AODYO INSTRUMENTS

anyma

User manual

VERSION 1.1.0
OCTOBER 26, 2022
www.aodyo.com

AODYO INSTRUMENTS

www.aodyo.com

Contents

Overview 5

Front panel (7) • *Rear panel (9)* • *Included accessories (10)* • *First steps (11)* • *Playing with a wind controller (12)* • *Updating your Anyma Phi (13)*

The matrix 15

Introduction (15) • *The* Animate *line (16)* • *External controls (A/B/C/D) (17)* • *Morphing (19)*

The anatomy of a patch 21

Introduction (21) • *Oscillators (22)* • *Effects (24)* • *Audio signal path (26)* • *Mappings (28)* • *Modulators (29)* • *Velocity Envelope and expression (31)* • *Connecting the synth to the matrix (32)* • *Patch structure (32)* • *The bank (34)*

Editing patches 35

Zooming on a parameter (35) • *Jumping to a module (39)* • *The module editor screen (39)* • *Using the PC/Mac editor (40)*

Tuning and microtonality 45

Importing and exporting tunings (46) • *Importing tunings using the MTS-ESP Suite (47)* • *Creating and editing tunings (48)* • *Preset tunings (51)*

Settings 55

While in the matrix (55) • *While in a module editor screen (56)* • *Patch commands (56)* • *MIDI settings (57)* • *Tuning (62)* • *Remote control (64)* • *System settings (65)*

Shortcuts 67

Global actions (67) • *While in the matrix (68)* • *In a* Related mappings *screen (68)* • *In a* Module editor *screen (68)* • *In the* settings menu *(69)* • *On startup (73)*

MIDI implementation 75

Basic information (75) • *Control change list (76)* • *Matrix remote control (76)* • *NRPN remote control (78)* • *MIDI Tuning Standard (79)* • *Routing limitations (80)* • *MIDI patch dump and format (82)*

Release notes 83

Complementary information 95

Technical features (95) • *Credits and licenses (96)* • *Trademarks (100)* • *Disclaimer (100)*

Appendix: Module reference 103

General (105) • *Oscillators (128)* • *Effects (187)* • *Modulators (244)* • *Mappings (318)*

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.

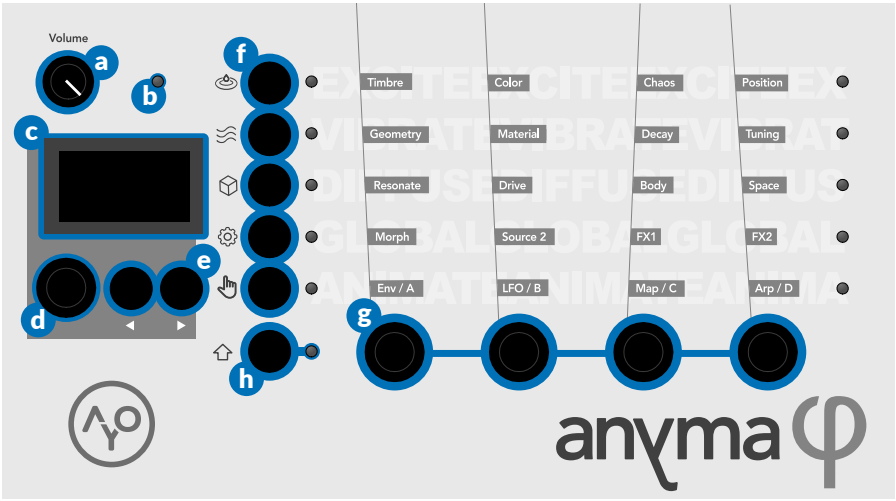
The Anyma Phi is a *monophonic* synthesizer, meaning it is designed to play a single voice at a time; that is, a melody played by a single instrument player or sequencer, generally without chords or harmony[1].

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 ^Y^o 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:

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.
4 *Arp* **Arpeggiator**
Disable or enable the internal arpeggiator of 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[2].

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] 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[3]. 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[4].

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[5]
D	CC 77	

[3] 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.

[4] 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.

[5] 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[6], 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.

[6] 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?

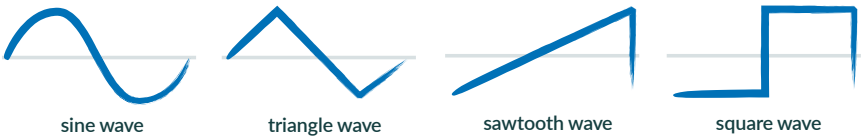
Oscillators

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).

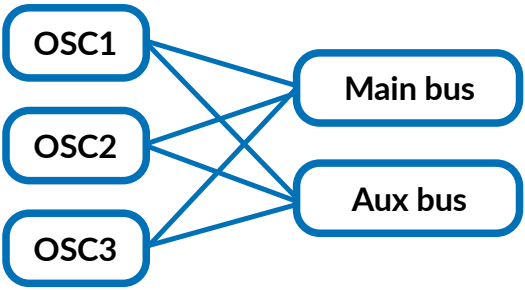
In the Anyma Phi, you can use up to five different effect modules, with an additional final *Reverb* effect that applies to the final sound as a whole. Now, these effects don't do anything in isolation, as they need to take sound from somewhere. So how are oscillators and effects interconnected?

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.

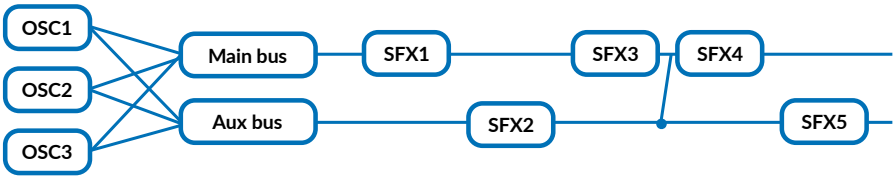
Your Anyma Phi can be seen as a *semi-modular* synth: the signal path is predetermined as well, but there are a few degrees of freedom as to how the signal flows, so that it is generally possible to achieve exactly what you want.

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.

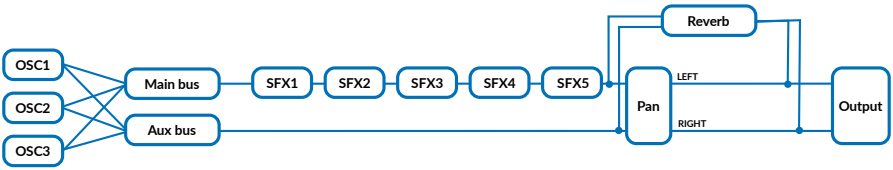


Now, each of the five effects (SFX1, SFX2, SFX3, SFX4, SFX5) are at a fixed place on the signal path, but they can be placed on either bus,

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.

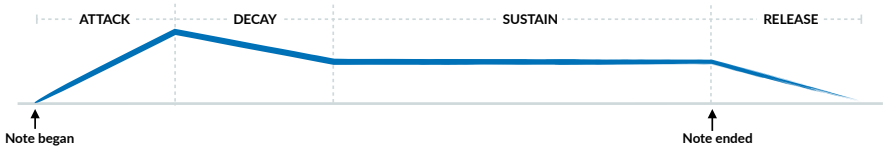
Modulators

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 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 **0SC1**, 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* D, 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*.

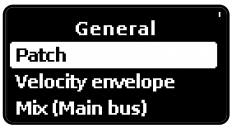
If you're content with them, you can **save** the patch using ⇧ + ▶, so that your modifications will not be forgotten once you turn off your Anyma Phi or switch to another patch.

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.

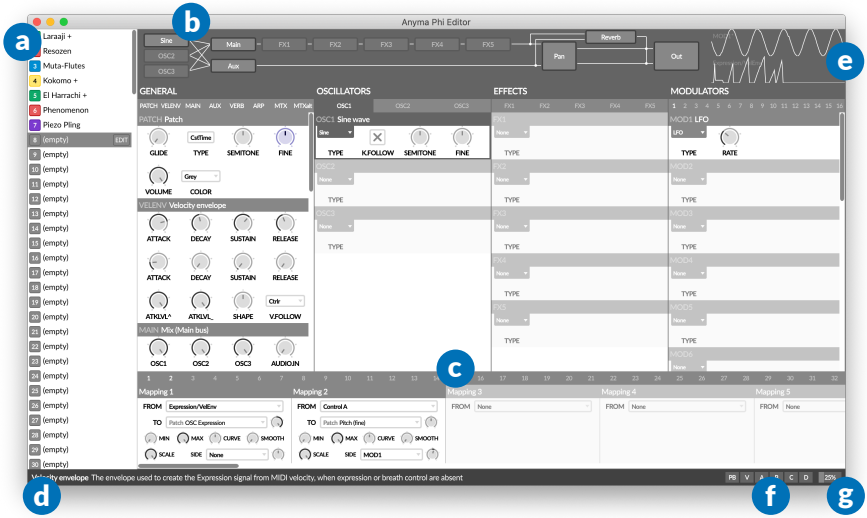
Connecting the editor

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.

Tuning and microtonality

You can tune your **Anyma Phi** to any reference frequency or tuning system, such as just intonation, historic or non-Western tunings, or those of modern microtonal and xenharmonic music.

In this chapter, you will learn how to interact with tunings, how they can be transferred in and out of your **Anyma Phi**, as well as how you can create and edit them directly on your device.

Up to 8 tunings can be stored, from *TUN1* to *TUN8*. Each tuning has a name and a definition, and can be created directly on the Anyma Phi or can be imported from other systems via the MIDI Tuning Standard. Take a look at the relevant section of the *Settings* chapter to see what you can do with a tuning.

By default, the global tuning is set to *None*, which means your Anyma Phi will be tuned to the standard 12-tone equal temperament (12-TET, or 12edo), with A4 sounding at 440 Hz. You can select a tuning from *TUN1* to *TUN8* to act as the global tuning, which will be used by the synth instead of the 12-TET. Each patch can also follow a specific tuning (with the `TUNING` parameter in the Patch module), but by default the global tuning will be followed.

When selecting or editing a specific tuning in the settings menu, you will be able to listen to its effect by playing your MIDI keyboard or controller, even if it is not selected as the global tuning or as the patch tuning. This will last until you exit the tuning menu.

Importing and exporting tunings

If you already have your tunings ready, either in the Scala (`.scl` or `.kbm`), Anamark (`.tun`), or MTS (`.syx`) format, you can import them to your Anyma Phi. Conversely, if you have created a tuning on your device, you might want to export it to another synth or to a computer.

The way to import and export tunings into your Anyma Phi is using the MIDI Tuning Standard (MTS). It defines a format of MIDI SysEx messages to represent full tuning tables and tuning changes. A full tuning table describes an exact target pitch for each of the 128 possible MIDI notes.

To import your tunings, you must first convert them to the MTS format. A few utilities exist to do this, such as *Scala*, or *the Scala to MTS Conversion webpage from microtonalsoftware.com*.

The result will be a `.syx` file that contains the MIDI SysEx message that should be transferred to the Anyma Phi to import the tuning. On Windows, you can use *MIDI-OX*, and on Mac you can use *SysEx Librarian*. With either program, you just need to select the `.syx` file, select the first MIDI port of your Anyma Phi, and send the message. If successful, your Anyma Phi will display a short message like *Received TUN1*. If you wish to keep the tuning permanently, you will need to go to the *Tuning* section of the settings menu and use the *Save* command.

Some utilities allow you to select the destination tuning program number; that is, which tuning will be chosen between *TUN1* (0) and *TUN8* (7). If there is no such option, then you can assume *TUN1* will be overwritten and you will have to copy/paste tunings around to reorder your list.

To export a tuning, you can use either of the three *Bulk dump* commands available in the tuning menu of each mapping, in the settings.

For more details on what is possible to remotely control the tunings of your Anyma Phi, please refer to the relevant section of the MIDI implementation chapter.

Importing tunings using the MTS-ESP Suite

To facilitate the creation and edition of tunings, you can use ODDSOUND's *MTS-ESP Suite* and connect it to your Anyma Phi.

Assuming the MTS-ESP Suite is installed, first add an instance of the *MTS-ESP Master* plug-in onto a track in your DAW or plug-in host. The Master plug-in is where you will be able to create and edit your tunings.

Then, add a new track with an instance of the *MTS-ESP MIDI Client* plug-in, which will be responsible for automatically sending any changes done in the *Master* instance to your Anyma Phi.

Set up the *MIDI Client* as follows:

- **Method:** MTS SYSEX
- **Output:** Anyma Phi
- **Auto send:** On
- **Throttle:** 100 ms (lower values can result in audible clicks)
- **Device ID:** 0 (or the same number as the *Univ. SysEx ch.* in the MIDI settings of your Anyma Phi)
- **Format:** Bulk
- **Bank:** Off
- **Prog:** 0 (will go to *TUN1*; select the destination as needed)
- **Checksum:** Calculate

Now, with these options in place, your Anyma Phi will automatically receive the tuning (by default, on *TUN1*) and changes in the *MTS-ESP Master* plug-in will be automatically taken into account by your Anyma Phi.

To ensure you can hear the changes in real time, you will need to go to the *Tuning* settings in your Anyma Phi, select the relevant tuning, and check *Global tuning*.

Later, when you are satisfied with the tuning, remember to *Save* it on your Anyma Phi.

Creating and editing tunings

In addition to importing and exporting tunings, there are several options that can be used to define and edit tunings directly on your Anyma Phi. Each tuning has a *type* that describes how it is defined, among the following options:

- *Reference tone*: Shift the pitch globally by tuning a specific note to a specific frequency, in Hz (e.g., A4 = 440 Hz).
- *12-tone octave scale*: Each of the 12 notes in the chromatic scale can be shifted up or down by a number of cents.
- *Tuning table*: A full keyboard mapping, unambiguously defining a specific pitch or frequency for each possible key (which is the format of a MTS dump).
- *Scale*: An arbitrary scale where you can define the number of degrees, the interval for each degree (as a ratio or a pitch in cents), and a note that is degree 0.
- *n-EDO*: The octave is divided into *n* equal degrees.

In the following, you will learn about each type of tuning definition in more detail.

Reference tone

This is the simplest kind of tuning. It amounts to setting the pitch of the synth so that a specific note is tuned to a specific frequency, shifting the frequency of all other notes as well. For instance, you may want to tune A4 to 442 Hz instead of the typical 440 Hz.

Typically, you will select the reference note (e.g., A4), and then tune it to a specific frequency. Different degrees of granularity are available. Shortcuts are detailed in the relevant section of the Shortcuts chapter.

12-tone octave

This is the form of tuning that is mostly encountered in synths with basic microtonal support. It allows you to shift each note of the chromatic scale (C, C#, D, D#, etc.) by a specific amount of cents (between -127 and 127), with a resolution of 1 cent. You can also select a reference tone (see the above subsection) in addition to the scale.

MIDI learn can be used to quickly select a note. See the relevant section of the Shortcuts chapter for details.

Tuning table

This is the most exhaustive way to define a tuning: each of the 128 possible MIDI notes is mapped to a specific target pitch, with a resolution of 0.0061 cents. Tunings received via MTS messages are defined in this way. The target pitches can be displayed in semitones or as frequencies in Hz.

You can use MIDI learn to quickly select a note, or directly tune a note by playing it on your controller (including pitch-bend). You can also edit the pitch with different degrees of granularity. See the relevant section of the Shortcuts chapter for details.

Scale

This is the most useful way of specifying a general repeating scale. Select a number of degrees, and for each degree, set its interval from the base degree (degree 0, or 1/1).

Each degree is set:

- either as an interval, in cents, separating it from the base degree,
- or as a frequency ratio[1] between its frequency and that of the base degree.

The last degree is always considered as the formal octave (often 2/1).

In addition, you can also select a note that will be mapped to degree 0, as well as a reference tone to shift the entire scale.

You can use MIDI learn to quickly select a degree. You can edit the intervals and ratios with different degrees of granularity. See the relevant section of the Shortcuts chapter for details.

[1] More exactly, a positive rational number greater than 1; i.e., $q = a/b$, with $a, b \in \mathbb{N}^+$ and $a \geq b$.

n-EDO

This form of tuning divides an octave (here, always a 2/1 frequency ratio) into a specific number of equal divisions. When there are *n* divisions, the scale is known as *n*-EDO, or *n*edo. For instance, the usual 12-TET of Western music is also known as 12edo.

Here, selecting the number of equal divisions is enough to define the scale. In addition, you can also select a reference tone to shift the scale.

Preset tunings

The following preset tunings are offered when choosing the *Load preset* submenu. When not indicated otherwise, the tuning has 12 tones. All the following preset tunings start with C4 and have C4 as the reference note as well.

Non-Western tunings

Pelog
A heptatonic scale used in Javanese gamelan.
Slendro
A pentatonic scale used in Javanese gamelan.
Rāgas (Bageshri, Bhairavi, Kafi, Todi, Yaman)
A series of heptatonic tunings that can be used in Hindustani rāgas.
22 Shruti
A tuning system for Indian music composed of 22 shruti or intervals.
Hirajōshi
A pentatonic japanese scale used for the tuning of the koto.

Well temperaments

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

Bach-Kellner
A temperament with 7 perfect fifths.

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

Kirnberger III
A temperament with a single perfect third.

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

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

Meantone temperament

1/4 comma meantone
The most commonly used meantone temperament in the 16th and 17th centuries.

1/3 comma meantone (Salinas)
A temperament first described by Francisco de Salinas in *De musica libri septem* (1577).

1/5 comma meantone (Verheijen)
A temperament attributed to Abraham Verheijen and Michelangelo Rossi.

1/6 comma (tritonic)
A meantone temperament making the tritone pure.

2/7 comma (Zarlino)
A temperament described by Gioseffo Zarlino in *Le istitutioni harmoniche* (1558).

Just intonation

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

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

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

7-limit tritone
A *basic just intonation* tuning with a septimal tritone.

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

SuperJust
Wendy Carlos' *SuperJust intonation scale*.

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

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

Centaura subh.
The original subharmonic version of *Centaura*.

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

Non-octave tunings

The following tunings do not include octaves.

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

Beta scale
The Beta scale proposed by *Wendy Carlos*, with 63.8 cents per step.
Gamma scale
The Gamma scale proposed by *Wendy Carlos*, with 35.1 cents per step.
Bohlen-Pierce equal
The equal-tempered form of the *Bohlen-Pierce scale*.
Bohlen-Pierce just
The just form of the *Bohlen-Pierce scale*.

Equal temperament

Pentatonic
A tuning system where the octave is divided in 5 equal steps (5edo) of 240 cents each.
Whole-tone
A tuning system where the octave is divided in 6 equal steps (6edo) of 200 cents each.
Heptatonic
A tuning system where the octave is divided in 7 equal steps (7edo) of approx. 171.4 cents each.
19edo
A tuning system where the octave is divided in 19 equal steps of approx. 63.2 cents each.
31edo
A tuning system where the octave is divided in 31 equal steps of approx. 38.7 cents each.
53edo
A tuning system where the octave is divided in 53 equal steps of approx. 22.6 cents each.

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).

Reset
Reset the current matrix (normal or alternate version, depending on which is active in the matrix screen) by setting all its macros to 0%.

Autofill
Tries to create mappings to the macros of the matrix that haven't been mapped yet, based on the contents of the patch.

Unmap all
Removes all the mappings related to matrix macros.

While in a module editor screen

The following commands do not work when editing a module in the *General* section.

Copy
Copy the current module to the clipboard.

Paste
Paste the current module in the clipboard, replacing the current module.

Swap with...
Swap the current module with another one of the same family (e.g., OSC1 with OSC3).

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.

Rename
Rename the patch. See the relevant section of the Shortcuts chapter for help.

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.

Randomize
Replaces the current patch by a random one.

Compactify
Regroups all the effects, modulators, and mappings so that there is no free slot in between.

Reload
Reload the patch from permanent memory, undoing all the unsaved modifications.

Dump to USB-Device
Send the current patch as a string of SysEx MIDI messages that can be captured with a MIDI utility. The captured messages can then be sent back to the Anyma Phi to replace the current patch.

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

Vel. curve
Set a global velocity curve applied to ongoing MIDI notes, with different strengths of logarithmic and exponential shapes.

Default: Linear

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

Default: ±2st

Expr. detect
Set the condition for detecting whether to use the expression CC or the velocity envelope. *Never* means the expression CC will be ignored, and the velocity envelope will always be used instead. Values *100ms* to *20s* mean that at the moment an expression CC is received, it will be used instead of the velocity envelope, but once the specified delay has passed and no further expression CC is received, it will revert to the velocity envelope until another expression CC is received. *Prg.Chg.* means the expression CC will be used from the moment it is received, but when switching to another patch the synth will revert to using the velocity envelope by default, until another expression CC is received.

Default: 100ms

Univ. SysEx ch.
Select the Universal SysEx channel to which the Anyma can respond.

Default: 0

CC mapping

For more details on how to select MIDI CCs using *MIDI learn*, please refer to the relevant section of the Shortcuts chapter.

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

Volume control CC
Set which MIDI message (*CC* or *Aftertouch*) is used as a second layer of volume control, for further dimming global output volume defined by the volume knob. The volume level set through MIDI will not be remembered after power-off.

Default: CC 7

Hold pedal CC
Set which MIDI message (*CC* or *Aftertouch*) is used to as the hold pedal control, sustaining all ringing notes until the control is off. The hold pedal state can also be used in patches.

Default: CC *64*

CC Output Channel
Set the MIDI channel used to output CC values from patches (mapping destinations *CC Out 1* to *4* in the *Matrix* module). You can use this to target a specific synth in your setup.

Default: 1

CC Output To USB device / DIN / USB host
Whether to output MIDI CC values from patches to each port (USB device, DIN, USB host). You can use this to target a specific synth in your setup.

Default: ○ Disabled

CC Output 1 to 4
Set which MIDI message (*CC* or *Aftertouch*) is output based on mapping destinations *CC Out 1* to *4* (in the *Matrix* module) in the current patch.

Default: CCs 70, 71, 72, 73

MIDI routing

More details about the limitations of the routing options are available in the related section of the MIDI implementation chapter.

Safe merge
When disabled, merging occurs as fast as possible, but there is a possibility data loss when simultaneous messages are merged. When enabled, all the input data is reinterpreted before being merged, guaranteeing zero interleaved or garbled MIDI message in the output. However, this comes at the cost of a possible slight increase of latency and a limit on SysEx message length. Limitations and tradeoffs are explained in the related section of the MIDI implementation chapter

Default: ○ Disabled

DIN Thru
When enabled, all the data received from the MIDI DIN input port will be recopied to the MIDI DIN output port (*soft thru*).

Default: ○ Disabled

DIN > USB Host
When enabled, all the data received from the MIDI DIN input port will be recopied to the USB host port.

Default: ○ Disabled

USB Host > DIN
When enabled, all the data received from the USB host port will be recopied to the MIDI DIN output port.

Default: ○ Disabled

Clock sync

External clock
Accept realtime MIDI Clock messages from any port.

Default: ● Enabled

Internal clock Tempo
Set the tempo (in BPM) of the internal clock that is used when no external clock is detected.

Default: 120 BPM

Tuning

By default, the global tuning is set to *None*, which means your Anyma Phi will be tuned to the standard 12-tone equal temperament (12-TET, or 12edo), with A4 sounding at 440 Hz. To set another tuning as the global tuning, select it from the list, and once in the tuning menu, select *Global tuning*. Each patch can also follow a specific tuning (with the `TUNING` parameter in the Patch module), but by default the global tuning will be followed.

When selecting or editing a specific tuning, you will be able to listen to its effect by playing your MIDI keyboard or controller, even if it is not selected as the global tuning or as the patch tuning. This will last until you exit the tuning menu.

The tuning menu offers the following options for each tuning:

Global tuning
When selected, the tuning is selected as the global tuning. To select another one, go back to the tuning selection menu using ◀, select the tuning you

want, and enable its *Global tuning* option. Global tunings can also be selected via MIDI.

Default: ○ Disabled

Type

Select the type of tuning definition to use, among the following: *Reference tone*, *12-note octave scale*, *Tuning table*, *Scale*, and *EDO*. Confirm with ▶ or D, and cancel your choice with ◀. Whenever possible, the tuning will be preserved.

Default: None

Edit

Edit the tuning. The tuning editor is different depending on the type of tuning definition you chose, and will be detailed below.

Load preset

Load a preset tuning, among non-western, well temperaments, meantone scales, just intonation, non-octave scales, and equal temperaments. A list of preset tunings is available in the dedicated chapter.

Copy

Copy the tuning to the clipboard.

Paste

Paste the tuning in the clipboard, replacing the current one. However, the changes will not be automatically saved, allowing you to return to the previously saved version by using the *Reload* command.

Rename

Rename the tuning. See the relevant section of the Shortcuts chapter for help.

Save
Save the tuning. This command is only available when there are unsaved modifications.

Reload
Forget all unsaved modifications, returning to the previously saved version of the tuning.

Reset
Reset the tuning to an empty one. However, the changes will not be automatically saved, allowing you to return to the previously saved version by using the *Reload* command.

Bulk dump via...
Send a MIDI Tuning Standard bulk dump of the tuning, using the Universal SysEx channel defined in the MIDI settings. The bulk dump can be sent either to the USB-MIDI device port, to the MIDI DIN output port, or to the USB-MIDI host port.

For more details on how tuning works and how to import, export, create, and edit tunings, please read the Tuning chapter.

Remote control

Matrix CC
Accept specific MIDI CCs (if not used otherwise) to remotely control the matrix parameters. A detailed list is available in the Matrix remote control section.

Default: ○ Disabled

NRPN
Accept specific MIDI NRPNs (if NRPN CCs are not used otherwise) to remotely control any parameter of the patch. A general description is available in the

NPRN remote control section, and parameter addresses can be found in the module reference.

Default: ○ Disabled

System settings

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

Default: 4

Warn if unsaved
Ask for confirmation before switching to another patch (using D or a MIDI program change[1]) if there are unsaved changes in the current patch. When asked, press ▶ to discard unsaved changes and switch to the other patch, or press ◀ to cancel and go back to the unsaved patch.

Default: ○ Disabled

Randomize empty
When loading an empty patch, start from a random patch instead of the default empty one. Don't forget to save your patch if you like the result!

Default: ○ Disabled

Large type
Display the name and current value of a parameter for a short time when modifying it or pressing its encoder.

Default: ○ Disabled

[1] You will not be asked for confirmation when switching to another patch from the PC/Mac editor.

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.

Default: *Minimum* 156, *Range* 93

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 ◀.

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).
Keep ⇧ **pressed for more half a second**
Display the amount of resources used by the current patch, as a progress bar. The resources must stay under 100% to avoid glitches and dropouts.

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.

More shortcuts are available in some specific situations, which are detailed below.

When selecting a MIDI CC

Play a MIDI CC using a controller
When *MIDI learn* is enabled (cf.), change the selected CC to the one played.
Press
Toggle *MIDI learn*.

When renaming something

The existing full name appears in small on the upper left corner of the display, as well as in large letters in the middle of the display.

A blinking cursor allows you to move among the existing characters and insert new ones, scrolling the large-lettered name in the middle if needed. When the cursor is at the end of the name, a semitransparent letter appears to indicate that another character can be appended.

Turn D
Change the character at the cursor location.
Press/hold ◀ or ▶
Move the cursor to the previous or next character, respectively.
Press D, or press ▶ when the cursor is one character past the end
Confirm the changes and return to the previous screen.
Press ◀ when the cursor is at the beginning
Cancel any change and return to the previous screen.
Press ⇧
Switch between uppercase and lowercase for the character at the cursor location.
Press/hold ⇧ + ◀
Remove the character before the cursor (backspace).
Press/hold ⇧ + ▶
Remove the character at the cursor (delete).
Turn ⇧ + D
Insert a character before the cursor.

When selecting a note in a tuning

Play a MIDI note
When *MIDI learn* is enabled (cf. ☝), select the note that is being played (in the 12-TET).
Press ☝
Toggle *MIDI learn*.

When selecting a reference frequency in a tuning

Press ⇧
Reset the frequency to that of the reference note (in the 12-TET).
Turn D
Change the frequency by 1 Hz.

Turn 1
Change the frequency by 0.1 Hz.
Turn 2
Change the frequency by 0.01 Hz.
Turn 3
Change the frequency by 0.001 Hz.
Play a MIDI note
When *MIDI learn* is enabled, select the frequency of the note that is being played (in the 12-TET).
Press
Toggle *MIDI learn*.

When tuning a note in a tuning table

Press ⇧
Reset the target pitch to the default value (the pitch of the note in the 12-TET).
Press ▶
Toggle the displayed unit between semitones and Hz.
Turn D
Change the target pitch by 1 semitone (independently of the displayed unit).
Turn 1
Change the target pitch by approx. 10 cents (independently of the displayed unit).
Turn 2
Change the target pitch by approx. 1 cent (independently of the displayed unit).
Turn 3
Change the target pitch by 0.0061 cent, which is the finest resolution available (independently of the displayed unit).
Play a MIDI note
When *MIDI learn* is enabled (cf.), change the target pitch to the note that is being played (in the 12-TET), taking pitch-bend into account too.

Press
Toggle *MIDI learn*.

When tuning a degree in a scale

Press
Reset the degree to the value of the preceding one.

Press
When the rightmost field is selected, toggle between tuning the degree as an interval from degree zero in cents, or as an exact frequency ratio. Note that a small amount of precision can be lost when doing the conversion.

WITH AN INTERVAL IN CENTS

Turn
Change the interval by 100 cents.

Turn
Change the interval by approx. 10 cents.

Turn
Change the interval by approx. 1 cent.

Turn
Change the interval by 0.0061 cent, which is the finest resolution available.

WITH A FREQUENCY RATIO

Press **or**
Select the numerator or the denominator, depending on the currently selected field.

Turn
Change the selected field (numerator or denominator), making sure that the ratio is always superior to 1/1.

Turn
Change the numerator, making sure that the ratio is always superior to 1/1.

Turn
Change the denominator, making sure that the ratio is always superior to 1/1.

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.

Power on while holding **+**
Reset the settings to their default values, as well as all the patches, without asking for confirmation.

MIDI implementation

Basic information

Function	Transmitted	Recognized	Remarks
Basic channel	1-16	1-16	Default: all.
Note number		0-127	
Note On Velocity		Yes	
Note Off Velocity		No	
Program Change	Yes	Yes	
Channel Aftertouch		Yes	
Poly (Key) Aftertouch		No	
Pitch Bend		Yes	
Control Change	No	Yes	See Control change list and Matrix remote control.
Clock	No	Yes	
Start sequence	No	Yes	Resets sync triggers.
Universal SysEx	Yes	Yes	MIDI Tuning Standard and Identity Request.
System Exclusive	Yes	Yes	Private. Manufacturer ID: `00h 21h 33h`.
RPNs	No	Yes	See MIDI Tuning Standard.
NRPNs	No	Yes	See NRPN remote control.

Control change list

The following control changes can be configured in the settings.

CC	Control
2, 11	Expression
1	Control A
75, 74	Control B
76, AT[1]	Control C
12, 84	Control D
7	Volume
64	Hold pedal

In addition, if not mapped to expression or Control A/B/C/D, the following control changes are supported:

CC	Action
120 *All Sound Off*	Turn off any ringing note.
121 *Reset All Controllers*	Reset all the controls in the above table.
123 *All Notes Off*	Turn off any ringing note if the hold pedal isn't pressed.

Matrix remote control

The following control changes can be received when the *Matrix CC Remote control* setting is enabled. This is unidirectional: changes in matrix parameters will not generate MIDI CCs to external controllers.

[1] Channel Aftertouch.

CC	Matrix parameter
16	*Timbre*
17	*Color*
18	*Chaos*
19	*Position*
20	*Geometry*
21	*Material*
22	*Decay*
23	*Tuning*
24	*Resonate*
25	*Drive*
26	*Body*
27	*Space*
28	*Morph*
29	*Source 2*
30	*FX1*
31	*FX2*
102	*Timbre* (alt. matrix)
103	*Color* (alt. matrix)
104	*Chaos* (alt. matrix)
105	*Position* (alt. matrix)
106	*Geometry* (alt. matrix)
107	*Material* (alt. matrix)
108	*Decay* (alt. matrix)
109	*Tuning (fine)* (alt. matrix)
110	*Resonate* (alt. matrix)
111	*Drive* (alt. matrix)
112	*Body* (alt. matrix)
113	*Space* (alt. matrix)
115	*Source 2* (alt. matrix)

CC	Matrix parameter
116	⚙ *FX1* (alt. matrix)
117	⚙ *FX2* (alt. matrix)
118	👆 *Arpeggiator Enable*

NRPN remote control

Any parameter of the current patch can be controlled by sending NRPNs if the *NRPN Remote control* setting is enabled. This is unidirectional: changes in patch parameters will not generate MIDI NRPNs to external controllers.

As per the MIDI specification, parameter changes must be sent as follows:

- **NRPN MSB** *(CC 99)*: Module number
- **NRPN LSB** *(CC 98)*: Parameter number
- **NRPN Value** *(CC 6)*: Parameter value

Data increment and decrement messages are not implemented.

The following example MIDI message sequence describes how to use NRPNs to setup 0SC2 in the patch as a square wave oscillator, with a pulse width of 50%.

MIDI message	Description
CC 99 (*NRPN MSB*) 9	Select module number 9 (0SC2)
CC 98 (*NRPN LSB*) 0	Select module parameter 0 (TYPE)
CC 6 (*NPRN Value*) 13	Set TYPE of 0SC2 to *Square wave*
CC 99 (*NRPN MSB*) 9	Select module number 9 (0SC2)

MIDI message	Description
CC 98 (*NRPN LSB*) 4	Select module parameter 4 (PWM)
CC 6 (*NRPN Value*) 127	Set PWM of OSC2 to *100%*
CC 6 (*NRPN Value*) 64	Set PWM of OSC2 to *50%*

The module reference details the appropriate NRPN and values for every accessible module and parameter.

MIDI Tuning Standard

Basic support for MIDI Tuning Standard messages allow MIDI-based modification of any existing tuning via Universal SysEx, as well as the selection of the global tuning (via RPN).

Messages with an invalid tuning program number are ignored. A tuning program number of 0 corresponds to *TUN1*, and so on.

The following details the supported messages and the expected behavior.

Universal SysEx

The Universal SysEx channel can be selected in the MIDI settings.

Bulk Dump Request *(08h 00h, non-realtime)*
Upon reception, sends a *Bulk Dump Reply* message with the full tuning table corresponding to the given number.

Bulk Dump Reply *(08h 01h, non-realtime)*
Upon reception, replaces the tuning at the given number with the full table described in the bulk dump. The changes are not saved until the tuning is explicitly saved in the *Tuning* menu.

Single Note Tuning Change *(08h 02h, realtime)*
Upon reception, changes the tuning at the given number to reflect all the note changes expressed in the message, possibly converting the tuning into a full table. The changes are not saved until the tuning is explicitly saved in the *Tuning* menu.

RPN

Tuning Program Select *(03h 00h)*
Upon reception, set the global tuning to the given number. The change is saved.

Data entry messages are implemented, but data increment and decrement are not implemented.

Routing limitations

When the Anyma Phi must merge two or more streams of MIDI data into one, which occurs for instance when at least one of the MIDI Routing settings is enabled, some tradeoffs and limitations apply. It is important to be aware of them so that you can choose the right option for your use case.

If the *Safe merge* option is disabled, the merging occurs at the byte level, regardless of the structure of the actual MIDI messages that constitute the streams. This provides the shortest latency possible, ensures that the data is transmitted exactly as it is sent, and without any limitation regarding SysEx length.

It is the best option for the simplest cases, where only one input is sending data to the merged stream at a any given time. But this also means that when several inputs routed to the same output are sending data at

the same time, there is a risk that MIDI messages interleave such that it is not possible to read them back correctly at the receiving end. This is more likely to happen when the emitting devices output MIDI at a different speed.

If the *Safe merge* option is enabled, the merging occurs at the message level, meaning that the Anyma Phi will ensure that each MIDI message received will never be interleaved with another one when both are merged. To ensure this, however, means accepting several other limitations:

- a message might not be transmitted exactly as it was sent[2] (e.g., running status might disappear);
- if two messages arrive approximately at the same time from two inputs competing for the same stream, their relative order in the merged stream might not be conserved, possibly meaning increased latency;
- SysEx messages with more than 1024 bytes are not supported, and will be dropped from the merged stream, as well as incomplete SysEx messages.

Thus, depending on your own setup, you might want to choose either of these options. In practice, we recommend that *Safe merge* be disabled by default, and to enable it only if witnessing or suspecting interleaved or garbled MIDI messages in your setup.

[2] No information is ever lost (this is a *safe* merge after all), but a different form for the same information might be used, possibly leading to increased thoughput and latency.

MIDI patch dump and format

The *Patch Dump to USB-Device* command sends a series of SysEx messages containing the current patch to the USB Device port. These messages can then be captured into a file using a MIDI tool such as *MIDI-OX* on PC and *SysEx Librarian* on Mac.

The files containing the patch dump can then be replayed to the Anyma via the USB Device port to replace the currently selected patch. They can also be imported inside the PC/Mac editor, just like another patch.

As of v1.0, when exporting a patch from the PC/Mac editor, the created file has a format similar to a patch dump, which also means that any patch you download can be loaded into your Anyma by replaying its content using a MIDI tool, without needing the editor.

Release notes

v1.1.0

- Added 4 New WaveTable OSC (WaveTable, 2D WaveTable, Interpolated WaveTable, 4-Voices WaveTable)
- Added 4 Extra Virtual Analog OSC (Folded Sine/Tri, Dual HardSync Osc, Triple Oscillator, SawSwarm)
- Added 1 new Noise OSC (circuit Bent Toy)
- Added new LUT16 Modulator
- "Other" oscillator category renamed as "Paraphony" for easier understanding
- Modulator list reorganization (New LFO, Envelope & Sequencer categories)

v1.0.0

- There is now an *Arpeggiator* module to turn polyphonic note input into arpeggios.
- Added a 16-step *Sequencer* modulator.
- Added a *Delay (sync)* effect and a *LFO (sync)* modulator (in the *Change* group), which are tempo-synced versions of the same modules.
- Now supports up to 8 tunings (*TUN1* to *TUN8*) that can be imported/exported via MTS SysEx messages. You can select a global tuning, and each patch can select a specific tuning among *TUN1* to

TUN8. The on-board tuning editor allows you to define a tuning in several ways (reference note, 12-tone octave scale, full tuning table, scale, and n-EDO).

- Added three new matrix commands:
  - *Reset* sets all the macros of the matrix (or its alternate version) to 0%.
  - *Autofill* tries to add mappings for the unmapped macros in the matrix, based on your patch.
  - *Unmap all* removes all the mappings related to the matrix.
- Added three new patch commands:
  - *Randomize* creates a random patch.
  - *Compactify* removes the free space in the effects, modulator, and mappings sections.
  - *Reload* removes unsaved changes.
- Added an option to randomize empty patches.
- Added more options to the *Envelope follower* modulator (noise floor, gain).
- Added a *Drum trigger* modulator that generates an expression signal suitable for triggering percussive oscillators.
- Added a *Timbre follower* modulator that extracts the overall brightness of an audio signal.
- Added a new oscillator group, the *Exciters*, which are meant to be used with a resonator effect. Three exciters are available:
  - *Bow* simulates the raw sound of a bow scratching a material.
  - *Wind* simulates a variety of continuous blowing, breathing, or wind noises.
  - *Strike* simulates a variety of impulses and percussive noises:

hammers, mallets, sticks, plectrums, or bouncing particles (using the *Mallet* parameter).

- Added a *Snare drum* effect, in the *Resonator* group, that simulates the propagation and reflection of the input waveform exciting a snare drum.
- Added a *Drawbar organ* oscillator, as well as two *Rotating speaker* effects.
- Added a *Granular processor* effect that creates audio textures and soundscapes by combining short segments (or grains) of the input.
- Added three dynamics effects: *Compressor*, *Dynamics booster*, and *Noise gate*.
- Added a *Virtual analog* oscillator with continuous waveform selection.
- Added a *FM Operator* effect (in the *Modulation* group) that uses the input as a modulation for its own internal virtual analog oscillator, for making mixed FM sounds.
- Added a *Dirty formant filter* effect that works pretty much like the *Artin VFOF* oscillator, but applied to any audio input.
- Added a *Ping-pong delay* effect, together with a *sync* variant.
- Added a few new value, gate and trigger sources related to polyphonic note input (*Paraphonic* group), to assist in making paraphonic patches.
- Added a new *Replicant* oscillator that copies the previous oscillator in the list but allows for changes in expression input and pitch, which can be useful to build paraphonic patches.
- Added a few new controller triggers (new note played, last note released, note transition).

- You can now rename patches and tunings directly from your Anyma Phi. When renaming, turn D to select a letter, use ◀ and ▶ to move the cursor, and press D to confirm.
- Added a *Dump patch* command to send the current patch as a SysEx message chain through the USB device port without needing the editor. A patch can then be dumped back to the Anyma as SysEx to replace the current one.
- Changed the format of exported patches in the PC/Mac editor to allow the files to be directly dumped as SysEx. Patches saved with previous versions can still open without issues, but patches saved with this version cannot be open in previous versions.
- A few modulators can now output useful triggers into up to 4 custom trigger channels (*TC1* to *TC4*), which can then be used a trigger sources for percussive oscillators or other modulators like envelopes. Look for parameters named *TC.OUT* in modulators like *DAHDSR Envelope* or *Compare*.
- Added a *Compare* modulator that determines whether a signal goes above or below a threshold.
- Added a *Logic* modulator group with five modulators providing different tools to work with triggers and gates (Gate combinator, Trigger combinator, Gate to trigger, Gate delay, Trigger delay).
- Added a *Slow LFO* variant of the *Advanced LFO* modulator (in the *Change* group) with periods between a second and a full day.
- Added a *Change polarity* modulator that turns unipolar (0% to 100%) values into bipolar ones (-100% to 100%), and vice-versa.
- Added a new modulator group, *Physics*, that provides physical models and simulations:

  - *Bouncing ball* simulates the movement of a ball thrown in the air and bouncing over a hard surface.
  - *Ball impact* simulates up to 8 independent bouncing balls thrown in the air and bouncing over a hard surface, and outputs their velocity where there are impacts, making it more suitable for modulating the expression of percussive oscillators.
  - *Spring-damper system* simulates a mass attached to the input with a damped spring, making a "springy" or "bouncy" version of the input.
- Added a new modulator group, *Algorithmic*, that provides various algorithm-based utilities to generate sequences:
  - The *Euclidean sequencer* generates a large variety of rhythms whose beats are as evenly spaced as possible.
  - The *Hexadecimal sequencer* generates a sequence based on the binary representation of a 32-bit hexadecimal number, with each of its 8 digits controllable individually.
- Added a new modulator group, *Chaos*, that provides various ways to generate chaotic sequences:
  - *Logistic map*, *Tent map*, and *Circle map*, each with controllable parameters.
  - *Discrete chaotic map*, which allows continuous switching between Duffing, Gingerbreadman, Bernoulli and Hénon maps.
  - *Cellular automaton*, with 32 cells, selectable rule and initial state, as well as trigger outputs for 3 different cells, whose indices can modulate.
- A few modules now have an *Oversampling* parameter that allows

increases in sound quality at the expense of a higher patch load.
- When selecting a CC in the *CC mapping* settings, your Anyma can now learn from the MIDI input. You can disable this feature by unchecking the *MIDI learn* setting at the bottom of the list, or by pressing the *Animate* matrix button while editing a CC. The same feature exists for tunings, where you can select specific scale degrees or notes using any MIDI input.
- Added four *CC Out* destinations in the *Matrix* module, that can be used so that every patch can output different MIDI CCs to an external device. The MIDI CCs, channel, and ports can be setup in the *CC mapping* settings.
- The volume knob now behaves more linearly.
- MIDI routing now offers a *Safe merge* option with a different set of tradeoffs.
- Added a *Velocity curve* setting to change the global velocity response of the synth.
- The *Reset* settings submenu can now reset the entire bank and settings of the Anyma.
- Improved the display and selection of mapping destination parameters on the Anyma.
- In the module editor screen, modules can now be copied, pasted and swapped.
- In the PC/Mac editor, modules can now be copied and pasted.
- Enabled drag-and-drop for empty modules in the PC/Mac editor.
- Now supports external MIDI clock sync, and provides an internal clock if no external clock is detected.
- Now supports external MIDI hold pedal (by default, CC 64).

- Added a *Remote display* option in the editor (*View* menu) allowing to mirror the display of the Anyma, allowing you to record clean videos of the display by capturing the editor window on your computer screen.
- Added a secret *Test tone* mode (hold any matrix button then press Shift) whose workings are left as an exercise to the reader.
- Fixed the *Sylphinet* preset of the *Windsyo* oscillator.
- Fixed the *Artin VOSM* oscillator.
- The Mac editor is now a Universal application, with native Apple Silicon support.
- Fixed some bugs leading to the Anyma Phi being frozen.
- Fixed some bugs leading to the PC/Mac editor not being able to communicate with the Anyma Phi.
- Added more feedback and confirmation options in the PC/Mac editor for bank operations.
- Added an *Expr. detect* option to choose the condition used by the synth to switch between expression CC input and velocity envelope.
- Limited Linux support: the editor is now also available for Linux x64, however due to limited resources Aodyo Instruments is not able to extensively test it and provide support for this particular version.
- Changed the order in the oscillator list to lump all pitch-insensitive oscillators at the beginning.

v0.8.107

- This is the first version compatible with Anyma Phis produced after November 2021: if you bought your Anyma after this date,

earlier versions than this one may not work at all.
- Added a *Delay* modulator.
- Now supports external MIDI volume control (by default, CC 7).
- Added flexible *MIDI routing* settings, including DIN soft thru (limitations are detailed in the *MIDI implementation* chapter of the user manual).
- Changing a parameter can display its value in large type, if the *Large type* setting is enabled.
- When switching to another patch while there are unsaved changes, you can be asked for confirmation, if the *Warn if unsaved* setting is enabled.
- Module editor screens now remember the last modified parameter of each module, until patch change or power-off.
- Matrix parameters can be remote controlled via MIDI CCs when the *Matrix CC Remote control* setting is enabled.
- All patch parameters can be remote controlled via MIDI NRPNs when the *NRPN Remote control* setting is enabled.
- On the device, you can now view the patch load by holding ⇧ for half a second.
- On the editor, when exporting a patch using the menu, the filename is pre-filled with the patch name.
- Fixed a bug where patch names and colors weren't correctly reported when switching to another patch.
- Fixed a bug where the editor wasn't responsive when attempting to save a patch with unmodified contents (e.g., after import or rename).
- Fixed a bug where patch renames were not saved until another

modification was done to the patch.
- Improved the behavior of percussive oscillators with wind controllers.
- Fixed formatting issues and links in the manual.
- Fixed a bug where adding or modifying a mapping from the Anyma was not reflected in the editor.
- Increased USB host compatibility with USB-MIDI devices advertising multiple device classes (e.g., Akai MPK Mini).
- Added dimensions and weight in the technical features section of the manual.
- Added a MIDI implementation chapter in the manual.

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
- 225x120x70mm, 0.9 kg
- 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
  - Support for microtonality, with 8 arbitrary tunings, fully editable on-board, and with MTS SysEx support, with both global tuning selection and per-patch
- Power supply requirements: 5V DC, 1000 mA, standard IEC 60130-10 Type A (5.5x2.1mm) barrel connector

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, as well as modifications from the *Parasites hacks* by Matthias Puech, available under the same license.

Copyright (c) 2012-2015 Emilie Gillet

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

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

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

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

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

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

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

Copyright (C) 2011 Paul Mineiro All rights reserved.

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

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

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

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

Contact: Paul Mineiro `paul@mineiro.com`

Portions of the internal software and the PC/Mac editor use the *Heatshrink* data compression library.

Copyright (C) 2013-2015, Scott Vokes `vokes.s@gmail.com`
All rights reserved.

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

THE SOFTWARE IS PROVIDED “AS IS” AND THE AUTHOR DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL

> THE AUTHOR BE LIABLE FOR ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

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*.

> Copyright (c) 2001, Jason Kottke (jason@kottke.org), with Reserved Font Name Silkscreen.
>
> Copyright (c) 2010-2020, tyPoland Lukasz Dziedzic (latofonts.com), with Reserved Font Name Lato.
>
> Copyright (c) 2016, Dave Gandy (fontawesome.com), with Reserved Font Name Font Awesome.

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**.

HOW TO READ NRPN MENTIONS

In this reference, MIDI NRPNs may be indicated as `<MSB>:<LSB>`[a], and ranges of possible numbers are indicated as `[min-max]`[b].
A complete NRPN assignment is indicated as `<MSB>:<LSB> = <value>`.
When a module has several instances, such as oscillators, effects, modulators, and mappings, its first parameter (*LSB* 0) is always its type.

[a] For instance, `9:1` refers to *MSB* 9 and *LSB* 1, which is the second parameter of the OSC2 module.
[b] For instance, `[0-4]` means that any value from 0 to 4 (included) can be used.

Module name	**Module number** (NRPN MSB)
Patch `PATCH`	0
Velocity envelope `VELENV`	1
Mix (Main bus) `MAIN`	2
Mix (Aux bus) `AUX`	3
Reverb `VERB`	4
Arpeggiator `ARP`	5
Matrix `MTX`	6
Matrix (alt.) `MTXalt`	7
Oscillators `OSC1 to OSC3`	8 to 10
Effects `SFX1 to SFX5`	11 to 15

Module name	**Module number** (NRPN MSB)
Modulators `MOD1 to MOD16`	16 to 31
Mappings `MAP1 to MAP32`	32 to 63

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
Arpeggiator Turns polyphonic note input into arpeggios	ARP
Matrix Macros presented in the matrix	MTX
Matrix (alt.) Alternate version of the matrix (for morphing)	MTXalt

Patch (PATCH)

Global patch parameters.

Volume `VOLUME`

The volume of the entire patch.

0 to 100%

NRPN 0:4

Patch color `COLOR`

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

NRPN 0:5 = [0-8]

- *Grey*

  NRPN = 0
- *Red*

  NRPN = 1
- *Orange*

  NRPN = 2
- *Yellow*

  NRPN = 3
- *Green*

  NRPN = 4
- *Teal*

  NRPN = 5
- *Blue*

  NRPN = 6
- *Violet*

  NRPN = 7
- *Pink*

  NRPN = 8

Pitch (semitones) `SEMI`

The global pitch correction on the controller.

-64 to +63 semitones

NRPN 0:2

Pitch (fine) `FINE`

The global pitch correction on the controller.

-100 to +100 cents

NRPN 0:3

Glide time `GLIDE`

The time taken to smoothly transition between two notes.

0 to 5 seconds

NRPN 0:0

Glide type `TYPE`

The kind of smooth transition between notes to use.

NRPN 0:1 = [0-1]

- *Constant-rate*

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

  NRPN = 0

- *Constant-time*

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

  NRPN = 1

Tuning `TUNING`

The tuning to use for this patch.

NRPN 0:6

Tune pitch modulations `TUN.MOD`

Whether pitch modulations (like bend) are affected by the tuning.
This parameter is only effective when using a tuning that is not the default equal temperament (or *12-TET*). It allows you to apply the tuning:

- either over the input note, before anything else like pitch-bend, `SEMI/FINE`, or mappings,
- or at the very last moment, after all modulations and mappings have been applied, when the synth needs to determine the frequencies of oscillators.

When this parameter is enabled, tuning is applied at the very last moment, affecting the interpretation of all pitch modulations.

For instance, with the default 12-TET tuning, if `SEMI` is set to +2 semitones, playing a C on the controller will result in hearing a D. However, with a tuning where the interval between two consecutive notes is doubled (200 *cents* instead of 100), and if this parameter is enabled, you would hear a E, because the amount "2 semitones" would be interpreted in the context of the tuning (so 400 cents), and not as "2 semitones as defined in the equal temperament". Similarly, with a tuning where the distance between two consecutive notes can vary, the actual pitch difference heard would not be the same everywhere on the keyboard.

NRPN 0:7

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
NRPN 1:0

Decay time (MaxVel) DECAY
The time needed to go from 1 to the sustain level in the decay phase.
0 to 8 seconds
NRPN 1:1

Sustain level (MaxVel) SUSTAIN
The output level during the sustain phase.
0 to 100%
NRPN 1:2

Release time (MaxVel) RELEASE
The time needed to go from the sustain level to 0 in the release phase.
0 to 8 seconds
NRPN 1:3

Attack time (MinVel) `ATTACK`

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

0 to 8 seconds

NRPN 1:4

Decay time (MinVel) `DECAY`

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

0 to 8 seconds

NRPN 1:5

Sustain level (MinVel) `SUSTAIN`

The output level during the sustain phase.

0 to 100%

NRPN 1:6

Release time (MinVel) `RELEASE`

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

0 to 8 seconds

NRPN 1:7

Attack level (MaxVel) `ATKLVL^`

The output level at the end of the attack phase.

0 to 100%

NRPN 1:8

Attack level (MinVel) `ATKLVL_`

The output level at the end of the attack phase.

0 to 100%

NRPN 1:9

Shape `SHAPE`

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

NRPN 1:10

Velocity follow `V.FOLLOW`

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

NRPN 1:11 = [0-2]

- *Controller*

  Follow the velocity of the controller.

  NRPN = 0

- *Min. velocity*

  Use a fixed velocity of 1.

  NRPN = 1

- *Max. velocity*

  Use a fixed velocity of 127.

  NRPN = 2

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%
NRPN 2:0

OSC2 Level `OSC2`
The level of OSC2 sent to the Main bus.
0 to 100%
NRPN 2:1

OSC3 Level `OSC3`
The level of OSC3 sent to the Main bus.
0 to 100%
NRPN 2:2

Audio input Level `AUDIO.IN`
The level of the audio input sent to the Main bus.
0 to 100%
NRPN 2:3

Level `LEVEL`

The output level of the Main bus.

0 to 100%

NRPN 2:4

Reverb Send `REVERB`

The level of the Main bus sent to the reverb.

0 to 100%

NRPN 2:5

Pan `PAN`

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

100% Left to 100% Right

NRPN 2:6

Audio input Source `AIN.SRC`

An external source for the audio input.

NRPN 2:7 = [0-3]

- *White noise*

  A random audio source.

  NRPN = 0

- *Left channel*

  The left channel of the audio input.

  NRPN = 1

- *Right channel*

  The right channel of the audio input.

  NRPN = 2

— *Audio input (sum)*

Both channels of the audio input summed.

NRPN = 3

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%
NRPN 3:0

OSC2 Level `OSC2`
The level of OSC2 sent to the Aux bus.
0 to 100%
NRPN 3:1

OSC3 Level `OSC3`
The level of OSC3 sent to the Aux bus.
0 to 100%
NRPN 3:2

Audio input Level `AUDIO.IN`
The level of the audio input sent to the Aux bus.
0 to 100%
NRPN 3:3

Level `LEVEL`

The output level of the Aux bus.

0 to 100%

NRPN 3:4

Reverb Send `REVERB`

The level of the Aux bus sent to the reverb.

0 to 100%

NRPN 3:5

Pan `PAN`

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

100% Left to 100% Right

NRPN 3:6

Audio input Source `AIN.SRC`

An external source for the audio input.

NRPN 3:7 = [0-3]

- *White noise*

  A random audio source.

  NRPN = 0

- *Left channel*

  The left channel of the audio input.

  NRPN = 1

- *Right channel*

  The right channel of the audio input.

  NRPN = 2

— *Audio input (sum)*

Both channels of the audio input summed.

NRPN = 3

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.
NRPN 4:0

Send `SEND`
The level of input.
0 to 100%
NRPN 4:2

Dry/wet `DRY/WET`
The balance between the pre-effect (dry) and post-effect (wet) signals.
NRPN 4:3

Time `TIME`
The amount of reverberation time.
0 to 100%
NRPN 4:4

Diffusion `DIFF`
The density of the reverberation.
0 to 100%
NRPN 4:5

Damping `DAMPING`

The amount of damping.

0 to 100%

NRPN 4:6

Arpeggiator (ARP)

Turns polyphonic note input into arpeggios.

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

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

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

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

Mode `MODE`

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

NRPN 5:0 = [0-14]

— *Up*

From lowest to highest note.

NRPN = 0

— *Down*

From highest to lowest note.

NRPN = 1

— *Up/Down*

Performs the Up and Down patterns in sequence.

NRPN = 2

— *Down/Up*

Performs the Down and Up patterns in sequence.

NRPN = 3

— *Up and Down*

Performs the Up and Down patterns in sequence, repeating the lowest/highest notes.

NRPN = 4

— *Down and Up*

Performs the Down and Up patterns in sequence, repeating the lowest/highest notes.

NRPN = 5

— *Converge*

Starts from the extreme notes and alternates towards the middle.

NRPN = 6

— *Diverge*

Starts from the middle notes and alternates towards the extreme notes.

NRPN = 7

— *Converge/Diverge*

Performs the Converge and Diverge patterns in sequence.

NRPN = 8

— *Thumb Up*

Alternates the lowest note with the Up pattern.

NRPN = 9

— *Thumb Up/Down*

Alternates the lowest note with the Up/Down pattern.

NRPN = 10

— *Pinky Up*

Alternates the Up pattern with the highest note.

NRPN = 11

— *Pinky Up/Down*

Alternates the Up/Down pattern with the highest note.

NRPN = 12

— *Order*

Plays the notes in the order they have been entered.

NRPN = 13

— *Random*

Plays the notes in random order.

NRPN = 14

Rate `RATE`

The speed at which to arpeggiate.

NRPN 5:1

Gate `GATE`

The relative amount of time each arpeggiated note is heard.

NRPN 5:2

Swing `SWING`

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

NRPN 5:3

Repeats `REPEAT`

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

1 to 8

NRPN 5:4

Distance `DISTANCE`

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

-64 to +63 semitones

NRPN 5:5

Offset `OFFSET`

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

NRPN 5:6

Arpeggiator enabled `Enabled`

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

NRPN 5:7

Matrix (MTX)

Macros presented in the matrix.

Timbre `TIMBRE`
0 to 100%
NRPN 6:0

Color `COLOR`
0 to 100%
NRPN 6:1

Chaos `CHAOS`
0 to 100%
NRPN 6:2

Position `POS`
0 to 100%
NRPN 6:3

Geometry `GEOM`
0 to 100%
NRPN 6:4

Material `MATER`
0 to 100%
NRPN 6:5

Decay `DECAY`
0 to 100%
NRPN 6:6

Resonate RESO
0 to 100%
NRPN 6:8

Drive DRIVE
0 to 100%
NRPN 6:9

Body BODY
0 to 100%
NRPN 6:10

Space SPACE
0 to 100%
NRPN 6:11

Morph MORPH
A continuous interpolation between Matrix and Matrix Alt.
0 to 100%
NRPN 6:12

Source 2 SRC2
0 to 100%
NRPN 6:13

FX 1 FX1
0 to 100%
NRPN 6:14

FX 2 FX2
0 to 100%
NRPN 6:15

Matrix (alt.) (`MTXalt`)

Alternate version of the matrix (for morphing).

Timbre `TIMBRE`
0 to 100%
NRPN 7:0

Color `COLOR`
0 to 100%
NRPN 7:1

Chaos `CHAOS`
0 to 100%
NRPN 7:2

Position `POS`
0 to 100%
NRPN 7:3

Geometry `GEOM`
0 to 100%
NRPN 7:4

Material `MATER`
0 to 100%
NRPN 7:5

Decay `DECAY`
0 to 100%
NRPN 7:6

Resonate RES0
0 to 100%
NRPN 7:8

Drive DRIVE
0 to 100%
NRPN 7:9

Body BODY
0 to 100%
NRPN 7:10

Space SPACE
0 to 100%
NRPN 7:11

Source 2 SRC2
0 to 100%
NRPN 7:13

FX 1 FX1
0 to 100%
NRPN 7:14

FX 2 FX2
0 to 100%
NRPN 7:15

Oscillators

Module name	Label
Bow Bowing noise generator	`Bow`
Wind Wind noise generator	`Wind`
Strike Percussive noise generator	`Strike`
White noise A simple white noise generator	`Noise`
Artin NOIS Filtered noise	`NOIS`
Artin TWNQ Resonant noise	`TWNQ`
Artin CLKN Random sample generator	`CLKN`
Artin CLOU Granular cloud generator	`CLOU`
Artin PRTC Particle system simulator	`PRTC`
Artin QPSK Telecommunication data generator	`QPSK`
Artin TOY* Circuit-bent toy	`TOY\*`
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 IUSimple flute	`FLUT`
Drawbar organ Classic drawbar organ sound	`ORGAN`
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`

Module name	Label
Sawtooth wave Very rich tone with many harmonics	Saw
Virtual analog A virtual analog oscillator with smooth waveform transition	VA
Artin SUB Waveform with sub-oscillator	SUB
Artin FOLD Folded sine/triangle	FOLD
Artin SYNC Dual hard-synced waveforms	SYNC
Artin X3 Triple oscillator	X3
Artin SAWSWARM Seven detuned sawtooths	SAWSWARM
Artin BUZZ One to many sine waves	BUZZ
Artin VOSM Voice simulator	VOSM
Artin VOWL Early speech synthesizer	VOWL
Artin VFOF FoF vowel simulator	VFOF
Artin HARM Additive synthesizer	HARM
Artin WTBL Wavetable synthesizer	WTBL
Artin WMAP 2D wavetable synthesizer	WMAP
Artin WLIN Interpolated wavetable synthesizer	WLIN
Artin WTx4 Four-voice wavetable synthesizer	WTx4
Replicant A copy of the previous oscillator	Replica

Bow

NRPN [8-10]:0 = 1

Bowing noise generator.

Simulates the raw sound of a bow scratching a material.

Timbre `TIMBRE`

0 to 100%

NRPN [8-10]:1

Bow pressure `PRESSR`

0 to 100%

NRPN [8-10]:2

Wind

NRPN [8-10]:0 = 2

Wind noise generator.

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

Timbre `TIMBRE`
0 to 100%
NRPN [8-10]:1

Color `COLOR`
0 to 100%
NRPN [8-10]:2

Flow `FLOW`
0 to 100%
NRPN [8-10]:3

Strike

NRPN [8-10]:0 = 3

Percussive noise generator.

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

Timbre `TIMBRE`
0 to 100%
NRPN [8-10]:1

Mallet type `MALLET`
The kind of percussive noise generated (samples, impulses, bouncing particles).
0 to 100%
NRPN [8-10]:2

Trigger `TRIGGER`
A trigger used to produce the noise.
NRPN [8-10]:3

White noise (`Noise`)

NRPN [8-10]:0 = 4

A simple white noise generator.

Artin NOIS (`NOIS`)

NRPN [8-10]:0 = 5

Filtered noise.

White noise filtered through a resonant multi-mode filter.

Pitch source `PITCH`
The pitch source to follow.
NRPN [8-10]:1

Pitch (semitones) `SEMI`
The pitch correction on the oscillator.
-64 to +63 semitones
NRPN [8-10]:2

Pitch (fine) `FINE`
The pitch correction on the oscillator.
-100 to +100 cents
NRPN [8-10]:3

Filter resonance `RESO`
0 to 100%
NRPN [8-10]:4

Filter mode `FILTER`
Cross-fades between the low-pass and high-pass outputs of the filter.
0 to 100%
NRPN [8-10]:5

Artin TWNQ (TWNQ)

NRPN [8-10]:0 = 6

Resonant noise.

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

Pitch source `PITCH`
The pitch source to follow.
NRPN [8-10]:1

Pitch (semitones) `SEMI`
The pitch correction on the oscillator.
-64 to +63 semitones
NRPN [8-10]:2

Pitch (fine) `FINE`
The pitch correction on the oscillator.
-100 to +100 cents
NRPN [8-10]:3

Filter resonance `RESO`
0 to 100%
NRPN [8-10]:4

Peak distance `PEAK`
0 to 100%
NRPN [8-10]:5

Artin CLKN (CLKN)

NRPN [8-10]:0 = 7

Random sample generator.

Produces a looped random sample with variable bit depth.

Pitch source `PITCH`
The pitch source to follow.
NRPN [8-10]:1

Pitch (semitones) `SEMI`
The pitch correction on the oscillator.
-64 to +63 semitones
NRPN [8-10]:2

Pitch (fine) `FINE`
The pitch correction on the oscillator.
-100 to +100 cents
NRPN [8-10]:3

Cycle length `CYCLE`
The length of the generated sample.
0 to 100%
NRPN [8-10]:4

Quantization `BITS`
The quantization level.
0 to 100%
NRPN [8-10]:5

Trigger `TRIGGER`

A trigger used to select a new random sample.

NRPN [8-10]:7

Artin CLOU (CL0U)

NRPN [8-10]:0 = 8

Granular cloud generator.

Produces random noise grains at various temporal intervals.

Pitch source `PITCH`
The pitch source to follow.
NRPN [8-10]:1

Pitch (semitones) `SEMI`
The pitch correction on the oscillator.
-64 to +63 semitones
NRPN [8-10]:2

Pitch (fine) `FINE`
The pitch correction on the oscillator.
-100 to +100 cents
NRPN [8-10]:3

Grain density/overlap `DENSITY`
The density and overlap of the grains.
0 to 100%
NRPN [8-10]:4

Grain randomization `RANDOM`
The amount of randomization of the grains.
0 to 100%
NRPN [8-10]:5

Artin PRTC (PRTC)

NRPN [8-10]:0 = 9

Particle system simulator.

Produces random noise droplets at various temporal intervals.

Pitch source `PITCH`
The pitch source to follow.
NRPN [8-10]:1

Pitch (semitones) `SEMI`
The pitch correction on the oscillator.
-64 to +63 semitones
NRPN [8-10]:2

Pitch (fine) `FINE`
The pitch correction on the oscillator.
-100 to +100 cents
NRPN [8-10]:3

Particle density/overlap `DENSITY`
The density and overlap of the particles.
0 to 100%
NRPN [8-10]:4

Particle randomization `RANDOM`
The amount of randomization of the particles.
0 to 100%
NRPN [8-10]:5

Artin QPSK (QPSK)

NRPN [8-10]:0 = 10

Telecommunication data generator.

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

Pitch source `PITCH`
The pitch source to follow.
NRPN [8-10]:1

Pitch (semitones) `SEMI`
The pitch correction on the oscillator.
-64 to +63 semitones
NRPN [8-10]:2

Pitch (fine) `FINE`
The pitch correction on the oscillator.
-100 to +100 cents
NRPN [8-10]:3

Baud rate `BAUD`
0 to 100%
NRPN [8-10]:4

Data modulation `MOD`
0 to 100%
NRPN [8-10]:5

Trigger `TRIGGER`
A trigger used to start a new data frame.
NRPN [8-10]:7

Artin TOY* (TOY*)

NRPN [8-10]:0 = 11

Circuit-bent toy.

Pitch source `PITCH`
The pitch source to follow.
NRPN [8-10]:1

Pitch (semitones) `SEMI`
The pitch correction on the oscillator.
-64 to +63 semitones
NRPN [8-10]:2

Pitch (fine) `FINE`
The pitch correction on the oscillator.
-100 to +100 cents
NRPN [8-10]:3

Clock rate `CLK.RATE`
Lo-fi -> Hi-fi.
0 to 100%
NRPN [8-10]:4

Bit-crushing `BITCRUSH`
Simulates glitches and short-circuits.
0 to 100%
NRPN [8-10]:5

Modal resonator (`Modal`)

NRPN [8-10]:0 = 12

Vibrating structure simulator.

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

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

Pitch source `PITCH`
The pitch source to follow.
NRPN [8-10]:1

Pitch (semitones) `SEMI`
The pitch correction on the oscillator.
-64 to +63 semitones
NRPN [8-10]:2

Pitch (fine) `FINE`
The pitch correction on the oscillator.
-100 to +100 cents
NRPN [8-10]:3

Harmonic structure `STRUCT`
The nature of the structure (plate, bar, string…).
0 to 100%
NRPN [8-10]:4

Brightness `BRIGHT`

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

0 to 100%

NRPN [8-10]:5

Damping `DAMPING`

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

0 to 100%

NRPN [8-10]:6

Position `POSITION`

The position of the excitation point on the structure.

0 to 100%

NRPN [8-10]:7

Voice count `VOICES`

The number of voices.

1 to 3

NRPN [8-10]:8

Quality `QUALITY`

Changes sound quality.

NRPN [8-10]:9 = [0-4]

- *Lowest*

  NRPN = 0
- *Low*

  NRPN = 1
- *Medium*

  NRPN = 2

— *High*

NRPN = 3

— *Highest*

NRPN = 4

Trigger `TRIGGER`

A trigger used to excite the resonator.

NRPN [8-10]:11

String resonator (`String`)

NRPN [8-10]:0 = 13

Vibrating string simulator.

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

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

Pitch source `PITCH`
The pitch source to follow.
NRPN [8-10]:1

Pitch (semitones) `SEMI`
The pitch correction on the oscillator.
-64 to +63 semitones
NRPN [8-10]:2

Pitch (fine) `FINE`
The pitch correction on the oscillator.
-100 to +100 cents
NRPN [8-10]:3

Structure `STRUCT`
Accentuates the non-linear response of the string.
0 to 100%
NRPN [8-10]:4

Brightness `BRIGHT`
The type of material used to excite the string.
0 to 100%
NRPN [8-10]:5

Damping `DAMPING`

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

0 to 100%

NRPN [8-10]:6

Position `POSITION`

The position of the excitation point on the string.

0 to 100%

NRPN [8-10]:7

Voice count `VOICES`

The number of voices.

1 to 3

NRPN [8-10]:8

Trigger `TRIGGER`

A trigger used to excite the resonator.

NRPN [8-10]:11

Windsyo

NRPN [8-10]:0 = 14

Complex reed-based physical models.

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

Pitch source `PITCH`
The pitch source to follow.
NRPN [8-10]:1

Pitch (semitones) `SEMI`
The pitch correction on the oscillator.
-64 to +63 semitones
NRPN [8-10]:2

Pitch (fine) `FINE`
The pitch correction on the oscillator.
-100 to +100 cents
NRPN [8-10]:3

Loss filter `LSS.FLT`
-100 to 100%
NRPN [8-10]:4

Noise `NOISE`
0 to 100%
NRPN [8-10]:5

Growl `GROWL`

0 to 100%

NRPN [8-10]:6

AM depth `AM.DEPTH`

-100 to 100%

NRPN [8-10]:7

Preset `PRESET`

NRPN [8-10]:8 = [0-3]

- *Reed One*

  NRPN = 0
- *Flute*

  NRPN = 1
- *Duduk*

  NRPN = 2
- *Sylphinet*

  NRPN = 3

Artin PLUK (PLUK)

NRPN [8-10]:0 = 15

Simple plucked string.

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

Pitch source `PITCH`
The pitch source to follow.
NRPN [8-10]:1

Pitch (semitones) `SEMI`
The pitch correction on the oscillator.
-64 to +63 semitones
NRPN [8-10]:2

Pitch (fine) `FINE`
The pitch correction on the oscillator.
-100 to +100 cents
NRPN [8-10]:3

Damping `DAMPING`
The amount by which the vibration is damped.
0 to 100%
NRPN [8-10]:4

Pluck position `POSITION`
Low -> High.
0 to 100%
NRPN [8-10]:5

Trigger `TRIGGER`

A trigger used to pluck the string.

NRPN [8-10]:7

Artin BOWD (**BOWD**)

NRPN [8-10]:0 = 16

Simple bowed string.

Simulates a bowed string using a physical model.

Pitch source `PITCH`
The pitch source to follow.
NRPN [8-10]:1

Pitch (semitones) `SEMI`
The pitch correction on the oscillator.
-64 to +63 semitones
NRPN [8-10]:2

Pitch (fine) `FINE`
The pitch correction on the oscillator.
-100 to +100 cents
NRPN [8-10]:3

Friction `FRICTION`
0 to 100%
NRPN [8-10]:4

Bow position `POSITION`
Low -> High.
0 to 100%
NRPN [8-10]:5

Artin BLOW (BLOW)

NRPN [8-10]:0 = 17

Simple single-reed wind.

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

Pitch source `PITCH`
The pitch source to follow.
NRPN [8-10]:1

Pitch (semitones) `SEMI`
The pitch correction on the oscillator.
-64 to +63 semitones
NRPN [8-10]:2

Pitch (fine) `FINE`
The pitch correction on the oscillator.
-100 to +100 cents
NRPN [8-10]:3

Air pressure `PRESSR`
0 to 100%
NRPN [8-10]:4

Instrument geometry `GEOM`
Determines how the instrument's body acts on the resulting timbre.
0 to 100%
NRPN [8-10]:5

Artin FLUT (**FLUT**)

NRPN [8-10]:0 = 18

IUSimple flute.

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

Pitch source `PITCH`
The pitch source to follow.
NRPN [8-10]:1

Pitch (semitones) `SEMI`
The pitch correction on the oscillator.
-64 to +63 semitones
NRPN [8-10]:2

Pitch (fine) `FINE`
The pitch correction on the oscillator.
-100 to +100 cents
NRPN [8-10]:3

Air pressure `PRESSR`
0 to 100%
NRPN [8-10]:4

Instrument geometry `GEOM`
Determines how the instrument's body acts on the resulting timbre.
0 to 100%
NRPN [8-10]:5

Drawbar organ (ORGAN)

NRPN [8-10]:0 = 19

Classic drawbar organ sound.

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

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

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

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

Pitch source `PITCH`
The pitch source to follow.
NRPN [8-10]:1

Pitch (semitones) `SEMI`
The pitch correction on the oscillator.
-64 to +63 semitones
NRPN [8-10]:2

Pitch (fine) `FINE`
The pitch correction on the oscillator.
-100 to +100 cents
NRPN [8-10]:3

Gain `GAIN`

0 to 100%

NRPN [8-10]:4

Key click level `CLICK`

0 to 100%

NRPN [8-10]:5

Percussion level `PERC`

-100 to 100%

NRPN [8-10]:6

Preset `PRESET`

0 to 100%

NRPN [8-10]:7

Preset/custom mix `MIX`

0 to 100%

NRPN [8-10]:8

Sub-fundamental `16'`

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

NRPN [8-10]:9

Sub-third `5.1/3'`

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

NRPN [8-10]:10

Fundamental `8'`

The drawbar controlling the amplitude of the fundamental harmonic.

NRPN [8-10]:11

2nd harmonic `4'`
The drawbar controlling the amplitude of the second harmonic.
NRPN [8-10]:12

3rd harmonic `2.2/3'`
The drawbar controlling the amplitude of the third harmonic.
NRPN [8-10]:13

4th harmonic `2'`
The drawbar controlling the amplitude of the fourth harmonic.
NRPN [8-10]:14

5th harmonic `1.3/5'`
The drawbar controlling the amplitude of the fifth harmonic.
NRPN [8-10]:15

6th harmonic `1.1/3'`
The drawbar controlling the amplitude of the sixth harmonic.
NRPN [8-10]:16

8th harmonic `1'`
The drawbar controlling the amplitude of the eighth harmonic.
NRPN [8-10]:17

Artin BELL (BELL)

NRPN [8-10]:0 = 20

Additive bell sound synthesizer.

Simulates a bell using a bank of decaying sine waves.

Pitch source `PITCH`
The pitch source to follow.
NRPN [8-10]:1

Pitch (semitones) `SEMI`
The pitch correction on the oscillator.
-64 to +63 semitones
NRPN [8-10]:2

Pitch (fine) `FINE`
The pitch correction on the oscillator.
-100 to +100 cents
NRPN [8-10]:3

Damping `DAMPING`
0 to 100%
NRPN [8-10]:4

Inharmonicity `INHARM.`
0 to 100%
NRPN [8-10]:5

Trigger `TRIGGER`
A trigger used to ring the bell.
NRPN [8-10]:7

Artin DRUM (DRUM)

NRPN [8-10]:0 = 21

Additive metal drum synthesizer.

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

Pitch source `PITCH`
The pitch source to follow.
NRPN [8-10]:1

Pitch (semitones) `SEMI`
The pitch correction on the oscillator.
-64 to +63 semitones
NRPN [8-10]:2

Pitch (fine) `FINE`
The pitch correction on the oscillator.
-100 to +100 cents
NRPN [8-10]:3

Damping `DAMPING`
0 to 100%
NRPN [8-10]:4

Brightness `BRIGHT`
0 to 100%
NRPN [8-10]:5

Trigger `TRIGGER`
A trigger used to strike the drum.
NRPN [8-10]:7

Artin KICK (KICK)

NRPN [8-10]:0 = 22

808-style kick drum.

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

Pitch source `PITCH`
The pitch source to follow.
NRPN [8-10]:1

Pitch (semitones) `SEMI`
The pitch correction on the oscillator.
-64 to +63 semitones
NRPN [8-10]:2

Pitch (fine) `FINE`
The pitch correction on the oscillator.
-100 to +100 cents
NRPN [8-10]:3

Resonant decay `DECAY`
0 to 100%
NRPN [8-10]:4

Tone `TONE`
0 to 100%
NRPN [8-10]:5

Trigger `TRIGGER`
A trigger used to produce a kick sound.
NRPN [8-10]:7

Artin CYMB (CYMB)

NRPN [8-10]:0 = 23

808-style cymbal.

Produces the typical cymbal sounds found on analog drum machines.

Pitch source `PITCH`
The pitch source to follow.
NRPN [8-10]:1

Pitch (semitones) `SEMI`
The pitch correction on the oscillator.
-64 to +63 semitones
NRPN [8-10]:2

Pitch (fine) `FINE`
The pitch correction on the oscillator.
-100 to +100 cents
NRPN [8-10]:3

Filter `FILTER`
The cutoff of the filters applied to the signal.
0 to 100%
NRPN [8-10]:4

Tone `TONE`
0 to 100%
NRPN [8-10]:5

Artin SNAR (SNAR)

NRPN [8-10]:0 = 24

808-style snare drum.

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

Pitch source `PITCH`
The pitch source to follow.
NRPN [8-10]:1

Pitch (semitones) `SEMI`
The pitch correction on the oscillator.
-64 to +63 semitones
NRPN [8-10]:2

Pitch (fine) `FINE`
The pitch correction on the oscillator.
-100 to +100 cents
NRPN [8-10]:3

Tone `TONE`
0 to 100%
NRPN [8-10]:4

Noise mix `SNAPPY`
0 to 100%
NRPN [8-10]:5

Trigger `TRIGGER`
A trigger used to produce a snare sound.
NRPN [8-10]:7

Sine wave (`Sine`)

NRPN [8-10]:0 = 25

Pure tone without any harmonics.

Pitch source `PITCH`
The pitch source to follow.
NRPN [8-10]:1

Pitch (semitones) `SEMI`
The pitch correction on the oscillator.
-64 to +63 semitones
NRPN [8-10]:2

Pitch (fine) `FINE`
The pitch correction on the oscillator.
-100 to +100 cents
NRPN [8-10]:3

Triangle wave (`Tri`)

NRPN [8-10]:0 = 26

Soft tone with some odd harmonics.

Pitch source `PITCH`
The pitch source to follow.
NRPN [8-10]:1

Pitch (semitones) `SEMI`
The pitch correction on the oscillator.
-64 to +63 semitones
NRPN [8-10]:2

Pitch (fine) `FINE`
The pitch correction on the oscillator.
-100 to +100 cents
NRPN [8-10]:3

Square wave (`Square`)

NRPN [8-10]:0 = 27

Harsh, rich tone with many odd harmonics.

Pitch source `PITCH`
The pitch source to follow.
NRPN [8-10]:1

Pitch (semitones) `SEMI`
The pitch correction on the oscillator.
-64 to +63 semitones
NRPN [8-10]:2

Pitch (fine) `FINE`
The pitch correction on the oscillator.
-100 to +100 cents
NRPN [8-10]:3

Pulse width `PWM`
The proportion of time the square is “on” during its period.
0 to 100%
NRPN [8-10]:4

Sawtooth wave (`Saw`)

NRPN [8-10]:0 = 28

Very rich tone with many harmonics.

Pitch source `PITCH`
The pitch source to follow.
NRPN [8-10]:1

Pitch (semitones) `SEMI`
The pitch correction on the oscillator.
-64 to +63 semitones
NRPN [8-10]:2

Pitch (fine) `FINE`
The pitch correction on the oscillator.
-100 to +100 cents
NRPN [8-10]:3

Virtual analog (VA)

NRPN [8-10]:0 = 29

A virtual analog oscillator with smooth waveform transition.

Pitch source `PITCH`
The pitch source to follow.
NRPN [8-10]:1

Pitch (semitones) `SEMI`
The pitch correction on the oscillator.
-64 to +63 semitones
NRPN [8-10]:2

Pitch (fine) `FINE`
The pitch correction on the oscillator.
-100 to +100 cents
NRPN [8-10]:3

Waveform `WAVE`
The waveform to use (sine, triangle, sawtooth, square, pulse).
NRPN [8-10]:4

Artin SUB (SUB)

NRPN [8-10]:0 = 30

Waveform with sub-oscillator.

Pitch source `PITCH`
The pitch source to follow.
NRPN [8-10]:1

Pitch (semitones) `SEMI`
The pitch correction on the oscillator.
-64 to +63 semitones
NRPN [8-10]:2

Pitch (fine) `FINE`
The pitch correction on the oscillator.
-100 to +100 cents
NRPN [8-10]:3

Pulse width `PWM`
The proportion of time the square is “on” during its period.
0 to 100%
NRPN [8-10]:4

Sub-oscillator `SUB`
-2 octaves -> No sub -> -1 octave.
0 to 100%
NRPN [8-10]:5

Waveform `WAVE`
NRPN [8-10]:6 = [0-1]

— *Square*

NRPN = 0

— *Sawtooth*

NRPN = 1

Artin FOLD (FOLD)

NRPN [8-10]:0 = 31

Folded sine/triangle.

Pitch source `PITCH`
The pitch source to follow.
NRPN [8-10]:1

Pitch (semitones) `SEMI`
The pitch correction on the oscillator.
-64 to +63 semitones
NRPN [8-10]:2

Pitch (fine) `FINE`
The pitch correction on the oscillator.
-100 to +100 cents
NRPN [8-10]:3

Folding strength `FOLD`
Amplifies more so that the waveform folds back onto itself.
0 to 100%
NRPN [8-10]:4

Waveform `WAVEFORM`
Sine -> Triangle.
0 to 100%
NRPN [8-10]:5

Artin SYNC (SYNC)

NRPN [8-10]:0 = 32

Dual hard-synced waveforms.

Pitch source `PITCH`
The pitch source to follow.
NRPN [8-10]:1

Pitch (semitones) `SEMI`
The pitch correction on the oscillator.
-64 to +63 semitones
NRPN [8-10]:2

Pitch (fine) `FINE`
The pitch correction on the oscillator.
-100 to +100 cents
NRPN [8-10]:3

Sync frequency `SYNC`
The frequency interval between master and slave waveform.
0 to 100%
NRPN [8-10]:4

Balance `BALANCE`
Master -> Slave.
0 to 100%
NRPN [8-10]:5

Waveform `WAVEFORM`
NRPN [8-10]:6 = [0-1]

— *Square*

NRPN = 0

— *Sawtooth*

NRPN = 1

Pulse width PWM

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

0 to 100%

NRPN [8-10]:7

Artin X3 (X3)

NRPN [8-10]:0 = 33

Triple oscillator.

Pitch source `PITCH`
The pitch source to follow.
NRPN [8-10]:1

Pitch (semitones) `SEMI`
The pitch correction on the oscillator.
-64 to +63 semitones
NRPN [8-10]:2

Pitch (fine) `FINE`
The pitch correction on the oscillator.
-100 to +100 cents
NRPN [8-10]:3

Pitch 2 `PITCH2`
The relative pitch of the second oscillator.
-64 to +63 semitones
NRPN [8-10]:4

Pitch 3 `PITCH3`
The relative pitch of the third oscillator.
-64 to +63 semitones
NRPN [8-10]:5

Waveform `WAVEFORM`
NRPN [8-10]:6 = [0-4]

— *Square*

NRPN = 0

— *Sawtooth*

NRPN = 1

— *Triangle*

NRPN = 2

— *Sine*

NRPN = 3

— *Ring-modulated sine*

NRPN = 4

Artin SAWSWARM (SAWSWARM)

NRPN [8-10]:0 = 34

Seven detuned sawtooths.

Pitch source `PITCH`
The pitch source to follow.
NRPN [8-10]:1

Pitch (semitones) `SEMI`
The pitch correction on the oscillator.
-64 to +63 semitones
NRPN [8-10]:2

Pitch (fine) `FINE`
The pitch correction on the oscillator.
-100 to +100 cents
NRPN [8-10]:3

Detune amount `DETUNE`
The amount of detuning between each sawtooth.
0 to 100%
NRPN [8-10]:4

High-pass filter `FILTER`
The amount of high-pass filtering applied on the resulting sound.
0 to 100%
NRPN [8-10]:5

Artin BUZZ (BUZZ)

NRPN [8-10]:0 = 35

One to many sine waves.

Pitch source `PITCH`
The pitch source to follow.
NRPN [8-10]:1

Pitch (semitones) `SEMI`
The pitch correction on the oscillator.
-64 to +63 semitones
NRPN [8-10]:2

Pitch (fine) `FINE`
The pitch correction on the oscillator.
-100 to +100 cents
NRPN [8-10]:3

Waveform `WAVE`
Single sine -> Many sines.
0 to 100%
NRPN [8-10]:4

Detune `DETUNE`
The amount of space between each sine wave.
-100 to +100 cents
NRPN [8-10]:5

Artin VOSM (VOSM)

NRPN [8-10]:0 = 36

Voice simulator.

Produces sounds inspired by early voice simulators.

Pitch source `PITCH`
The pitch source to follow.
NRPN [8-10]:1

Pitch (semitones) `SEMI`
The pitch correction on the oscillator.
-64 to +63 semitones
NRPN [8-10]:2

Pitch (fine) `FINE`
The pitch correction on the oscillator.
-100 to +100 cents
NRPN [8-10]:3

Second formant `FMT2`
The frequency of the second formant.
-100 to 100%
NRPN [8-10]:4

Third formant `FMT3`
The frequency of the third formant.
-100 to 100%
NRPN [8-10]:5

Carrier shape `CARRIER`

The shape of the carrier.

0 to 100%

NRPN [8-10]:6

Artin VOWL (VOWL)

NRPN [8-10]:0 = 37

Early speech synthesizer.

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

Pitch source `PITCH`
The pitch source to follow.
NRPN [8-10]:1

Pitch (semitones) `SEMI`
The pitch correction on the oscillator.
-64 to +63 semitones
NRPN [8-10]:2

Pitch (fine) `FINE`
The pitch correction on the oscillator.
-100 to +100 cents
NRPN [8-10]:3

Vowel `VOWEL`
A -> E -> I -> O -> U.
0 to 100%
NRPN [8-10]:4

Age `AGE`
Old -> Young.
0 to 100%
NRPN [8-10]:5

Consonant `CONSON.`

The shape of the consonant to be produced when triggered.

0 to 100%

NRPN [8-10]:6

Consonant length `CONS.LEN`

The duration of a consonant when triggered.

0 to 100%

NRPN [8-10]:7

Consonant trigger `CNS.TRIG`

A trigger used to produce a consonant.

NRPN [8-10]:11

Artin VFOF (VF0F)

NRPN [8-10]:0 = 38

FoF vowel simulator.

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

Pitch source `PITCH`
The pitch source to follow.
NRPN [8-10]:1

Pitch (semitones) `SEMI`
The pitch correction on the oscillator.
-64 to +63 semitones
NRPN [8-10]:2

Pitch (fine) `FINE`
The pitch correction on the oscillator.
-100 to +100 cents
NRPN [8-10]:3

Vowel `VOWEL`
A -> E -> I -> O -> U.
0 to 100%
NRPN [8-10]:4

Age `AGE`
Old -> Young.
0 to 100%
NRPN [8-10]:5

Artin HARM (HARM)

NRPN [8-10]:0 = 39

Additive synthesizer.

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

Pitch source `PITCH`
The pitch source to follow.
NRPN [8-10]:1

Pitch (semitones) `SEMI`
The pitch correction on the oscillator.
-64 to +63 semitones
NRPN [8-10]:2

Pitch (fine) `FINE`
The pitch correction on the oscillator.
-100 to +100 cents
NRPN [8-10]:3

Central frequency `FREQ`
The frequency of the central harmonic.
0 to 100%
NRPN [8-10]:4

Bandwidth `BW`
The distribution of harmonics and presence of the fundamental.
-100 to 100%
NRPN [8-10]:5

Artin WTBL (`WTBL`)

NRPN [8-10]:0 = 40

Wavetable synthesizer.

Pitch source `PITCH`
The pitch source to follow.
NRPN [8-10]:1

Pitch (semitones) `SEMI`
The pitch correction on the oscillator.
-64 to +63 semitones
NRPN [8-10]:2

Pitch (fine) `FINE`
The pitch correction on the oscillator.
-100 to +100 cents
NRPN [8-10]:3

Sweep `SWEEP`
Travels through the selected wavetable.
0 to 100%
NRPN [8-10]:4

Wavetable `WAVETBL`
Selects the current wavetable.
NRPN [8-10]:5

Artin WMAP (WMAP)

NRPN [8-10]:0 = 41

2D wavetable synthesizer.

Pitch source `PITCH`
The pitch source to follow.
NRPN [8-10]:1

Pitch (semitones) `SEMI`
The pitch correction on the oscillator.
-64 to +63 semitones
NRPN [8-10]:2

Pitch (fine) `FINE`
The pitch correction on the oscillator.
-100 to +100 cents
NRPN [8-10]:3

X scanning `X`
The horizontal position in the grid.
0 to 100%
NRPN [8-10]:4

Y scanning `Y`
The vertical position in the grid.
0 to 100%
NRPN [8-10]:5

Artin WLIN (`WLIN`)

NRPN [8-10]:0 = 42

Interpolated wavetable synthesizer.

Pitch source `PITCH`
The pitch source to follow.
NRPN [8-10]:1

Pitch (semitones) `SEMI`
The pitch correction on the oscillator.
-64 to +63 semitones
NRPN [8-10]:2

Pitch (fine) `FINE`
The pitch correction on the oscillator.
-100 to +100 cents
NRPN [8-10]:3

Wavetable `WAVETBL`
Selects the current wavetable.
0 to 100%
NRPN [8-10]:4

Interpolation `LERP`
The interpolation method used for playback.
0 to 100%
NRPN [8-10]:5

Artin WTx4 (`WTx4`)

NRPN [8-10]:0 = 43

Four-voice wavetable synthesizer.

Pitch source `PITCH`
The pitch source to follow.
NRPN [8-10]:1

Pitch (semitones) `SEMI`
The pitch correction on the oscillator.
-64 to +63 semitones
NRPN [8-10]:2

Pitch (fine) `FINE`
The pitch correction on the oscillator.
-100 to +100 cents
NRPN [8-10]:3

Morph `MORPH`
Morphs through wavetables.
0 to 100%
NRPN [8-10]:4

Detune/chord `CHORD`
Detunes the oscillators or selects a fixed chord.
0 to 100%
NRPN [8-10]:5

Replicant (`Replica`)

NRPN [8-10]:0 = 44

A copy of the previous oscillator.

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

Pitch source `PITCH`
The pitch source to follow.
NRPN [8-10]:1

Pitch (semitones) `SEMI`
The pitch correction on the oscillator.
-64 to +63 semitones
NRPN [8-10]:2

Pitch (fine) `FINE`
The pitch correction on the oscillator.
-100 to +100 cents
NRPN [8-10]:3

Effects

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

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

Modal resonator (`Modal`)

NRPN [11-15]:0 = 1

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.
NRPN [11-15]:1

Send `SEND`
The level of input sent into the effect.
0 to 100%
NRPN [11-15]:2

Dry/wet `DRY/WET`
The balance between the pre-effect (dry) and post-effect (wet) signals.
NRPN [11-15]:3

Harmonic structure `STRUCT`
The nature of the structure (plate, bar, string).
0 to 100%
NRPN [11-15]:4

Brightness `BRIGHT`
The type of material (nylon, wood, steel, glass).
0 to 100%
NRPN [11-15]:5

Damping `DAMPING`

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

0 to 100%

NRPN [11-15]:6

Position `POSITION`

The position of the excitation point on the structure.

0 to 100%

NRPN [11-15]:7

Voice count `VOICES`

The number of voices.

1 to 3

NRPN [11-15]:8

Pitch source `PITCH`

The pitch source to follow.

NRPN [11-15]:9

Pitch (semitones) `SEMI`

The pitch correction on the effect.

-64 to +63 semitones

NRPN [11-15]:10

Pitch (fine) `FINE`

The pitch correction on the effect.

-100 to +100 cents

NRPN [11-15]:11

Quality `QUALITY`

Changes sound quality.

NRPN [11-15]:12 = [0-4]

- *Lowest*

  NRPN = 0

- *Low*

  NRPN = 1

- *Medium*

  NRPN = 2

- *High*

  NRPN = 3

- *Highest*

  NRPN = 4

String resonator (`String`)

NRPN [11-15]:0 = 2

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.
NRPN [11-15]:1

Send `SEND`
The level of input sent into the effect.
0 to 100%
NRPN [11-15]:2

Dry/wet `DRY/WET`
The balance between the pre-effect (dry) and post-effect (wet) signals.
NRPN [11-15]:3

Structure `STRUCT`
Accentuates the non-linear response of the string.
0 to 100%
NRPN [11-15]:4

Brightness `BRIGHT`
The brightness of the input.
0 to 100%
NRPN [11-15]:5

Damping `DAMPING`

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

0 to 100%

NRPN [11-15]:6

Position `POSITION`

The position of the excitation point on the strings.

0 to 100%

NRPN [11-15]:7

Voice count `VOICES`

The number of voices.

1 to 3

NRPN [11-15]:8

Pitch source `PITCH`

The pitch source to follow.

NRPN [11-15]:9

Pitch (semitones) `SEMI`

The pitch correction on the effect.

-64 to +63 semitones

NRPN [11-15]:10

Pitch (fine) `FINE`

The pitch correction on the effect.

-100 to +100 cents

NRPN [11-15]:11

Snare drum (`Snare`)

NRPN [11-15]:0 = 3

A physical model of a snare drum.

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

The physical model is derived from a paper by Philipp Schmalfuß et al. (2020)[1] from the *Center for Haptic Audio Interaction Research (CHAIR)*.

Bus `BUS`
Whether to apply the effect on the Main or Aux bus.
NRPN [11-15]:1

Send `SEND`
The level of input sent into the effect.
0 to 100%
NRPN [11-15]:2

Dry/wet `DRY/WET`
The balance between the pre-effect (dry) and post-effect (wet) signals.
NRPN [11-15]:3

Head tuning `TUNING`
The amount of tension on the drum head.
0 to 100%
NRPN [11-15]:4

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

Buzz BUZZ

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

0 to 100%

NRPN [11-15]:5

Damping DAMPING

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

-100 to 100%

NRPN [11-15]:6

Position POS

The position of the hit on the drum head.

0 to 100%

NRPN [11-15]:7

State-variable filter (SVF)

NRPN [11-15]:0 = 4

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.
NRPN [11-15]:1

Dry/wet `DRY/WET`
The balance between the pre-effect (dry) and post-effect (wet) signals.
NRPN [11-15]:3

Filter mode `MODE`
The type of filter reponse.
NRPN [11-15]:4 = [0-3]

- *Low-pass*
  A low-pass filter (-12 dB/oct).
  NRPN = 0
- *High-pass*
  A high-pass filter (-12 dB/oct).
  NRPN = 1
- *Band-pass*
  A band-pass filter (-12 dB/oct).
  NRPN = 2

— *Notch*

A notch filter.

NRPN = 3

Cutoff frequency `CUTOFF`

The cutoff frequency of the filter.

12.26 Hz to 18.82 kHz

NRPN [11-15]:5

Resonance `RESO`

The resonance of the filter.

0 to 100%

NRPN [11-15]:6

Keyfollow `K.FOLLOW`

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

-100 to 100%

NRPN [11-15]:7

Ladder filter (`Ladder`)

NRPN [11-15]:0 = 5

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.
NRPN [11-15]:1

Dry/wet `DRY/WET`
The balance between the pre-effect (dry) and post-effect (wet) signals.
NRPN [11-15]:3

Filter mode `MODE`
The type of filter reponse.
NRPN [11-15]:4 = [0-10]

- *Low-pass LP1*

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

  NRPN = 0
- *Low-pass LP2*

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

  NRPN = 1
- *Low-pass LP3*

  A low-pass filter (-18 dB/oct).

  NRPN = 2
- *Low-pass LP4*

  A low-pass filter (-24 dB/oct).

  NRPN = 3

— *High-pass HP1*

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

NRPN = 4

— *High-pass HP2*

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

NRPN = 5

— *High-pass HP3*

A high-pass filter (-18 dB/oct).

NRPN = 6

— *High-pass HP4*

A high-pass filter (-24 dB/oct).

NRPN = 7

— *Band-pass BP1*

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

NRPN = 8

— *Band-pass BP2*

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

NRPN = 9

— *Notch*

A notch filter.

NRPN = 10

Cutoff frequency CUTOFF

The cutoff frequency of the filter.

12.26 Hz to 18.82 kHz

NRPN [11-15]:5

Resonance RESO

The resonance of the filter.

0 to 100%
NRPN [11-15]:6

Keyfollow `K.FOLLOW`
The amount by which the filter cutoff follows the controller's pitch.
-100 to 100%
NRPN [11-15]:7

Simple EQ (**EQ**)

NRPN [11-15]:0 = 6

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.
NRPN [11-15]:1

Send `SEND`
The level of input sent into the effect.
0 to 100%
NRPN [11-15]:2

Dry/wet `DRY/WET`
The balance between the pre-effect (dry) and post-effect (wet) signals.
NRPN [11-15]:3

Low-pass `LOWPASS`
The low-pass filter frequency.
0 to 100%
NRPN [11-15]:4

High-pass `HIPASS`
The high-pass filter frequency.
0 to 100%
NRPN [11-15]:5

Peak frequency `PEAKFRQ`

The peaking filter frequency.

0 to 100%

NRPN [11-15]:6

Peak Q `PEAK.Q`

The peaking filter Q factor.

0 to 100%

NRPN [11-15]:10

Peak gain `PEAKGAIN`

The peaking filter gain.

0 to 100%

NRPN [11-15]:11

Dirty formant filter (`FOFlt`)

NRPN [11-15]:0 = 7

Old-school FoF-based formant filter.

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

Bus `BUS`
Whether to apply the effect on the Main or Aux bus.
NRPN [11-15]:1

Send `SEND`
The level of input sent into the effect.
0 to 100%
NRPN [11-15]:2

Dry/wet `DRY/WET`
The balance between the pre-effect (dry) and post-effect (wet) signals.
NRPN [11-15]:3

Vowel `VOWEL`
A -> E -> I -> O -> U.
0 to 100%
NRPN [11-15]:4

Age `AGE`
Old -> Young.
0 to 100%
NRPN [11-15]:5

VCA

NRPN [11-15]:0 = 8

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

Bus BUS
Whether to apply the effect on the Main or Aux bus.
NRPN [11-15]:1

Level LEVEL
The level of the input signal.
0 to 100%
NRPN [11-15]:2

Tremolo (`Treml`)

NRPN [11-15]:0 = 9

Change the amplitude of the input in a periodic way.

Bus BUS

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

NRPN [11-15]:1

Rotation speed SPEED

The speed at which the amplitude changes.

0 to 100%

NRPN [11-15]:2

Amount AMOUNT

The amount of amplitude modulation.

0 to 100%

NRPN [11-15]:3

Noise gate (`Gate`)

NRPN [11-15]:0 = 10

Attenuates the input when the signal is below a threshold.

Bus `BUS`
Whether to apply the effect on the Main or Aux bus.
NRPN [11-15]:1

Dry/wet `DRY/WET`
The balance between the pre-effect (dry) and post-effect (wet) signals.
NRPN [11-15]:3

Sidechain `SIDECH`
Whether to use the other bus as a sidechain input.
NRPN [11-15]:2

Threshold level `THRESH`
The level over which the gate opens.
NRPN [11-15]:4

Attack time `ATTACK`
The time to reach the input level when the gate opens.
NRPN [11-15]:5

Hold time `HOLD`
The minimum time during which the gate stays open.
NRPN [11-15]:6

Release time `RELEASE`
The time to reach the attenuation level when the gate closes.
NRPN [11-15]:7

Hysteresis amount HYST

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

NRPN [11-15]:8

Attenuation level ATTN

The level of attenuation applied when the gate is closed.

NRPN [11-15]:9

Invert INVERT

Whether to invert the gate.

NRPN [11-15]:11

Lookahead time LOOKAHD

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

NRPN [11-15]:10 = [0-2]

- *None*

  NRPN = 0
- *1.5ms*

  NRPN = 1
- *10ms*

  NRPN = 2

Compressor (Comp)

NRPN [11-15]:0 = 11

Compress the input signal.

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

Bus `BUS`
Whether to apply the effect on the Main or Aux bus.
NRPN [11-15]:1

Dry/wet `DRY/WET`
The balance between the pre-effect (dry) and post-effect (wet) signals.
NRPN [11-15]:3

Sidechain `SIDECH`
Whether to use the other bus as a sidechain input.
NRPN [11-15]:2

Threshold level `THRESH`
The level above which compression will occur.
NRPN [11-15]:4

Compression ratio `RATIO`
The amount of compression.
NRPN [11-15]:5

Attack time `ATTACK`
The time for compression to start reacting when above the threshold.
NRPN [11-15]:6

Release time RELEASE

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

NRPN [11-15]:7

Knee width KNEE

Makes compression happen more gradually around the threshold point.

NRPN [11-15]:8

Makeup gain MAKEUP

Recovers the amount of level lost by compression.

NRPN [11-15]:9

Lookahead time LOOKAHD

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

NRPN [11-15]:10 = [0-2]

- *None*

  NRPN = 0
- *1ms*

  NRPN = 1
- *10ms*

  NRPN = 2

Dynamics booster (`Boost`)

NRPN [11-15]:0 = 12

Tame or boost the signal by compressing it.

Changes the dynamics of the input.

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

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

Bus `BUS`
Whether to apply the effect on the Main or Aux bus.
NRPN [11-15]:1

Sensitivity `SENS`
Whether the booster reacts to soft signals.
0 to 100%
NRPN [11-15]:2

Boost amount `AMOUNT`
Tames the signal to the left, boosts it to the right.
NRPN [11-15]:3

Cross-fader (`XFader`)

NRPN [11-15]:0 = 13

Balances between two inputs.

Bus `BUS`
Whether to apply the effect on the Main or Aux bus.
NRPN [11-15]:1

Mix `MIX`
The cross-fading amount (this bus on the left, the other bus on the right).
0 to 100%
NRPN [11-15]:3

Cross-fader with drive (XFadeD)

NRPN [11-15]:0 = 14

Cross-fader with drive controls.

Bus `BUS`
Whether to apply the effect on the Main or Aux bus.
NRPN [11-15]:1

Gain `GAIN`
The gain applied to the input signal on this bus.
0 to 400%
NRPN [11-15]:2

Gain 2 `GAIN2`
The gain applied to the input signal on the other bus.
0 to 400%
NRPN [11-15]:6

Dry/wet `DRY/WET`
The balance between the pre-effect (dry) and post-effect (wet) signals.
NRPN [11-15]:7

Mix `MIX`
The amount of cross-fading after applying drive (this bus on the left, the other bus on the right).
0 to 100%
NRPN [11-15]:3

Oversampling `OVERSPL`
The oversampling factor.
NRPN [11-15]:5 = [0-2]

- *1x*

  NRPN = 0
- *2x*

  NRPN = 1
- *4x*

  NRPN = 2

Rotary speaker (stereo) (`RotryS`)

NRPN [11-15]:0 = 15

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

Bus for input `BUS`

NRPN [11-15]:1

Rotation speed `SPEED`

The speed of the rotating speakers.

0 to 100%

NRPN [11-15]:2

Amount `AMOUNT`

The intensity of the effect.

0 to 100%

NRPN [11-15]:3

Amplifier (AMP)

NRPN [11-15]:0 = 16

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.
NRPN [11-15]:1

Dry/wet `DRY/WET`
The balance between the pre-effect (dry) and post-effect (wet) signals.
NRPN [11-15]:3

Gain `GAIN`
The gain applied to the input signal.
0 to 400%
NRPN [11-15]:2

Oversampling `OVERSPL`
The oversampling factor.
NRPN [11-15]:4 = [0-2]

- *1x*
  NRPN = 0
- *2x*
  NRPN = 1
- *4x*
  NRPN = 2

Overdrive (`Drive`)

NRPN [11-15]:0 = 17

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.
NRPN [11-15]:1

Dry/wet `DRY/WET`
The balance between the pre-effect (dry) and post-effect (wet) signals.
NRPN [11-15]:3

Drive `DRIVE`
The level of overdrive applied to the input signal.
0 to 100%
NRPN [11-15]:2

Oversampling `OVERSPL`
The oversampling factor.
NRPN [11-15]:4 = [0-2]

- *1x*
  NRPN = 0
- *2x*
  NRPN = 1
- *4x*
  NRPN = 2

Bitcrusher (`Crush`)

NRPN [11-15]:0 = 18

Reduces the resolution of the audio signal.

Bus `BUS`
Whether to apply the effect on the Main or Aux bus.
NRPN [11-15]:1

Dry/wet `DRY/WET`
The balance between the pre-effect (dry) and post-effect (wet) signals.
NRPN [11-15]:3

Depth `DEPTH`
The resolution of the output signal.
NRPN [11-15]:2

Sample rate reduction `REDUX`
The downsampling factor of the output signal.
NRPN [11-15]:4

Cross-folder (`XFold`)

NRPN [11-15]:0 = 19

Wavefolds two inputs together.

Bus `BUS`
Whether to apply the effect on the Main or Aux bus.
NRPN [11-15]:1

Dry/wet `DRY/WET`
The balance between the pre-effect (dry) and post-effect (wet) signals.
NRPN [11-15]:3

Gain `GAIN`
The gain applied to the input signal on this bus.
0 to 400%
NRPN [11-15]:2

Gain 2 `GAIN2`
The gain applied to the input signal on the other bus.
0 to 400%
NRPN [11-15]:6

Wavefold amount `FOLD`
The amount of wavefolding of both signals.
0 to 100%
NRPN [11-15]:4

Oversampling `OVERSPL`
The oversampling factor.
NRPN [11-15]:5 = [0-2]

- *1x*

  NRPN = 0
- *2x*

  NRPN = 1
- *4x*

  NRPN = 2

Ring modulator (`RingMod`)

NRPN [11-15]:0 = 20

Ring-modulates two inputs together.

Bus `BUS`
Whether to apply the effect on the Main or Aux bus.
NRPN [11-15]:1

Dry/wet `DRY/WET`
The balance between the pre-effect (dry) and post-effect (wet) signals.
NRPN [11-15]:3

Gain `GAIN`
The gain applied to the input signal on this bus.
0 to 400%
NRPN [11-15]:2

Gain 2 `GAIN2`
The gain applied to the input signal on the other bus.
0 to 400%
NRPN [11-15]:6

Gain boost `BOOST`
The amount of gain applied to the resulting signal.
0 to 100%
NRPN [11-15]:4

Analog `ANALOG`
Whether ring-modulation is analog or digital.
NRPN [11-15]:7

Oversampling OVERSPL

The oversampling factor.

NRPN [11-15]:5 = [0-2]

- *1x*

  NRPN = 0

- *2x*

  NRPN = 1

- *4x*

  NRPN = 2

XOR modulator (XOR)

NRPN [11-15]:0 = 21

XORs two inputs together bit by bit.

Bus `BUS`
Whether to apply the effect on the Main or Aux bus.
NRPN [11-15]:1

Dry/wet `DRY/WET`
The balance between the pre-effect (dry) and post-effect (wet) signals.
NRPN [11-15]:3

Gain `GAIN`
The gain applied to the input signal on this bus.
0 to 400%
NRPN [11-15]:2

Gain 2 `GAIN2`
The gain applied to the input signal on the other bus.
0 to 400%
NRPN [11-15]:6

Bits to XOR `BITS`
Selects which bits are XOR'ed together.
0 to 100%
NRPN [11-15]:4

Oversampling `OVERSPL`
The oversampling factor.
NRPN [11-15]:5 = [0-2]

- *1x*

  NRPN = 0
- *2x*

  NRPN = 1
- *4x*

  NRPN = 2

CMP modulator (CMP)

NRPN [11-15]:0 = 22

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.
NRPN [11-15]:1

Dry/wet `DRY/WET`
The balance between the pre-effect (dry) and post-effect (wet) signals.
NRPN [11-15]:3

Gain `GAIN`
The gain applied to the input signal on this bus.
0 to 400%
NRPN [11-15]:2

Gain 2 `GAIN2`
The gain applied to the input signal on the other bus.
0 to 400%
NRPN [11-15]:6

Operator 0P

Selects the comparison operator to use.

0 to 100%

NRPN [11-15]:4

Oversampling 0VERSPL

The oversampling factor.

NRPN [11-15]:5 = [0-2]

- *1x*

  NRPN = 0

- *2x*

  NRPN = 1

- *4x*

  NRPN = 2

Chorus

NRPN [11-15]:0 = 23

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.
NRPN [11-15]:1

Send `SEND`
The level of input sent into the effect.
0 to 100%
NRPN [11-15]:2

Dry/wet `DRY/WET`
The balance between the pre-effect (dry) and post-effect (wet) signals.
NRPN [11-15]:3

Delay time `TIME`
The time between the original and the delayed signal.
0 to 26.5 ms
NRPN [11-15]:4

Feedback `FEEDBK`
The amount of feedback from the delayed signal.
0 to 100%
NRPN [11-15]:5

Modulation rate `MOD.RATE`

The speed of the oscillator modulating delay time.

0.07 to 14.80 Hz

NRPN [11-15]:6

Modulation amount `MOD.AMT`

The intensity of the modulation of the delay time.

0 to 100%

NRPN [11-15]:7

Phaser

NRPN [11-15]:0 = 24

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.
NRPN [11-15]:1

Send SEND
The level of input sent into the effect.
0 to 100%
NRPN [11-15]:2

Dry/wet DRY/WET
The balance between the pre-effect (dry) and post-effect (wet) signals.
NRPN [11-15]:3

Rate RATE
The speed of phase shifting.
0.07 to 14.80 Hz
NRPN [11-15]:4

Feedback FEEDBACK
The amount of feedback from the phase-shifted signals.
0 to 100%
NRPN [11-15]:5

Pitch-shifter (`PShiftr`)

NRPN [11-15]:0 = 25

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.
NRPN [11-15]:1

Send `SEND`
The level of input sent into the effect.
0 to 100%
NRPN [11-15]:2

Dry/wet `DRY/WET`
The balance between the pre-effect (dry) and post-effect (wet) signals.
NRPN [11-15]:3

Pitch `PITCH`
The relative pitch for the shifted signal.
NRPN [11-15]:4

Window time `TIME`
The time of the window over which to shift the signal.
20 to 1000 ms
NRPN [11-15]:5

Rotary speaker (`Rotary`)

NRPN [11-15]:0 = 26

Simulation of a rotary speaker.

Bus `BUS`
Whether to apply the effect on the Main or Aux bus.
NRPN [11-15]:1

Rotation speed `SPEED`
The speed of the rotating speakers.
0 to 100%
NRPN [11-15]:2

Amount `AMOUNT`
The intensity of the effect.
NRPN [11-15]:3

FM Operator (`FM Op`)

NRPN [11-15]:0 = 27

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

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

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

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

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

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

[2] Any oscillator can be used, but the *Virtual analog* oscillator could be the best starting point, as it provides the classic waveforms used in FM synthesis.

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

Bus `BUS`
Whether to apply the effect on the Main or Aux bus.
NRPN [11-15]:1

Modulation index `INDEX`
0 to 100%
NRPN [11-15]:2

Modulation scale `SCALE`
The scale factor applied to the modulation index.
1 to 100
NRPN [11-15]:3

Waveform `WAVE`
The waveform to use (sine, triangle, sawtooth, square, pulse).
NRPN [11-15]:4

Pitch source `PITCH`
The pitch source to follow.
NRPN [11-15]:5

Pitch (semitones) `SEMI`
The pitch correction on the effect.

-64 to +63 semitones
NRPN [11-15]:6

Pitch (fine) `FINE`
The pitch correction on the effect.
-100 to +100 cents
NRPN [11-15]:7

Self-modulation `SELFMOD`
The amount to which the oscillator modulates itself.
0 to 100%
NRPN [11-15]:8

Damping `DAMP`
Limits the modulation index of higher notes.
0 to 100%
NRPN [11-15]:9

Through-zero `TZ`
Whether the operator can support negative frequencies.
NRPN [11-15]:10

Oversampling `OVERSPL`
The oversampling factor.
NRPN [11-15]:11 = [0-2]

- *1x*
  NRPN = 0
- *2x*
  NRPN = 1

— *4x*

NRPN = 2

Delay

NRPN [11-15]:0 = 28

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.
NRPN [11-15]:1

Send `SEND`
The level of input sent into the effect.
0 to 100%
NRPN [11-15]:2

Dry/wet `DRY/WET`
The balance between the pre-effect (dry) and post-effect (wet) signals.
NRPN [11-15]:3

Delay time `TIME`
The time between the original and the delayed signal.
0 to 1000 ms
NRPN [11-15]:4

Feedback `FEEDBK`
The amount of feedback from the delayed signal.
0 to 100%
NRPN [11-15]:5

Damping `DAMPING`

The amount of damping of the feedback signal.

0 to 100%

NRPN [11-15]:6

Delay (sync) (`Delay.S`)

NRPN [11-15]:0 = 29

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

Bus `BUS`
Whether to apply the effect on the Main or Aux bus.
NRPN [11-15]:1

Send `SEND`
The level of input sent into the effect.
0 to 100%
NRPN [11-15]:2

Dry/wet `DRY/WET`
The balance between the pre-effect (dry) and post-effect (wet) signals.
NRPN [11-15]:3

Delay time `TIME`
The time between the original and the delayed signal.
NRPN [11-15]:4

Feedback `FEEDBK`
The amount of feedback from the delayed signal.
0 to 100%
NRPN [11-15]:5

Damping `DAMPING`
The amount of damping of the feedback signal.
0 to 100%
NRPN [11-15]:6

Ping-pong delay (`PngDly`)

NRPN [11-15]:0 = 30

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

Send `SEND`
The level of input sent into the effect.
0 to 100%
NRPN [11-15]:2

Dry/wet `DRY/WET`
The balance between the pre-effect (dry) and post-effect (wet) signals.
NRPN [11-15]:3

Bus balance `BALANCE`
NRPN [11-15]:1

Delay time `TIME`
The time between the original and the delayed signal.
0 to 1000 ms
NRPN [11-15]:4

Feedback `FEEDBK`
The amount of feedback from the delayed signal.
0 to 100%
NRPN [11-15]:5

Damping `DAMPING`
The amount of damping of the feedback signal.
0 to 100%
NRPN [11-15]:6

Ping-pong delay (sync) (`PingDlS`)

NRPN [11-15]:0 = 31

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

Send `SEND`
The level of input sent into the effect.
0 to 100%
NRPN [11-15]:2

Dry/wet `DRY/WET`
The balance between the pre-effect (dry) and post-effect (wet) signals.
NRPN [11-15]:3

Bus balance `BALANCE`
NRPN [11-15]:1

Delay time `TIME`
The time between the original and the delayed signal.
NRPN [11-15]:4

Feedback `FEEDBK`
The amount of feedback from the delayed signal.
0 to 100%
NRPN [11-15]:5

Damping `DAMPING`
The amount of damping of the feedback signal.
0 to 100%
NRPN [11-15]:6

Reverb

NRPN [11-15]:0 = 32

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.
NRPN [11-15]:1

Send `SEND`
The level of input sent into the effect.
0 to 100%
NRPN [11-15]:2

Dry/wet `DRY/WET`
The balance between the pre-effect (dry) and post-effect (wet) signals.
NRPN [11-15]:3

Time `TIME`
The amount of reverberation time.
0 to 100%
NRPN [11-15]:4

Diffusion `DIFF`
The density of the reverberation.
0 to 100%
NRPN [11-15]:5

Damping `DAMPING`

The amount of damping.

0 to 100%

NRPN [11-15]:6

Granular processor (`Grains`)

NRPN [11-15]:0 = 33

Creates audio textures by combining short segments of the input.

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

Bus `BUS`
Whether to apply the effect on the Main or Aux bus.
NRPN [11-15]:1

Send `SEND`
The level of input sent into the effect.
0 to 100%
NRPN [11-15]:2

Dry/wet `DRY/WET`
The balance between the pre-effect (dry) and post-effect (wet) signals.
NRPN [11-15]:3

Position `POS`
How far back in time to select grains.
0 to 100%
NRPN [11-15]:4

Grain size `SIZE`
The size of the grains.
0 to 100%
NRPN [11-15]:5

Grain density `DENSITY`
The amount of grains to generate.
0 to 100%
NRPN [11-15]:6

Randomize grains `RANDOM`
Whether to play the grains at constant rate or at random.
NRPN [11-15]:12

Grain shape `SHAPE`
The envelope shape of the grains (hard/smooth square, ramp up/down, triangle).
0 to 100%
NRPN [11-15]:8

Diffusion `DIFF`
The amount of transient smearing in the grains.
0 to 100%
NRPN [11-15]:9

Reverse grains `REVERSE`
Whether to reverse playback of the grains.
NRPN [11-15]:10

Pitch `PITCH`
The amount of transposition to apply to the grains.
NRPN [11-15]:11

Feedback `FEEDBK`
The amount of output fed back into the input.
0 to 100%
NRPN [11-15]:7

Modulators

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

Module name	Label
Circle map Unfold Arnold's circle map sequence on each trigger	CRCMAP
Discrete chaotic map Apply a specific chaotic map equation on each trigger	DC.MAP
Cellular automaton Use specific bits of a running cellular automaton	CELL
Gate combinator Perform successive operations on a series of gates; e.g., (G1 and G2) or G3	GATEC
Trigger combinator Perform successive operations on a series of triggers; e.g., (T1 and T2) or T3	TRIGC
Gate to trigger Converts gate transitions into triggers	GT>TRIG
Gate delay Delay the gate signal by a given time offset	GDELAY
Trigger delay Delay the trigger signal by a given time offset	TDELAY
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
Delay Delay the signal by a given time offset	DELAY
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
Compare Determine when the input value goes above or below a threshold	CMPR
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., min(I1+I2,I3)	CALC

Simple LFO (LFO)

NRPN [16-31]:0 = 1

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
NRPN [16-31]:1

Advanced LFO (LF0+)

NRPN [16-31]:0 = 2

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
NRPN [16-31]:1

Trigger TRIGGER
A trigger used to reset the LFO.
NRPN [16-31]:2

Shape SHAPE
The shape of the oscillator waveform.
NRPN [16-31]:3 = [0-4]

- *Sine*

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

  NRPN = 0
- *Triangle / Sawtooth*

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

  NRPN = 1
- *Square*

  Variation adjusts pulse width.

  NRPN = 2

— *Stepped triangle*

Variation adjusts the number of steps (quantization).

NRPN = 3

— *Noise*

Variation adjusts the amount of interpolation between noise samples.

NRPN = 4

Shape variation `SHP.VAR`

Sweeps through variations of the oscillator waveform.

0 to 100%

NRPN [16-31]:4

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

NRPN [16-31]:5

Phase on reset `PHASE`

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

0 to 100%

NRPN [16-31]:6

Trigger output (reset) `TC.OUT`

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

NRPN [16-31]:7

Slow LFO (SLOWFO)

NRPN [16-31]:0 = 3

Low-frequency oscillator with very long periods of time.

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

Period PERIOD
The time to generate a single cycle of the waveform.
1 day to 1 second
NRPN [16-31]:1

Trigger TRIGGER
A trigger used to reset the LFO.
NRPN [16-31]:2

Shape SHAPE
The shape of the oscillator waveform.
NRPN [16-31]:3 = [0-4]

- *Sine*

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

  NRPN = 0
- *Triangle / Sawtooth*

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

  NRPN = 1
- *Square*

  Variation adjusts pulse width.

  NRPN = 2

— *Stepped triangle*

Variation adjusts the number of steps (quantization).

NRPN = 3

— *Noise*

Variation adjusts the amount of interpolation between noise samples.

NRPN = 4

Shape variation `SHP.VAR`

Sweeps through variations of the oscillator waveform.

0 to 100%

NRPN [16-31]:4

Phase on reset `PHASE`

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

0 to 100%

NRPN [16-31]:6

Trigger output (reset) `TC.OUT`

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

NRPN [16-31]:7

LFO (sync) (`LFO.S`)

NRPN [16-31]:0 = 4

Low-frequency oscillator synced to the tempo.

Period `PERIOD`
The time to generate a single cycle of the waveform.
NRPN [16-31]:1

Trigger `TRIGGER`
A trigger used to reset the LFO.
NRPN [16-31]:2

Shape `SHAPE`
The shape of the oscillator waveform.
NRPN [16-31]:3 = [0-4]

- *Sine*

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

  NRPN = 0
- *Triangle / Sawtooth*

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

  NRPN = 1
- *Square*

  Variation adjusts pulse width.

  NRPN = 2
- *Stepped triangle*

  Variation adjusts the number of steps (quantization).

  NRPN = 3

— *Noise*

Variation adjusts the amount of interpolation between noise samples.

NRPN = 4

Shape variation `SHP.VAR`

Sweeps through variations of the oscillator waveform.

0 to 100%

NRPN [16-31]:4

Phase on reset `PHASE`

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

0 to 100%

NRPN [16-31]:6

Trigger output (reset) `TC.OUT`

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

NRPN [16-31]:7

Envelope (ENV)

NRPN [16-31]:0 = 5

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
NRPN [16-31]:4

Decay time DECAY
The time needed to go from 1 to the sustain level in the decay phase.
0 to 8 seconds
NRPN [16-31]:5

Sustain level SUSTAIN
The output level during the sustain phase.
0 to 100%
NRPN [16-31]:6

Release time RELEASE
The time needed to go from the sustain level to 0 in the release phase.
0 to 8 seconds
NRPN [16-31]:7

DAHDSR Envelope (DAHDSR)

NRPN [16-31]:0 = 6

DAHDSR envelope generator.

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

Gate GATE
The gate used to control the envelope.
NRPN [16-31]:1

Retrigger RETRIG
A trigger source used to reset the envelope.
NRPN [16-31]:2

Shape SHAPE
The shape of the envelope (Log -> Lin -> Exp).
NRPN [16-31]:3

Attack time ATTACK
The time needed to go from 0 to 1 in the attack phase.
0 to 8 seconds
NRPN [16-31]:4

Decay time DECAY
The time needed to go from 1 to the sustain level in the decay phase.
0 to 8 seconds
NRPN [16-31]:5

Sustain level `SUSTAIN`

The output level during the sustain phase.

0 to 100%

NRPN [16-31]:6

Release time `RELEASE`

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

0 to 8 seconds

NRPN [16-31]:7

Delay time `DELAY`

The time before the envelope starts its attack phase.

0 to 8 seconds

NRPN [16-31]:8

Hold time `HOLD`

The time before the envelope starts its decay phase.

0 to 8 seconds

NRPN [16-31]:9

Hard reset `HARD.RST`

Whether the envelope is reset to zero when restarted.

NRPN [16-31]:10

Trigger output (sustain) `TC.OUT`

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

NRPN [16-31]:11

Step sequencer (STEPSEQ)

NRPN [16-31]:0 = 7

Change a value according to a predetermined pattern over time.

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

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

Step count `STEPS`
The number of steps in the sequence.
1 to 16
NRPN [16-31]:1

Step trigger `TRIG`
A trigger used to advance by one step.
NRPN [16-31]:2

Reset trigger `RST.TRIG`
A trigger used to reset the sequence to the beginning.
NRPN [16-31]:3

Step 1 `1`
0 to 100%
NRPN [16-31]:4

Step 2 `2`
0 to 100%
NRPN [16-31]:5

Step 3 3
0 to 100%
NRPN [16-31]:6

Step 4 4
0 to 100%
NRPN [16-31]:7

Step 5 5
0 to 100%
NRPN [16-31]:8

Step 6 6
0 to 100%
NRPN [16-31]:9

Step 7 7
0 to 100%
NRPN [16-31]:10

Step 8 8
0 to 100%
NRPN [16-31]:11

Step 9 9
0 to 100%
NRPN [16-31]:12

Step 10 10
0 to 100%
NRPN [16-31]:13

Step 11 11
0 to 100%
NRPN [16-31]:14

Step 12 12
0 to 100%
NRPN [16-31]:15

Step 13 13
0 to 100%
NRPN [16-31]:16

Step 14 14
0 to 100%
NRPN [16-31]:17

Step 15 15
0 to 100%
NRPN [16-31]:18

Step 16 16
0 to 100%
NRPN [16-31]:19

Euclidean sequencer (EUCLID)

NRPN [16-31]:0 = 8

Generate euclidean rhythms.

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

Beats `BEATS`
The number of beats in the sequence.
NRPN [16-31]:1

Steps `STEPS`
The number of steps in the sequence.
NRPN [16-31]:2

Trigger `TRIGGER`
A trigger used to advance to the next step in the sequence.
NRPN [16-31]:3

Rotation `SHIFT`
The number of steps by which to shift the sequence.
NRPN [16-31]:4

Reset trigger `RST.TRIG`
A trigger used to reset to the initial step.
NRPN [16-31]:6

Trigger output `TC.OUT`
A trigger output for when the current step has a beat.
NRPN [16-31]:7

Hex sequencer (HEXSEQ)

NRPN [16-31]:0 = 9

Generate rhythms from hexadecimal numbers.

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

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

Digit 0 `DIGIT0`
A digit of the hexadecimal number.
NRPN [16-31]:1

Digit 1 `DIGIT1`
A digit of the hexadecimal number.
NRPN [16-31]:2

Trigger `TRIGGER`
A trigger used to advance to the next step in the sequence.
NRPN [16-31]:3

Digit 2 `DIGIT2`
A digit of the hexadecimal number.
NRPN [16-31]:4

Digit 3 `DIGIT3`
A digit of the hexadecimal number.
NRPN [16-31]:5

Reset trigger `RST.TRIG`
A trigger used to reset to the initial step.
NRPN [16-31]:6

Trigger output `TC.OUT`
A trigger output for when the current step has a beat.
NRPN [16-31]:7

Digit 4 `DIGIT4`
A digit of the hexadecimal number.
NRPN [16-31]:8

Digit 5 `DIGIT5`
A digit of the hexadecimal number.
NRPN [16-31]:9

Digit 6 `DIGIT6`
A digit of the hexadecimal number.
NRPN [16-31]:10

Digit 7 `DIGIT7`
A digit of the hexadecimal number.
NRPN [16-31]:11

Envelope follower (FOLLOW)

NRPN [16-31]:0 = 10

Transform an audio signal into a smoothed value.

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

Input IN

The audio input source to use.

NRPN [16-31]:1 = [0-4]

- *None*

  NRPN = 0
- *Audio input (Left)*

  NRPN = 1
- *Audio input (Right)*

  NRPN = 2
- *Main bus (post-FX)*

  NRPN = 3
- *Aux bus (post-FX)*

  NRPN = 4

Rise time RISE

0 to 5 seconds

NRPN [16-31]:2

Fall time FALL

0 to 5 seconds

NRPN [16-31]:3

Noise floor FLOOR

0 to 100%

NRPN [16-31]:4

Gain `GAIN`

0 to 400%

NRPN [16-31]:5

Timbre follower (TIMBRE)

NRPN [16-31]:0 = 11

Extract the brightness from an audio signal.

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

Input `IN`
The audio input source to use.
NRPN [16-31]:1 = [0-4]

- *None*
  NRPN = 0
- *Audio input (Left)*
  NRPN = 1
- *Audio input (Right)*
  NRPN = 2
- *Main bus (post-FX)*
  NRPN = 3
- *Aux bus (post-FX)*
  NRPN = 4

Rise time `RISE`
0 to 5 seconds
NRPN [16-31]:2

Fall time `FALL`
0 to 5 seconds
NRPN [16-31]:3

Noise floor `FLOOR`
0 to 100%
NRPN [16-31]:4

Gain `GAIN`

0 to 400%

NRPN [16-31]:5

Sensitivity `SENS`

0 to 100%

NRPN [16-31]:6

Drum trigger (DRUM)

NRPN [16-31]:0 = 12

Derive a signal suitable for percussive sound triggering.

This is another envelope follower more suitable for percussion.

Input `IN`
The audio input source to use.
NRPN [16-31]:1 = [0-4]

- *None*
  NRPN = 0
- *Audio input (Left)*
  NRPN = 1
- *Audio input (Right)*
  NRPN = 2
- *Main bus (post-FX)*
  NRPN = 3
- *Aux bus (post-FX)*
  NRPN = 4

Attack time `ATTACK`
NRPN [16-31]:2

Release time `RELEASE`
NRPN [16-31]:3

Threshold `THRESH`
0 to 100%
NRPN [16-31]:4

Gain `GAIN`

0 to 400%

NRPN [16-31]:5

Trigger output `TC.OUT`

The trigger channel to use as an output.

NRPN [16-31]:6

Xform (XFORM)

NRPN [16-31]:0 = 13

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.
NRPN [16-31]:1

Minimum `MIN`
The minimum value of the input to consider.
0 to 100%
NRPN [16-31]:2

Maximum `MAX`
The maximum value of the input to consider.
0 to 100%
NRPN [16-31]:3

Curve `CURVE`
The amount of curvature to apply to the value.
-100 to 100%
NRPN [16-31]:4

Curve type `CURVE`

The kind of curve to apply to the value.

NRPN [16-31]:5 = [0-1]

- *Log/Exp*

  Log -> Linear -> Exp.

  NRPN = 0

- *Sigmoid*

  Sigmoid Horiz. -> Linear -> Sigmoid Vert.

  NRPN = 1

Rise time `RISE`

The amount of smoothing when the value increases.

0 to 5 seconds

NRPN [16-31]:6

Fall time `FALL`

The amount of smoothing when the value decreases.

0 to 5 seconds

NRPN [16-31]:7

Curve (CURVE)

NRPN [16-31]:0 = 14

Apply a curve to a value.

Input `IN`
The value to use.
NRPN [16-31]:1

Shape `SHAPE`
The shape of the curve to apply.
NRPN [16-31]:2 = [0-1]

- *Log/Exp*
  Log -> Linear -> Exp.
  NRPN = 0
- *Sigmoid*
  Sigmoid Horiz. -> Linear -> Sigmoid Vert.
  NRPN = 1

Amount `AMOUNT`
The amount of curvature to apply.
-100 to 100%
NRPN [16-31]:3

Quantize (QTZ)

NRPN [16-31]:0 = 15

Reduce the resolution of a value.

Input `IN`
The value to use.
NRPN [16-31]:1

Steps `STEPS`
The number of different possible values of the output.
1 to 127
NRPN [16-31]:2

Change polarity (POLAR)

NRPN [16-31]:0 = 16

Make a unipolar value bipolar, and vice-versa.

Input `IN`
The value to use.
NRPN [16-31]:1

Polarity change `CHANGE`
NRPN [16-31]:2 = [0-1]

- *Unipolar to bipolar*
  NRPN = 0
- *Bipolar to unipolar*
  NRPN = 1

Smooth (SMOOTH)

NRPN [16-31]:0 = 17

Smooth out the variations of a value.

Input `IN`
An input value.
NRPN [16-31]:1

Rise time `RISE`
The amount of smoothing when the value increases.
0 to 5 seconds
NRPN [16-31]:2

Fall time `FALL`
The amount of smoothing when the value decreases.
0 to 5 seconds
NRPN [16-31]:3

Shape `SHAPE`
Whether smoothing is linear or exponential.
NRPN [16-31]:4 = [0-1]

- *Exponential smoothing*
  Uses a low-pass filter.
  NRPN = 0
- *Linear smoothing*
  Uses linear interpolation.
  NRPN = 1

Link times `LINK`

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

NRPN [16-31]:5

Reset trigger `RST.TRIG`

A trigger used to reset smoothing.

NRPN [16-31]:7

Accumulate (ACCU)

NRPN [16-31]:0 = 18

Accumulate a value or variations over time.

Input `IN`
An input value.
NRPN [16-31]:1

Gain `GAIN`
An attenuation factor for the input.
0 to 25%
NRPN [16-31]:2

High-pass filtering `HIPASS`
The amount of high-pass filtering on the input.
0 to 100%
NRPN [16-31]:3

Overflow behavior `OVRFLOW`
The behavior when the accumulator overflows.
NRPN [16-31]:4 = [0-3]

- *Saturate*

  Keep the value to the maximum until it is reset.

  NRPN = 0
- *Reset*

  Reset to zero upon overflow.

  NRPN = 1

— *Wrap*

Wrap around the value range.

NRPN = 2

— *Fold*

Go back towards zero upon overflow.

NRPN = 3

Reset trigger `RST.TRIG`

A trigger used to reset the accumulator.

NRPN [16-31]:5

Trigger output (overflow) `TC.OUT`

A trigger output for when the accumulator overflows.

NRPN [16-31]:7

Lookup table (LUT16)

NRPN [16-31]:0 = 19

Change a value according to a predetermined shape.

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

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

Input `IN`
The value to use.
NRPN [16-31]:1

Table size `SIZE`
The number of values in the table.
1 to 16
NRPN [16-31]:2

Interpolate `LERP?`
Whether to interpolate in-between values.
NRPN [16-31]:3

Value 1 `1`
0 to 100%
NRPN [16-31]:4

Value 2 `2`
0 to 100%
NRPN [16-31]:5

Value 3 3
0 to 100%
NRPN [16-31]:6

Value 4 4
0 to 100%
NRPN [16-31]:7

Value 5 5
0 to 100%
NRPN [16-31]:8

Value 6 6
0 to 100%
NRPN [16-31]:9

Value 7 7
0 to 100%
NRPN [16-31]:10

Value 8 8
0 to 100%
NRPN [16-31]:11

Value 9 9
0 to 100%
NRPN [16-31]:12

Value 10 10
0 to 100%
NRPN [16-31]:13

Value 11 11
0 to 100%
NRPN [16-31]:14

Value 12 12
0 to 100%
NRPN [16-31]:15

Value 13 13
0 to 100%
NRPN [16-31]:16

Value 14 14
0 to 100%
NRPN [16-31]:17

Value 15 15
0 to 100%
NRPN [16-31]:18

Value 16 16
0 to 100%
NRPN [16-31]:19

Bouncing ball (BOUNCE)

NRPN [16-31]:0 = 20

Simulate the movement of a single bouncing ball.

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

Simulation speed `SPEED`
The speed of the simulation.
0 to 100%
NRPN [16-31]:1

Bounciness `BOUNCE`
The restitution coefficient, representing how the ball bounces when hitting the ground.
0 to 100%
NRPN [16-31]:2

Trigger `TRIGGER`
A trigger used to throw the ball.
NRPN [16-31]:3

Initial height `HEIGHT`
The height of the ball when thrown.
NRPN [16-31]:4

Force `FORCE`
The force at which the ball is thrown.
0 to 100%
NRPN [16-31]:5

Trigger output `TC.OUT`

A trigger output for when a ball hits the ground.

NRPN [16-31]:7

Ball impact (BALL+)

NRPN [16-31]:0 = 21

Simulate the impact of several independent bouncing balls.

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

Simulation speed `SPEED`
The speed of the simulation.
0 to 100%
NRPN [16-31]:1

Bounciness `BOUNCE`
The restitution coefficient, representing how a ball bounces when hitting the ground.
0 to 100%
NRPN [16-31]:2

Trigger `TRIGGER`
A trigger used to throw another ball in the air.
NRPN [16-31]:3

Initial height `HEIGHT`
The height of a ball when thrown.
NRPN [16-31]:4

Throw force `FORCE`
The force at which a ball is thrown.
0 to 100%
NRPN [16-31]:5

Impact fall time `FALL`

The amount of damping of the impulse after impact.

0 to 5 seconds

NRPN [16-31]:6

Trigger output `TC.OUT`

A trigger output for when a ball hits the ground.

NRPN [16-31]:7

Spring-damper system (SPRING)

NRPN [16-31]:0 = 22

Simulate a spring-damper system attached to the input.

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

Input `IN`
An input value.
NRPN [16-31]:1

Damping `DAMP`
The amount of resistance of the spring.
0 to 100%
NRPN [16-31]:2

Stiffness `STIFF`
The stiffness of the spring.
0 to 100%
NRPN [16-31]:3

Logistic map (LGSMAP)

NRPN [16-31]:0 = 23

Unfold the logistic equation on each trigger.

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

Reset trigger `RST.TRIG`
A trigger used to reset to the initial value.
NRPN [16-31]:1

Parameter (r) `r`
The r parameter of the logistic equation.
NRPN [16-31]:2

Trigger `TRIGGER`
A trigger used to advance to the next value in the sequence.
NRPN [16-31]:3

Center output `CENTER`
Whether to center around (r-1)/r, making the output bipolar.
NRPN [16-31]:4

Tent map (TNTMAP)

NRPN [16-31]:0 = 24

Unfold the tent map sequence on each trigger.

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

Reset trigger `RST.TRIG`
A trigger used to reset to the initial value.
NRPN [16-31]:1

Parameter (mu) `mu`
The mu parameter of the tent map equation.
NRPN [16-31]:2

Trigger `TRIGGER`
A trigger used to advance to the next value in the sequence.
NRPN [16-31]:3

Center output `CENTER`
Whether to center around (mu-1)/mu, making the output bipolar.
NRPN [16-31]:4

Circle map (CRCMAP)

NRPN [16-31]:0 = 25

Unfold Arnold's circle map sequence on each trigger.

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

Reset trigger `RST.TRIG`
A trigger used to reset to the initial value.
NRPN [16-31]:1

Coupling strength (K) `K`
The K parameter of the circle map equation.
0 to 100%
NRPN [16-31]:2

Trigger `TRIGGER`
A trigger used to advance to the next value in the sequence.
NRPN [16-31]:3

Natural frequency (Omega) `OMEGA`
The Omega parameter of the circle map equation.
0 to 100%
NRPN [16-31]:4

Discrete chaotic map (DC.MAP)

NRPN [16-31]:0 = 26

Apply a specific chaotic map equation on each trigger.

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

Reset trigger `RST.TRIG`
A trigger used to reset to the initial value.
NRPN [16-31]:1

Sequence equation `EQN`
The equation to use to compute the next value of the sequence.
NRPN [16-31]:2

Trigger `TRIGGER`
A trigger used to advance to the next value in the sequence.
NRPN [16-31]:3

Initial value `INIT`
The value when the reset trigger occurs.
0 to 100%
NRPN [16-31]:4

Cellular automaton (CELL)

NRPN [16-31]:0 = 27

Use specific bits of a running cellular automaton.

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

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

Reset trigger `RST.TRIG`
A trigger used to reset to the initial state.
NRPN [16-31]:1

Rule `RULE`
The Wolfram code for the rule used to advance the cells.
NRPN [16-31]:2

Trigger `TRIGGER`
A trigger used to advance to the next value in the sequence.
NRPN [16-31]:3

Initial state `INIT`
The state when the reset trigger occurs.
NRPN [16-31]:4

Cell A `CELL.A`
A specific cell number.
NRPN [16-31]:6

Trig. out. (Cell A) `TRIG.A`
The trigger channel to which to output changes to the cell.
NRPN [16-31]:7

Cell B `CELL.B`
A specific cell number.
NRPN [16-31]:8

Trig. out. (Cell B) `TRIG.B`
The trigger channel to which to output changes to the cell.
NRPN [16-31]:9

Cell C `CELL.C`
A specific cell number.
NRPN [16-31]:10

Trig. out. for Cell C `TRIG.C`
The trigger channel to which to output changes to the cell.
NRPN [16-31]:11

Gate combinator (GATEC)

NRPN [16-31]:0 = 28

Perform successive operations on a series of gates; e.g., (G1 and G2) or G3.

Gate output `GT.OUT`

The gate channel to use as output and as a base input.

NRPN [16-31]:1

Operation 1 `OP1`

The operation to apply to the base input and input 1.

NRPN [16-31]:2 = [0-8]

- *Replace (=)*

  NRPN = 0
- *OR*

  NRPN = 1
- *AND*

  NRPN = 2
- *XOR*

  NRPN = 3
- *IMPLY*

  NRPN = 4
- *NOR*

  NRPN = 5
- *NAND*

  NRPN = 6
- *XNOR*

  NRPN = 7

— *NIMPLY*

NRPN = 8

Input 1 `IN1`

An input gate.

NRPN [16-31]:3

Operation 2 `OP2`

The operation to apply to the previous result and input 2.

NRPN [16-31]:4 = [0-8]

— *Replace (=)*

NRPN = 0

— *OR*

NRPN = 1

— *AND*

NRPN = 2

— *XOR*

NRPN = 3

— *IMPLY*

NRPN = 4

— *NOR*

NRPN = 5

— *NAND*

NRPN = 6

— *XNOR*

NRPN = 7

— *NIMPLY*

NRPN = 8

Input 2 `IN2`
An input gate.
NRPN [16-31]:5

Operation 3 `OP3`
The operation to apply to the previous result and input 3.
NRPN [16-31]:6 = [0-8]

- *Replace (=)*
  NRPN = 0
- *OR*
  NRPN = 1
- *AND*
  NRPN = 2
- *XOR*
  NRPN = 3
- *IMPLY*
  NRPN = 4
- *NOR*
  NRPN = 5
- *NAND*
  NRPN = 6
- *XNOR*
  NRPN = 7
- *NIMPLY*
  NRPN = 8

Input 3 `IN3`
An input gate.
NRPN [16-31]:7

Trigger combinator (TRIGC)

NRPN [16-31]:0 = 29

Perform successive operations on a series of triggers; e.g., (T1 and T2) or T3.

Trigger output `TC.OUT`
The trigger channel to use as output and as a base input.
NRPN [16-31]:1

Operation 1 `OP1`
The operation to apply to the base input and input 1.
NRPN [16-31]:2 = [0-4]

- *=*
  NRPN = 0
- *OR*
  NRPN = 1
- *AND*
  NRPN = 2
- *XOR*
  NRPN = 3
- *IMPLY*
  NRPN = 4

Input 1 `IN1`
An input trigger.
NRPN [16-31]:3

Operation 2 `OP2`
The operation to apply to the previous result and input 2.
NRPN [16-31]:4 = [0-4]

— =

NRPN = 0

— *OR*

NRPN = 1

— *AND*

NRPN = 2

— *XOR*

NRPN = 3

— *IMPLY*

NRPN = 4

Input 2 `IN2`

An input trigger.

NRPN [16-31]:5

Operation 3 `OP3`

The operation to apply to the previous result and input 3.

NRPN [16-31]:6 = [0-4]

— =

NRPN = 0

— *OR*

NRPN = 1

— *AND*

NRPN = 2

— *XOR*

NRPN = 3

— *IMPLY*

NRPN = 4

Input 3 `IN3`

An input trigger.

NRPN [16-31]:7

Gate to trigger (GT>TRIG)

NRPN [16-31]:0 = 30

Converts gate transitions into triggers.

Gate input `INPUT`
The gate source to use as an input.
NRPN [16-31]:1

Trigger output (rising edge) `TC.OPEN`
The trigger channel to output to when the gate opens.
NRPN [16-31]:2

Trigger output (falling edge) `TC.CLOS`
The trigger channel to output to when the gate closes.
NRPN [16-31]:3

Gate delay (GDELAY)

NRPN [16-31]:0 = 31

Delay the gate signal by a given time offset.

Gate `GATE`
The gate source to delay.
NRPN [16-31]:1

Delay time `TIME`
The time by which the input is delayed.
0 to 1000 ms
NRPN [16-31]:2

Trigger delay (TDELAY)

NRPN [16-31]:0 = 32

Delay the trigger signal by a given time offset.

Trigger `TRIG`
The trigger source to delay.
NRPN [16-31]:1

Delay time `TIME`
The time by which the input is delayed.
0 to 1000 ms
NRPN [16-31]:2

Impulse (`IMPULSE`)

NRPN [16-31]:0 = 33

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.
NRPN [16-31]:1

Trigger `TRIGGER`
A trigger used to generate the impulse from the current input value.
NRPN [16-31]:2

Fall time `FALL`
The time for the impulse to fall down to zero.
0 to 5 seconds
NRPN [16-31]:3

Count (COUNT)

NRPN [16-31]:0 = 34

Count occurrences of a trigger.

Trigger `TRIGGER`
A trigger.
NRPN [16-31]:1

Steps `STEPS`
The number of steps to count.
1 to 127
NRPN [16-31]:2

Wrap-around `WRAP`
Whether to wrap around when the number of steps is reached.
NRPN [16-31]:3

Reset trigger `RST.TRIG`
A trigger used to reset the count to zero.
NRPN [16-31]:4

Trigger output (end) `TC.OUT`
A trigger output for when the number of steps is reached.
NRPN [16-31]:7

Time (`TIME`)

NRPN [16-31]:0 = 35

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.
NRPN [16-31]:1

Time `TIME`
The time to reach 100% of the output value.
0 to 8 seconds
NRPN [16-31]:2

Trigger output (end) `TC.OUT`
A trigger output for when 100% of the output value is reached.
NRPN [16-31]:3

Delay (DELAY)

NRPN [16-31]:0 = 36

Delay the signal by a given time offset.

Input `IN`
An input value.
NRPN [16-31]:1

Delay time `TIME`
The time by which the input is delayed.
0 to 1000 ms
NRPN [16-31]:2

Latch (LATCH)

NRPN [16-31]:0 = 37

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.
NRPN [16-31]:1

Trigger `TRIGGER`
A trigger used to capture the input value.
NRPN [16-31]:2

Minimum (`MIN`)

NRPN [16-31]:0 = 38

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.
NRPN [16-31]:1

Trigger `TRIGGER`
A trigger used to reset to the input value.
NRPN [16-31]:2

Maximum (MAX)

NRPN [16-31]:0 = 39

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.
NRPN [16-31]:1

Trigger `TRIGGER`
A trigger used to reset to the input value.
NRPN [16-31]:2

Compare (CMPR)

NRPN [16-31]:0 = 40

Determine when the input value goes above or below a threshold.

Input `IN`
An input value used for comparison.
NRPN [16-31]:1

Upper threshold `THRES>`
The value above which the signal must go.
0 to 100%
NRPN [16-31]:2

Lower threshold `THRES<`
The value below which the signal must go.
0 to 100%
NRPN [16-31]:3

Gate output `GT.OUT`
A gate output for when the signal is above the upper threshold.
NRPN [16-31]:4

Gate operation `GT.OP`
The operation to use to combine the new gate output to the existing value.
NRPN [16-31]:5 = [0-8]

- *Replace (=)*
  NRPN = 0
- *OR*
  NRPN = 1

- *AND*

  NRPN = 2
- *XOR*

  NRPN = 3
- *IMPLY*

  NRPN = 4
- *NOR*

  NRPN = 5
- *NAND*

  NRPN = 6
- *XNOR*

  NRPN = 7
- *NIMPLY*

  NRPN = 8

Trigger output (above) `TC.OUT>`

A trigger output for when the signal goes above the upper threshold.

NRPN [16-31]:6

Trigger output (below) `TC.OUT<`

A trigger output for when the signal goes below the lower threshold.

NRPN [16-31]:7

Clamp (CLAMP)

NRPN [16-31]:0 = 41

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.
NRPN [16-31]:1

Minimum `MIN`
The minimum value.
0 to 100%
NRPN [16-31]:2

Maximum `MAX`
The maximum value.
0 to 100%
NRPN [16-31]:3

Wrap (WRAP)

NRPN [16-31]:0 = 42

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.
NRPN [16-31]:1

Minimum `MIN`
The minimum value.
0 to 100%
NRPN [16-31]:2

Maximum `MAX`
The maximum value.
0 to 100%
NRPN [16-31]:3

Fold (FOLD)

NRPN [16-31]:0 = 43

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.
NRPN [16-31]:1

Minimum `MIN`
The minimum value.
0 to 100%
NRPN [16-31]:2

Maximum `MAX`
The maximum value.
0 to 100%
NRPN [16-31]:3

Interpolate (LERP)

NRPN [16-31]:0 = 44

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.
NRPN [16-31]:1

Input 2 `IN2`
The end value.
NRPN [16-31]:2

Interpolation position `LERP`
The mix between both values.
NRPN [16-31]:3

Interpolate (4-point) (LERP4)

NRPN [16-31]:0 = 45

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%
NRPN [16-31]:1

P2 Position `P2.POS`
The position of point P2.
0 to 100%
NRPN [16-31]:2

Interpolation position `LERP`
The mix between the values defined at 0%, P1, P2, and 100%.
NRPN [16-31]:3

Start value `0%`
The value when the interpolation is at 0%.
-100 to 100%
NRPN [16-31]:4

Value at P1 `P1%`
The value when the interpolation is at P1.
-100 to 100%
NRPN [16-31]:5

Value at P2 `P2%`

The value when the interpolation is at P2.

-100 to 100%

NRPN [16-31]:6

End value `100%`

The value when the interpolation is at 100%.

-100 to 100%

NRPN [16-31]:7

Calculate (CALC)

NRPN [16-31]:0 = 46

Perform successive operations on a series of values; e.g., min(I1+I2,I3).

Input 1 `IN1`

An input value.

NRPN [16-31]:1

Operation 1 `OP1`

The operation to apply to inputs 1 and 2.

NRPN [16-31]:2 = [0-4]

- *Add*

  Add both operands.

  NRPN = 0

- *Subtract*

  Subtract both operands.

  NRPN = 1

- *Multiply*

  Multiply both operands.

  NRPN = 2

- *Minimum*

  Take the minimum of both operands.

  NRPN = 3

- *Maximum*

  Take the maximum both operands.

  NRPN = 4

Input 2 `IN2`

An input value.

NRPN [16-31]:3

Operation 2 `OP2`

The operation to apply to the previous result and input 3.

NRPN [16-31]:4 = [0-4]

- *Add*

  Add both operands.

  NRPN = 0

- *Subtract*

  Subtract both operands.

  NRPN = 1

- *Multiply*

  Multiply both operands.

  NRPN = 2

- *Minimum*

  Take the minimum of both operands.

  NRPN = 3

- *Maximum*

  Take the maximum both operands.

  NRPN = 4

Input 3 `IN3`

An input value.

NRPN [16-31]:5

Operation 3 `OP3`

The operation to apply to the previous result and input 4.

NRPN [16-31]:6 = [0-4]

— *Add*

Add both operands.

NRPN = 0

— *Subtract*

Subtract both operands.

NRPN = 1

— *Multiply*

Multiply both operands.

NRPN = 2

— *Minimum*

Take the minimum of both operands.

NRPN = 3

— *Maximum*

Take the maximum both operands.

NRPN = 4

Input 4 `IN4`

An input value.

NRPN [16-31]:7

Mappings

Source `FROM`

The value that will be used to modulate a parameter.

NRPN [32-63]:0

Destination module `TO`

The module where the destination parameter is to be found.

NRPN [32-63]:1

Destination parameter `TO.PARAM`

The parameter to be modulated.

0 to 64

NRPN [32-63]:2

Amount `AMOUNT`

The amount by which the source modulates the destination parameter.

-100 to 100%

NRPN [32-63]:3

Minimum value `MIN`

The minimum source value.

0 to 100%

NRPN [32-63]:4

Maximum value `MAX`

The maximum source value.

0 to 100%

NRPN [32-63]:5

Curve amount `CURVE`

The amount of curvature to apply to the source value.

-100 to 100%

NRPN [32-63]:6

Smoothing amount `SMOOTH`

The amount of smoothing to apply to the source value.

0 to 5 seconds

NRPN [32-63]:7

Scale `SCALE`

The scale of the mapping amount.

0 to 100%

NRPN [32-63]:8

Sidechain `SIDE`

A value that can be used to modulate the amount.

NRPN [32-63]:9

Sidechain amount `SIDE.AMT`

How much the sidechain modulates the amount.

-100 to 100%

NRPN [32-63]:10

©2021 Aodyo.
All rights reserved.

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

contact@aodyo.com
www.aodyo.com

AODYO INSTRUMENTS