BUILDING IN PRIMARY

Document authored by: Adam Hanley

Software version: 6.0.1 (11/2015)

NATIVE INSTRUMENTS GmbH
Schlesische Str. 29-30
D-10997 Berlin
Germany
www.native-instruments.de

NATIVE INSTRUMENTS North America, Inc.
6725 Sunset Boulevard
5th Floor
Los Angeles, CA 90028
USA
www.native-instruments.com

NATIVE INSTRUMENTS K.K.
YO Building 3F
Jingumae 6-7-15, Shibuya-ku,
Tokyo 150-0001
Japan
www.native-instruments.co.jp

NATIVE INSTRUMENTS UK Limited
18 Phipp Street
London EC2A 4NU
UK
www.native-instruments.com

Table of Contents

1 Welcome to Building in Primary

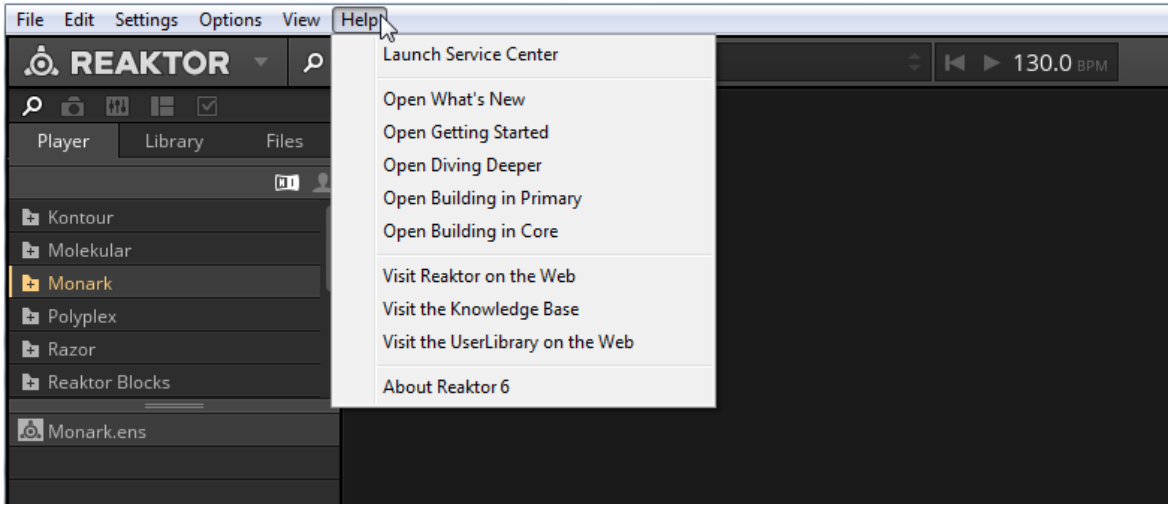
Welcome to Building in Primary, a tutorial for building in REAKTOR's Primary level. It describes the functions in REAKTOR that relate to building custom instruments, effects, or sequencers, enabling you to modify existing Ensembles or create your own.

This document assumes you are already familiar with the REAKTOR interface, as well as loading and playing with Ensembles. At the least, you should have read the chapters in the REAKTOR 6 Diving Deeper document regarding navigation in the Structure View.

This document begins with an introduction to basic functions and concepts, followed with a collection of tutorials. These tutorials introduce building concepts through common examples.

The final sections are guides to additional building features available in REAKTOR, as well as some general tutorials for general building tasks (like skinning controls, and preparing Ensembles for the KOMPLETE PLAYER Browser).

1.1 The REAKTOR 6 Documentation



The REAKTOR documentation is accessible from the *Help* menu

The documentation for REAKTOR 6 is divided into five separate documents, guiding you from loading and playing pre-built Ensembles to building your own Instruments.

- **REAKTOR 6 What Is New** is written for users who are already familiar with previous versions of REAKTOR and only describes the latest features in brief.
- **REAKTOR 6 Getting Started** is for new users. It is the only document needed for users who intend to use REAKTOR for loading and playing pre-built REAKTOR instruments and effects.
- **REAKTOR 6 Diving Deeper** expands on the concepts introduced in the Getting Started document. It provides more detail on subjects like Snapshots (REAKTOR's preset system), and introduces advanced topics like OSC control and combining multiple Instruments in one Ensemble.

- **REAKTOR 6 Building in Primary** shows you how to build your own Instruments in REAKTOR's Primary level. It focuses on a series of tutorials that guide you through building your first synthesizers, effects, and sequencers.
- **REAKTOR 6 Building in Core** describes the Core level of REAKTOR with its low-level building features, which can be used for implementing custom DSP algorithms. It includes reference information about the Core Macro Library, an comprehensive collection of DSP building blocks.

With the exception of the What Is New document, each of the documents listed above builds on the knowledge of the previous documents. While it is not necessary to read all of every document, some of the more advanced documents, like Building in Primary, assume knowledge from the previous documents.

1.2 Manual Conventions

This section introduces you to the signage and text highlighting used in this manual.

- Text appearing in (drop-down) menus (such as *Open...*, *Save as...* etc.) and paths to locations on your hard disk or other storage devices is printed in *italics*.
- Text appearing elsewhere (labels of buttons, controls, text next to checkboxes etc.) is printed in blue. Whenever you see this formatting applied, you will find the same text appearing somewhere on the screen.
- Important names and concepts are printed in **bold**.
- References to keys on your computer's keyboard you'll find put in square brackets (e.g., "Press [Shift] + [Enter]").

▶ Single instructions are introduced by this play button type arrow.

→ Results of actions are introduced by this smaller arrow.

An indented, gray paragraph contains additional, contextual information.

Furthermore, this manual uses particular formatting to point out special facts and to warn you of potential issues. The icons introducing these notes let you see what kind of information is to be expected:

The speech bubble icon indicates a useful tip that may often help you to solve a task more efficiently.

The exclamation mark icon highlights important information that is essential for the given context.

The red cross icon warns you of serious issues and potential risks that require your full attention.

2 Introductory Topics

2.1 Browsing and Loading Modules and Macros

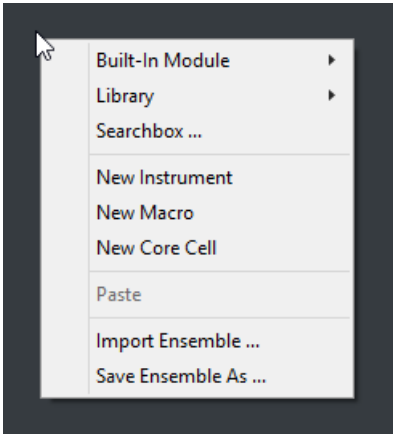
The Primary level of REAKTOR uses two types of building-blocks to create structures, these are:

- **Modules**: These are the lowest level of building-block in the REAKTOR Primary level.
- **Macros**: These are containers used to group together sub-structures of other Modules and Macros.

There are a few different ways to browse for and load Modules and Macros.

2.1.1 Using the Context Menu

Clicking the right mouse button (or [Ctrl] + click on OSX) on any empty area in the structure view will open the context menu.



The Context Menu.

For the purposes of browsing and loading Modules and Macros, you will only need to use the following entries:

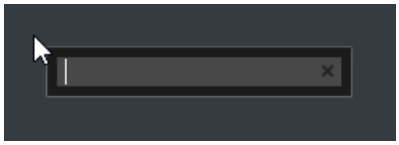
- *Built-In Module*: Opens a sub-menu from which you can browse for Modules. Once you have located the Module you wish to load, click on its name in the menu.
- *Library*: Opens a sub-menu from which you can browse for and load Macros, and Instruments.
- *Open Searchbox*: Opens the Searchbox, which is covered in section ↑2.1.2, Using the Searchbox ↑2.1.2, Using the Searchbox.
- *New Macro*: Creates a new empty Macro.

The Context Menu is a useful thing to use when you are unsure of the name of the Module or Macro you wish to load.

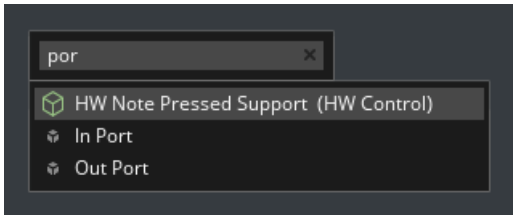
2.1.2 Using the Searchbox

When you are familiar with the Modules and Macros available in REAKTOR, you may find it faster to use the Searchbox to browse for and load content.

1. The fastest way to open the Searchbox is to press the [Enter] key when in the structure view.
2. Alternatively, you can open the Searchbox by selecting *Open Searchbox* in the Context Menu.
3. With the Searchbox now displayed, you can search for a Module or Macro by typing its name.



4. The Searchbox will autocomplete your entry.



5. When the Module/Macro you wish to load appears, you can load it by clicking on its name.
6. If the Module/Macro you want is the first or only entry that appears, you can load it by pressing [Enter].
7. You can also select from the list by using the arrow keys, before clicking [Enter].

When you next open the Searchbox, it will display a list of your most recently loaded Modules and Macros. You can then load any of the Modules/Macros in this list as you would normally.

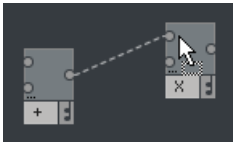
You can also search for mathematical functions by typing their symbol rather than their full name. For example, typing "*" will load the Multiply Module and typing "+" will load the Add Module.

You can also search for mathematical functions by typing their symbol rather than their full name. For example, typing "*" will load the Multiply Module and typing "+" will load the Add Module.

2.2 Connecting Modules and Macros

REAKTOR uses wires to connect the input and output Ports of the Modules and Macros.

► Connecting two Ports is as simple as clicking on one port, dragging a wire to another Port, and then releasing the mouse to create the connection.



However, there are some rules to making connections:

- Wires can only be used to create a connection between an output Port and an input Port.
- An input can only ever accept one connection at a time, but an output can make multiple connections.
- Event outputs can connect to Audio inputs, but not vice versa.

2.2.1 Creating Ports with Wires

Some Modules and all Macros will allow you to make new ports by holding the [Ctrl]/[Cmd] key while dragging a wire.

Modules that can accept additional ports are identified by the ... below their lowest port.

2.3 Event Signals and Audio Signals

The Primary level of REAKTOR uses two main signal types:

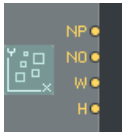
- **Audio**: Audio signals run at the audio rate (typically 44.1kHz or 48kHz). Anything that produces audio (like an oscillator), processes audio (e.g. a filter), or requires a high level of fidelity (e.g. an envelope), should use Audio signals.
- **Event**: Event signals run at a slower rate, known as the Control Rate, the default value of which is 400Hz. Since they run at a slower rate, event signals require less CPU, but have less fidelity. This makes them ideal for signals which we do not directly hear. For exam-

ple, the pitch input for an oscillator uses Event signals, as our ears cannot perceive changes in pitch at rates faster than the Control Rate. Event signals are also useful when creating displays or sequencers.

Certain ports are predefined to send or receive one kind of signal. These ports are color-coded:

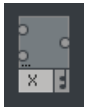
- Light Grey ports are Audio ports.

- Yellow ports are Event ports.



Some Modules can operate at either the Audio Rate or the Control Rate. Most Math Modules can operate at either rate.

- Dark Grey ports show that the Module can operate at either the Audio Rate or the Control Rate.



- The processing rate of these Modules depends on how the ports are connected:
  - If all ports are connected to other Event ports, then the Module will run at the Control Rate.
  - If any of the ports are connected to an Audio port, then the Module will run at the Audio Rate.

There are also Modules with hollow or black Ports. This shows that the Module can accept Event signals or Table References.

Table References will be explored in detail in the tutorial in section ↑8, Drag and Drop Sampler ↑8, Drag and Drop Sampler.

2.4 Mono and Poly Modes

Many Modules have two modes of operation: Monophonic (mono) and Polyphonic (poly).

These modes are a common stumbling block for users who are new to REAKTOR, as their function is not always obvious.

- Monophonic Modules will only process a single signal.
- Polyphonic Modules can process a separate signal for each voice.

The easiest way to think of these concepts is how you would think of synthesizers:

- Monophonic synthesizers can only produce a single note at a time.
- Polyphonic synthesizers can produce several notes at once.

The mono/poly modes in REAKTOR work much in the same way.

Another way to think about it is like this: a monophonic Module is a single item, whereas a polyphonic Module is like a simplified representation of several duplicate Modules, one for each voice.

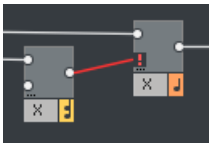
You can see if a Module is set to mono or poly mode by looking at the icon to the bottom right:

- If the icon is of a single note, then the Module is monophonic.

- If the icon is of two notes, one on top of the other, then the Module is polyphonic.

It is possible to connect the output of a monophonic Module to the input of a polyphonic Module, but it is not possible to connect the output of a polyphonic Module directly to monophonic Module.

Connecting a polyphonic Module to a monophonic Module will produce an error and the connection will not work. A connection like this will be colored red and an "!" will mark the input of the monophonic Module.



A Polyphonic Module Connected to a Monophonic Module

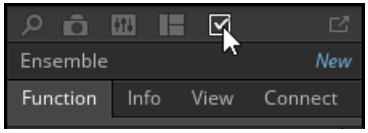
This kind of warning is reserved exclusively for this kind of error.

2.4.1 Setting the Number of Polyphonic Voices

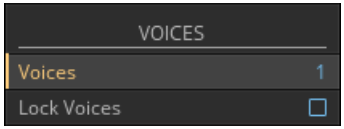
By default, the Ensemble that is created when you select File -> *New Ensemble* has its number of voices set to 1. This effectively makes everything in the Ensemble monophonic, regardless of the individual mono/poly settings of the Modules. This is because a polyphonic module that only uses one voice is functionally the same as a monophonic Module.

For some cases, like creating a polyphonic synthesizer, you will need to change the number of voices.

1. Create a New Ensemble.
2. Select the Ensemble by clicking in an empty space, or on the Ensemble Header.
3. Open the Ensemble's Properties in the Side Pane.

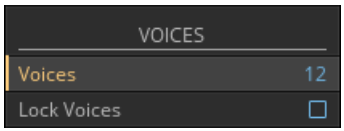


4. In the Function tab is an area called VOICES.
5. Locate the Voices parameter, which should be set to 1.



6. Click and drag on this number to change it.

7. Alternatively, you can also double-click on it and type in a specific number (in the range of 1 to 1024).



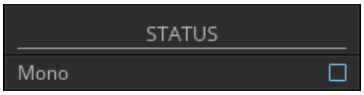
Lock Voices

The Lock Voices option below the Voices parameter disables the editing of the properties related to polyphony. If your Ensemble requires a specific number of voices to function correctly, checking this box will help to avoid accidental editing of the Voices parameter.

2.4.2 Setting the Mono State of a Module

Any Module that can be either monophonic or polyphonic will have an option in its properties for this setting.

The Mono setting is located in the STATUS section of the Function tab.



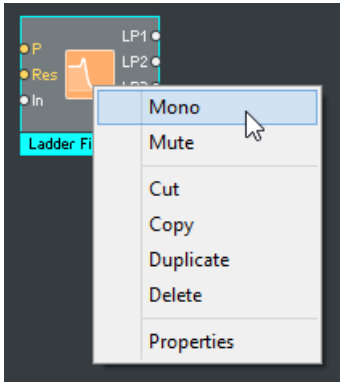
The Mono Option.

- Checking this box will set the Module to be monophonic.
- Unchecking this box will set the Module to be polyphonic.

It is also possible to set the mono/poly state of a Module without entering the properties:

1. Right-click on the Module to open a menu.

2. Select *Mono* to set the Module monophonic mode.



3. Re-opening the menu and unchecking the *Mono* option will set the Module to polyphonic mode.

2.4.3 Polyphony and Macros

Macros also have a *Mono* option in their menu; however the functionality is a little different.

A Macro is never truly monophonic or polyphonic, but checking the *Mono* option will set all Modules inside the Macro to be monophonic.

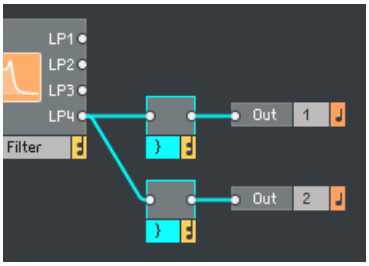
In other words, setting the mono/poly state of a Macro is a shortcut to changing the settings of all the Modules within the Macro, rather than being a state of the Macro itself.

However, it is then possible to enter the Macro and set a single Module to be polyphonic, without altering the mono/poly state of the Macro.

2.4.4 Voice Combiners

Voice Combiners are Modules that can be used to merge a polyphonic signal into a monophonic signal.

The outputs of REAKTOR are monophonic, since they can only output a single audio signal. Therefore, Voice Combiners are important to add before the final output of any Ensemble that uses polyphonic signals.



Audio Voice Combiners.

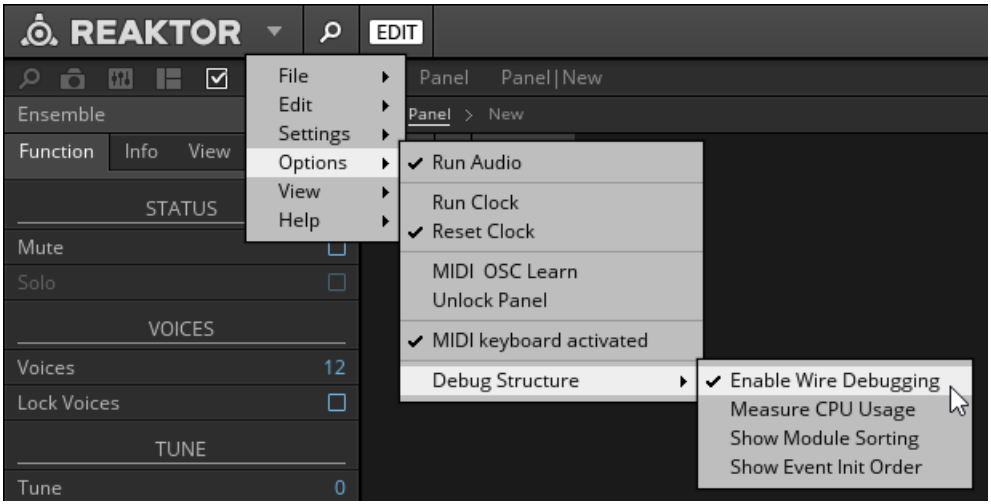
Voice Combiner Modules merge polyphonic signals into a monophonic signal.

In some cases it may be necessary to use voice combiners before the Ensemble output ports, for example, if you are using effects like Delay or Reverb.

2.5 Debugging

Debugging is always an important part of development, and REAKTOR is no exception. As such, REAKTOR has a handful of built-in tools for debugging.

► The debugging tools are located in the Options menu, under *Debug Structure*.

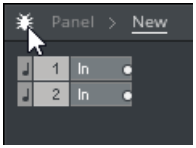


2.5.1 Wire Debugging

When debugging in REAKTOR, it will be common for you to want to see the value of a signal as it passes from Module to Module, and Wire Debugging is quick and easy way of doing this.

There are two ways of activating Wire Debugging:

► In the structure view, click on the bug icon to the left of the breadcrumb navigation.



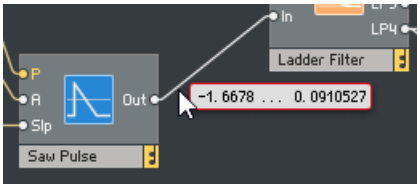
Or…

1. Enter the Options Menu.
2. Select *Debug Structure* to display the debugging tools.
3. Click on the *Enable Wire Debugging* option.

With Wire Debugging active, you can hover the mouse over any wire in the structure and see its current value.

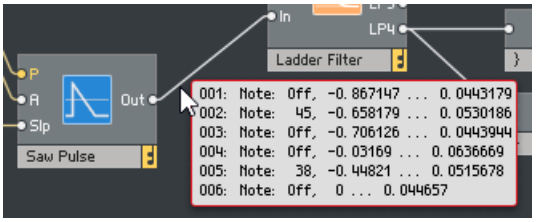
Wire Debugging.

If the value is changing faster than the display refresh rate (i.e. if it is an audio signal), then you will be shown the minimum and maximum values that pass through the wire, like this:



Debugging an Audio Signal.

If the wire is polyphonic, then you will be given a list of the values on each voice:



Debugging a Polyphonic Signal.

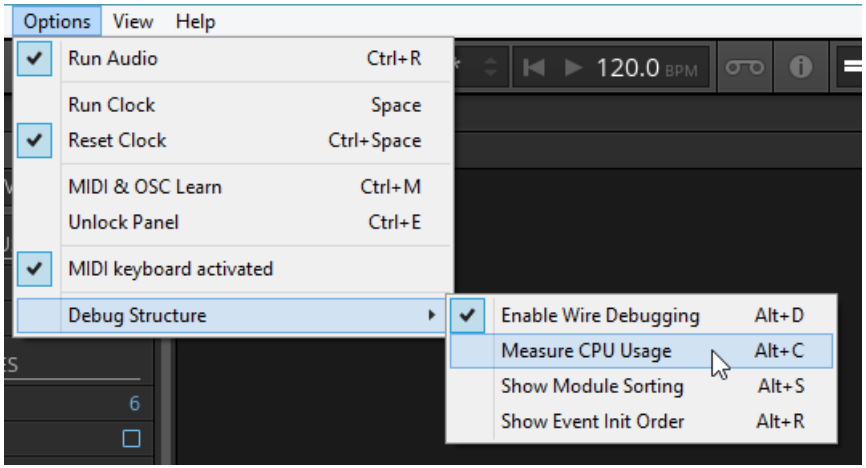
2.5.2 CPU Usage

In REAKTOR, it is possible to measure the CPU usage of the individual elements in your structure. This can help you locate any areas that may be causing CPU spikes, or that could be optimized.

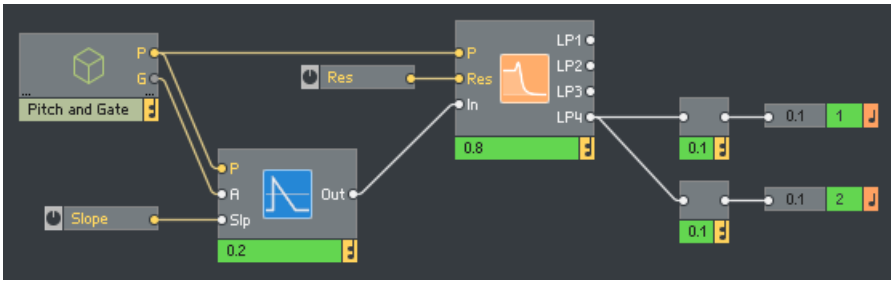
To toggle the Measure CPU Usage option on or off:

1. Enter the Options Menu.
2. Select *Debug Structure* to display the debugging tools.

3. Click on the *Measure CPU Usage* option.



With this option activated, you will be able to see numbers in place of the names of some of the Modules and Macros.



Measuring CPU Usage.

These numbers show the CPU usage of the Modules and Macros.

If a Module or Macro does not have a number, this means that its CPU usage is less than 0.1%

All audio outputs will be muted while the *Measure CPU Usage* option is active. They will return to normal when the option is deactivated.

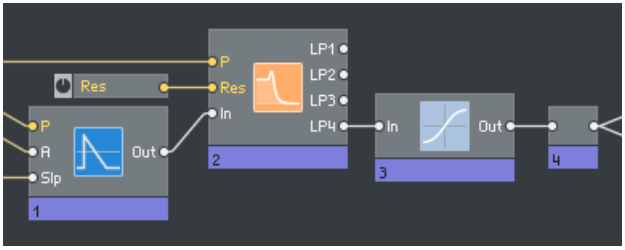
2.5.3 Module Sorting

In Primary Level when performing signal processing with Audio signals, the order of operations becomes important. This is particularly crucial if you have feedback in your Structure.

You can use the Module Sorting Debug Tool to see in which order the operations are performed on your Audio signal.

To toggle the Measure CPU Usage option on or off:

1. Enter the Options Menu.
2. Select *Debug Structure* to display the debugging tools.
3. Click on the *Show Module Sorting* option.



The Module Sorting of Audio Modules.

With this Debug Tool active, a purple display for each Module shows its place in the sequence in which the operations are performed on the Audio signal.

Note that only Audio Modules have Module Sorting numbers, Event Modules have no Module sorting number.

2.5.4 Event Initialization Order

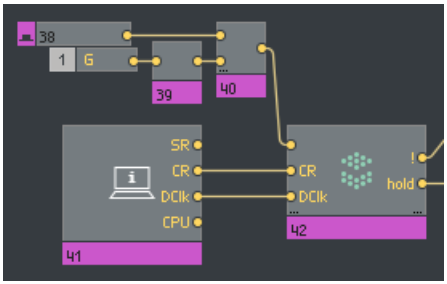
When you load an Ensemble, it goes through a process called Initialization. During this procedure, the Modules are initialized in a certain order and each only once.

At the moment of its Initialization, the Module looks at all values that have arrived at its input ports and then sends the appropriate values and/or Events from its output port(s). In more complex Structures, Event Initialization becomes a crucial part of your Structure. Some Modules require Events to arrive in a certain order at their input ports before functioning in the intended way.

To see the order in which Modules that process or send Events are initialized, use the Event Initialization Order Debug Tool.

To toggle the Event Initialization Order option on or off:

1. Enter the Options Menu.
2. Select *Debug Structure* to display the debugging tools.
3. Click on the *Show Event Init Order* option.



Modules displaying their Event Init Order.

With this Debug Tool active, a magenta display for each Module shows its place in the sequence in which Event functions are initialized.

3 Subtractive Synthesizer

3.1 Overview

REAKTOR is a visual programming environment, allowing you to build musical tools specific to your own needs. This tutorial will introduce the basic skills needed to build your own Ensembles from the components in the Macro Library using a subtractive synthesizer as an example.

3.1.1 Previous Knowledge

This tutorial assumes you have read all of the documentation up to this point and that you are familiar with REAKTOR's interface and basic functionality, including how to navigate in the Structure View.

The tutorial also assumes that you have set up your Audio and MIDI devices and are able to play REAKTOR Ensembles from the Factory Library.

3.1.2 The Theory

In its most basic form, a subtractive synthesizer is an oscillator routed through a filter. A harmonically rich waveform, like a saw wave, has its harmonics subtracted by the filter, hence the term subtractive synthesis.

However, even early synthesizer designers knew that a single oscillator and a single filter were not enough on their own and that modulation was required to change the sound over time. Amplifiers, LFOs (Low Frequency Oscillators) and Envelopes were added to give sound designers and musicians more flexibility.

In this tutorial, you will use the Macro Library to build a subtractive synthesizer, learning the basics of building in REAKTOR along the way.

3.2 Tutorial

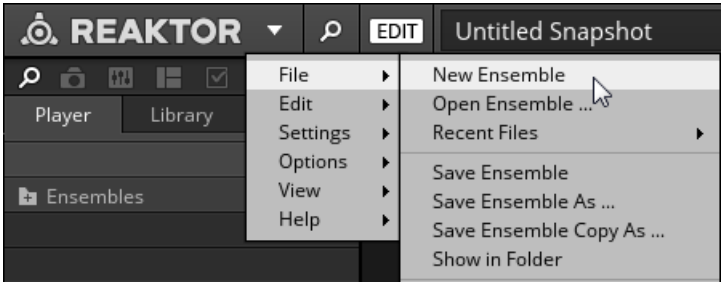
3.2.1 Setting Up

Before you begin building, you need to have REAKTOR running in Edit Mode.

► If Edit Mode is not already enabled, activate it by clicking on the EDIT button in the toolbar.

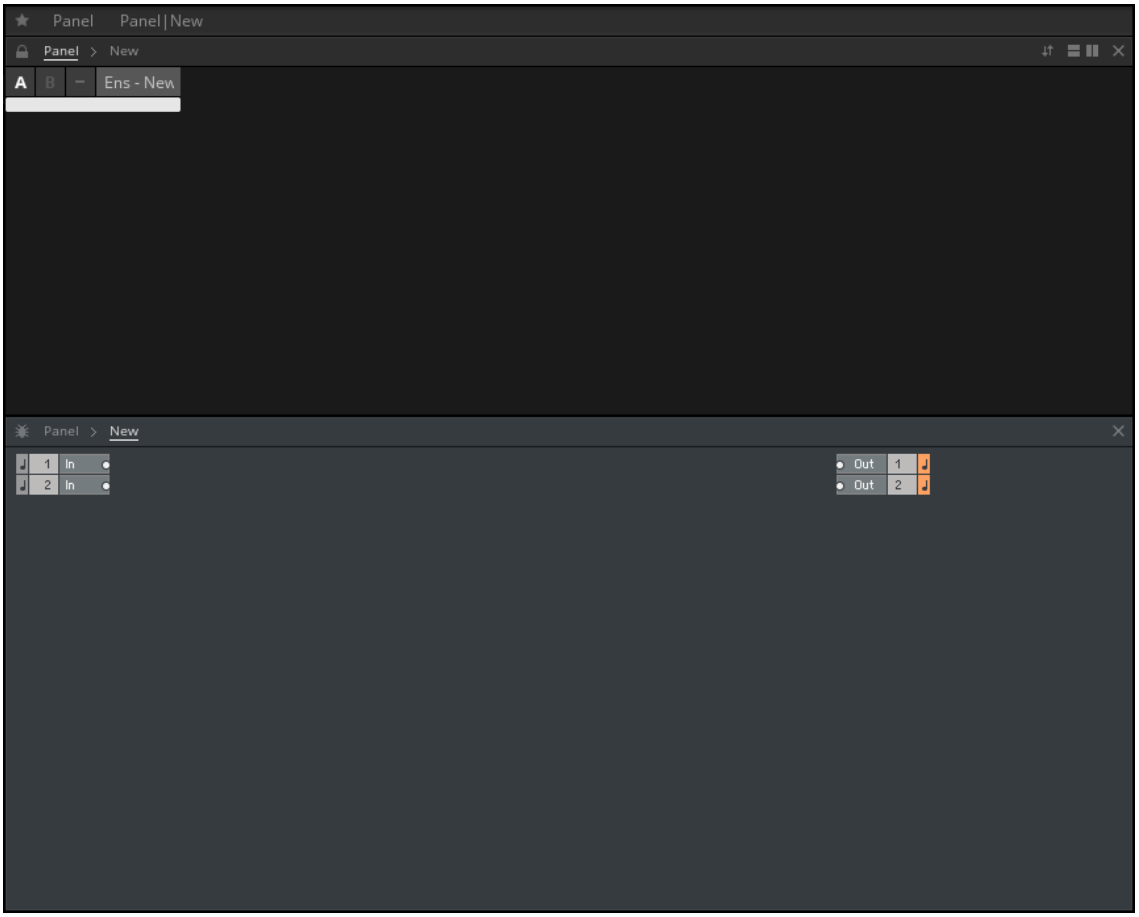
Next, you need to create a new empty Ensemble.

► Create a new empty Ensemble by entering the File menu and selecting *New Ensemble*.



► Alternatively, you can use the shortcut [Ctrl] + [N] ([Cmd] + [N] on OSX)

When you create a new Ensemble in Edit Mode, it loads with the screen already split horizontally, with the Panel View on top and the Structure View below.



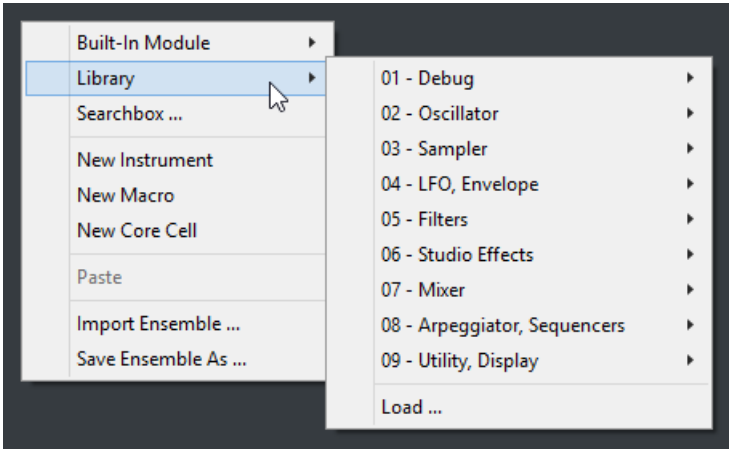
The New Ensemble, as it appears when first loaded.

This kind of split-screen setup is useful when building something like a synthesizer, as you can view the Panel and the Structure at the same time, allowing you to edit them both without navigating back and forth.

3.2.2 Building a MIDI Controlled Oscillator

For this synthesizer, there will be one oscillator, the amplitude of which will be controlled with an AHDSR envelope. So the first step towards building this will be to create these elements.

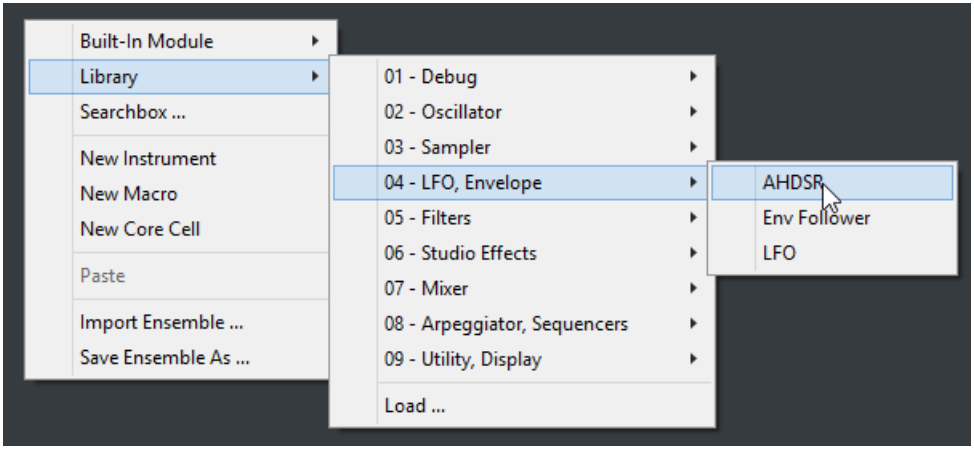
1. In the Structure View, right-click on an empty space to open the Context Menu.
2. Navigate to the Library entry to view the Macro Library.



3. In the *02 - Oscillator* sub-menu, select *OSC Sync*.

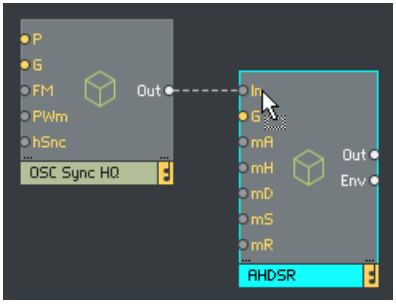
→ The *OSC Sync* Macro will be inserted into the Structure and will also appear on the Ensemble Panel.

► To create an AHDSR Envelope, repeat the above process, only this time select the *AHDSR* entry in the *04 - LFO, Envelope* sub-menu.



With the two elements created, the next step is to wire them together.

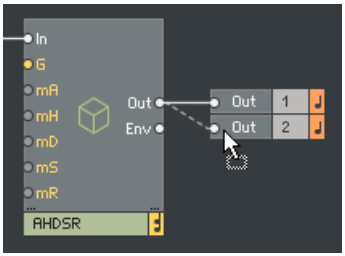
► Click and drag with the mouse to create a wire from the Out Port of the *OSC Sync* Macro to the In Port of the *AHDSR* Macro.



The AHDSR Macro has its own internal amplifier, so connecting the oscillator to the envelope in this way means that the envelope will control the output volume of the oscillator.

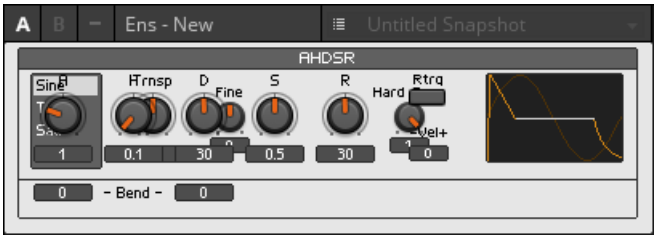
The output of the *AHDSR* Macro will be the final output of the synthesizer for now, so the next step will be to connect it to the Ensemble outputs. However, there are two Ensemble Out Ports, one for each of the stereo channels (left and right). To produce a sound from both speakers, the output of the AHDSR should be connected to both ports. Fortunately, REAKTOR allows you to make multiple connections from one output port to multiple input ports.

► Create connections from the Out Port of the *AHDSR* Macro and to the input ports of both of the Ensemble Out Ports.



With the two Macros created and connected, it is a good time to arrange the controls on the Ensemble Panel.

Looking at the Panel view, you can see that the *AHDSR* Macro was created on top of the *OSC Sync* Macro.

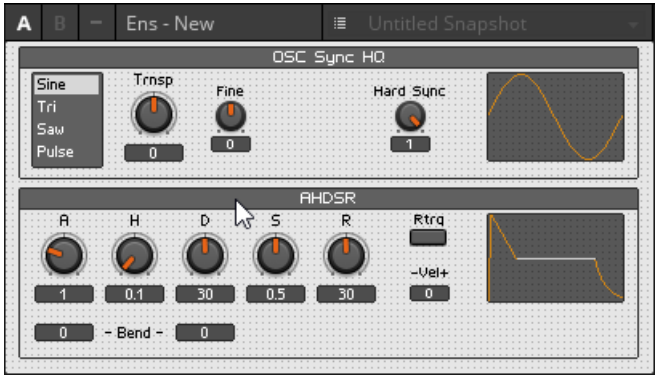


The Panel View

In order to make the Ensemble usable, you should re-arrange the Macros so that they are not obscuring each other in this way.

1. Unlock the Ensemble Panel by clicking on the padlock icon.

2. Click on the title of the *AHDSR* Macro to select the whole Macro and all of the controls contained within it.
3. Move the Macro so that it is no longer overlapping with the OSC Sync Macro.



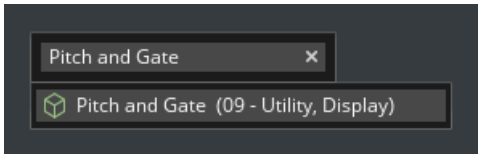
4. When you are finished, lock the Panel by clicking on the padlock again.

Whichever Macro was last selected in the Structure View will be the Macro that is on the top layer on the Panel. This means that the OSC Sync Macro could be on top of the AHDSR Macro, but the principal of moving the Macros remains the same.

With the Macros connected and the Panel arranged, the next step is to make the instrument playable. To do this you will need to connect MIDI Pitch and Gate messages to the Ensemble Structure. The MIDI Pitch will control the pitch of the oscillator and the Gate will be used to trigger the envelope.

The Macro Library includes a *Pitch and Gate* Macro that you can use for this. To speed up the process, you can use the Searchbox to load this Macro.

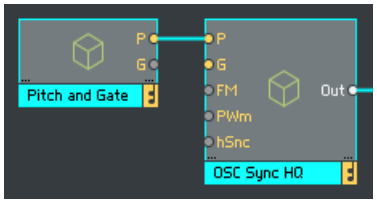
1. In the Structure View, press [Enter] to open the Searchbox.
2. Type "Pitch and Gate" into the Searchbox.



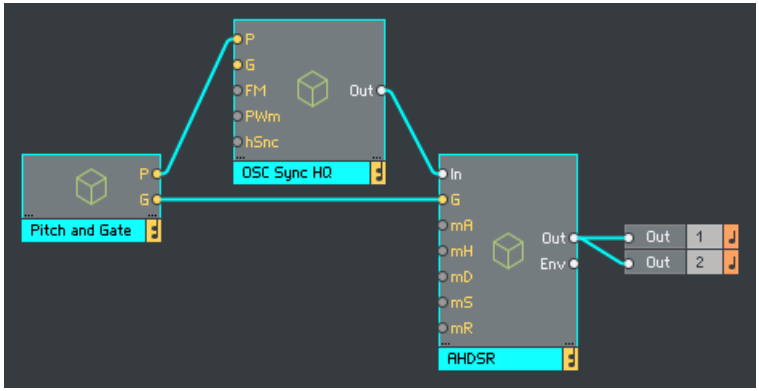
3. Press [Enter] to insert the *Pitch and Gate* Macro into the Structure.

REAKTOR's Modules and Macro Library use a number of standard abbreviations for Port names. A port named P is always for pitch, and a G Port is always for gate signals.

1. Since the MIDI Pitch will control the pitch of the oscillator, connecting the two Macros to accomplish this is a case of connecting the two P ports to each other.



2. The same logic applies to the G Ports of the *Pitch and Gate* Macro and the *AHDSR* Macro.

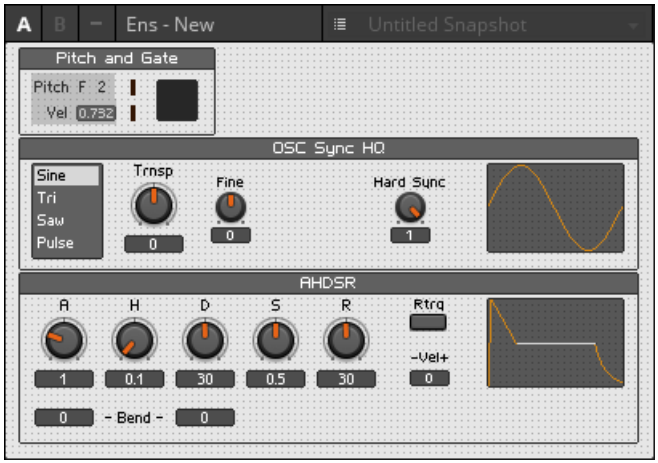


→ With the Pitch and Gate connections made, play the MIDI keyboard to hear a sound.

As a final step, you will need to re-arrange the Ensemble Panel to accommodate the Pitch and Gate Macro.

1. Unlock the Ensemble Panel.

2. Move the *Pitch and Gate* Macro so that it is not overlapping with the other Macros.



3. When you are finished, lock the Panel.

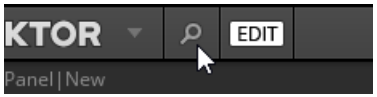
You have just created a basic synthesizer that you can play with a MIDI Keyboard.

3.2.3 Adding the Filter

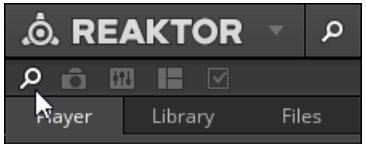
The synthesizer you have created is only the first part of a subtractive synthesizer: the oscillator. The next step is to add the second part: a filter.

For this part of the tutorial, you will use the Library tab in the Side Pane to load the filter Macro.

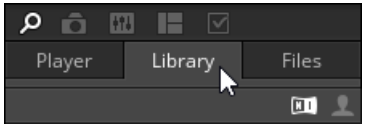
1. If the Side Pane is not already open, open it now by clicking on the magnifying glass icon in the toolbar.



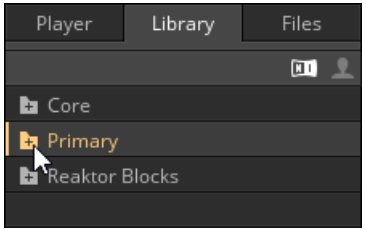
2. With the Side Pane open, click on the magnifying glass tab to open the browser.



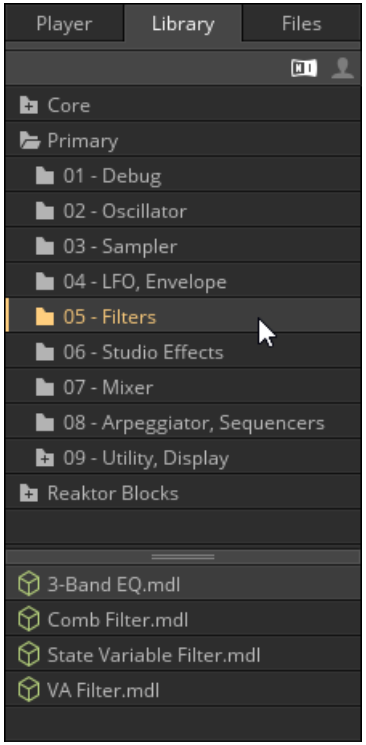
3. Click on the Library tab to display the Builder Library content.



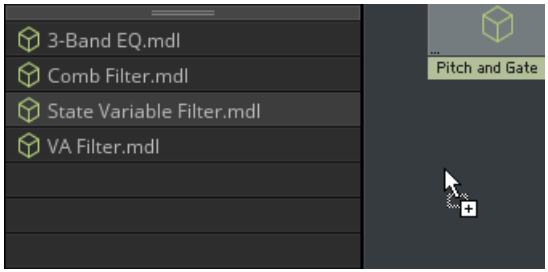
4. Open the Primary folder by clicking on the folder icon.



5. Click on the 05 - Filters folder to display its contents in the lower half of the Side Pane.



6. Click and drag the State Variable Filter into the Ensemble Structure.

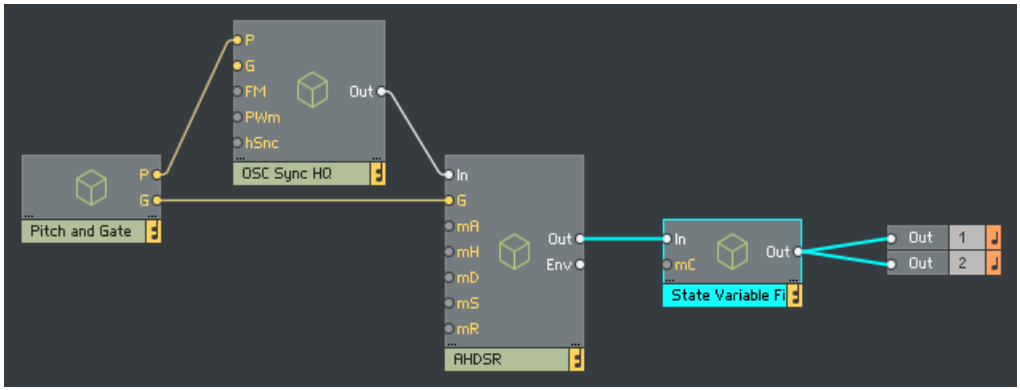


7. Release the mouse to insert the *State Variable Filter* Macro into the Structure.

→ The *State Variable Filter* Macro will be added wherever you dropped it.

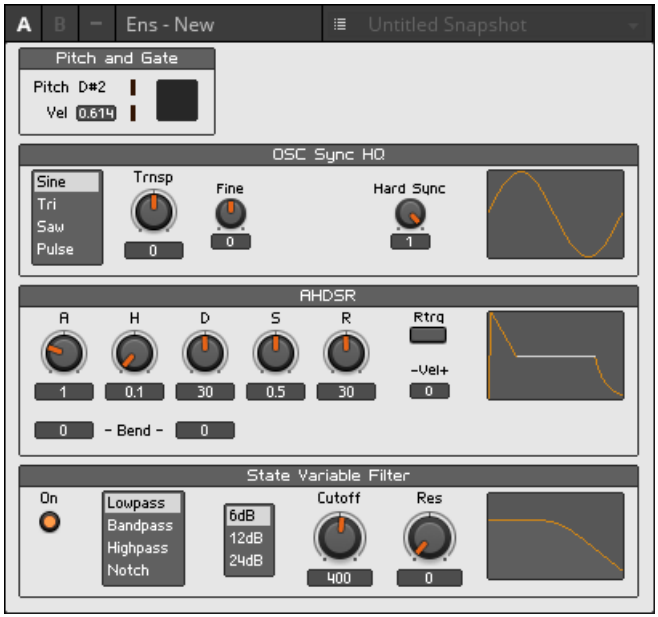
The *State Variable Filter* should be connected to the end of the Ensemble Structure, in order for it to process the audio coming from the oscillator.

1. Remove the wires connecting the *AHDSR* Macro to the Ensemble Outputs by selecting them and pressing the [Delete] key.
2. Connect the *State Variable Filter* as illustrated below.



3. Return to the Panel View and Unlock the Panel.

4. Move the *State Variable Filter* so that it is not obscuring any other Macros.



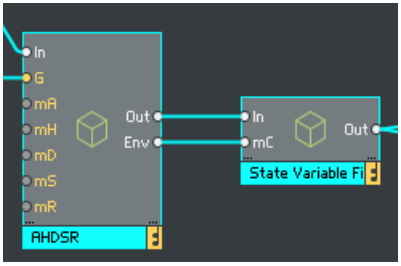
5. Lock the Panel.

→ Congratulations! You have built a playable subtractive synthesizer in REAKTOR.

3.2.4 Modulating the Filter

A static filter, like the one that is currently in your synthesizer, does not sound very interesting. However, with REAKTOR you can always build more! By connecting the envelope to the filter, you will be able to modulate the filter to alter the timbre of the synthesizer over time.

► The most simple way of doing this is to connect the Env output of the *AHDSR* Macro to the mC (modulate Cutoff) input of the *State Variable Filter*.



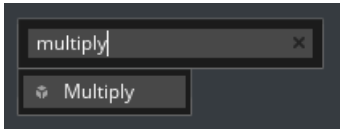
However, this doesn't allow you to control the modulation amount.

To add control for the modulation amount, you will need to start using **Modules**, which are the basic building blocks in REAKTOR.

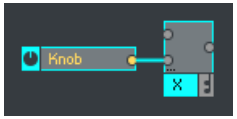
The basic skills required to use Modules are not much different from those you have developed already.

To control the modulation amount, there needs to be a multiplier between the envelope and the filter, and a knob to control the multiplication amount.

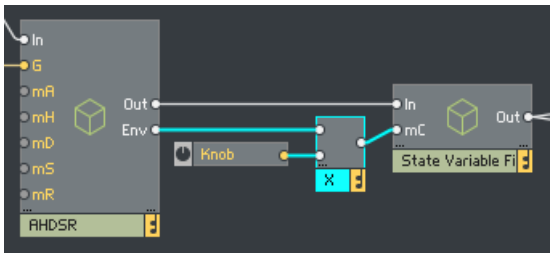
1. In the Structure View, open the Searchbox by pressing [Enter]
2. Type "Multiply" and press [Enter] to create a *Multiply* Module.



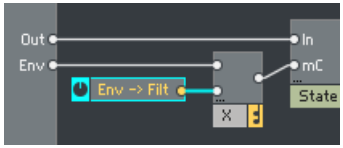
3. Open the Searchbox again and type "Knob" to create a *Knob* Module.
4. Connect the *Knob* to the *Multiply* Module.



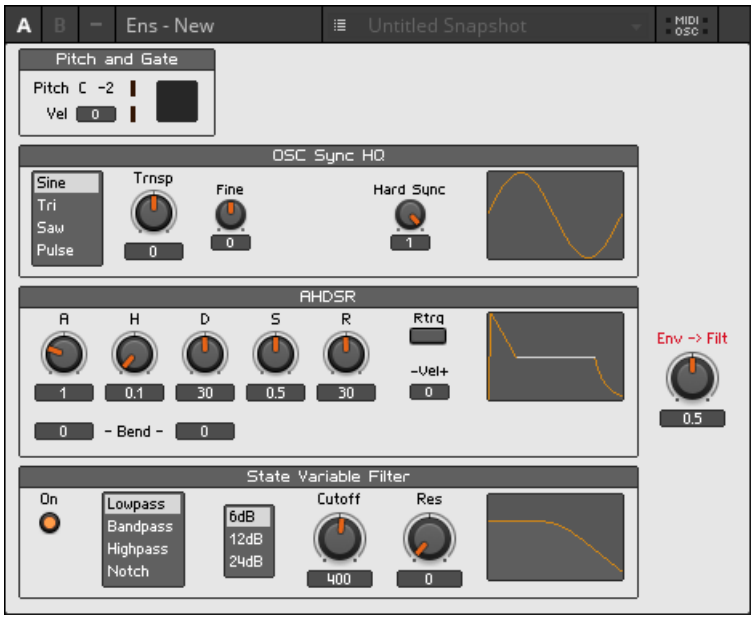
5. Connect the *Multiply* Module to the Ensemble Structure between the *AHDSR* Macro and the *State Variable Filter* Macro, as illustrated below.



6. Double-click on the *Knob* to re-name it.
7. Re-name the Knob "Env -> Filt"



8. In the Panel View, move the new Env -> Filt knob so that it is not obscuring any other controls.



→ Turn the Env -> Filt knob to control the intensity of the modulation from the envelope to the filter.

3.2.5 Adding Polyphony

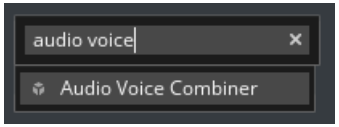
The synthesizer you have created is monophonic; it can only play one note at a time, making it impossible to play chords.

By increasing the number of Voices in the Ensemble, you can make the synthesizer polyphonic, i.e. capable of producing several notes at once. The number of Voices directly affects the number of notes you can play at once, so an Ensemble with 3-Voice polyphony can play up to three notes at any one time.

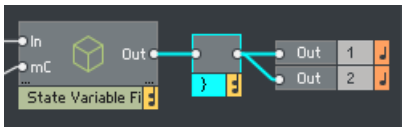
Before you change the number of voices, you will need to create an *Audio Voice Combiner*.

> The importance of Voice Combiners, as well as their use in this scenario, is described in section ↑2.4.4, Voice Combiners ↑2.4.4, Voice Combiners.

1. In the Structure View, open the Searchbox by pressing [Enter]
2. Type "Audio Voice" into the Searchbox and the *Audio Voice Combiner* Module will appear in the results.



3. Press [Enter] to insert the *Audio Voice Combiner* into the Structure. It will appear as a Module labeled with the } symbol.
4. Connect the *Audio Voice Combiner* between the *State Variable Filter* Macro and the Ensemble *Out* Ports.

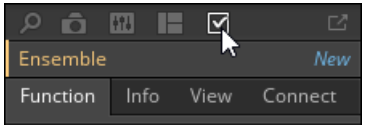


→ With the *Audio Voice Combiner* connected, changing the number of voices will not cause any errors.

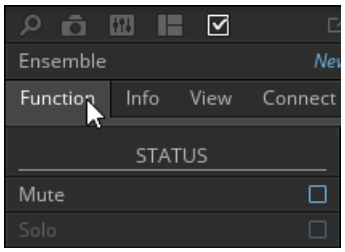
The next step to add polyphony to the synthesizer is to increase the number of Voices in the Ensemble. This can be done in the Ensemble Properties.

1. Select the Ensemble by either clicking on an empty space in the Structure View, or by clicking on the Ensemble Header.

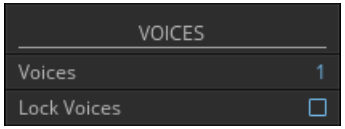
2. Click on the Properties tab (the checkbox icon) in the Side Pane.



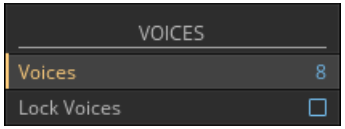
3. The Properties should open on the Function tab, but if it does not, click on the Function tab.



4. In the Function tab, locate the section called VOICES.



5. The Voices parameter in this section is currently set to 1, making the Ensemble monophonic.
6. Change the Voices parameter in this section either by clicking on the value and dragging to a new value, or by double-clicking on the value and entering a new value with the keyboard.
7. Set the Voices parameter to 8.



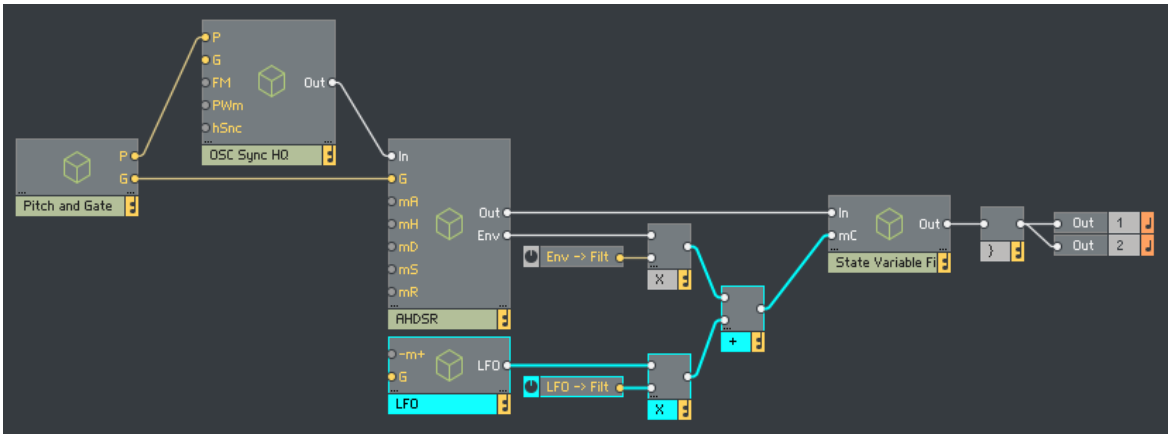
→ You can now play up 8 notes at once.

Playing several notes at once may cause REAKTOR's output to overload, so you should reduce the master output level in the Toolbar to avoid this.

Congratulations! You have created a polyphonic subtractive synthesizer in REAKTOR.

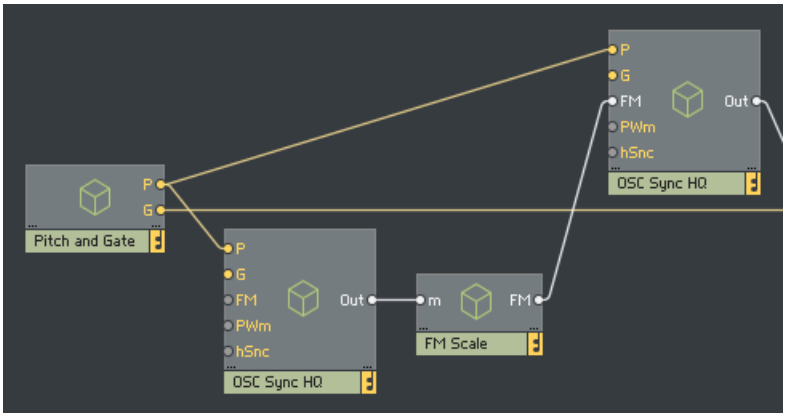
3.3 Going Further

By connecting more Macros to your synthesizer, you will be able to expand it in any number of ways. For example, you could connect an LFO to also modulate the filter cutoff.



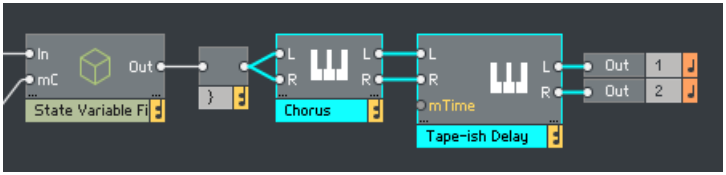
Adding LFO Modulation

You could add a second oscillator for Frequency Modulation (FM)



An FM Structure

You can even add some effects after the Audio Voice Combiner.



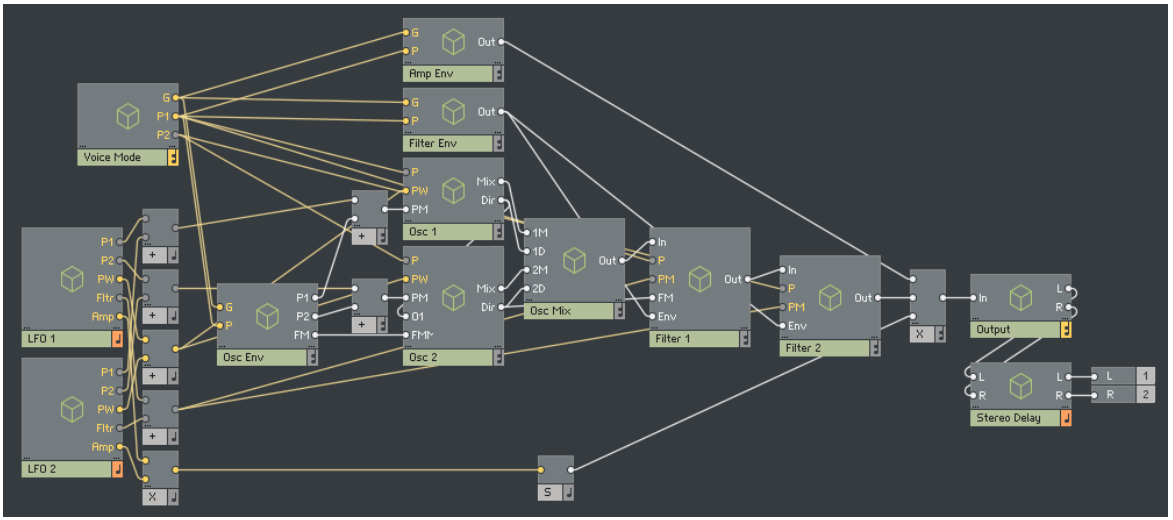
Adding Chorus and Delay Effects

By experimenting with different Macros and signal routing, you can create any number of different sound tools.

3.3.1 Related Factory Content

2-Osc is a relatively simple subtractive synthesizer Ensemble that can be found in the Factory Library in the Synthesizers folder.

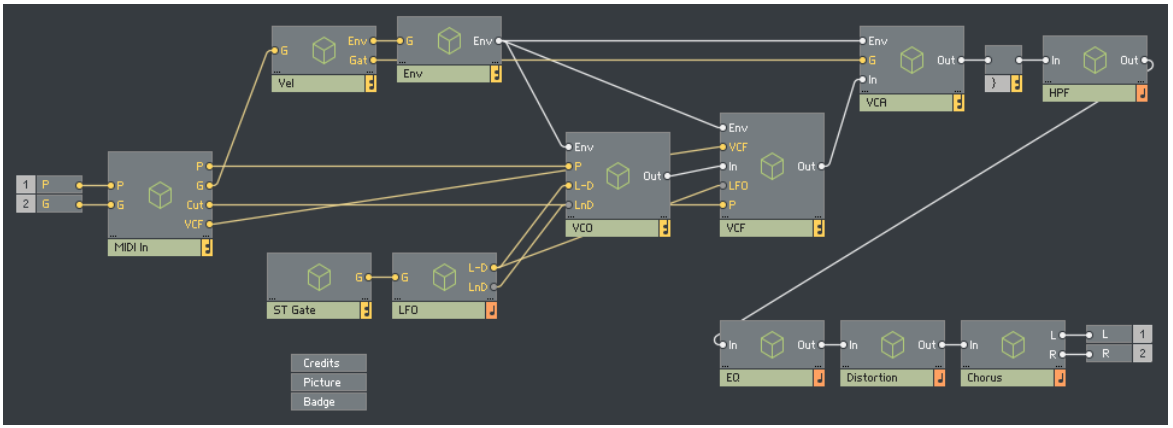
Its structure is organized into Macros like the synthesizer you have built, and these Macros are well labeled, making it easy to navigate.



The 2-Osc Structure

This not only makes it a good example for learning REAKTOR, but also makes it an easy Ensemble to customize.

Junatik is another simple subtractive synthesizer with a Structure that is also easy to read.



The Junatik Structure

4 Echo Effect Macro

4.1 Overview

Macros are a very useful way of containing sections of a larger Instrument or Ensemble project. This tutorial will lead you through the process of building your own Macros with Built-In Modules, using an echo effect as the example.

4.1.1 Previous Knowledge

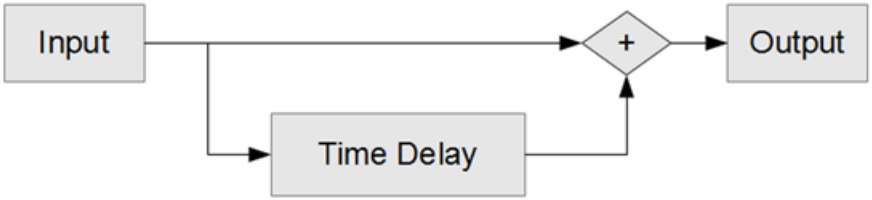
This tutorial assumes you have read the previous chapters of this document, and so are familiar with the following concepts:

- How to navigate REAKTOR's Structure View.
- The difference between Audio and Event signals.
- Polyphony in REAKTOR.
- How to create wires between ports.

4.1.2 The Theory

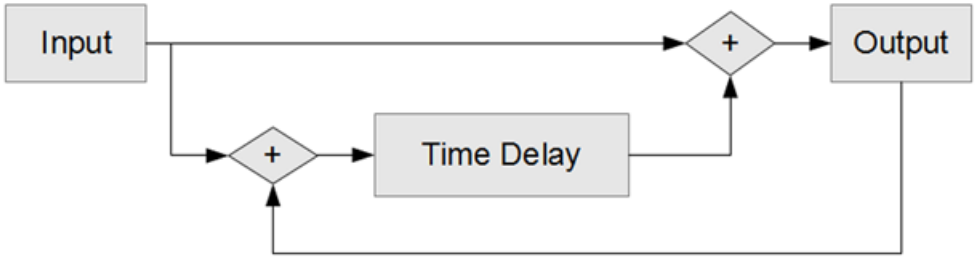
In its most simple form, an echo effect takes an audio signal and adds a time delay to it; this delayed signal is then mixed with the original signal to produce something resembling an echo.

A signal flow diagram of this effect is as follows:



The Signal Flow Diagram of a Basic Echo Effect

It is then possible to add repeating, decaying echoes by feeding the delayed signal back into the delay input, like this:



The Signal Flow Diagram of an Echo Effect with Feedback

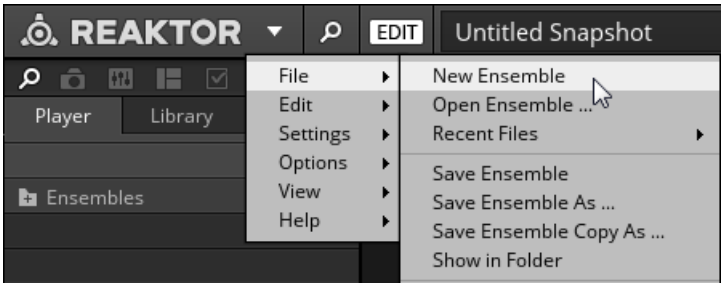
This tutorial will guide you through the process of building the simple echo effect using a delay module, before modifying it to include feedback and tempo-sync options.

4.2 Tutorial

4.2.1 Setting Up

For this tutorial you will need to start with an empty Ensemble:

► Create a new empty Ensemble by entering the File menu and selecting *New Ensemble*.

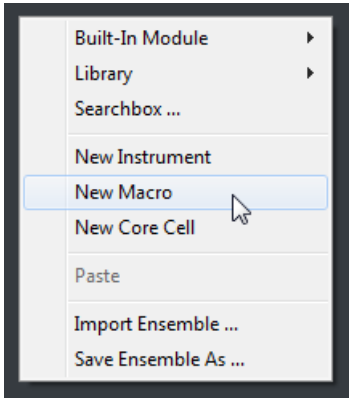


► Alternatively, you can use the shortcut [Ctrl] + [N] ([Cmd] + [N] on OSX)

Next you can create an empty Macro inside which you will build your echo effect.

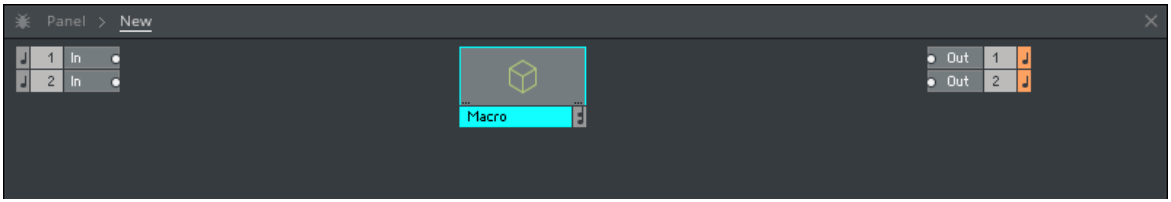
To create an empty macro:

1. Right-click in the Ensemble Structure to open the Context Menu.
2. Click on *New Macro*.



→ You have now created a new empty Macro.

Your structure should now look something like this:

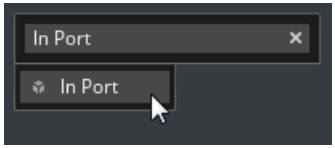


How your structure should currently look

In order for the Macro to receive and send audio signals, you will need to create input and output ports.

You can create ports in two ways:

1. Enter the Macro by double-clicking on it.
2. Press [Enter] to open the Searchbox.
3. Type "In Port" into the Searchbox.



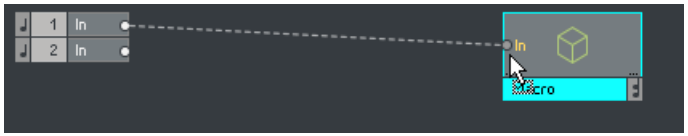
4. Press [Enter] to create an *In Port* Module. This Module also creates an input port on the Macro container.
5. Repeat steps 2, 3, and 4, only this time enter "Out Port" into the Searchbox.

→ The Macro will now have one input port and one output port.

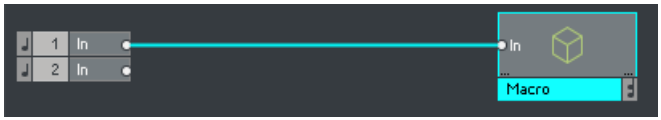
Alternatively, you can create ports without even entering the Macro:

1. If you are not already there, navigate to the top level of the Ensemble structure.
2. Click on the output of the top *In* Module.
3. While holding down both the mouse button and the [Ctrl] key on your keyboard (or [Cmd] key for OSX), drag a cable to the left side of the Macro.

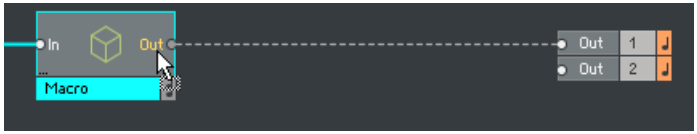
4. When you reach the edge of the Macro, an In port should appear.



5. When this happens, release the mouse button and then release the [Ctrl]/[Cmd] key. This will create a wire from the output of the In Module to a new In port on the Macro.



6. Repeat this process, only this time drag the wire from the input of the top *Out* Module to the right edge of the Macro.



→ The Macro will now have an input port, and output port, and wires connecting these ports to the Ensemble's first audio inputs and outputs.

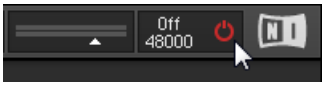
► With the Macro's ports created, Wire them to the Ensemble inputs and outputs like this:

To keep things organized, re-name the Macro:

1. Double-click on the Macro's name.
2. Type in a logical name, like "Echo FX"
3. Press [Enter] or click away from the Macro to set these changes.

While working on an Ensemble's structure, it is often a good idea to deactivate the audio engine while you work. If your computer is connected to speakers and an active microphone, creating an effect could cause feedback between them.

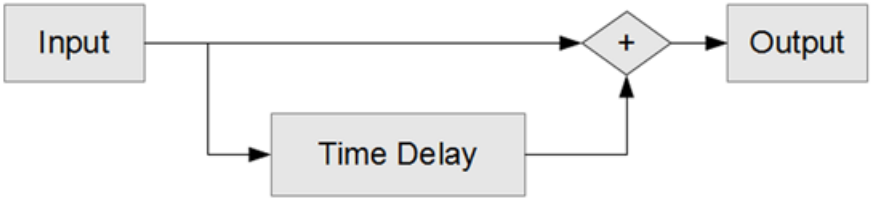
► Deactivate REAKTOR's audio engine by clicking on the power button to the top right.



4.2.2 Building the Simple Echo Effect

► If you aren't already in the Echo FX Macro's structure, enter it now by double-clicking on the Macro.

Since REAKTOR is a visual programming environment, you can use the signal flow diagram from earlier as a template for what you will need to build in the structure:



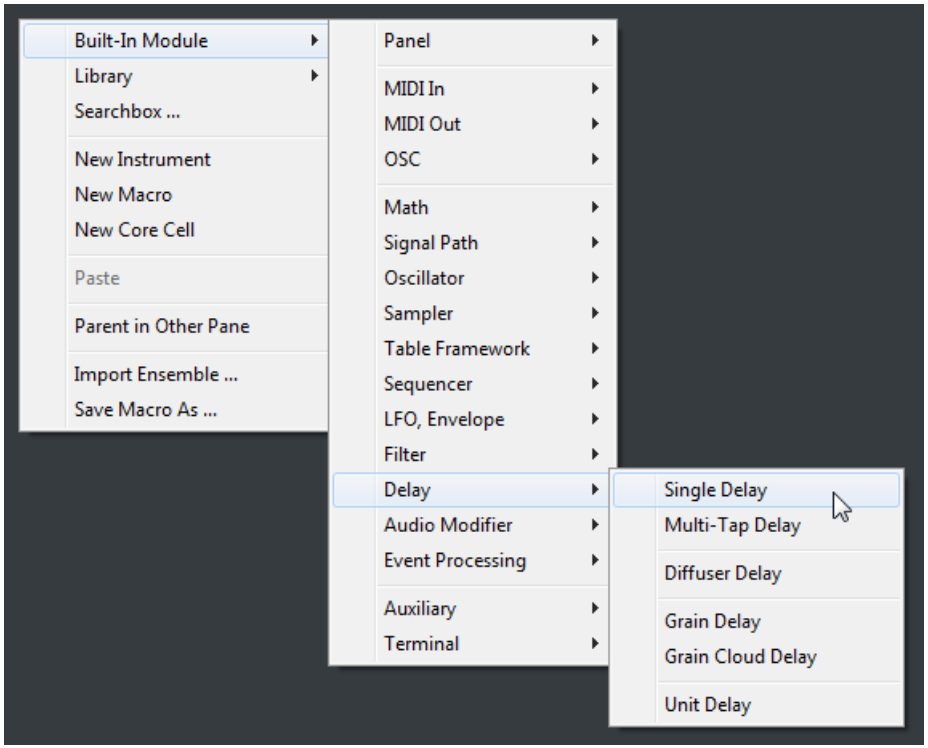
The Echo Effect Signal Flow Diagram

To build this structure you will need three Modules: *Single Delay*, *Multiply*, and *Add*.

The *Multiply* Module will be used to set the volume level of the delayed signal and the *Add* Module will be used to add the input signal to the delayed signal.

1. Right-click to open the Context Menu.
2. Navigate to *Built-In Module* and then select *Delay*.

3. Select and click on *Single Delay* to create a *Single Delay* Module.



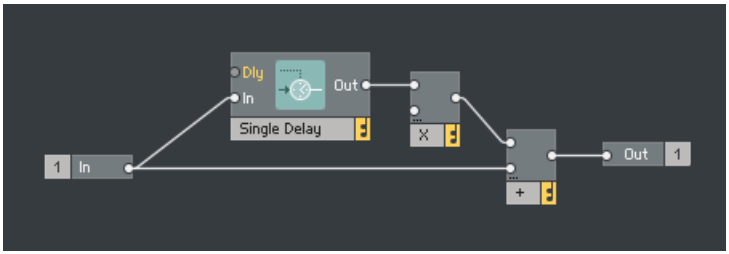
4. Open the menu again and navigate to *Built-in Module* followed by *Math*.
5. Click on *Add*.
6. Return to the *Math* section of the menu and click on *Multiply*.

→ You now have all the basic Modules you need to build the echo.

Alternatively you can open the Searchbox by pressing the [Enter] key, and then type "single delay" to locate the module directly. You can also search for mathematic modules by typing the corresponding symbol into the Searchbox, so you can type * to find the Multiply Module, and type + to find the Add Module.

If you haven't done so already, now is a good time to deactivate REAKTOR's audio engine, or mute the main output.

► Wire the Modules together like this:

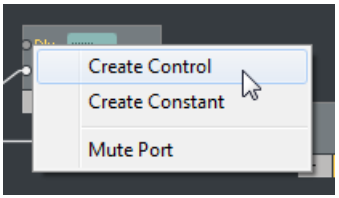


Adding Controls

You will need two knobs for this effect, one to control the delay time, and one for the delay output level.

You can create knobs in the same way you created the other Modules, either through the menu or the Searchbox, but there is also a shortcut:

► right-click on the Dly port of the Single Delay and select *Create Control*.



→ A *Knob* Module has been created with a logical name and useful minimum and maximum values.

Although the properties of this newly created *Knob* Module are good for most cases, you may want to change the resolution of the knob to get more precise delay times.

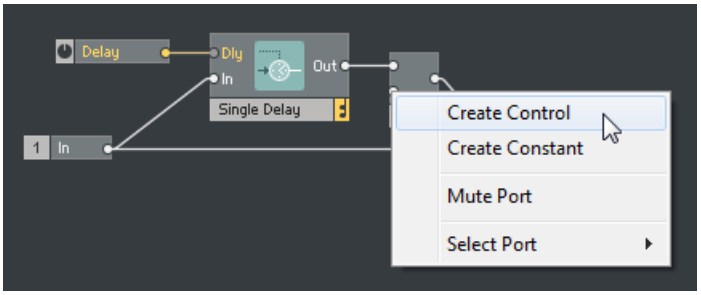
1. Select the Delay *Knob* Module.
2. Navigate to the Properties window.
3. Select the Function tab.
4. In the RANGE section, check the Step Size parameter.

5. Double-click on this value to edit it.

6. Set it to 1.

→ This has decreased the step size of the control from 10ms to 1ms.

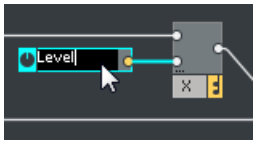
► Use the same technique with the empty port of the *Multiply* Module.



Since the *Multiply* module ports can have many uses with many possible value ranges, the *Knob* Module we have been given here is less suited to our purposes, but we can change that.

First, re-name it to "Level":

1. Double -click on the knob and type "Level" as the new name.



2. Press [Enter] or click away from the knob to accept these changes.

Fortunately, the Max, Min, and Step Size parameters for this *Knob* Module are all set to values that work for this tutorial, so they do not need to be changed.

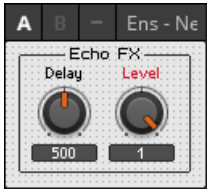
Finally, you'll need to arrange these controls on the Instrument Panel. Return to the Panel view and have a look at your work...

The Current Look of the Ensemble Panel

All of the knobs are stacked on top of each other, which makes them rather difficult to use. You need to move the knobs to make them user-friendly:

1. Unlock the Panel by clicking on the padlock button.

2. Move the knobs so they are no longer overlapping.



3. When you have finished positioning the knobs where you want them, lock the panel by clicking on the padlock button again.

Testing the Effect

The easiest way to test an effect is to use REAKTOR's File Player.

> If you do not know how to use the File Player, you can read the relevant chapter in the REAKTOR 6 Diving Deeper document to learn more about it.

1. Load an audio file (wav or aiff format) into the File Player (ideally something short and looped).
2. Select the loop option.
3. Press the play button.
4. If you deactivated the audio engine, or muted the main output, you will now need to reactivate the audio engine and/or unmute the main output.

→ You should be able to hear the audio file you loaded with a delayed signal added on top.

As the audio plays, you will be able to edit the parameters of the echo effect by using the two knobs you created:

- Move the Delay knob to control the delay time (the time between the input signal and the delayed signal).
- Move the Level knob to control the output level of the delayed signal.

4.2.3 Adding Feedback

Your echo effect is currently a simple slap-back style single delay, which is good for some uses, but by adding a feedback loop you can create repeating echoes.

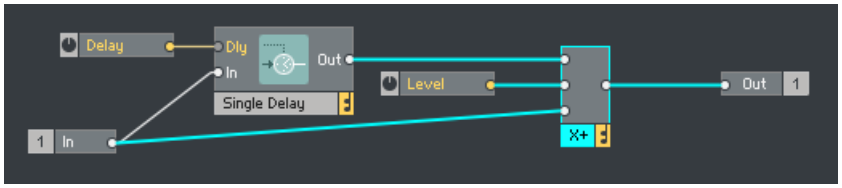
Rather than introducing more *Multiply* and *Add* Modules, you can save space and keep a tidier structure by using the *Mult/Add, a*b+c* Module.

The *Mult/Add* Module outputs the result of the formula (a*b)+c, with the three input ports corresponding to the variables a, b, and c. The top port is the a value, and the bottom port is the c value.

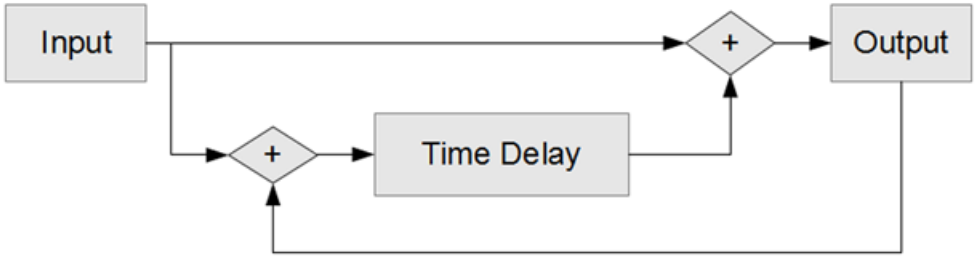
Remember that when you are editing the structure, it is advised that you deactivate the audio engine, or mute the main output.

You can easily replace the *Multiply* and *Add* Modules in your current echo effect with a single *Mult/Add* Module:

1. Enter the structure view of the Echo FX Macro.
2. Select the *Multiply* and *Add* Modules.
3. Press the [Delete] key to remove them.
4. Create a *Mult/Add* Module either by using the menu or the Searchbox.
5. Wire the *Mult/Add* Module into the structure like this:



Looking back to the signal flow diagram from earlier, you can see the modifications that need to happen to your current effect in order to add a feedback loop:



The Feedback Signal Flow Diagram

You will need to take the delayed signal, multiply it by a level control, add it to the input, and then feed the resulting signal back into the delay module. You can do this easily with another *Mult/Add* and a *Knob*:

1. Create a *Mult/Add* Module and a *Knob* Module and wire them into the structure like this:

Do not worry about the Z over the input of the Mult/Add Module, this just shows that there is feedback in the structure, and a 1 sample delay has automatically been added because of it.

2. To keep things readable, rename the new knob to "Feedback".
3. Return to the panel view and move the newly created knob so that it is no longer sitting on top of any other controls:

→ You have added a feedback path to your echo effect.

The newly added Feedback knob will control the amount of signal fed from the output of the delay back into its input. This effectively controls the number of repeating echoes in the effect.

Before you test the updated echo effect, make sure the feedback knob is not set too high, otherwise the effect could overload.

4.2.4 Adding Tempo Sync

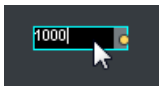
Currently, you can only set the delay time of your effect in milliseconds, but for a lot of creative echo effects the user is able to sync the delay to the host tempo. In order to do this, you will need to retrieve the tempo information somehow. Fortunately, REAKTOR includes a Built-in Module that allows you to do exactly that: the *Tempo Info* Module.

The *Tempo Info* Module outputs the host tempo in Beats per Second (BPS). You will need to convert this to Milliseconds per Beat (µPB) using the following formula:

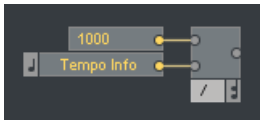
```
µPB = 1000/BPS
```

1. Create a *Tempo Info* Module.
2. Create a *Constant* Module.
3. Enter the *Constant* Module's properties and set the value to 1000.

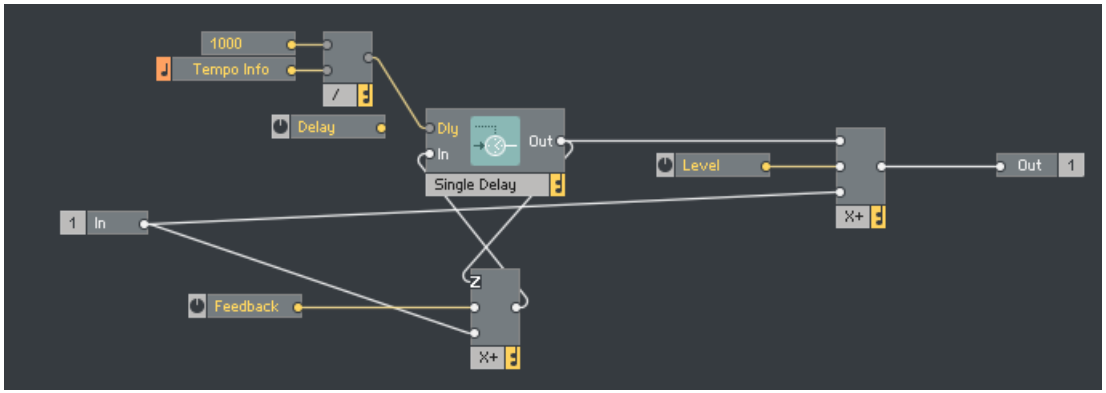
4. You can also set the value of a *Constant* by double-clicking on it, much like how you rename Macros and Knobs.



5. Create a *Divide* Module.
6. Re-create this formula in REAKTOR by using these Modules to build the following structure:



7. Connect the output of the *Divide* Module to the Dly input of the *Single Delay* Module, replacing the Delay knob.

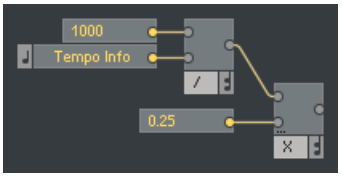


→ The echo time is now synched to the length of a quarter note.

Although the echo is tempo-synched, it is fixed at a single length.

You can create a delay that uses multiples of 16th note lengths quite easily:

► Take the µPB value and multiply it by 0.25 to get the milliseconds per 16th note value.



The Multiply Module can have more than two inputs. You can attach another wire to this Module using the same technique described at the start of this tutorial, in which you used the [Ctrl]/[Cmd] keys to create ports while dragging wires.

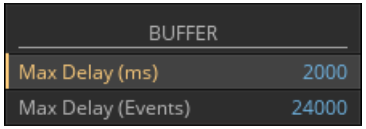
Modules and Macros that can have extra inputs created in this way are identified by the ellipsis (three dots) below its bottom input port.

If you play with the effect now, you may notice that the delay time does not change when you set the Delay knob to a value above 8. This is because the Single Delay Module has a maximum delay time buffer, which is 1000ms by default (equivalent to 2 beats at 120bpm).

As the delay essentially records the audio in order to play it back later as the delayed signal, the delay buffer assigns a section of memory for this recording. Larger buffer sizes can be useful for longer delay times, but will take up more RAM. The maximum setting for the delay buffer is 1000000ms (a little under 16.7 minutes) which equates to about 96MB of RAM for 16bit/48kHz audio.

You can change the buffer setting in the Single Delay's Properties:

1. In the Single Delay's properties, under the Function tab, find the BUFFER section.
2. In this section locate the Max Delay (ms) parameter.
3. Double-click the parameter's value and set it to 2000.



4. Press [Enter] to set this change.

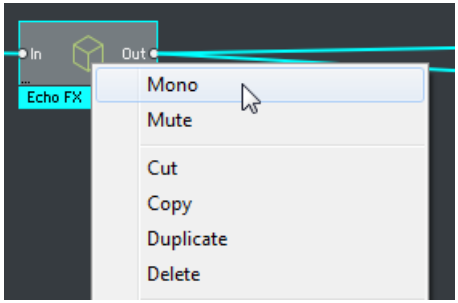
Congratulations, you have just created a tempo-synced echo effect in REAKTOR!

4.2.5 Saving and Loading the Macro

You can export your echo effect for use in other Ensembles by saving it as a Macro.

Before you save it, you'll need to make sure it remains as a monophonic structure, even in polyphonic Ensembles. To do this:

1. Navigate to the top level of your Ensemble.
2. Select the echo Macro and right-click on it.
3. Select the Mono option from the menu.

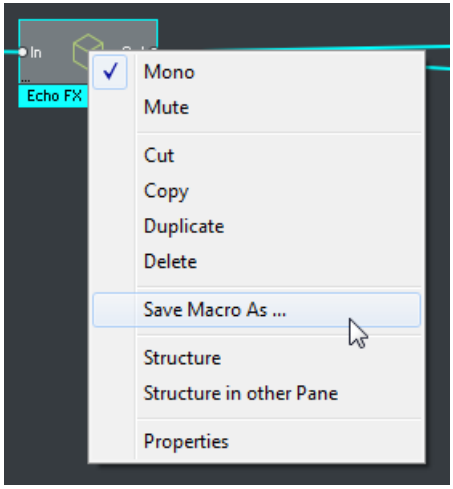


→ All Modules inside the Macro have been set to monophonic, so even when you load the Macro into a polyphonic Instrument, the echo effect will remain mono.

Next, export the effect by saving it as a Macro:

1. Select the echo Macro.

2. Right-click the Macro and select *Save Macro As …*



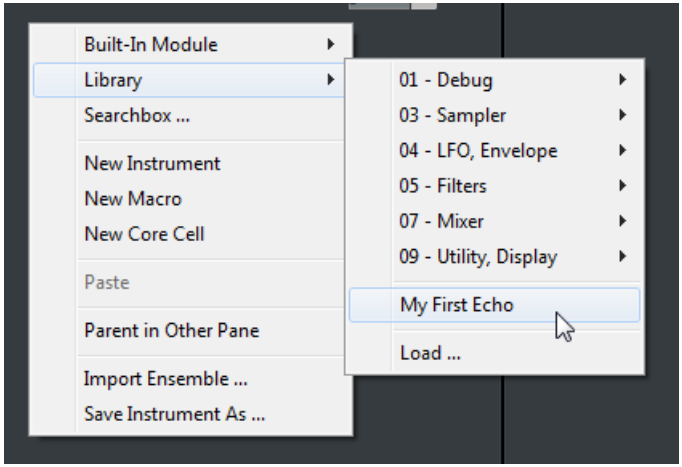
3. Select a save location for the Macro file (saving it to the default location makes it easier to locate in REAKTOR later).
4. Give it a unique name like "My First Echo".
5. Click Save.

→ You can now load your echo effect like any other Macro.

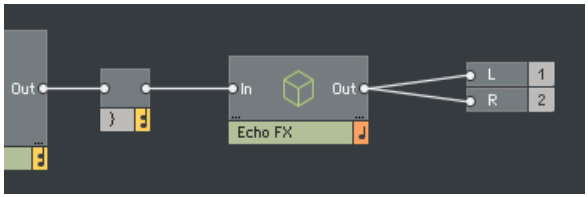
Using the subtractive synthesizer from the previous tutorial as an example, you can load and connect your echo effect like this:

1. Load the Ensemble you wish to use and enter its structure.
2. Rick-click on an empty space in the structure and select *Library*.

3. If you saved your effect in the default location, it should appear at the bottom of this menu.



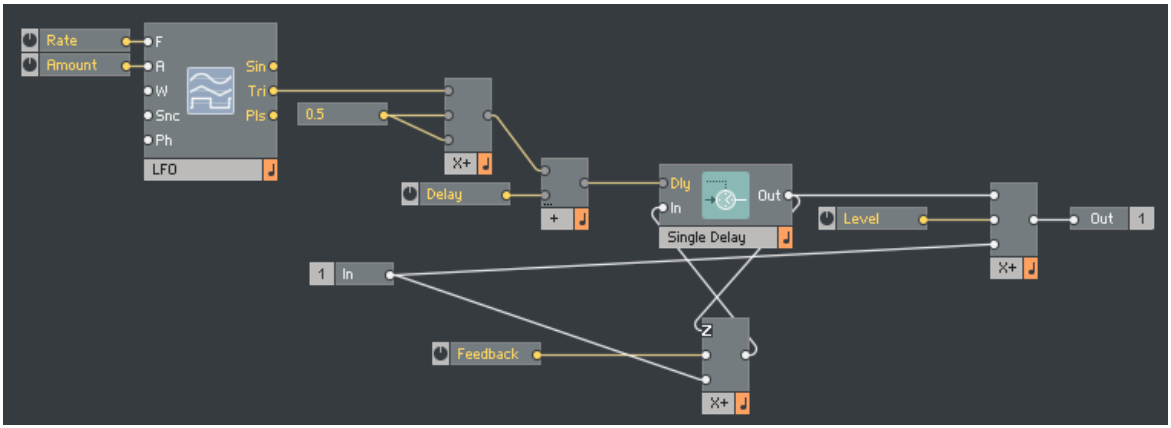
4. If you saved it somewhere else you can locate it by clicking on *Load ...*
5. Since the echo effect is monophonic, it will need to be placed after any *Audio Voice Combiners* in the structure, like this:



4.3 Going Further

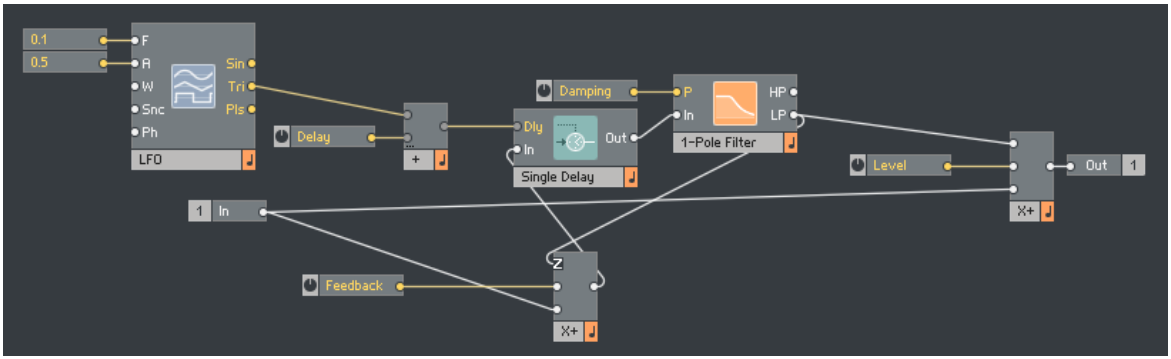
The delay effect is the basis for a few other effects including flanger and chorus.

You can create a flanger/chorus by modulating the delay time with an LFO:



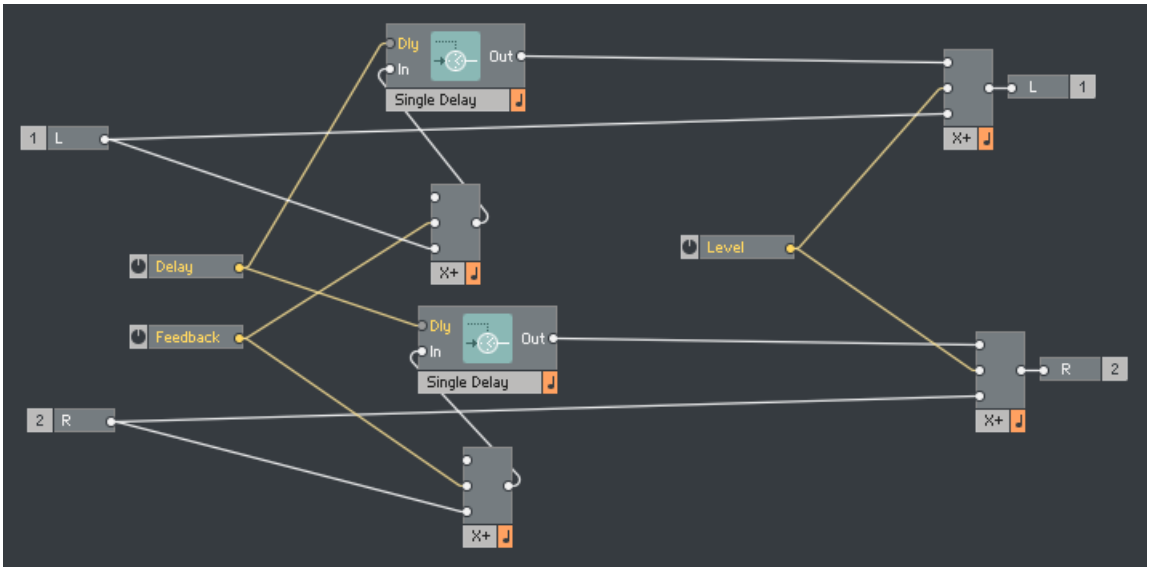
The structure of a simple chorus/flanger effect.

You could also create a vintage style delay by adding filters to the delay output/feedback chain, and slow, subtle modulation to the delay time:



The structure of a vintage style echo effect.

You can make any of these effects stereo by duplicating the macro structure:



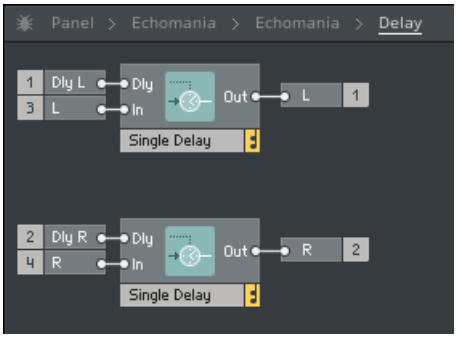
The structure of a basic stereo delay.

4.3.1 Related Factory Content

The following REAKTOR 6 Factory Library Ensembles were all built using simple delay units at their core. With what you have learnt here, you should be able to reverse engineer these Ensembles to take your knowledge further.

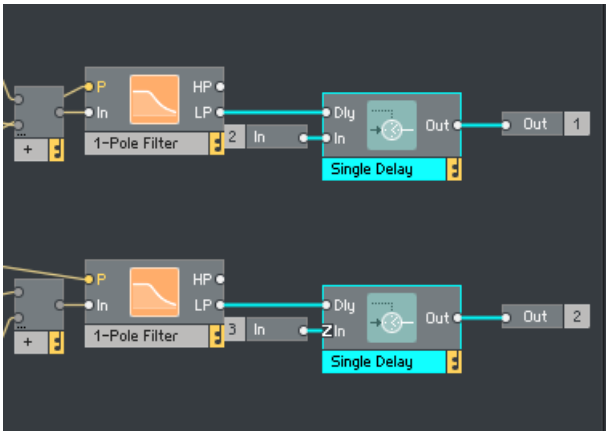
> The following delay based Ensembles can be found in the Effects section of the Factory Library.

Although it features LFOs and EQs, the *Echomania* Ensemble is ultimately driven by two *Single Delay* Modules.



The two Single Delay Modules in Echomania

Longflow is another effect that ultimately comes down to two *Single Delay* Modules.



The Single Delay Modules in Longflow

5 Basic Step Sequencer

5.1 Overview

Event signals in REAKTOR behave differently from audio signals: audio signals constantly send values at the audio rate, whereas event signals can have their timing controlled. Events can be triggered, and thus can be used to trigger other events.

This process of triggering timed events is ideal for creating step sequencers, which is what the following tutorial will cover.

5.1.1 Previous Knowledge

This tutorial assumes you have read the previous chapters of this document, and so are familiar with the following concepts:

- How to navigate around REAKTOR's Edit view
- Understanding the difference between audio and event signals
- Understanding the difference between monophonic and polyphonic signals
- How to create wires between ports
- How to create Macros and Modules

5.1.2 The Theory

A basic step sequencer sends a sequence of values at timed intervals. They are commonly used to create musical patterns by sequencing the pitch of a synthesizer's oscillators, or they can be used to modulate sound parameters to create rhythmic effects. It is also common to sync a sequencer to the tempo of a project.

Common hardware step sequencers have a row of knobs to set the sequence of values, with each knob representing a step in the sequence. A clock signal is used to progress the sequence: when a clock tick is received, the sequencer advances to the next step, outputting the current step's value.

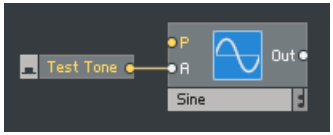
In REAKTOR, our clock signal will be created using timed events.

5.2 Tutorial

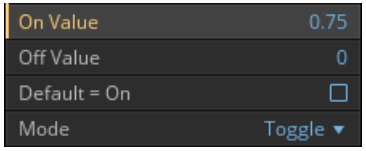
5.2.1 Setting Up

In order to test the sequencer, you will need to create a single oscillator:

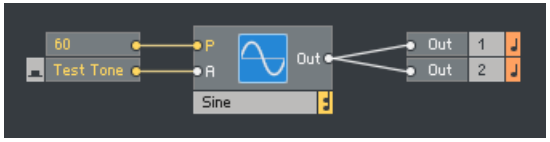
1. Create a new empty Ensemble.
2. Enter the structure and create a *Sine Oscillator* Module.
3. Create a button and attach it to the A (amplitude) input port of the *Sine Oscillator*.
4. Rename the button to "Test Tone"



5. In the button's Properties, change the On Value to 0.75. This sets the output amplitude of the *Sine Oscillator* when the button is on. An output amplitude of 1 can overload REAKTOR's output and should be avoided.



6. Create a Constant for the P (pitch) input of the oscillator by right-clicking on the port and selecting *Create Constant*.
7. Connect the output of the *Sine Oscillator* to the Ensemble's main Out ports.



→ The oscillator is now ready to be used. Pressing the Test Tone button activates the Oscillator with a MIDI pitch of 60 (middle C). We recommend to lower the volume of your loudspeakers before testing the oscillator.

Next, you will create a Macro inside which you will build your sequencer:

1. Create a new Macro.
2. Re-name it to "Sequencer".

5.2.2 Building the Basic Sequencer

As mentioned in the Overview, a sequencer uses a clock to move between steps in a sequence, so you will start building your sequencer by creating a clock source.

1. Enter the *Sequencer* Macro structure.
2. Create a *Clock Oscillator* Module.
3. Create a *Constant* at the A input port by right-clicking on the port and selecting *Create Constant*.
4. Create a *Knob* at the F input port using the same right-click method, but this time selecting *Create Control*.

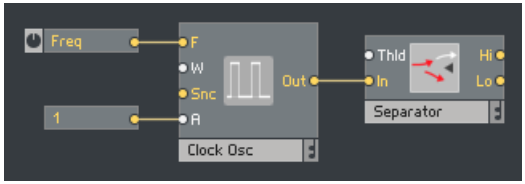
Using the right-click method to create constants and controls usually sets their Properties to commonly used values.

You have now created a clock source (the *Clock Oscillator*) which outputs events with a value of 1 at a rate specified by the frequency at the F input. Each of these events is followed by an event of 0, producing a kind of square wave with a frequency of F.

Since the sequencer structure will ignore the value of these events, only using them to trigger additional events, you will need to filter out the 0 value events, as these double the frequency of events.

1. Create a *Separator* Module. This Module splits events depending on whether or not their value is above or below a certain threshold (Thld).

2. For this case, the threshold should be 0, so you can leave the Thld port empty as it defaults to 0
3. Connect the Out port of the *Clock Oscillator* to the In port of the *Separator*.

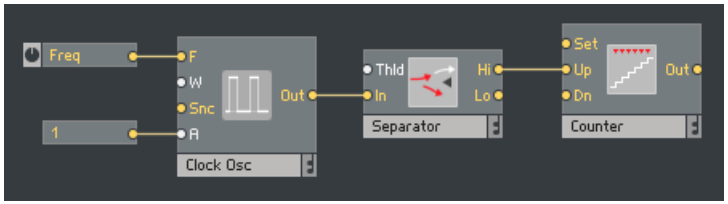


→ The Hi output port of the Separator Module will now act as the output of our clock.

Each event produced by this clock needs to trigger a value to increase by one. This value will be used to select the steps in the sequencer.

You can build this using the *Counter* Module.

1. Create a *Counter* Module.
2. Connect the Hi output of the *Separator* to the Up port of the *Counter*.



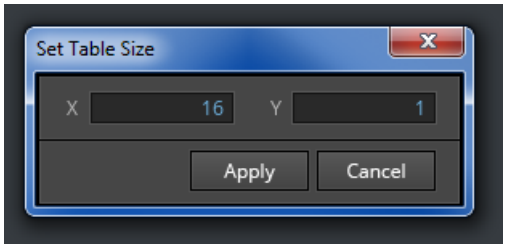
→ Every time the clock sends an event, the *Counter* will increase by one.

With this basic counting structure in place, you can move on to creating a Module for setting the values the *Counter* will address. For this tutorial, you will use a Module called the *Event Table*. This Module holds a table of values (if you are familiar with programming, you may like to think of it as a 2D array), and is perfect for quickly creating step sequences.

1. Create an *Event Table* Module.
2. In the *Event Table* Properties, locate the SIZE section in the Function tab.

3. Click on the Set button.

4. A window will open, asking you to set the table size for the event table.
5. You will be creating a 16 step sequencer, so set the X value to 16 and leave the Y value at 1.



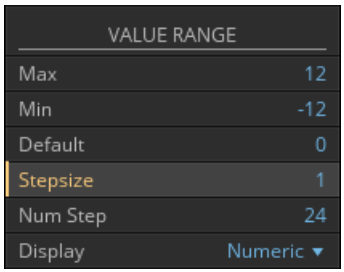
6. Click Apply.

→ You have created a table of 16 values.

Next, you will need to set the range of the values in the table. This sequencer will be used to control the pitch of the *Sine Oscillator* you created earlier, so the range should be +/- one octave (+/- 12 semi-tones).

1. In the *Event Table* Properties, locate the VALUE RANGE section in the Function tab.
2. Set the Max value to 12 and the Min value to -12.

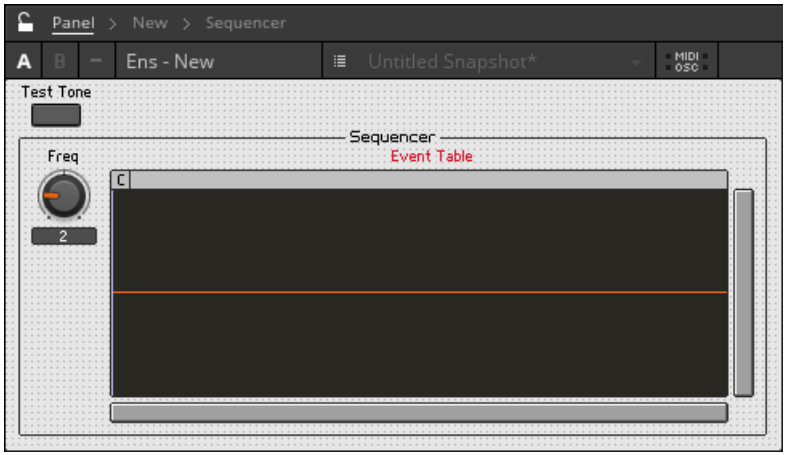
3. Set the Stepsize to 1. The Num Step property should automatically update to 24 when you do this.



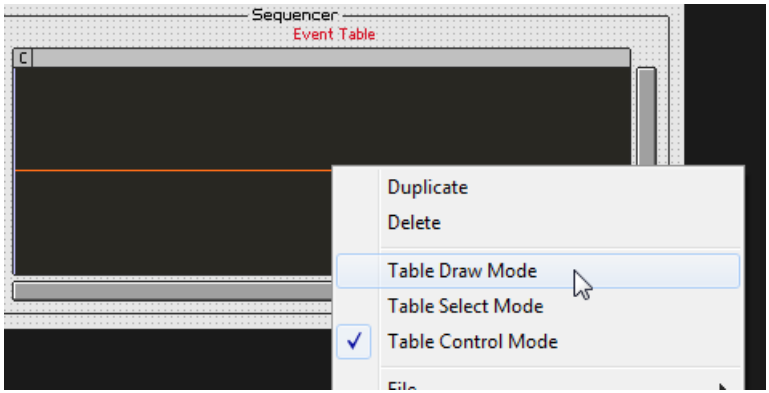
You have created a table of values with a possible range of -12 to +12. You now need to create a way of editing these values.

The *Event Table* has a panel element which shows the values in the table. With the following additional settings, you will also be able to edit them from the panel:

1. Go to the Ensemble Panel view.
2. Unlock the Panel and rearrange the controls so they are no longer on top of one another.



3. Right-click on the *Event Table* and click on the *Table Draw Mode* option.



4. Lock the Panel.

→ You will now be able to draw on the Event Table to edit the values in the sequencer.

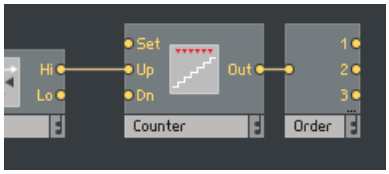
To complete the sequencer, you need to connect the *Counter* to the *Event Table.*

The output of the *Counter* will select an X position in the *Event Table* using the RX (Read X) input, and then trigger the *Event Table* to output the value at that position using the R (Read) input.

Triggering two inputs from a single output in this way can be sensitive to the ordering of events. In this case the RX input needs to receive the *Counter* output event before the R input.

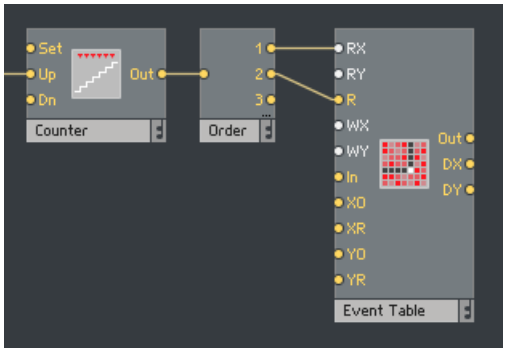
To control the order of the events, you can use an *Order* Module.

1. Navigate to the Sequencer structure.
2. Create an *Order* Module.
3. Connect the output of the *Counter* to the input of the *Order* Module.



4. Connect output 1 of the *Order* Module to the RX input of the *Event Table*.

5. Connect output 2 of the *Order* Module to the R input of the *Event Table*.



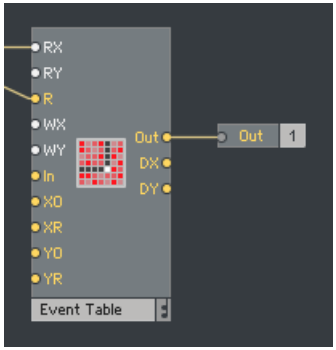
→ The *Order* Module now ensures that RX receives the *Counter* value before R.

Simple cases like this may not require an *Order* Module; however it is good practice to use them to ensure that events are processed as planned.

Testing the Sequencer

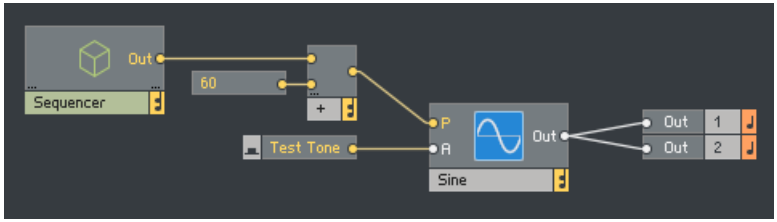
With your sequencer complete, you can connect it to the Sine Oscillator for testing:

1. Create an *Out Port* Module in the *Sequencer* Macro.
2. Connect the Out port of the *Event Table* to the input of the *Out Port* Module.

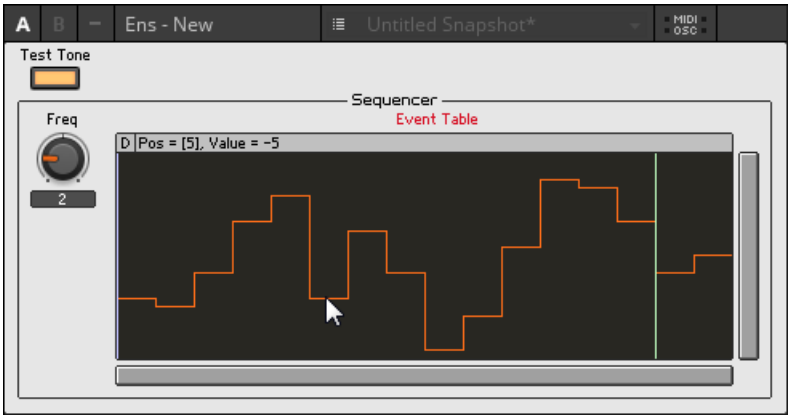


3. Navigate to the top level of the Ensemble structure.
4. Create an *Add* Module.

5. Use the *Add* Module to add the sequencer output to the *Constant* at the P input of the *Sine Oscillator*, like this:



When you return to the Ensemble Panel, you can draw a sequence by clicking and dragging on the *Event Table*. You can then listen to this sequence by activating the Test Tone button.



Editing and Testing the Sequencer

Changing the Freq value will alter the rate at which the sequence plays.

5.2.3 Adding Tempo Sync

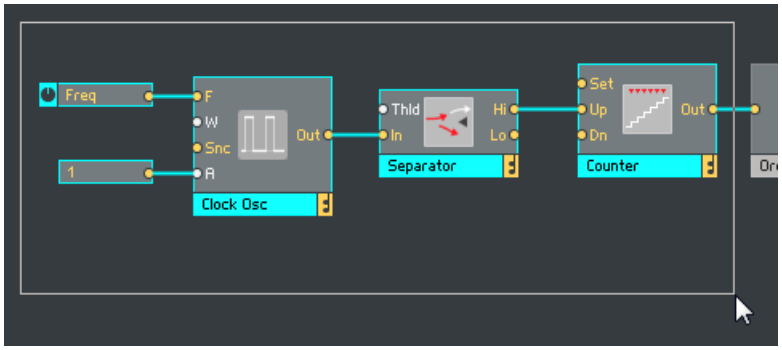
It is common for step sequencers like the one you have just created to be synchronized to the host tempo. This can be further improved by synchronizing to the song position, not just the tempo.

REAKTOR has a *Song Position* Module that allows you to access the current playback position of the host in 1/96 note resolution.

In this next step of the tutorial, you will replace the clock you previously created with this *Song Position* Module.

Before you build anything new, remove all of the Modules that are a part of the clock functionality:

1. Enter the structure of the *Sequencer* Macro.
2. Select the Modules pictured below. You can select them all at once by clicking and dragging with the mouse to create a box around them.



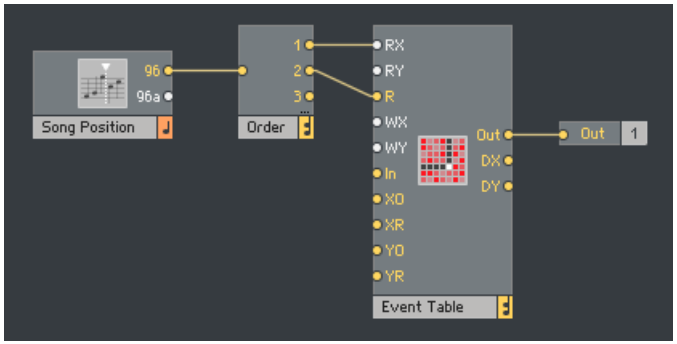
3. Delete them by pressing the [Delete] key on your keyboard.

Now you can replace the old clock structure with a *Song Position* Module.

1. Create a *Song Position* (MIDI In) Module.

The *Song Position* Module has two outputs: the top one outputs event signals and the lower one outputs an audio signal. Both outputs send the host song position at 1/96 note resolution.

2. Connect the event (top) output to the input of the *Order* Module. You are using the event output as it sends a single event with each song position value, and thus can also be used as a clock source.



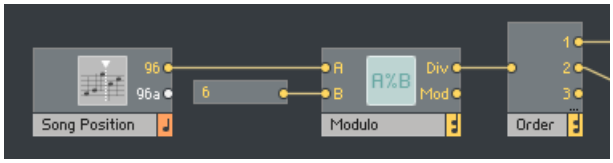
When you return to the panel and test the sequencer, you may notice that it is not running. This is because the song position will not change unless the transport is running, and if the song position does not change then the *Song Position* Module will not output any events.

► Start the transport by pressing the play button in the Toolbar, or by pressing [Space] on your keyboard.

With the transport running, the sequence will start playing. However, the sequence is running at a fast 1/96 rate, so you need to adjust the structure to control this.

By using a divider that truncates (rounds down), you can control the resolution of the *Song Position*, and thus the rate of the sequencer.

1. Create a *Modulo* Module. This Module outputs the truncated result of a division function.
2. Wire the event output of the *Song Position* Module to the A input of the *Modulo* Module.
3. Create a *Constant* and connect it to the B input of the *Modulo* Module.
4. In order to get a 1/16 rate, you need to divide the 1/96 position by 6 (96/6 = 16), so set the *Constant* to 6.
5. Connect the Div output of the *Modulo* Module to the input of the *Order* Module.



→ The sequencer now runs at 1/16 rate.

Even though you have reduced the resolution of the song position to 1/16 notes, the Modulo Module is still outputting events at a 1/96 rate. So instead of outputting the following values and events:

`0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, etc…`

It is now outputting the following at the same rate:

`0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, etc…`

To optimize the structure, the sequencer should filter out repeated values. Fortunately REAKTOR includes a *Step Filter* Module to do just that.

► Create a *Step Filter* Module and connect it between the *Modulo* and *Order* Modules, like this:

The Tol (tolerance) input of the Step Filter can remain disconnected since the default value works with this structure.

With the *Step Filter* connected, a new event will only be sent to the *Event Table* when the value changes.

Although this optimization did not change the final output of the Ensemble in any noticeable way, this technique is very important. If a structure uses events to trigger sounds, filtering repeated values ensures that the sounds are only triggered when the value changes.

The final step in creating this sequencer is to add control over the rate. Changing the value at the B input of the Modulo Module will do this, however a knob is not ideal as you do not need a large range of values, only those that produce commonly used musical note divisions.

The *List* Module allows you to create a menu with any number of entries, each with specified names and values. Using this Module, you can create a menu of musical note divisions, but output the number required to get that rate from the *Song Position* Module.

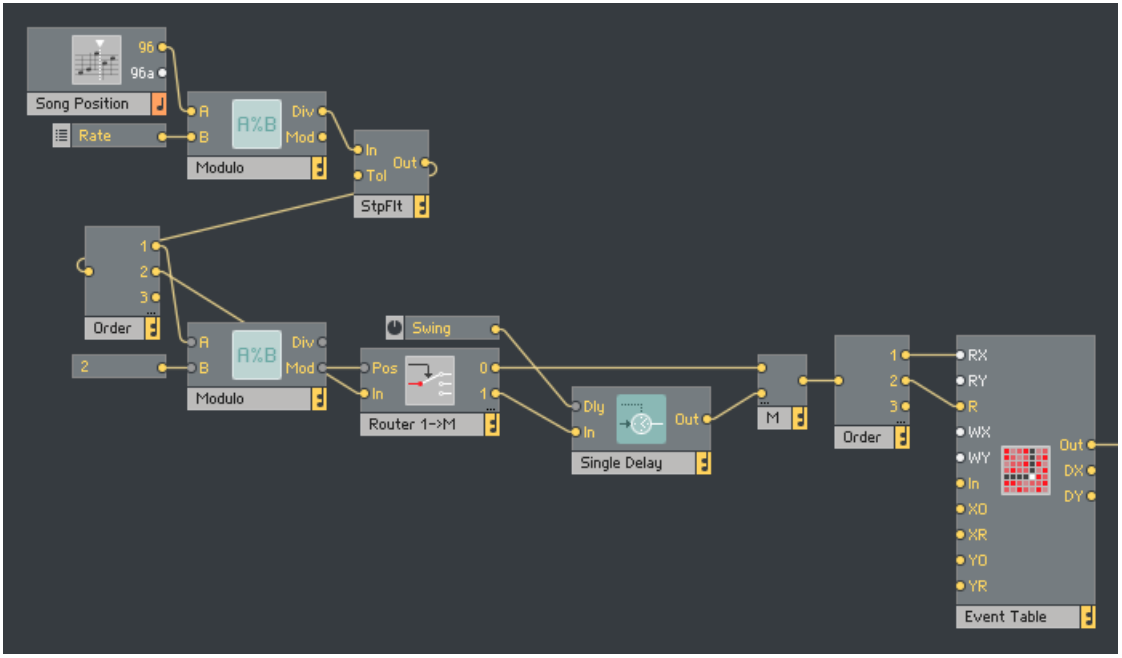
To find the values needed, just use the simple formula: value = 96*rate. For example, if you want quarter notes the value = 96 * (1/4) = 96/4 = 24.

→ You can now use the Rate Menu to specify the rate of the sequencer using musical note divisions.

5.3 Going Further

You can use a sequencer to control any parameter, not just pitch. Try saving the *Sequencer* Macro and importing it into other Ensembles. Be sure to set the Macro to Mono, and to change the Range values of the *Event Table* accordingly.

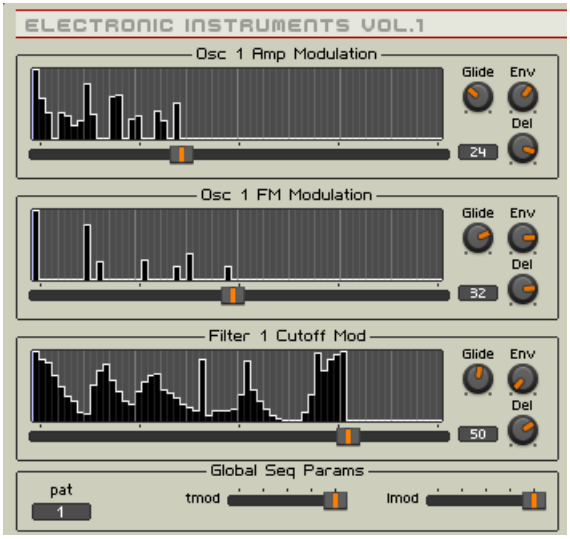
You can use further event handling and a *Single Delay* Module to create swing:



A Sequencer with Swing

5.3.1 Related Factory Content

Any sequencer in the Factory Library will be built on the basic concepts you have just learnt. However, there are two Ensembles in particular that uses Event Table Modules in their sequencers: *Atmotion* and *Vierring*. Both of these Ensembles can be found in the Sequenced Synthesizers section of the Factory Library.



The Step Sequencers of Atmotion

6 Advanced Step Sequencer

6.1 Overview

REAKTOR offers a number of Modules that you can combine to create custom controls.

Building on the ↑5, Basic Step Sequencer tutorial, this tutorial will guide you through using the advanced Panel Modules and techniques to create your own Step Sequencer interface.

6.1.1 Previous Knowledge

As well as assuming a basic understanding of building in REAKTOR, this tutorial also assumes that you have completed the tutorial in section ↑5, Basic Step Sequencer ↑5, Basic Step Sequencer.

The following will not only build on the Ensemble created in that tutorial, but will also assume all knowledge acquired to that point.

6.1.2 The Theory

By this stage you should have already created a ↑5, Basic Step Sequencer, however certain aspects of this sequencer are limited - the information in the sequencer is stored in the Ensemble, and not in the Snapshots.

Building on the ↑5, Basic Step Sequencer also makes a good starting point for learning some advanced concepts about REAKTOR's GUI (Graphical User Interface) capabilities.

Until now you will have been using simple, pre-made Panel Modules like knobs and buttons. While these are useful, sometimes you'll need a control with an option that is unavailable in those on offer, and this is why REAKTOR also offers additional advanced Panel Modules.

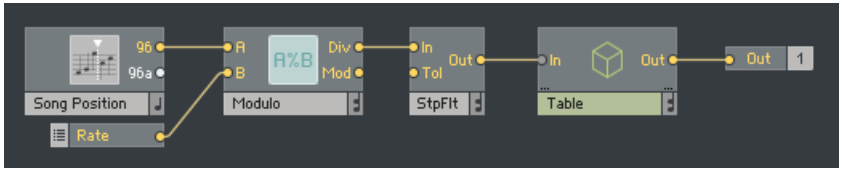
In this tutorial, you will replace the *Event Table* Module used in the ↑5, Basic Step Sequencer with your own table, built from scratch. This new table will allow you to save sequences in Snapshots, and not just in the Ensemble.

6.2 Tutorial

6.2.1 Setting Up

This tutorial will build on the ↑5, Basic Step Sequencer, focusing on replacing the *Event Table* Module.

1. If you do not have the Step Sequencer Ensemble loaded, open it now.
2. Enter the Sequencer Macro structure and delete the *Event Table* Module and the *Order* Module.
3. Create a new empty Macro.
4. Rename it to "Table".
5. Attach the Table Macro to the structure as follows:



6.2.2 Building the Sequencer Interface

Before you build any of the step sequencer functionality, you will create the interface elements needed.

1. Enter the Table Macro structure.
2. Create a *Mouse Area* Module.

The *Mouse Area* Module allows you to access the state and position of the user's mouse when it is over a defined area on the Panel. Because of this, the *Mouse Area* is a very useful Module when creating custom interfaces.

The *Mouse Area* sends the actions of the user to our structure, but this is only one part of the functionality of an interface control; the sequencer will also need a display to show the values of the sequencer to the user.

To ensure the display stays under the *Mouse Area* (so as not to block the user's input) you should create a new Macro to contain the display.

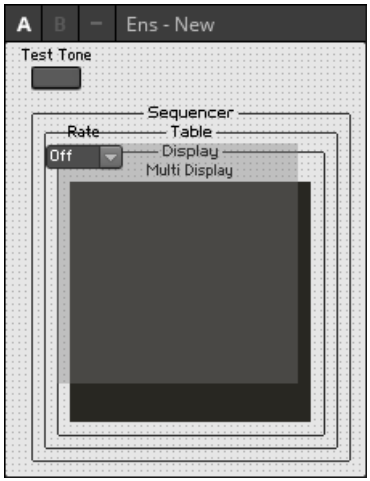
1. Create a new empty Macro.
2. Name this Macro "Display".
3. Enter this Macro and create a *Multi Display* Module.

→ Because the *Multi Display* is contained in a Macro one level below the *Mouse Area*, it will be placed under of the *Mouse Area* on the panel.

The *Multi Display* Module is a Panel element that can contain a number of objects, which can be made of basic shapes, like lines and rectangles. In this tutorial, the *Multi Display* will be used to create the bars of the sequencer.

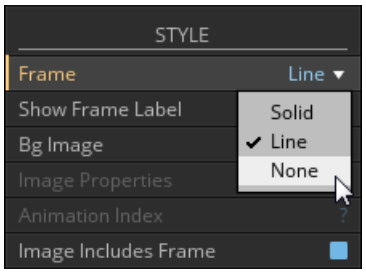
The *Mouse Area* and the *Multi Display* will be the only Panel elements you'll need for this tutorial, so now is a good time to adjust the Panel to make sure everything is in the right place.

1. Return to the Ensemble Panel and Unlock the Panel.
2. When you unlock the Panel, you will notice that the Mouse Area is shaded so that you can see where it is located.

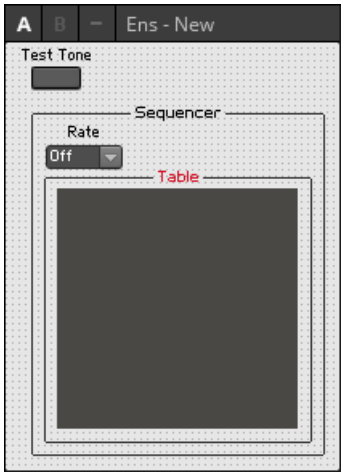


3. Select the *Multi Display* to show its properties.
4. Navigate to the View tab and uncheck the Show Label option to hide the label.
5. Select the Display Macro to show its properties.

6. Navigate to the View tab and set the Frame parameter to None. This will hide the frame of the Macro without hiding its contents.



7. Move the Table Macro so that it is no longer overlapping with the rate control.

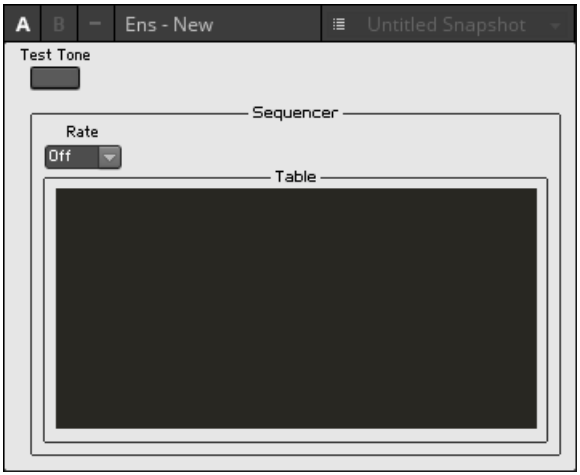


The interface is starting to come together, but the default square shape of the Table Module is not ideal for a 16-step sequencer.

To re-size the Table:

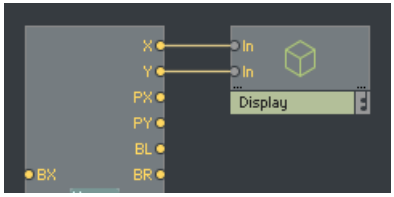
1. Select the *Mouse Area* Module.
2. Navigate to the View tab of its Properties.
3. Change the Width parameter to 300.
4. Repeat this process for the *Multi Display*.

→ The Table is now a 300 x 150 rectangle.

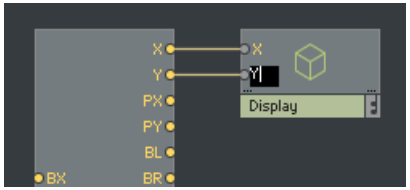


With the panel organized, now is time to connect the *Mouse Area* to the *Multi Display*:

1. Enter the Table Macro structure.
2. Connect the X and Y outputs of the *Mouse Area* to the Display Macro.



3. Rename the ports on the Display Macro to match the corresponding ports on the *Mouse Area*. You can do this by double-clicking on the ports and typing in the new names.



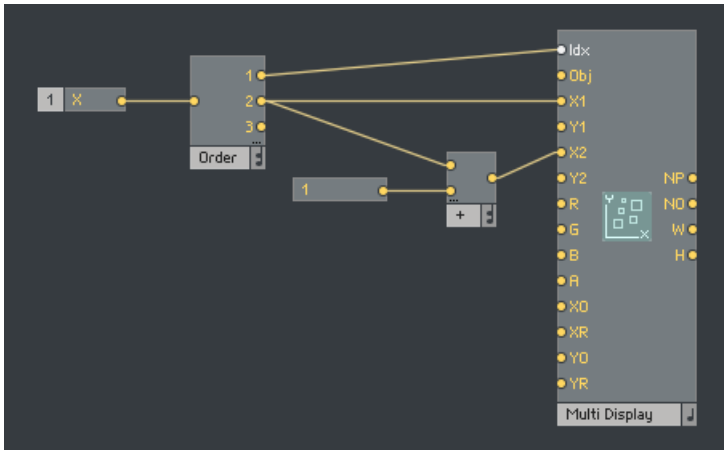
When the user interacts with the sequencer interface, the X position of the mouse will correspond to the step in the sequence, and the Y position will give the value at that step.

When using the *Multi Display*, you need to select which object you are editing by using the Idx (index) port. Once you have selected an object, you can set its parameters using the other input ports.

Therefore, the X position of the mouse should select the object index, and that the Y value will define the object height.

Using this information, you can connect the X and Y ports to the various input ports on the *Multi Display* Module:

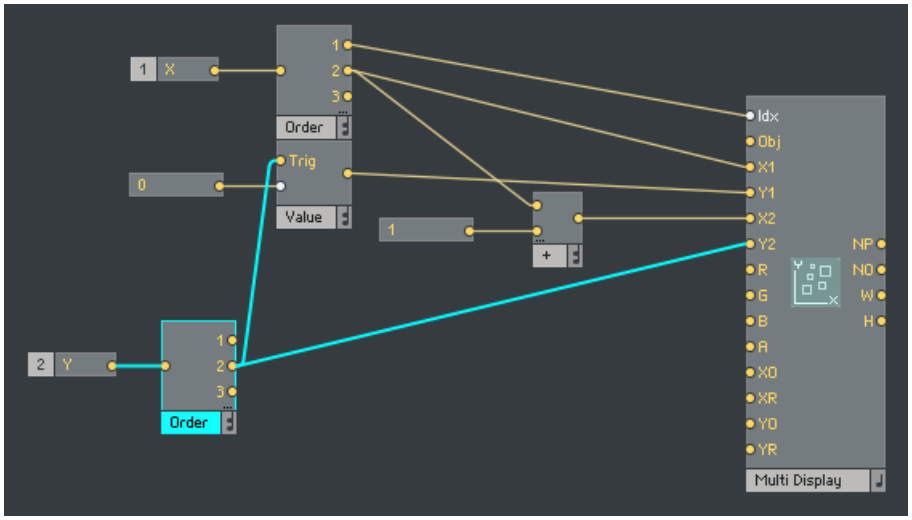
1. Enter the Display Macro.
2. Connect the X port to the *Order* Module. The *Order* Module ensures that the object index is set before the X co-ordinates are set.
3. Connect output 1 of the *Order* Module to the Idx (index) input of the *Multi Display*.
4. Connect output 2 to the X1 input
5. Create an *Add* Module with a *Constant* of 1 attached to the bottom input.
6. Wire the Modules as follows:



With the Modules connected in this way, the index will be selected, and then an object will be drawn from X1 to X2. X2 will be the same as X1 of the next object, so the bars should be placed next to each other.

The Y1 position will always be the center of the display, and therefore should always have a value of 0. Y2 will define the height of the bar.

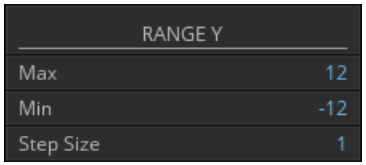
1. To ensure the Y values arrive after the Idx value is set, you will need to create another *Order* Module.
2. Create *Value* Module. When the *Value* Module receives an event at the Trig input, it outputs a new event with a value defined at the lower input.
3. Create a *Constant* and set it to 0.
4. Connect the Modules as follows:



You will notice that you have used the second output of the *Order* Module for the Y values. This is to ensure that the Y values are also processed after the Index is set.

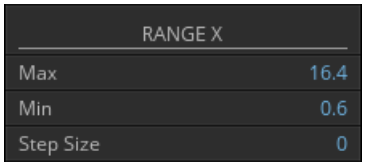
With everything wired up, you will need to set the properties of the *Mouse Area* and the *Multi Display* to produce the desired results.

1. Enter the *Mouse Area* properties.
2. The Y range will set the value of each step, which you'll want to be -12 to +12. So in the RANGE Y section, set the Max value to 12 and the Min value to -12.
3. Set the Step Size in the RANGE Y section to 1.

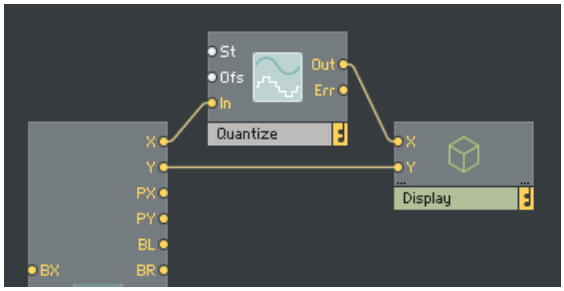


The X range will select the bar in step in the sequencer, so you will want a range of 1 to 16. However, this will not align with the display, as and X value of 1.5 will round up to 2 and select the next step even though you are still in the first section of the *Mouse Area*.

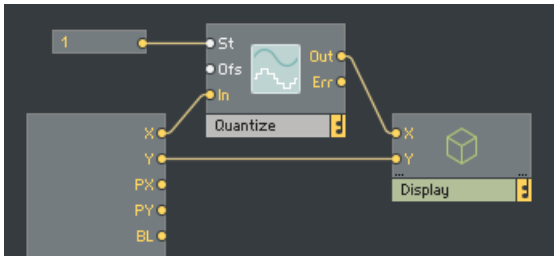
1. To compensate for this, set the Min value of the RANGE X section to 0.6 and the Max value to 16.4. Leave the Step Size at 0.



2. Then enter the Table structure and create a *Quantize* Module.
3. Connect the *Quantize* Module like so:



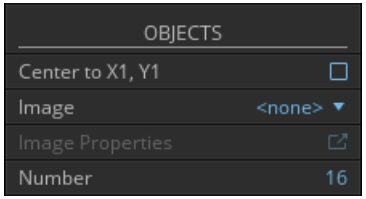
4. The *Quantize* Module needs to round the X values to the nearest whole number, so set the St (steps) to 1 by connecting a *Constant* to that port.



Now that the *Mouse Area* properties are defined, it is time to edit the *Multi Display*:

1. Enter the Display Macro and select the *Multi Display* to show its properties.

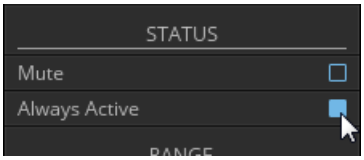
2. For the Table you'll need 16 objects in the display (one for each bar), so locate the OBJECTS section and set the Number to 16.



3. In the RANGE section, set the X Origin to 1 and the X Range to 16.
4. Set the Y Origin to -12 and the Y Range to 24 (the minimum value is -12 and the maximum is 12, this is why the range is 24).

The final step here is very important, and is a common stumbling block for first time builders who are unfamiliar with REAKTOR.

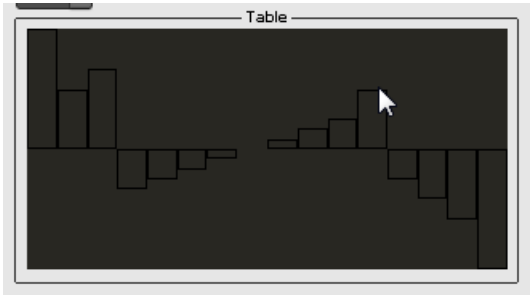
► Check the Always Active option in the Multi Display properties.



Checking this option ensures that the *Multi Display* will function, even though none of its outputs are connected to anything.

It is now time to return to the Panel and check your work.

► Click and drag the mouse on the table to draw in a sequence.



The bars of the Table are black outlines, which are not ideal, but you can change this easily:

1. Return to the *Multi Display* Module in the structure.
2. The *Multi Display* has 3 inputs for setting the object color, labeled R, G, and B for Red, Green, and Blue. They all accept values between 0 and 1.
3. Create a *Constant* of 1 and connect it to the G and B ports.

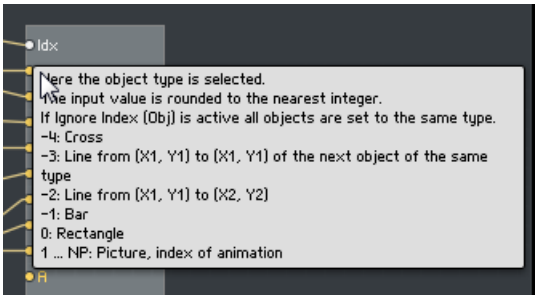
→ This will set the object colors to cyan (a mix of green and blue).

The *Multi Display* also has an input for setting the object type (Obj). Without anything connected to this input, the current object is a Rectangle.

1. Create a *Constant* with a value of -1.
2. Connect this to the Obj port of the *Multi Display*.

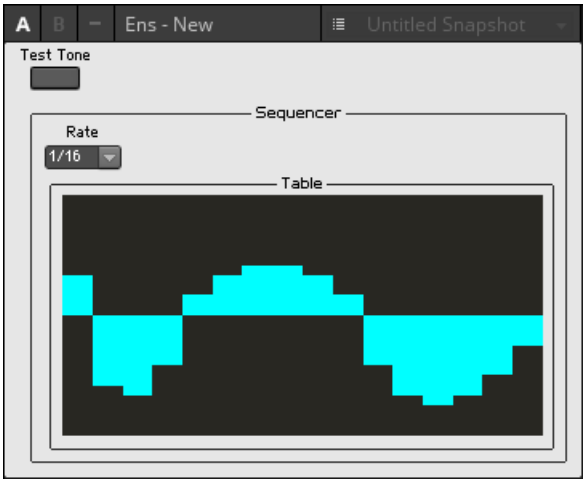
→ The objects are set to solid bars.

You can use Info Hints on ports to see the available options and value ranges. Using Info Hints on the Obj port gives you a list of the available object types and their corresponding values.



Using Info Hints on the Obj Port

Return to the Panel and take another look at the Sequencer.



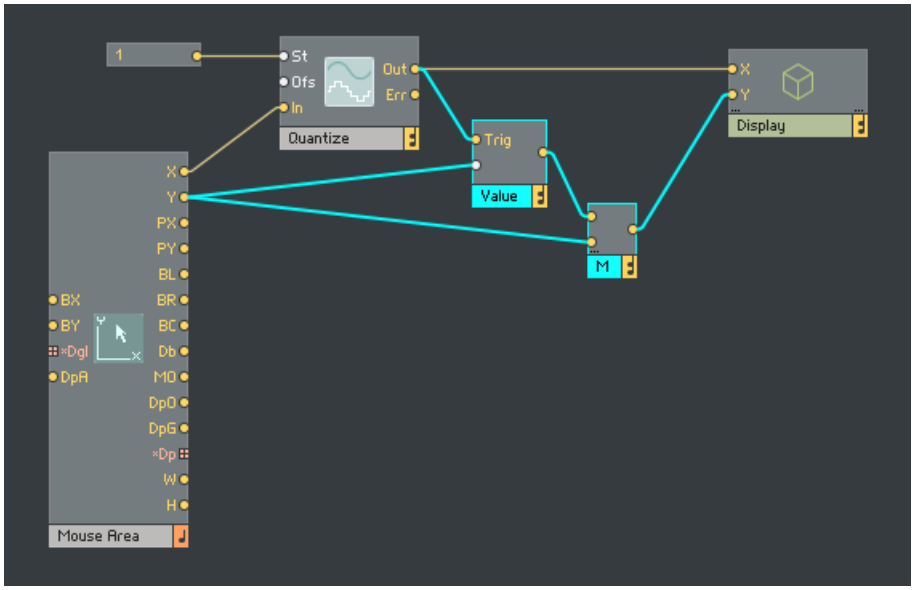
The Current Look of Your Sequencer

Before you move on to the next section, you need to do a little optimization of your sequencer. At the moment, dragging the mouse along the X axis, without moving in the Y axis, doesn't update the values of the steps of the sequencer; this is because the *Mouse Area* is not sending any events from the Y output, since the Y value is not changing.

You can fix this using a *Value* Module and a *Merge* Module:

1. Enter the Table Macro structure.
2. Create a *Value* Module and a *Merge* Module.

3. Connect these Modules to the structure like this:



→ Now the Y value will send an event when either it changes or when the X value changes.

6.2.3 Storing Values in a Snapshot

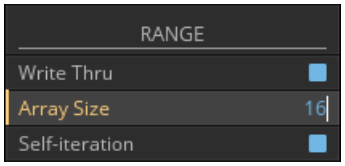
The sequencer you have created looks good and the interface is working, but it does not function like a sequencer yet.

Before you get the sequencer running, you will need a way of storing the values of the sequencer in a Snapshot.

REAKTOR includes a *Snap Value* Module that stores a value when a Snapshot is saved, and outputs that value when the Snapshot is recalled. For the Sequencer you will need to store 16 values in the snapshot, and this requires a *Snap Value Array*, which has similar functionality to the *Snap Value* Module, but can store an array of values, rather than just one.

1. Enter the Table Macro structure and create a *Snap Value Array* Module.
2. Open the *Snap Value Array*'s properties.

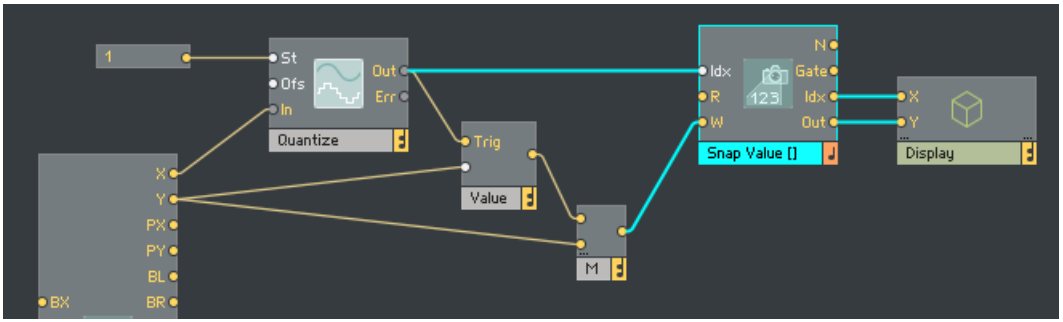
3. In the RANGE section, set the Array Size to 16.



→ You now have a *Snap Value Array* that can store 16 values.

The *Snap Value Array* needs to store the values sent by the *Mouse Area*, and send them to the Display.

► Connect the *Snap Value Array* in the structure like this:



As with the *Multi Display* Module, the *Snap Value Array* will use the X value of the *Mouse Area* to set the index (Idx), and the Y value to write (W) a value into this index.

You do not need an *Order* Module in this case because the additional Modules in the structure ensure that the X value reaches the *Snap Value Array* before the Y value.

When you set the Array Size, you may have noticed the options Write Thru and Self-iteration, which were both checked.

- The Write Thru option means that when a new value is written into the *Snap Value Array*, the index and value are automatically sent to the Module's corresponding outputs.
- Self-iteration means that the *Snap Value Array* will automatically send all of its values when a Snapshot is recalled, or when the Ensemble is initialized.

For this sequencer, both of these options should remain checked, as they allow the Display to update when a Snapshot is loaded and when the user edits the values of the sequencer from the interface.

6.2.4 Adding Playback

The Sequencer is nearly there: the interface is in place and its values can be stored and recalled with Snapshots.

However, the sequencer is neither connected to the clock, nor is it outputting its values to the Sine Oscillator.

Unfortunately, you cannot connect the clock directly to the *Snap Value Array*, as the single Idx port will cause it to interfere with the events coming from the *Mouse Area*. So, to add playback, you'll need to create a new *Event Table*.

This *Event Table* will not be accessed by the user, and is only used to temporarily hold the sequencer values for the clock to trigger. Accordingly, you will need to remove the *Event Table* from the panel.

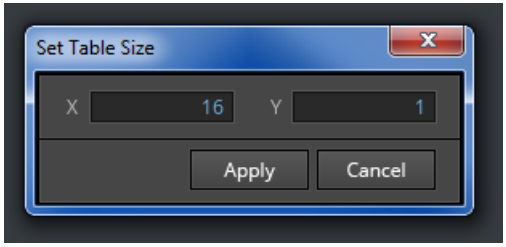
1. In the Table Macro, create an *Event Table*.
2. In the *Event Table*'s properties, navigate to the View tab.
3. At the bottom of the tab, in the VISIBLE section, uncheck the On option.

With the Event Table now hidden, you need to set it up to hold 16 values with a range of -12 to +12:

1. In the *Event Table* Properties, locate the SIZE section in the Function tab.
2. Click on the Set button.

3. A window will open, asking you to set the table size for the event table.

4. You will be creating a 16 step sequencer, so set the X value to 16 and leave the Y value at 1.



5. Click Apply.
6. Navigate to the VALUE RANGE section in the Function tab.
7. Set the Max value to 12 and the Min value to -12.

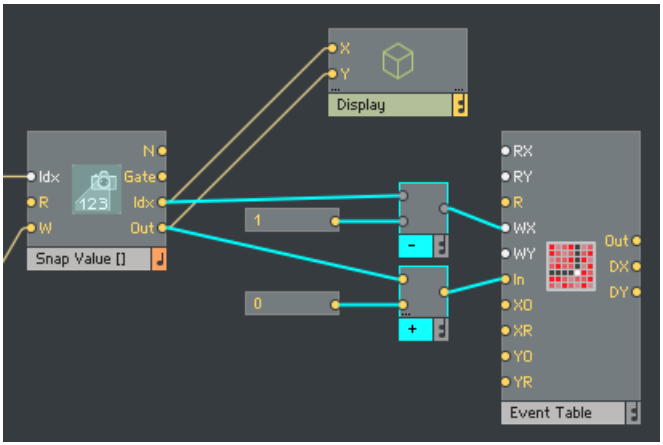
The *Snap Value Array* and the *Multi Display* use 1 indexed arrays, this means that the first array value is accessed with an index of 1. The *Event Table* uses a 0 based indexing system, so the first value is accessed with an index of 0.

When you connect the *Event Table* to the *Snap Value Array*, you will need to compensate for this.

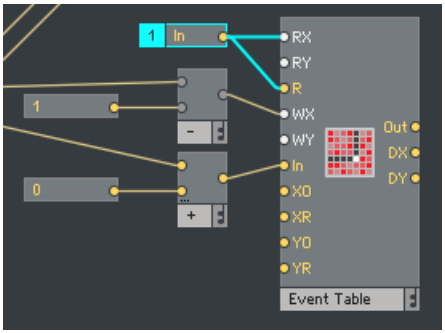
1. Create a *Subtract* Module and connect it to a *Constant* of 1 to its lower input.
2. Create another *Add* Module and connect it to a *Constant* of 0.

The -1 function will be used to adjust the index from the *Snap Value Array* to the *Event Table*. The +0 function is a dummy function used to ensure that the indices and the values have the same number of modules between the *Snap Value Array* and the *Event table*. This means that their timing will remain consistent.

► Wire these Modules into the structure as follows:

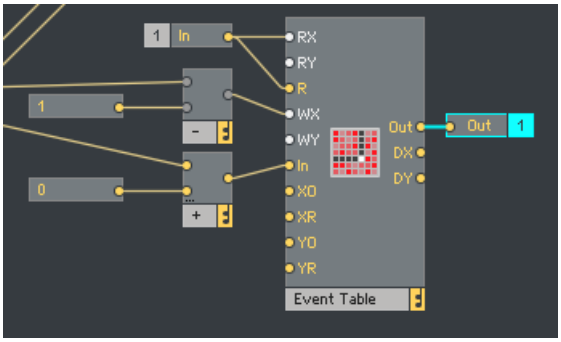


► Next, you can connect the clock so that it reads values from the *Event Table*.



An *Order* Module is not needed here. When the RX and R ports receive events at the same time, as they will do here, the *Event Table* automatically applies them in the correct order.

► Finally, connect the *Event Table* to the *Out* port of the Macro.

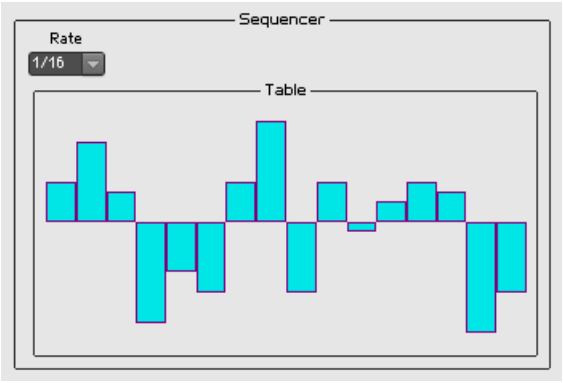


Congratulations! With the *Event Table* connected, the sequencer will be fully functional. You can now return to the Panel and use it as before.

6.3 Going Further

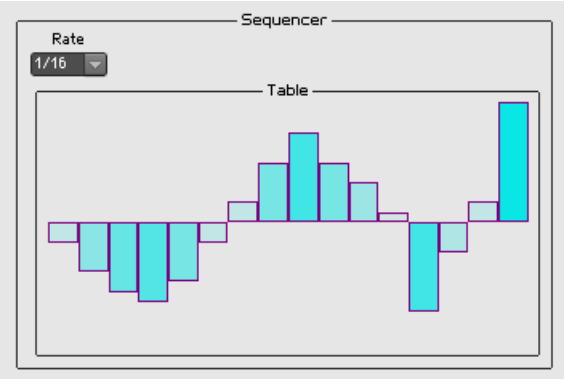
Although you only created a series of solid bars, the techniques you have learnt will allow you to create your own interface elements, with their own unique look.

For example, by using a transparent image as the background of a *Multi Display*, you can layer multiple *Multi Displays* to create more complex interfaces:



An Interface created by layering Multi Display Modules.

You can also access the color and opacity of individual objects in the *Multi Display* to create more responsive displays.



An Interface created by layering Multi Display Modules and changing the opacity of individual objects.

6.3.1 Related Factory Content

Although they tend to use *Poly Display* Modules rather than *Multi Displays*, any of the Ensembles from the Sequencers section of the Factory Library will use techniques similar to the ones you have learnt in this tutorial.

The SQX Ensemble

7 Additive Synthesizer

7.1 Overview

Using event signals in REAKTOR not only lets you control their timing, but also allows you to use a process called iteration.

> The idea of iteration was introduced in the previous tutorial with the *Snap Value Array* and its Self-iteration property.

For those of you with previous programming experience, iteration can be thought of as REAKTOR's version of a loop (with the *Iteration* Module functioning like a For Loop).

In REAKTOR, iteration can be used to generate a large number of events that are processed simultaneously, but are treated as if they came one after the other. This technique is very useful when using Modules which behave like arrays (like the *Multi Display* seen in the previous tutorial, ↑5, Basic Step Sequencer ↑5, Basic Step Sequencer)

This tutorial will teach you about iteration using the example of building an additive synthesizer.

7.1.1 Previous Knowledge

As well as assuming a basic understanding of building in REAKTOR, this tutorial also assumes that you have completed tutorial ↑5, Basic Step Sequencer ↑5, Basic Step Sequencer.

The following will not only build on the Ensemble created in that tutorial, but will also assume all knowledge acquired to that point.

7.1.2 Theory

An additive synthesizer creates sounds by combining large numbers of sine waves with different frequencies, amplitudes and phases. In the context of an additive synthesizer these sine waves are called partials or harmonics (throughout the rest of the tutorial they will be referred to as partials).

REAKTOR offers a Module created specifically for this purpose: the *Sine Bank*.

Rather than creating a large number of individual *Sine Oscillator* Modules, this one Module allows you to generate many sine waves with only a single Module. It is also optimized for this function, and is thus a more CPU-friendly option.

The *Sine Bank* Oscillator is at the heart of REAKTOR synthesizers LAZERBASS and RAZOR.

The best way to control the *Sine Bank* is to use iteration, as it can be used to set the properties of many sine waves in a single process.

For this tutorial, you will be guided through the process of building an additive synthesizer that uses 64 partials, the amplitudes of which are edited from the interface via a table similar to the one built in tutorial ↑5, Basic Step Sequencer ↑5, Basic Step Sequencer.

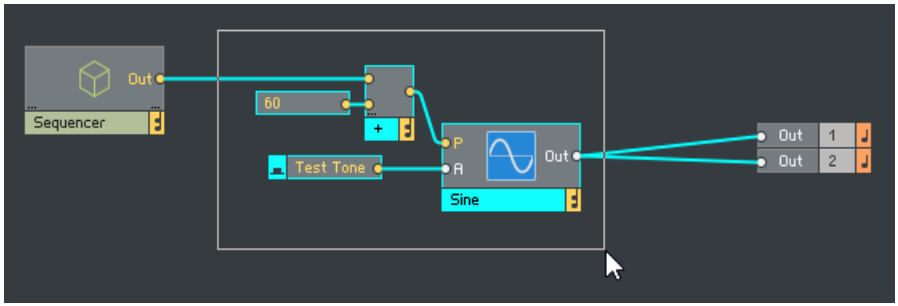
7.2 Tutorial

7.2.1 Setting Up

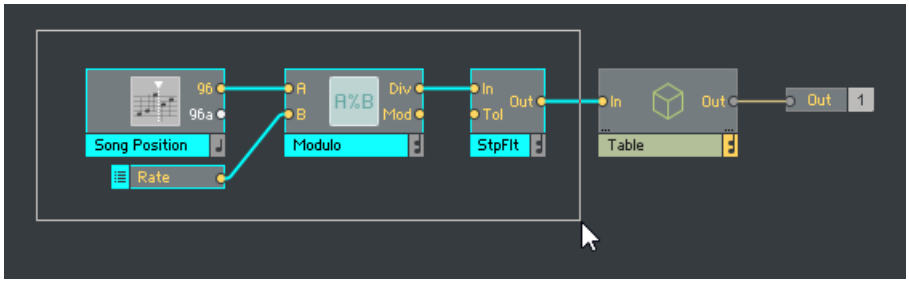
This tutorial will build on the ↑5, Basic Step Sequencer, as it already uses many of the Modules you will need. However, it also has a lot of Modules that you will not need, so the first thing to do is delete them:

1. If you do not have the Step Sequencer Ensemble loaded, open it now.

2. Enter the structure and delete the Modules highlighted below:

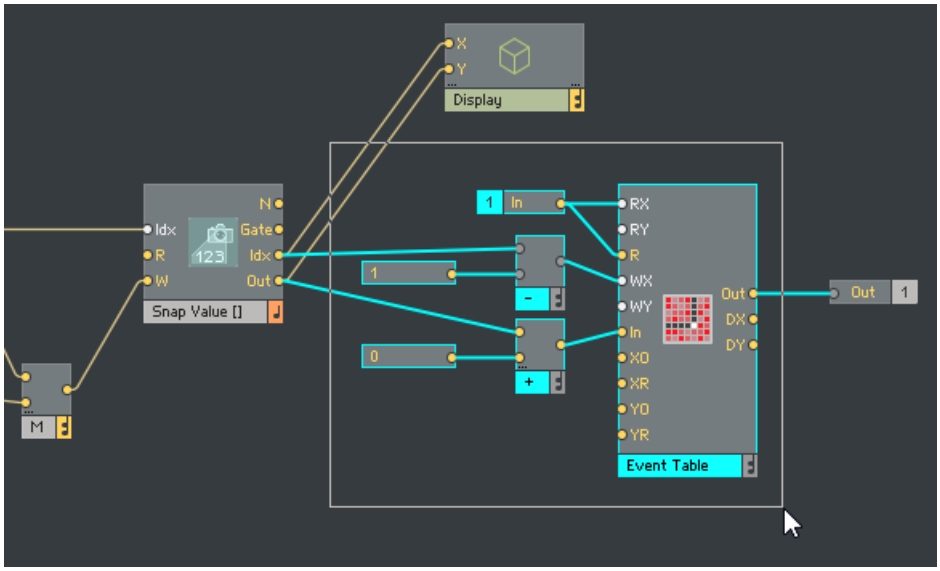


3. Enter the Sequencer Macro and delete all of the Modules that are used to generate the clock signal.



4. Rename the Sequencer Macro to "Additive Synth".

5. Enter the Table Macro and delete the *Event Table* and the Modules highlighted below:

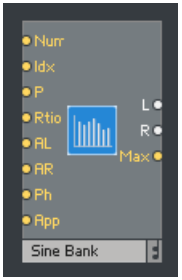


6. Rename the Table Macro to "Partials".

→ The Ensemble is now ready to be converted to an additive synthesizer.

7.2.2 The Sine Bank Oscillator

The *Sine Bank* can be a daunting Module, but it works on the same principals as the *Multi Display*, only with audio partials in place of visual objects.



The Sine Bank Module

Like the *Multi Display*, you need to select a partial using its index (Idx). With the index selected you can then set a variety of parameters for that partial. In the case of the sine bank, the available parameters are:

- Rtio: Sets the ratio of the partial. The frequency of the partial will be the main frequency (defined by the pitch (P) input) multiplied by the number set at this port.
- AL: Sets the amplitude of the partial in the left channel.
- AR: Sets the amplitude of the partial in the right channel.
- Ph: Sets the phase of the partial.

Where the *Sine Bank* differs from the *Multi Display* is the inclusion of the App (apply) port. When you set parameters for the partials in the *Sine Bank*, the changes need to be applied by sending an event to the App port before they take effect.

Although the P (pitch) input isn't set per partial, it is also not applied until an event is sent to the App port.

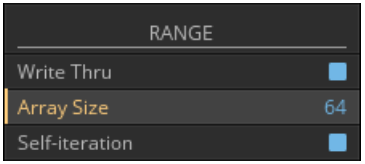
So, when creating this Ensemble, you will need to set the parameters for all of the partials, and then apply these settings. The App port will also need set every time the user plays a note, in order to update the pitch of the Module.

7.2.3 Building the Synthesizer

The first section of this tutorial for building the 64-partial additive synthesizer will explain how to update the table that you created in the ↑5, Basic Step Sequencer tutorial from 16 bars to 64 bars.

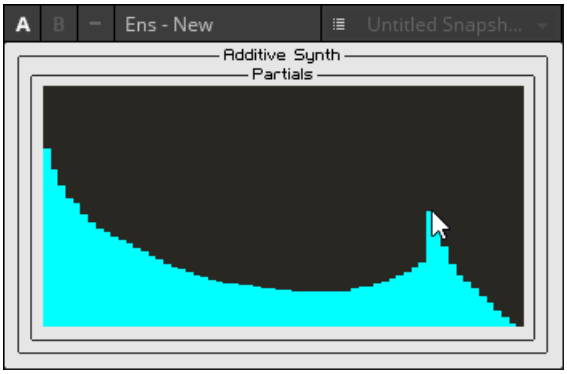
1. Select the *Mouse Area* Module to display its Properties.
2. Set the Max parameter in the RANGE X section to 64.4

3. Select the *Snap Value Array* Module to display its Properties.
4. Set the Array Size to 64



5. Enter the Display Macro and select the *Multi Display* to show its Properties.
6. In the OBJECTS section, set the Number parameter to 64
7. In the RANGE section, set the X Range to 64

If you now return to the Panel view, you will see that the table has many more bars in it, but still behaves as before.



The Table now with 64 Bars

With the interface updated, you can now connect this to the *Sine Bank* Module.

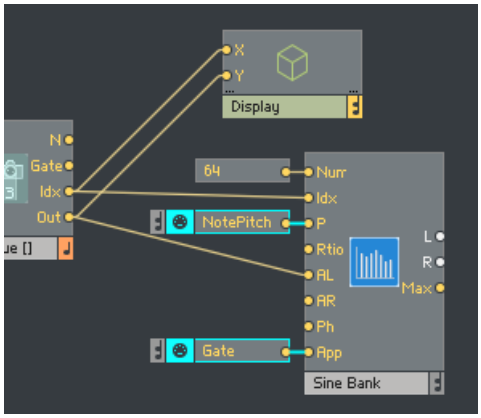
1. In the structure view, navigate to the Partials Macro.
2. Create a *Sine Bank* Module.
3. Create a *Constant* of 64 and attach it to the Num input.
4. Connect the *Snap Value Array* to the *Sine Bank* as shown below.

You only need to connect to the AL input as the additive synthesizer you are creating will be mono, and therefore only requires one output channel.

In order to make the *Sine Bank* playable, you'll need to add pitch and gate MIDI inputs.

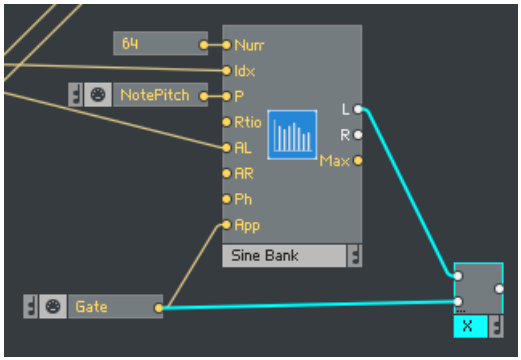
1. Create a *Note Pitch* (MIDI In) Module and a *Gate* (MIDI In) Module.

2. Connect these Modules to the *Sine Bank* as shown below.



Now the MIDI pitch input will define the pitch of the *Sine Bank*, and the MIDI gate will apply the settings. However, you still need a way of controlling the output with velocity.

1. Create a *Multiply* Module.
2. Use it to multiply the *Sine Bank* output with the Gate output, as illustrated below.



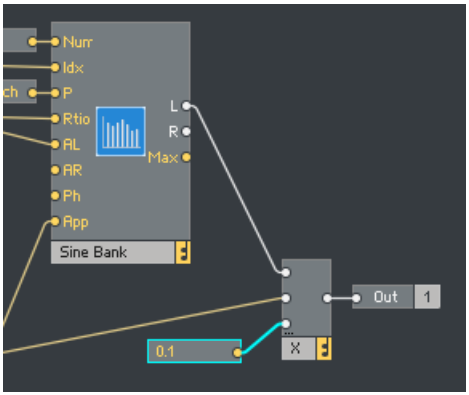
3. Attach the output of the *Multiply* Module to the Macro *Out* port.

→ When you play a MIDI keyboard, you should now be able to hear the output of the Sine Bank.

Return to the Panel and edit the table of partials. Each time you play a new note, you will hear the sound update according to the settings of the table.

The Output of this synthesizer may be quite loud, so you should adjust the output level of the synthesizer accordingly:

► Create a Constant of 0.1 and connect it to the Multiply Module at the synthesizers output.

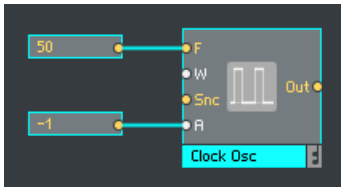


7.2.4 Adding Real-time Updates

The additive synthesizer you have built has one flaw: its sound does not update in real-time, it only updates when the user plays a new note.

To resolve this, you can add a clock that triggers the App port at regular intervals while the user's mouse is interacting with the interface.

1. Enter the structure and navigate to the Partials Macro.
2. Create a *Clock Oscillator*.
3. Create *Constants* at the F and A ports.
4. Set the *Constant* at the F Port to 50.
5. Set the *Constant* at the A Port to -1.

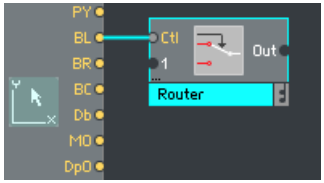


→ You now have a clock that will oscillate between -1 and 0 at a rate of 50 times per second.

The reason you want the clock value to be negative is that when you send an event higher than 0 to the App input of the Sine Bank, the phases of the partials will update. If the phases update while a sound is being played, then this will cause an audible click. Sending an event of 0 or lower to the App input will update all of the partial parameters except the phase, which should produce a less noticeable transition.

Next you will create a way to make sure the clock only updates the *Sine Bank* while the user's mouse is pressed on the table.

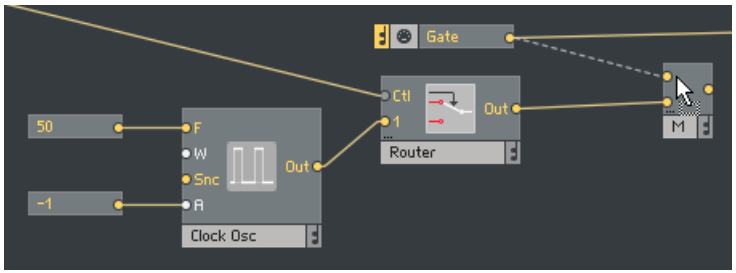
1. Create a *Router 1,2* Module.
2. Connect the BL output of the *Mouse Area* to the Ctl input of the *Router*.



The *Router* Modules act as switches that control the flow of signals. With the BL port of the *Mouse Area*, which outputs the pressed state of the left mouse button, connected to the Ctl input of the *Router*, the signal from the 1 input will only be sent to the output when the left mouse button is pressed.

1. Connect the *Clock Oscillator* to the 1 port of the *Router*.
2. Create a *Merge* Module.
3. Connect the output of the *Router* to the input of the *Merge* Module.

4. While holding the [Ctrl]/[Cmd] key, connect the *Gate* Module to the *Merge* Module in such a way that it creates another port in the *Merge* Module.



5. Connect the output of the *Merge* Module to the App input of the *Sine Bank*.

→ When you return to the Panel view, you will be able to hear the changes to the partials as you edit the values in the table.

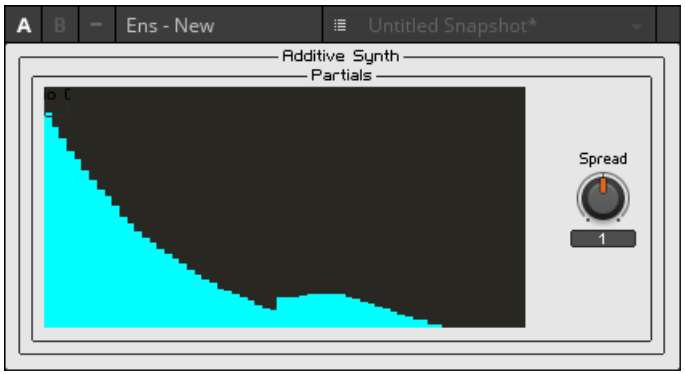
7.2.5 Adjusting the Partial Ratio Spread

By now you should have an additive synthesizer that you can edit in real-time, but you have not yet needed to use iteration.

This final section will add a knob that will alter the harmonic spread of the partials. Since this single knob will need to edit the ratio parameter of all of the partials, it will require iteration to do so.

1. Enter the structure of the Partials Macro.
2. Create a *Knob* Module.
3. In the *Knob's* properties, set the Max value to 2.
4. Re-name the *Knob* to "Spread".

5. Go to the Panel view and move the Spread knob so that it is not overlapping with the table.



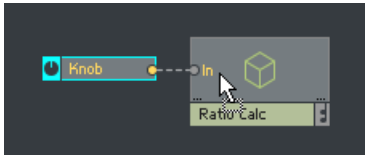
At the moment the ratios of the partials are set to their default settings, which is the Harmonic Series (partial 1 has a ratio of 1, partial 2 has a ratio of 2, and so on).

The Spread knob that you just created will be used to multiply the ratio values of the partials to control the spread of the harmonics.

The process that the knob will trigger will be rather complex. You can use a Macro to keep the structure of your synthesizer organized.

Creating the Ratio Calculation Macro

1. Create a New Macro.
2. Re-name it to "Ratio Calc"
3. While holding the [Ctrl]/[Cmd] key, connect the Spread knob to the Macro in such a way that it creates an input port in the Macro.



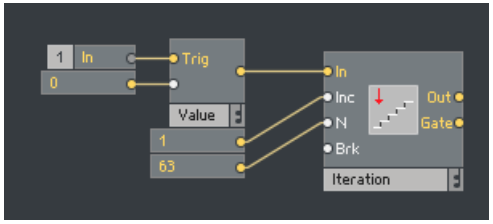
4. The Ratio Calc Macro will not have any controls, so it can be hidden from the Ensemble's Panel. Click on the Ratio Calc Macro to display its properties.
5. Click on the View Tab.

6. In the VISIBLE section, uncheck the On option.

The Iteration Loop

The process that needs to occur in the Ratio Calc Macro each time the user uses the Spread knob is a loop that generates values from 0 to 63, these values will then access the indices of the partials of the *Sine Bank*, and then also be used to calculate the ratios of the partials. You can build this with an *Iteration* Module:

1. Enter the Ratio Calc Macro.
2. Create an *Iteration* Module and a *Value* Module.
3. Connect these Modules with some additional *Constants* as illustrated below.



What will happen in the structure you have just created is the following:

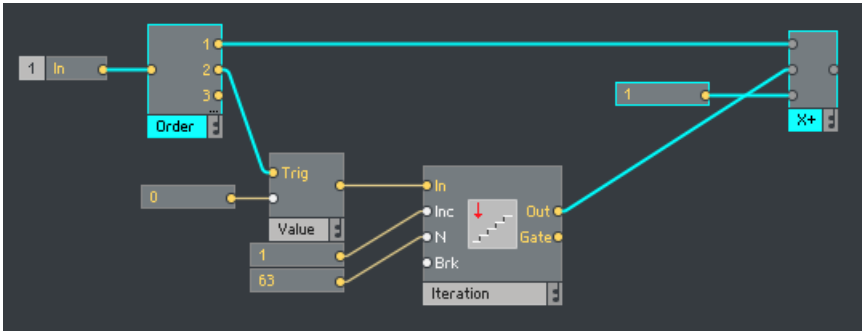
- When the Spread knob sends an event into the Macro, the *Value* Module will output an event with a value of 0.
- This event will trigger the *Iteration* Module to start its process.
- The iteration process will create 63 additional events (specified by the *Constant* at the N port).
- Each additional event will be of a value 1 higher than the last event (specified by the *Constant* at the Inc port).

So the *Iteration* Module has effectively become the loop that generates the values of 0 to 63 every time the knob is moved.

The Ratio Calculation

Now you can add the harmonic ratio spread calculation:

► Create the structure illustrated below.



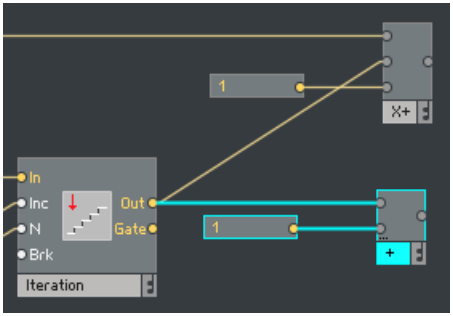
The calculation is very simple; it only needs one *Mult/Add* Module. The power of this process is that the calculation happens 64 times at an effectively instantaneous rate.

Note that you also need to add an *Order* Module. This ensures that all of the values coming from the *Iteration* Module are processed with the same value from the Spread knob.

Calculating Partial Indices

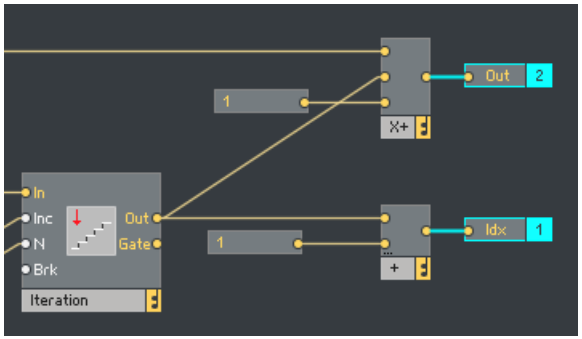
An additional calculation is needed to send the corresponding indices for the partials:

► Add the following Modules to your structure and connect them as illustrated below:



The final step in this Macro will be to create the outputs needed to send these values to the *Sine Bank*.

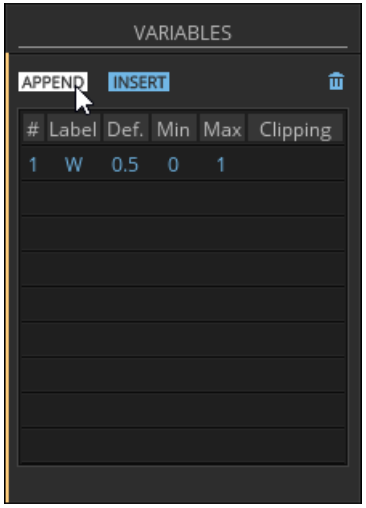
1. Create 2 *Out Port* Modules.
2. Re-name the first one "Idx".
3. Connect the *Out Ports* to the structure as follows:



Snapshot Storage

The values generated by the Spread knob will be stored in the Snap Value Array as another variable.

1. Navigate to the Partials Macro.
2. Select the *Snap Value Array* to display its Properties.
3. In the VARIABLES section, click the APPEND button.

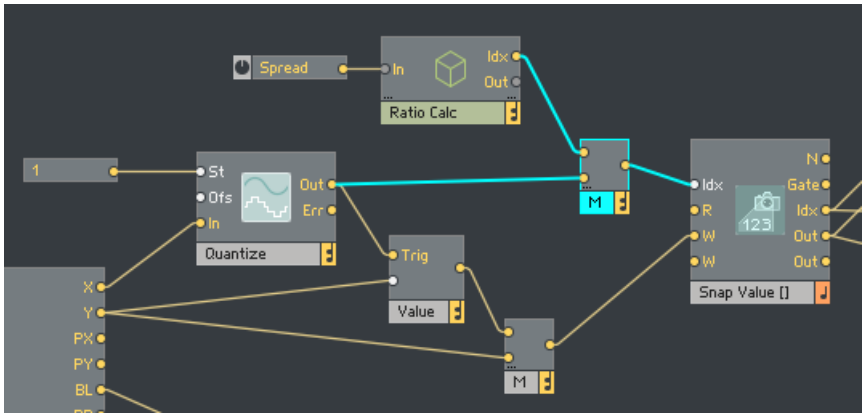


→ A new variable has been created and a new W input has been added to the *Snap Value Array* Module, coupled with a new Out port.

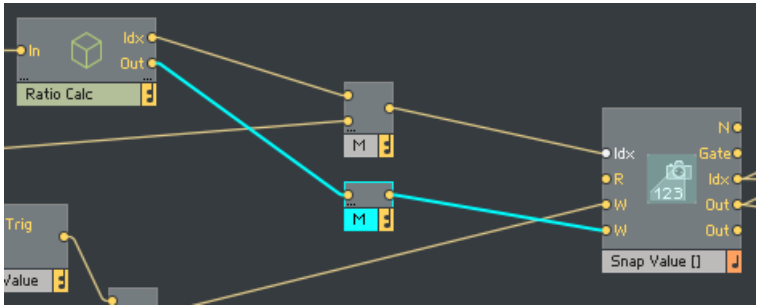
Connecting the Ratio Values to the Synthesizer

The next step is to connect the Ratio values that are generated in the Ratio Calc Macro to the *Sine Bank* via the *Snap Value Array*.

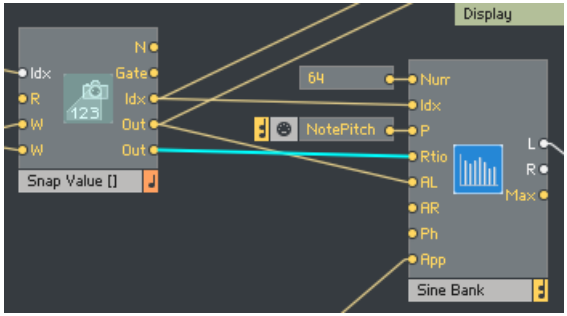
1. Connect the Idx output of the Ratio Calc Macro to the Idx input of the *Snap Value Array*, using a *Merge* Module to combine it with the current connection.



2. Connect the Out port of the Ratio Calc Macro to the second W input of the *Snap Value Array*, using a second *Merge* Module as a dummy to keep the event timings correct.



3. Connect the second Out port of the *Snap Value Array* to the Rtio input of the *Sine Bank* Module.

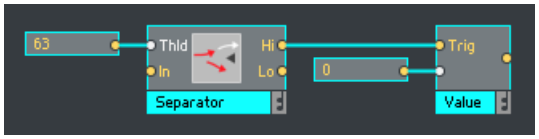


Applying the Ratio Setting Changes

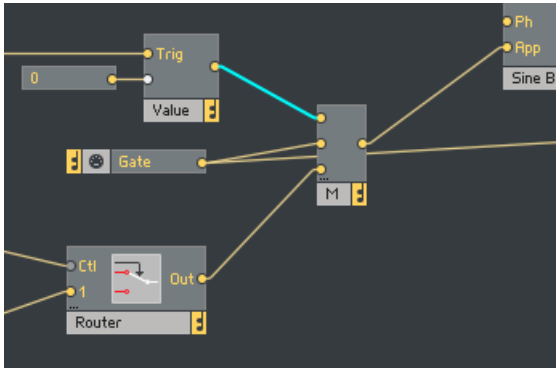
There needs to be an event sent to the App port of the *Sine Bank* in order for these changes to be heard in real-time.

The apply event should happen after the iteration has taken place. You can use a *Separator* Module to split the Idx signal from the Ratio Calc Macro so that the final Idx event from the *Iteration* Module triggers a 0 valued event to the App port.

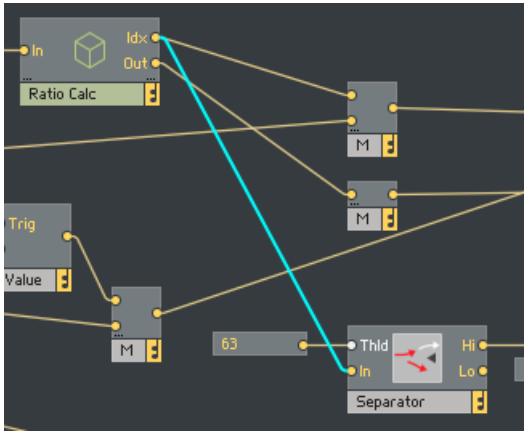
1. Create a *Separator* and set the threshold (Thld) to 63.
2. Create a *Value* Module and attach a *Constant* of 0 to its lower input.
3. Connect the Hi output of the *Separator* to the Trig input of the *Value* Module.



4. Connect the output of the *Value* Module to the *Merge* Module that leads to the *Sine Bank*'s App port.



5. Connect the Idx output of the Ratio Calc Macro to the In port of the *Separator*.



→ When the final index value (64) comes from the *Iteration* Module, it will also trigger an event with a value of 0, which will then apply the ratio changes in the *Sine Bank*.

Congratulations! You have created an additive synthesizer with controls for setting the amplitude of each partial, and for setting the harmonic spread of the partials, and all in real-time.

7.3 Going Further

The *Modal Bank* Filter Module shares many similarities with the *Sine Bank*. It has many of the same input ports and shares the same input paradigm; it simply creates a bank of resonant band-pass filters rather than a bank of sine wave oscillators.

The Modal Bank is the Module at the heart of REAKTOR PRISM and MIKRO PRISM.

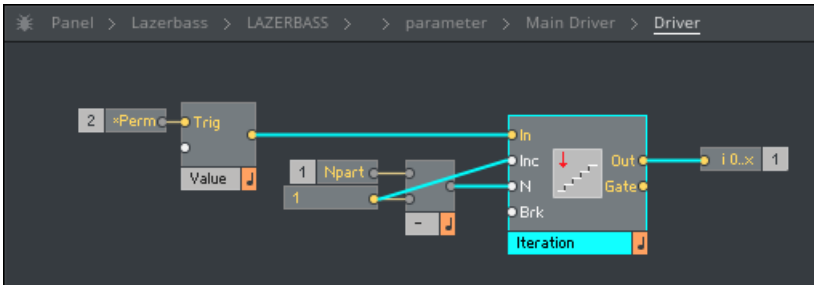
Additive synthesis and modal synthesis are both very powerful. By adding more calculations to the iteration process, you can create complex sounds that are not possible with any other forms of synthesis.

Try adapting your additive synthesizer to use the *Modal Bank*. Note that the *Modal Bank* requires an "exciter" signal (like a short click) to produce sound.

7.3.1 Related Factory Content

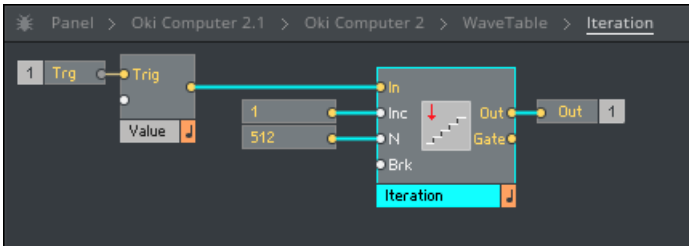
The *LAZERBASS* synthesizer uses a *Sine Bank* as its main sound generator. Almost all of the available controls on the interface apply calculations to parameters of the partials in the *Sine Bank*.

Although it is a large structure, you will be able to find *Iteration* Modules driving many of the parameter calculations.



An Iteration Module inside LAZERBASS.

As an example of how iteration can be used outside of additive synthesis, it is also used in *Oki Computer 2* to drive the calculations that create the wavetables.



The Iteration Module that drives the Wavetable creation.

8 Drag and Drop Sampler

8.1 Overview

Table References are a signal type that facilitates flexible and efficient sharing of data in the structure. Modules and Macros that make use of Table References are part of the Table Framework.

Reference information about the Table Framework can be found in section ↑16, Table Framework.

In the following tutorial, you will use Macros that are part of the Table Framework to create a drag and drop sampler with a waveform display. You will also look at the structure of these Macros to learn how Table References can be used.

This tutorial will also introduce Core Cells and how they interact with the Primary level of REAKTOR.

8.1.1 Previous Knowledge

This tutorial assumes you have read the tutorials up until this point. You should already know how to navigate the Structure View and should be familiar with Event processes.

8.1.2 Theory

A table is a two-dimensional array of data, and Table References allow you to access this data anywhere in the structure. These properties make the Table Framework ideal for working with samples.

Table References exist in the Primary level of REAKTOR, but are utilized by Core Cells.

In this tutorial, the Table Reference will be an audio sample. The Ensemble you create will use the Primary structure to load the sample, which will then have its data sent to Core Cells for playback and to calculate the display.

8.2 Tutorial

8.2.1 The Drag and Drop Area

To start, you will need a blank canvas:

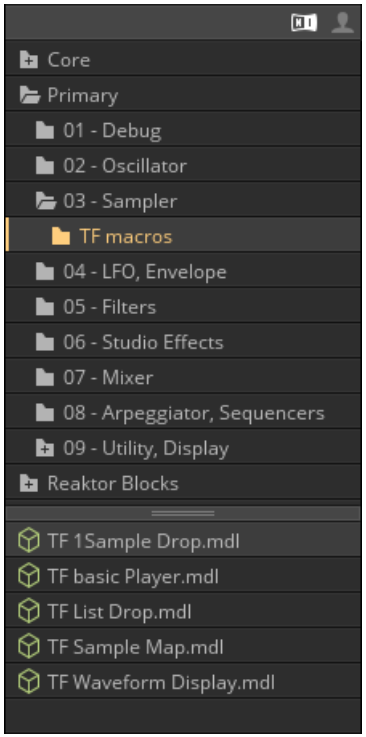
► Create a new Ensemble.

All of the Library Macros that make use of the Table Framework can be found in the **03 - Sampler** section of the Library. In this section is a sub-section called **TF-macros** that contains a number of building-blocks for creating your own samplers.

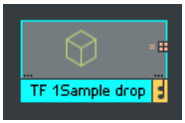
As a first step, you will create the Drag and Drop Area:

1. Open the Library Browser in the Side Pane.
2. Open the Primary folder, followed by the 03 - Sampler folder.

3. Click on the TF macros folder to display its contents in the lower half of the Side Pane.



4. Drag the *TF 1Sample Drop* Macro into the Structure.



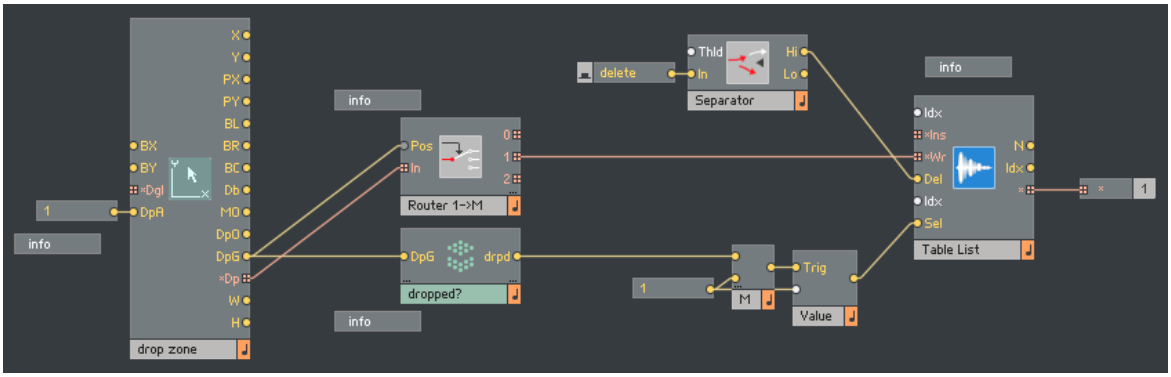
The TF 1Sample drop Macro

Looking at the Macro you have just loaded, you will notice a new port type. This kind of port is a Table Reference port and is used to send Table References (in this case audio sample data) around the Structure.

Table References do not send signals with single values in the same manner as Audio or Event ports, but rather send a reference to a table of data.

If you think of a sample as a table of data that is store in RAM, the Table Reference is a pointer to that data.

► Enter the *TF 1Sample drop* Macro by double-clicking on it.



The Structure of the TF 1Sample drop Macro

In the Structure of the Macro you will notice a number of Modules called info. These Modules contain information regarding the Modules to which they are closest.

To view the information in these Modules:

1. Click on the Module to select it.
2. Open the Properties View in the Sidepane.
3. Click on the Info tab.

→ The information in the Module will be displayed in the Properties.

While the Info tab is open, you can click on other Modules and the information will update accordingly.

As a brief overview of the Macro there are three important Modules:

The Mouse Area

The *Mouse Area* Module (renamed here to drop zone) is used to define the area where the user can drop an audio file.

- The 1 at the DpA input tells the Module that it should accept audio files for drag and drop.
- The *Dp (Drop) output sends the Table Reference whenever an audio file is dragged over the Mouse Area and whenever it is dropped.
- The DpG (Drop Gate) output is used to send the drag and drop state of the audio file.

The values from the *Mouse Area* are then sent to the next important Module...

The Router 1->M

Since the *Dp output sends Table References when an audio file is dragged over the *Mouse Area*, as well as when it is Dropped, the *Router 1->M* Module is used to control which *Dp Table Reference is sent to the rest of the Structure.

1. The *Router* only allows a *Dp Reference through when the value of the DpG is at 1.
2. The DpG outputs a 0 value generally, a 2 when an audio file is dragged over it, and a 1 when it is dropped.
3. Thus, the *Router* blocks the *Dp References until the file is definitely dropped on the *Mouse Area* by the user.

The Table List

The *Table List* is the Module that holds the Table data.

As the name suggests, it can hold several Tables in a list. The different Tables are selected using indices. But in this instance, it only holds a single Table, so the indices are left blank.

- The Table Reference is sent from the *Router* into the *Table List*, which stores it.
- A Core Cell (named dropped?) is used to trigger the output of the Table Reference from the *Table List* by sending a value to the Sel input of the *Table List*.
- The Table Reference is then sent out of the Macro and into the rest of the Structure.
- Note that an Event will also be sent to the Sel input during initialization because of the *Constant* attached to the *Merge* Module. This will ensure that the Table Reference will be sent to the Structure during initialization.

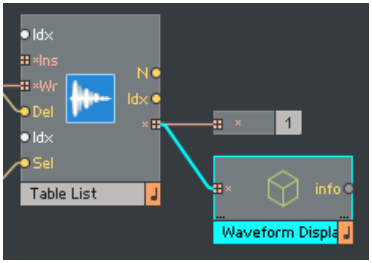
8.2.2 The Waveform Display

For the purposes of layering, you need to place the Waveform Display Macro inside the drop Macro.

► Insert the *TF Waveform Display* Macro into the Structure of the *TF 1Sample drop* Macro.

The *TF Waveform Display* has an input for Table References. It then uses the Table data to draw a waveform using a *Multi Display*.

► Connect the Table Reference output of the *Table List* to the Table Reference input of the *Waveform Display* Macro.



The Structure of the *Waveform Display* is more complicated than the *1Sample drop*, but the basic theory is that when a Table Reference is sent to its input, an iteration process is triggered. This process reads the data in the Table which is then converted it into signals that can be used by a *Multi Display* Module.

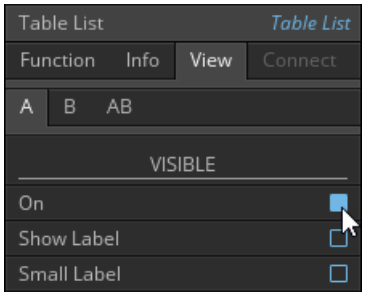
This is the same basic principal behind the Step Sequencer from section ↑6, Advanced Step Sequencer.

Next, you need to align these Macros on the Panel.

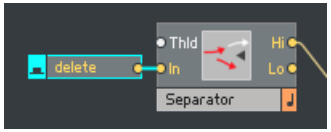
You do not need the *delete* button or the Table List on the Panel, so they should be hidden.

1. Select the *Table List* Module.
2. Open the View tab of its Properties.

3. Deactivate the On option in the VISIBLE section.



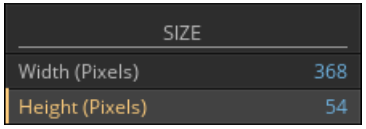
4. Repeat this process with the *delete* Button.



Resizing the Mouse Area

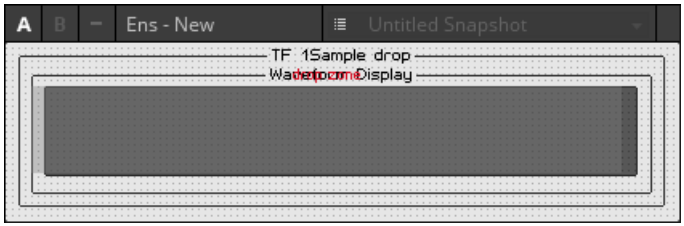
Next, you will enlarge the *Mouse Area* so that it covers the whole Waveform Display.

1. Select the *Mouse Area* Module.
2. Open the View tab of its Properties.
3. Set the Width (Pixels) parameter to 368.
4. Set the Height (Pixels) parameter to 54.



→ The *Mouse Area* is now the same size as the Waveform Display.

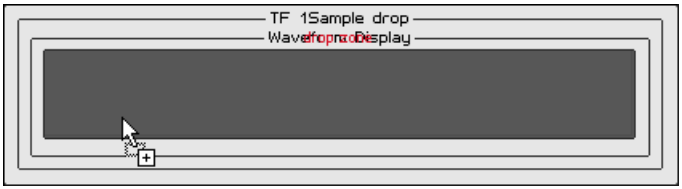
1. Unlock the Panel to see the Mouse Area as a grey shadow.



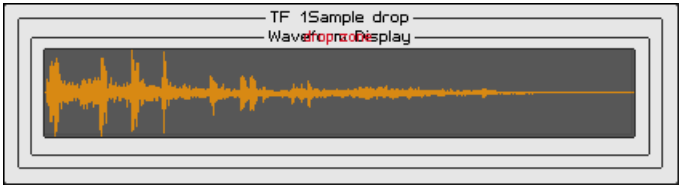
2. Move the Mouse Area until it covers the Waveform Display.
3. Lock the Panel.

The Panel elements of the Drag and Drop Sampler are complete.

► Drag and drop an audio file (wav or aiff) onto the Mouse Area.



→ The Waveform of the audio file will be displayed on the Panel.

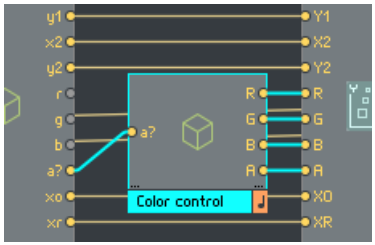


Changing the Waveform Color

It is possible to change the color of the display by opening some hidden controls.

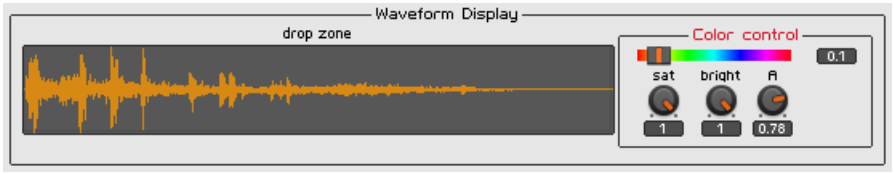
1. Enter the *Waveform Display* Macro Structure.

2. Select the *Color control* Macro.



3. Open the View tab of its Properties.
4. Activate the On option in the VISIBLE section.

→ Color controls will appear on the Panel.



1. Edit the color of the waveform using these controls.
2. When you are finished, return to the View tab of its Properties and deactivate the On option in the VISIBLE section.

8.2.3 Playback

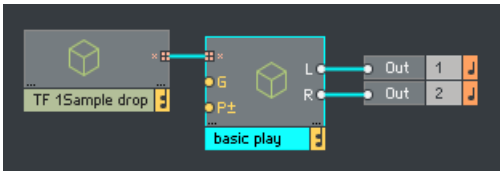
The Sampler can load samples and it can display their waveforms, but it cannot yet play the samples. You will need one final Macro for that.

1. Navigate to the top level of the Ensemble.
2. In the TF macros sub-section of the Library, load the *TF basic Player* Macro.

The TF basic Player Macro

The Basic Player plays the audio data from a Table when it receives a gate signal.

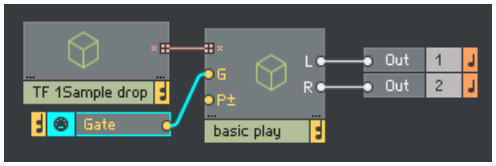
► Connect the *TF basic Player* Macro as shown below.



→ The Table Reference from the Table List in the *TF 1Sample drop* Macro will feed into the *TF basic Player* Macro.

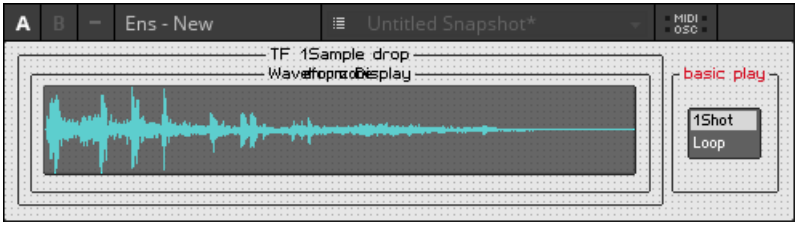
Next, you need to connect a gate to the *TF basic Player* Macro so that the user can trigger playback of the sample.

1. Create a *Gate (MIDI In)* Module.
2. Connect it to the G input of the *TF basic Player* Macro.



→ Play a MIDI key to trigger the sampler.

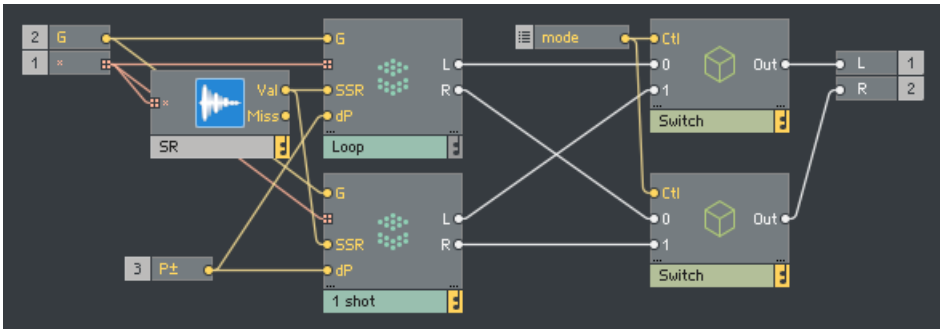
► In the Panel View, move the basic play Macro so that it is not obscuring the Waveform Display.



Basic Player Structure

Now is a good time to take a look into the Structure of the basic Player Macro.

► Double-click on the *basic Player* Macro to enter its Structure.



The Structure of the basic Player Macro

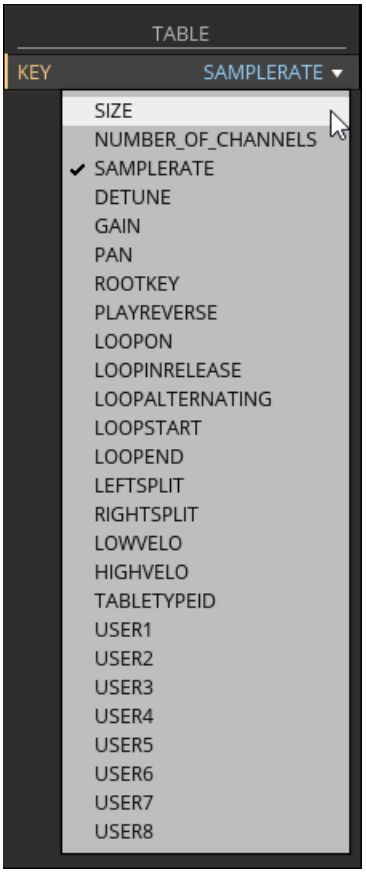
There are two Core Cells in this Structure, one called Loop and one called 1 shot, each one handling a different mode of sample playback.

This tutorial will not cover the contents of the Core Cells, their contents would be close to meaningless without some understanding of Core (read the REAKTOR 6 Building in Core document to learn about building Core Cells).

The Module that will be explored is the *Table Info* Module (renamed in this Structure to SR).

The *Table Info* Module reads meta-data from a Table and exports that data as an Event value. In this Structure, the *Table Info* Module is retrieving the sample rate (SR) of the audio file and sending that information to the Core Cells.

► The meta-data that the *Table Info* Module retrieves is set in its Properties.



8.3 Going Further

The Table Framework allows you to manipulate audio samples in your own custom structures, but it requires Core Cells for this, and so now is a good time to jump into the Core level of REAKTOR (see section ↑15, An Introduction to Core).

You can also read section ↑16, Table Framework for more detailed information about the Table Framework.

Related Factory Content

Everything in the **03 - Sampler** section of the Library uses the Table Framework. You can load in any of the Samplers and check out their Structures.

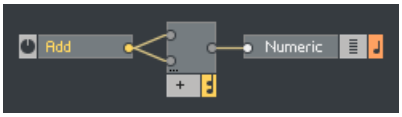
As a starting point, however, it is advised that you begin by looking into the other **TF macros**. These Macros are annotated with *info* Modules and are much easier to navigate and read.

9 A Quick Guide to Initialization

Initialization was mentioned briefly in section ↑2.5.4, Event Initialization Order ↑2.5.4, Event Initialization Order. It is the process that occurs immediately after you load an Ensemble.

Initialization is particularly important for Events signals and Event Modules, especially in complex structures. It is important to be mindful of the initialization process - an Ensemble that works fine while it is being built, may experience odd behavior when it is closed and re-loaded.

As an example, in normal circumstances the *Add* Module sends an Event when an Event arrives at either input port. Even in the example shown below, the *Add* Module will output two Events when the *Knob* sends one, although they will happen almost simultaneously.



However, during initialization, the *Add* Module will preemptively look at all the values at its inputs and send a single Event from its output port.

Initialization of the Order Module

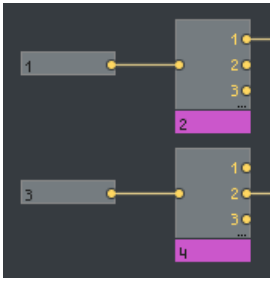
The *Order* Module exhibits special behavior during initialization.

You may have already used the *Order* Module in the earlier tutorials, but they did not explain initialization.

The *Order* Module only sends an Event from its 1 output port during the actual initialization procedure. The Events from the 2 and 3 output ports come later.

It is also important to note that this affects Modules that come before the *Order* Module.

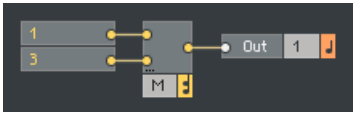
With the *Show Event Init Order* option active, you can see this in effect:



Initialization of the Merge Module

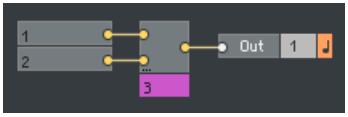
Another Module that has special behavior at initialization is the *Merge* Module. As with the *Add* Module example, it sends only one Event during initialization. As a result, the *Merge* Module cannot behave like it does during normal operation, that is, to forward all Events to its output. It only forwards the Event from the bottommost input port to its output port.

In the Structure shown below, the *Merge* Module sends the value “3” to its output during initialization.



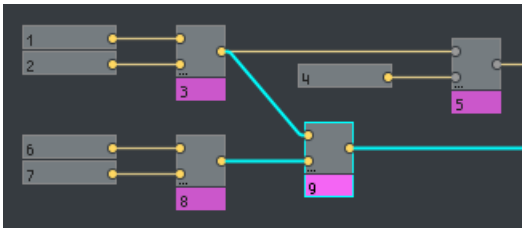
Another special property of the *Merge* Module is that its ports determine the initialization order of the Modules connected to them.

The rule is that the Modules are always initialized from top to bottom. So if you look at the example Structure again with the *Show Event Init Order* option active, you will see this illustrated:



Even though the *Merge* Module will only let the Event from the bottommost input port through, the initialization order can affect other parts of the Structure as well if the Modules lying upstream of the *Merge* Module have other connections as well.

In the following structure, the *Merge* Module is highlighted and *Show Event Init Order* is active.



It shows that everything connected to the top port of the *Merge* Module gets initialized first, and then the Modules connected to the lower port are initialized.

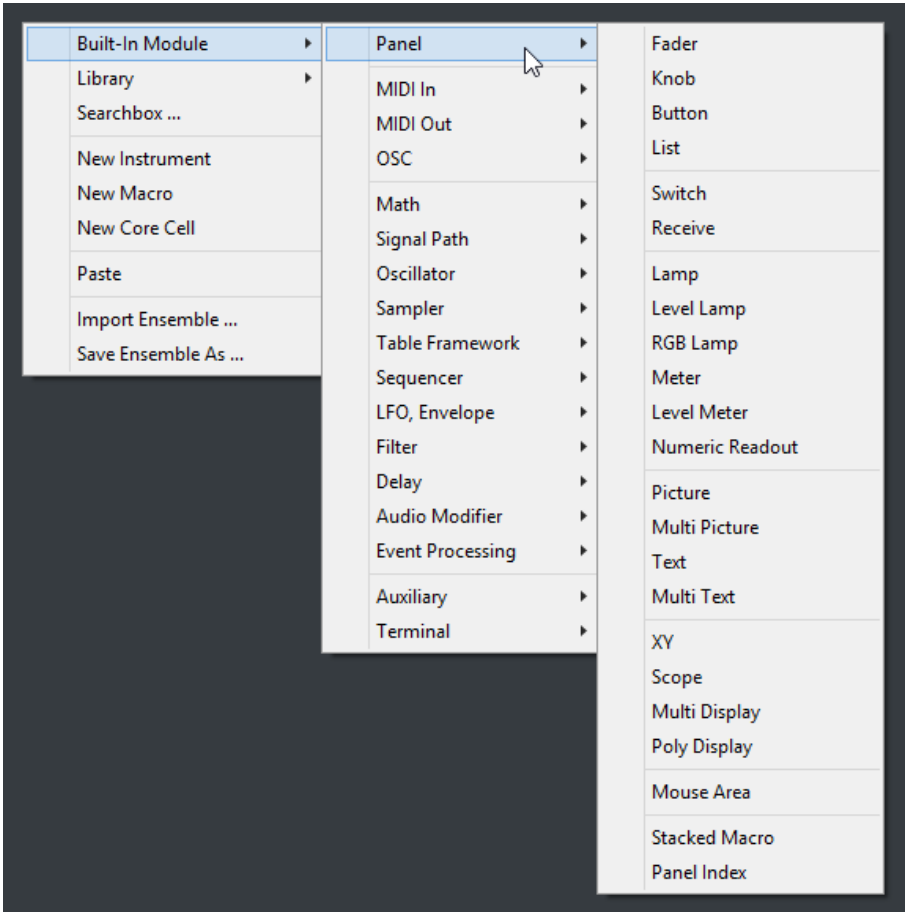
For simple structures, this behavior should not cause any problems, but should issues arise with the timing of Events during initialization, this is something to look out for.

10 Creating and Customizing Interfaces

REAKTOR gives you several ways to design and customize the graphical user interface (GUI) of your creations, from simply setting a custom color scheme to importing your own custom skins for controls. This section will cover all topics in this area.

10.1 Overview of Panel Elements

Any Module designed for use on the instrument panel can be found in the *Panel* section of the *Built-In Modules*.



The Panel Modules.

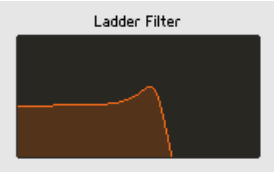
There are many Panel Modules, all of which are listed in the Module Reference in section ↑18.5, Panel ↑18.5, Panel.

There are 3 main types of Panel Modules:

- **Simple Controls** like *Knobs* and *Buttons*. These Modules are controls designed for common uses. *Knobs* and *Buttons* appear on many instrument and effect interfaces, and these Modules are the quickest and easiest way to create these controls.

- **Simple Displays** like the *Level Meter*. These Modules display information, but are not a control. In other words, they have inputs but no outputs.
- **Elements** like the *Mouse Area* or *Multi Display*. These Modules generally cannot be used on their own; they need to be combined with other modules to create a control or display. They allow for more flexibility, but are thus more complicated to use.

Some other Modules also have a display option. For example, the Filter Modules can be set to visible in their properties to display a frequency curve.



The Panel Display of the Ladder Filter.

The following categories of Modules have optional panel displays:

- **Samplers** display the waveform of the currently playing sample.
- **Envelopes** display the envelope plot.
- **Filters** display the frequency response of the filter.
  - This includes EQ Modules, but excludes the Modal Bank, Differentiator, and Integrator.
- The **Event Table** and **Audio Table** display the contents of the tables.

10.2 Customizing the Header

At the top of every new Ensemble or Instrument you create is the Header.

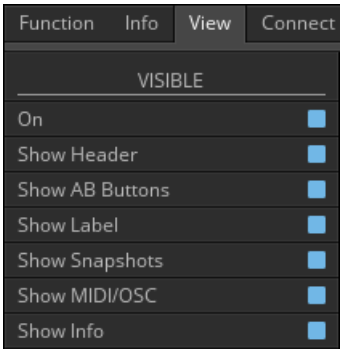
The Header of a New Ensemble.

By default it contains the following (from left to right):

- A/B view and minimize buttons.

- The Instrument/Ensemble name.
- Snapshot menu.
- MIDI and OSC input/output indicators.
- Polyphony voice display.

You can customize the contents of the Header in the View tab of the Properties.



The Header Display Options.

In the VISIBLE section of the View tab there is a list of checkboxes, each one controlling the visibility of each of the elements listed above.

- Show Header: Toggles the visibility of the entire Header on/off.
- Show AB Buttons: Shows/hides the A/B and minimize buttons.
- Show Label: Toggles Shows/hides the Instrument/Ensemble name label.
- Show Snapshots: Shows/hides the Snapshot menu.
- Show MIDI/OSC: Shows/hides the MIDI and OSC input/output indicators.
- Show Info: Shows/hides the Voices display.

If the number of Voices is set to 1, the Voices display is automatically hidden.

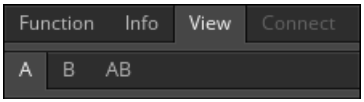
10.3 A/B Views

Each Instrument created in a REAKTOR Ensemble can have two panel views, labelled A and B.

► You can locate the A/B view controls in the Instrument Header.

The A/B views can be very useful for creating a main interface panel and something like an options or settings panel.

In every Module or Macro with a panel element, in the View tab of its Properties you will see 3 tabs: A, B, AB.



The A/B View Tabs.

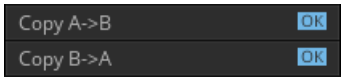
These allow you to give each Module a different set of panel properties for each panel view.

- Selecting A means you will be editing the values for the A view only.
- Selecting B means you will be editing the values for the B view only.
- Selecting AB means you will be editing the values for both the A view and the B view.

You have many options to completely change the look and position of a control between the A and B view. You can even make a control only visible in one view.

In some cases you may want the A and the B view to be the same. In this case you can copy the settings for all panel elements from one view to the other from the Instrument Properties:

1. Enter the Instrument Properties and select the View tab.
2. Find the COPY section at the bottom of the View tab. It holds two functions: Copy A->B and Copy B->A.

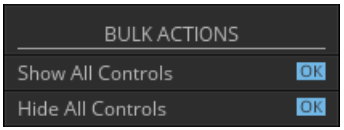


3. Click on the OK button beside the function you wish to perform.

→ The View settings from one panel will be copied to the other.

Every individual Module/Macro also has this option, so you can perform this function on a single control or display.

The View tab of the Instrument/Ensemble Properties also gives you BULK ACTIONS to hide or show all controls.



The Bulk Actions.

You can use these to hide all controls on the B view, and then go through the structure and only make a handful of Modules visible on the panel.

10.3.1 Examples in the Factory Library

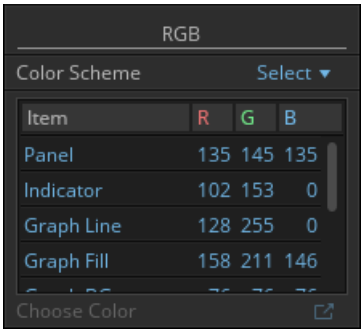
Some of the Ensembles in the Factory Library make use of the A/B Views. Here are a handful of examples:

- The *Steam Pipe 2* and *SubHarmonic* synthesizers both use the B view to display the reverb effect controls.
- The *Green Matrix* synthesizer uses the B view to display the modulation matrices.

10.4 Changing the Color Scheme

Any control with a built-in panel display can have its colors customized at the Ensemble/Instrument level. This means that for any Instrument in the Ensemble, you can set the indicator color for all knobs to a single setting, but you cannot set the color of an individual knob.

The Color Scheme settings are located in the View tab of the Instrument Properties in a table labeled RGB (Red Green Blue).



The RGB Controls.

Here, you can select the color of the following elements:

- Panel: The main background color of the Instrument.
- Indicator: Sets the color of the indicator of the knobs and sliders, as well as the on state of the buttons and switches.
- Graph Line: Sets the color of lines drawn in any element that uses a graph or plot (eg. Filter curves and Tables)
- Graph Fill: Sets the color of the fill area under the Graph Line.
- Graph BG: Sets the background color of the graph.
- Grid: The Event and Audio Tables can have a value grid overlaid on top of the graph. This parameter selects the color of the grid.

The following parameters only apply to Tables which have their Graph property set to *2D Color*:

- 2D Table Min: Sets the color to represent the minimum value.
- 2D Table Max: Sets the color to represent the maximum value.
- 2D Table Default: Sets the color to represent the default value (0).

10.4.1 Editing Colors

You can specify colors for any of the available elements in the table in two ways:

- Entering RGB values.

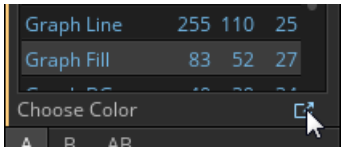
- Selecting a color using the Choose Color window.

To manually enter an RGB value:

1. Double-click on the value you wish to edit.
2. Enter a value between 0 and 255.
3. Click away or press [Enter] to apply the changes.

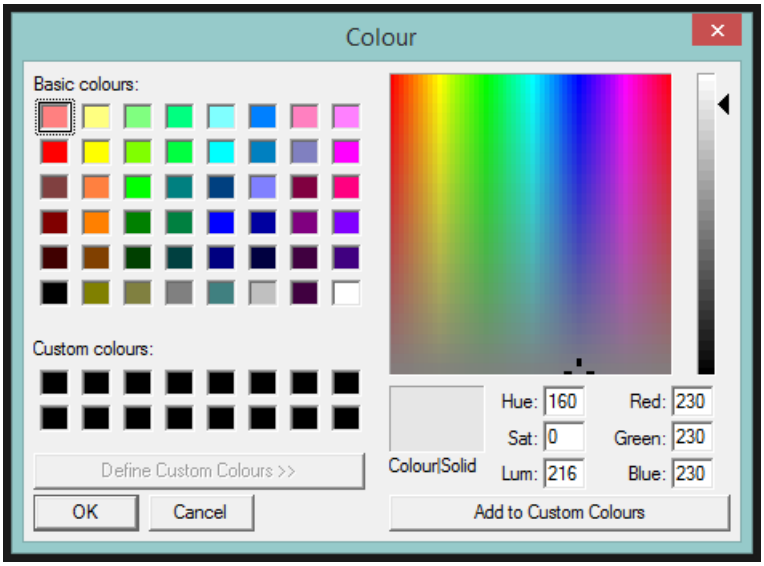
To select a color using the Color window:

1. Select the element you wish to edit from the RGB table.
2. Click on the open window icon next to Choose Color below the RGB table.



→ The Color window will open.

The Color window displays a number of options that allow you to select colors in a more intuitive way than entering numbers into a table.



The Color Window as it appears on the Windows Operating System

► Select a color of your choice and confirm the selection by clicking OK.

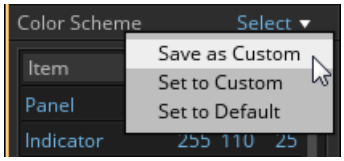
→ The values in the RGB table will update to those of the chosen color.

10.4.2 Saving a Custom Color Scheme

REAKTOR allows you to store a custom color scheme that you can store and recall for future creations.

Once you have created a color scheme you like you can store it by following these steps:

1. Open the Select menu next to Color Scheme.
2. Select Save as Custom.



→ The current color scheme will be stored as the Custom color scheme.

To recall the Custom color scheme:

1. Open the Select menu again.
2. Select Set to Custom.

→ The color scheme for the Instrument will be set to the last color scheme you saved as the Custom color scheme.

10.5 Skinning with Custom Images

Many of the Panel Modules can have their look altered by importing image files to skin the control or display.

You can also add custom images as the background of the Instrument.

10.5.1 Image Formatting

REAKTOR can accept BMP, PNG or TGA file formats for images.

It is not possible for REAKTOR to process images (for example: rotating them or adjusting their transparency) and so all frames of an animation should be rendered as images and then stitched together to form one long image, like a reel of film.

The frames of the animation can be arranged either horizontally or vertically, with the lowest value state in the 1st frame to the left/top of the file. As an example, an animation for a knob could look something like this:

An example of a knob animation in a filmstrip style image.

When an image like this is loaded into REAKTOR, you must specify the number of frames, and then REAKTOR can divide the image accordingly.

Faders are an exception as they can accept a single image, which is then used to replace the fader handle.

For creating knob animations like the one above, you can use a free program called KnobMan by g200kg.

Formatting Images for Buttons

Buttons can accept 2 or 4 state animations. A 2-state animation will give you the basic on/off states, but a 4-state animation will also allow you to customize mouse-pressed states.

A 2-state button animation will have its frames interpreted like this:

- Frame 1 = off
- Frame 2 = on

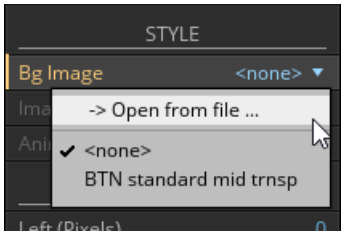
A 4-state button animation will have its frames interpreted like this:

- Frame 1 = off
- Frame 2 = on
- Frame 3 = off, mouse pressed
- Frame 4 = on, mouse pressed

10.5.2 Importing an Image

As a first example, you can import an image as a custom background for an Instrument:

1. Select the Instrument.
2. In the Properties window, select the View tab.
3. In the section labelled STYLE, open the Bg Image menu (which should currently display the text *<none>*)
4. Select *-> Open from file ...*



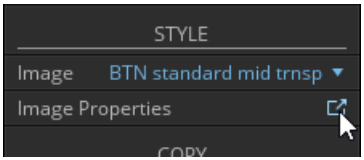
5. A dialog window will open allowing you to browse your file system for BMP, PNG and TGA image files.
6. Locate and select an image and click Open.
7. The Image Properties window will open, but you can ignore this for now.
8. Click Ok at the bottom of the Image Properties window.

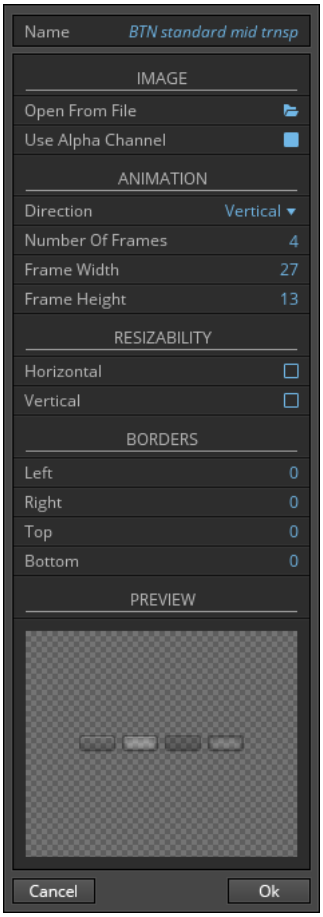
→ Your image will now be displayed as the instrument background.

10.5.3 The Image Properties Window

When loading an image for the first time, you will be presented with the Image Properties window. This is where you tell REAKTOR how to read the image file.

- You can also edit the properties of an image that has already been imported by clicking on the open window icon next to Image Properties in the STYLE section of the Properties' View tab.





The Image Properties Window

The Image Properties window has the following options:

- Name: By default the name of the imported image will be the name of the file you loaded, however you can change this by entering new text into this area.
- IMAGE:

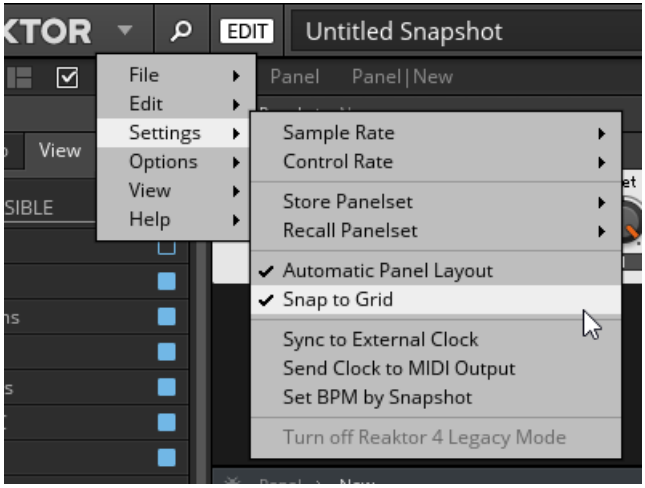
    - Open From File: If you have made adjustments to an image that is used in multiple places throughout an Instrument, you do not have to replace each one individually. You can use this option to open an image file to replace whichever image you had selected when you opened the Image Properties window.
    - Use Alpha Channel: This checkbox tells REAKTOR whether or not the imported image uses an alpha channel (i.e. transparency).
- ANIMATION:
    - Direction: Tells REAKTOR how the frames of the animation are orientated.
    - Number Of Frames: You can enter the number of frames in the animation here. If the imported image is not an animation, this should be set to 1.
    - Frame Height: Displays the pixel height of each frame in the animation. If the animation is vertical, you can manually enter the height of each frame and REAKTOR will calculate the number of frames.
    - Frame Width: Displays the pixel width of each frame in the animation. If the animation is horizontal, you can manually enter the width of each frame and REAKTOR will calculate the number of frames.
- RESIZABILITY: Certain Panel Modules can be resized by REAKTOR (for example, Macros and the backgrounds of the Multi Display and Poly Display). This section contains checkboxes to tell REAKTOR if the image is permitted to be resized in such circumstances.
- BORDERS: If an image is set to be resizable in any way, you are able to specify borders which will not be resized. This is especially useful for any image used as a background.
- PREVIEW: This area displays a preview of the image after it has been processed by REAKTOR.

10.6 The Panel Grid

By default, REAKTOR places controls on a 4 pixel grid. This can be useful for creating clean layouts with the built-in panel elements, but when working with custom skins, this restriction could be undesirable.

To turn the 4-pixel grid off:

1. Click on Settings in the menu bar.
2. Click on *Snap to Grid* to uncheck this option.



→ You will now be able to place controls anywhere on the panel.

Most controls have their position on the grid calculated from the top left corner of the element; however the Knob element is measured from its center.

10.7 Layering Panel Elements

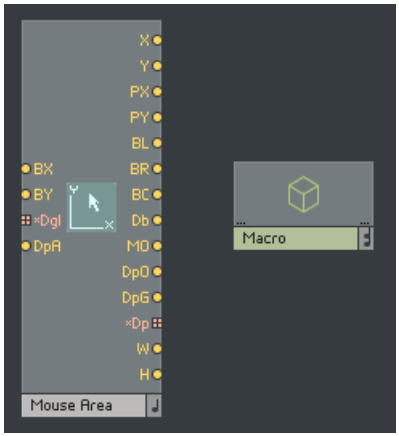
Sometimes it is necessary to layer one panel element on top of another; a common example would be placing a *Mouse Area* Module on top of a *Multi Display*.

When layering panel elements, REAKTOR uses the Macro order to decide which element should be placed on top of the others.

So, in the example case of placing a *Mouse Area* on top of a *Multi Display*, you would need to:

1. Create a Macro and place a *Mouse Area* Module inside it.

2. Create a new Macro inside the current Macro.



3. Place a *Multi Display* inside the new Macro.

→ Because the *Multi Display* is contained in a Macro one level below the *Mouse Area*, it will be placed under of the *Mouse Area* on the panel.

If you do not use Macros to order the Modules, REAKTOR will give top level priority to whichever Module is currently selected. This means that the panel element on top could change during use, which is often undesirable.

This technique is used in the ↑6, Advanced Step Sequencer ↑6, Advanced Step Sequencer tutorial.

10.8 Stacked Macros

Stacked Macros are a special kind of Macro that allows you to dynamically show and hide panel elements.

Each Macro placed inside a Stacked Macro can be considered as a page or tab on the panel. You can only view the panel elements contained in one of these Macros at any one time.

How you select which Macro is currently on display is by using the **Panel Index** Module.

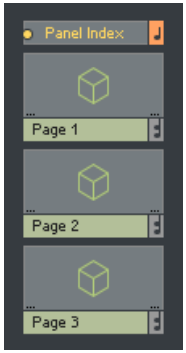
10.8.1 Using Stacked Macros

This short guide will show you how to use a Stacked Macro to create multiple pages of controls.

Firstly, you will need to create a few items in the structure:

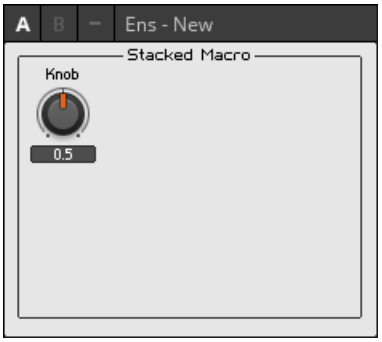
1. In an empty Ensemble, create a *Stacked Macro*. It is located in the *Panel* submenu of the Built-In Module menu.

2. Enter the *Stacked Macro* and create a *Panel Index* Module.
3. Create 3 empty Macros inside the *Stacked Macro* and call them Page 1, Page 2, and Page 3.



4. Inside the Page 1 Macro, create a *Knob* Module.
5. Inside the Page 2 Macro, create a *Fader* Module.
6. Inside the Page 3 Macro, create a *Button* Module.

When you return to the panel view, you should only be able to see the knob you created in the Page 1 Macro.

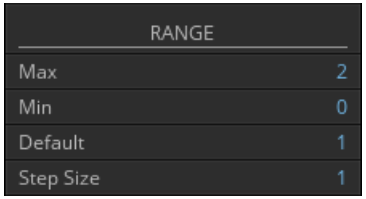


The Ensemble Panel

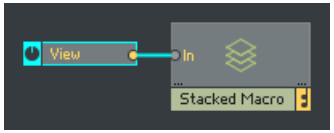
The *Fader* and *Button* are hidden because they are in different Macros.

To select between these Macros we need to create a control:

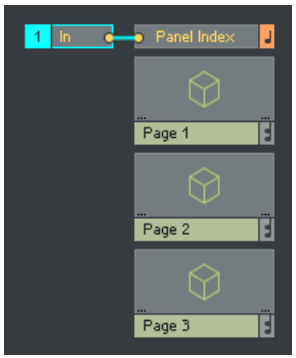
1. Enter the Ensemble structure and create a *Knob*.
2. In the Function tab of the *Knob* Properties, set the Min value of the knob to 0 and the Max value of the knob to 2.
3. Set the Step Size to 1.



4. Rename the knob to "View".
5. Connect the *Knob* to the *Stacked Macro*.



6. Inside the *Stacked Macro*, connect the *In Port* from the *View* knob to the input of the *Panel Index* Module.

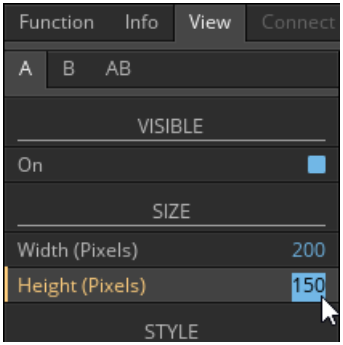


→ Upon returning to the panel view, you should be able to change which Macro, and therefore which control type, is displayed by using the View knob.

Note that when using the View knob, the order of the Macros displayed is the same as the order in which they were created.

When using the Stacked Macro you might notice that the Macro frame is much larger than required for the controls you have placed inside it. This is because a Stacked Macro may contain several Macros with different numbers and sizes of elements, and it would not be ideal for the frame to constantly resize as the user browses through the Macros.

- The size of the Stacked Macro frame is set in the View tab of the Stacked Macro Properties.



11 Automation

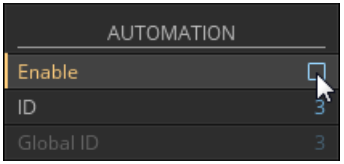
Automation is a way of controlling the parameters of an instrument or effect from a host program. When any new control (like a knob or button) is added to an Ensemble, it is automatically given an Automation ID and added to the automation list.

11.1 Customizing Automation Properties

Disabling Automation

In some cases, you may wish to remove the ability to automate a control (for example, if the control is used to change the display in some way).

1. Click on the control you wish to edit and open its properties.
2. Click on the Connect tab.
3. At the bottom of the Connect tab area is a section called AUTOMATION.
4. Click on the Enable checkbox so that it is empty.



→ The control will no longer be automatable and will be removed from the automation list.

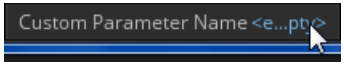
Automation Name

By default, the automation name for a control will be the name of the control combined with its location in the Instrument structure.

However, this is not always the best way to identify a parameter, so it is possible to change this name.

1. Open the control's properties and click on the Connect tab.
2. Navigate to the AUTOMATION section.

3. Click on the text in the section called Custom Parameter Name (it should currently say <empty>).



4. Enter the name you wish to give the control.

5. Once you are finished, press [Enter].

→ The control will now use the Custom Parameter Name as its automation name.

Automation ID

As well as an automation name, each parameter has an automation ID. This is a number used to identify the parameter and to order the parameters.

An ID is assigned when a new control is created, and it will automatically be the next number after the ID of the last control that was created.

Like the automation name, this ID number can be changed.

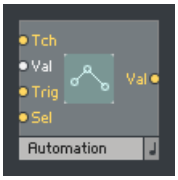
1. Open the control's properties and click on the Connect tab.
2. Navigate to the AUTOMATION section.
3. The parameter called ID will be displaying the current automation ID of the control.

4. Double-click on this number to enter a new number.

5. Once you have finished, press [Enter]

→ The newly entered number will now be used as the automation ID

11.2 The Automation Module



The Automation Module

The Automation module is a module that sends and receives automation data to/from the host. It can be used to add custom automation to an existing control, or can be used as part of a custom control element you may want to create from scratch.

The Automation Module also gives you the ability to dynamically change the automation name for the Module. This means that you could use this as part of a control whose function changes depending on a setting in the instrument. For example, a distortion effect might have a control that functions as a Drive control, but the effect may be able to change to a Bitcrusher, and the Drive control becomes a Bit-depth control.

11.2.1 Sending and Receiving Automation Values

The Automation Module has 3 input ports that relate to sending automation values, which are ordered from top to bottom in the order you should use them:

- Tch (touch): sends the touch-state of the parameter, in other words: this lets the host know whether or not the user is moving the connected control.
  - A value greater than 0 should be sent if the touch-state is active.
  - A value of 0 or less should be sent if the touch-state is inactive.
- Val (value): sets the current value of the module to be sent when an event arrives at the Trig input.
- Trig (trigger): an event at this input triggers a send of the value currently at the Val input. To send, the Tch input also needs to be set to active state. If Tch input is in inactive state, then this Trig input has no effect.

The module has a single output port, which sends a value event whenever it receives automation values from the host.

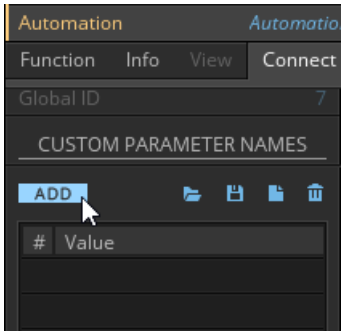
11.2.2 Dynamically Changing Automation Names

By default, the automation name for the Automation Module is the name given to the Module itself. Re-naming the Module will change the automation name.

However, there are some cases in which you may want to change the automation name dynamically.

First of all, you will need to create the name entries:

1. Select the Automation Module and open its properties.
2. Click on the CONNECT tab.
3. In the CUSTOM PARAMETER NAMES table, click the ADD button to add an entry, or click on the trashcan icon to delete an entry from the list.

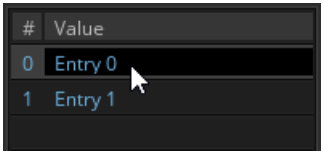


Each entry has an index (#) and a value (Value) field:

- The index is automatically set as an integer, starting from 0 for the first entry.
- The value is the automation name that is to be sent to the host.

To change a value:

► Double-click on the text you want to edit and then type the text you want to be associated with the corresponding index.



The Sel input selects which entry from the list is used as the automation parameter label:

- If the Sel input is not connected, then entry 0 is used.
- If the list is empty, then the original default automation name will be used (i.e. the module's name and its macro path). The Sel input will be ignored.
- If the Sel input is out of bounds, then it is clipped to 0 → n-1, where n is number of items in the list.

You can create and edit lists of automation parameter labels outside of REAKTOR, which will considerably speed up the process of creating and saving large lists. To open, save, or clear lists of automation parameter labels, use the CUSTOM PARAMETER NAMES table's additional buttons. From left to right these are:

- **Read from File**: Click the folder icon to open a file selection dialog that allows you to select a plain text file. The list of automation parameter labels will be filled with the content of this file.

This action will replace any existing list content.

- **Save to File**: Click the disk icon to open a file save dialog that allows you to save the current list of automation parameter labels as plain text file.

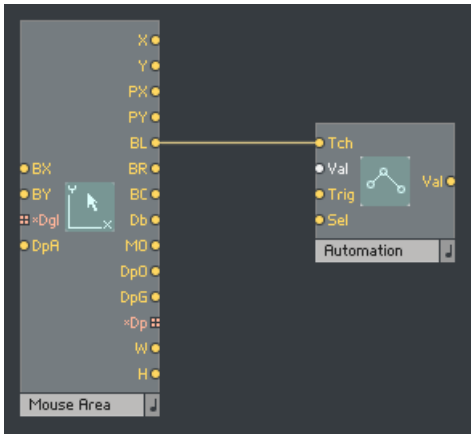
- **Clear List**: Click the document icon to clear the entire list of automation parameter labels.

The file format used by REAKTOR is plain ASCII text (*.txt or *.asc) with one line per entry. An entry index number is not required, as REAKTOR will create these numbers.

11.2.3 Custom Control Example

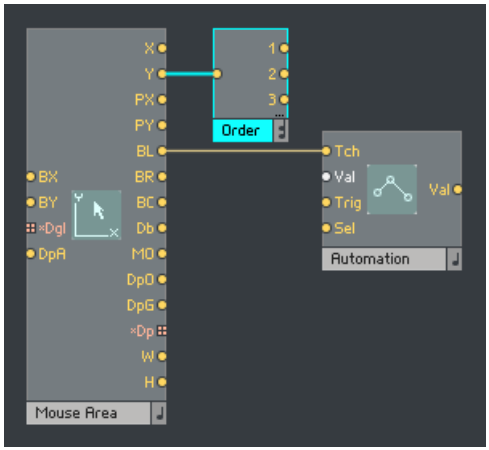
The following tutorial is a quick example designed to introduce you to the Automation Module. You will create a custom control using the Mouse Area Module and a Meter to mimic a fader.

1. Create a *Mouse Area* Module and an *Automation* Module.
2. The control will have its touch-state activated when the left mouse button is pressed, so connect the BL (button left) output of the *Mouse Area* Module to the Tch (touch) input of the *Automation* Module.



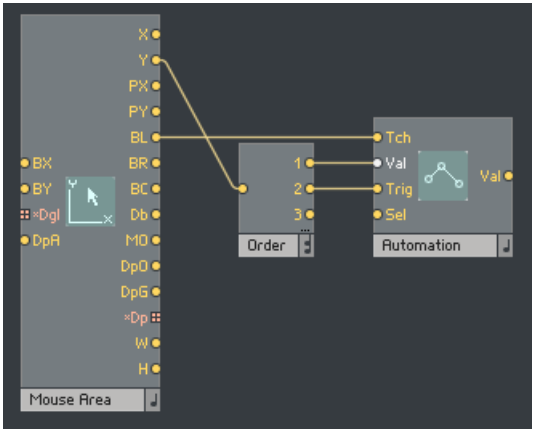
3. When using a software fader, the Y position of the mouse on the fader sets the value of the control. Use the Y output of the *Mouse Area* to send the value. However, the Y output also needs to trigger the value of the *Automation* Module and send it to the host. To insure a correct event order, create an *Order* Module.

4. Connect the Y output of the *Mouse Area* to the input of the *Order* Module.



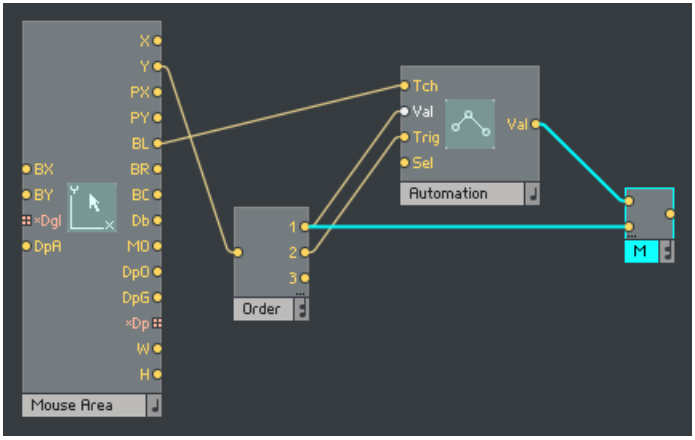
5. Connect output port 1 of the Order Module to the Val input of the Automation Module.
6. Then connect output port 2 to the Trig input.

→ The Structure should look like this:



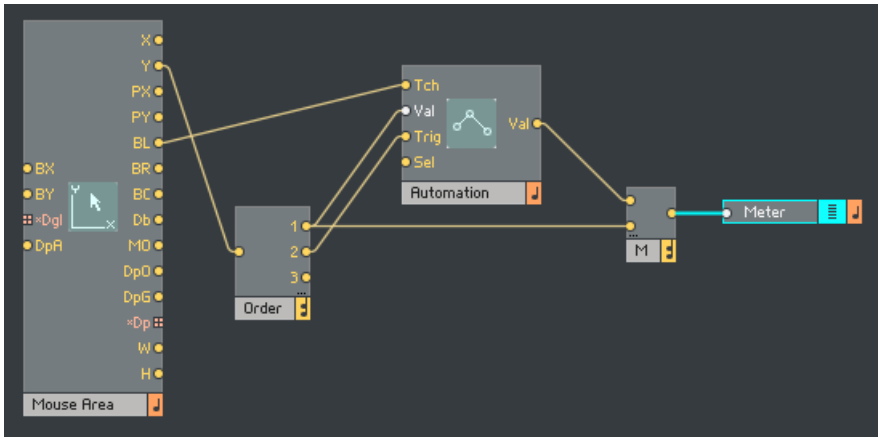
Since both the *Mouse Area* interaction and the automation received from the host will be used to drive the fader value, the Y output of the *Mouse Area* will need to be merged with the output of the *Automation Module*. This can be done be using a *Merge* Module.

► Create a *Merge* Module and connect output port 1 of the *Order* Module and the Val output of the *Automation* Module to its inputs.

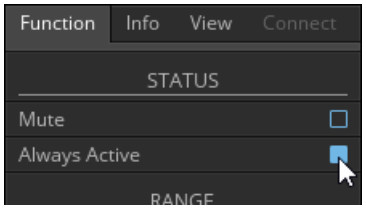


The final stage will be creating the fader itself, and for this you can use a *Meter* Module.

1. Create a *Meter* Module
2. Connect the output of the *Merge* Module to the input of the *Meter*.



3. In the *Meter*'s properties, make sure the Always Active option is checked.



→ Clicking and dragging in the *Mouse Area* will now change the level of the *Meter*.

Now that the control is working, you can save the Ensemble and open it in a host. If everything is connected correctly, writing and reading automation values will function as it does for any other REAKTOR control.

12 KOMPLETE KONTROL and MASCHINE 2 Integration

The following chapter will cover the features that relate to integrating your REAKTOR Ensembles into MASCHINE 2 and KOMPLETE KONTROL. This includes visibility in the sound browsers, and optimizing the panel.

Although this chapter will mostly refer only to KOMPLETE KONTROL, the same principals can all be applied to MASCHINE 2.

12.1 Snapshot Visibility

When using REAKTOR, you can browse for a number of different things: Ensembles, Modules, Snapshots, etc. Generally a Snapshot is considered to be a REAKTOR sound preset.

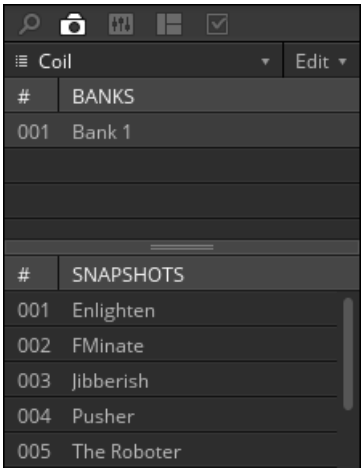
Since the KOMPLETE KONTROL Browser is used primarily to navigate sounds, it has been designed to scan for Snapshots within a REAKTOR Ensemble file. However, Snapshots need to be tagged so that KOMPLETE KONTROL can integrate them correctly.

12.1.1 Tagging Snapshots

Since a REAKTOR Ensemble can be an instrument, an effect, or even both, we need to communicate which of these the Snapshot is classed as so KOMPLETE KONTROL will know where to place it in the Browser.

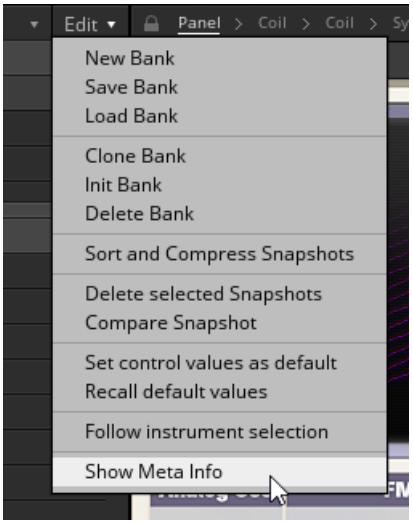
Note that KOMPLETE KONTROL can only open Snapshots tagged as instruments; whereas MASCHINE 2 can open both instrument and effect Snapshots.

Tagging is done in the Snapshot tab.

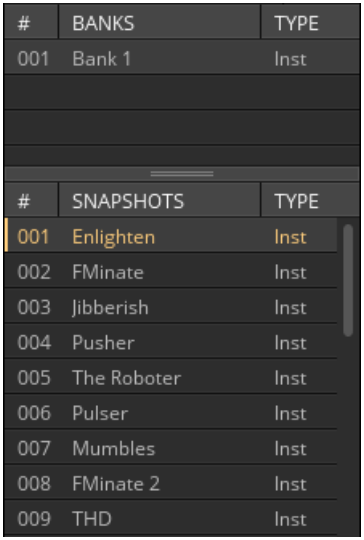


The Snapshot Tab

► To display Snapshot tags, enter the Edit menu and select *Show Meta Info*.



→ A new column will appear in the Snapshot browser displaying any tags applied to the Snapshots and Banks.



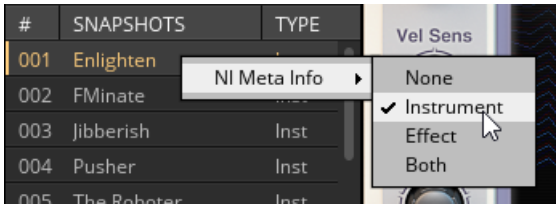
There are four kinds of tags:

- None: The sound will be hidden from the KOMPLETE KONTROL browser. This can be useful if your Ensemble includes an "Init" (i.e. blank) snapshot, which is not something you necessarily want to be displayed in the KOMPLETE KONTROL browser.
- Inst: An Instrument sound. Any Snapshot that generates audio, but does not alter incoming audio should be tagged as an Instrument sound.
- FX: An Effect sound. Any Snapshot that relies on, alters, or reacts to incoming audio should be tagged as an Effect sound.
- Both: A sound that can be either an Instrument or Effect. Any Snapshot that generates audio independently, but can also react to incoming audio somehow, should be tagged as Both.

Tagging a Single Snapshot

1. Right-click on the Snapshot in the Snapshot browser.

2. Enter the *NI Meta Info* sub-menu to display the available tags.

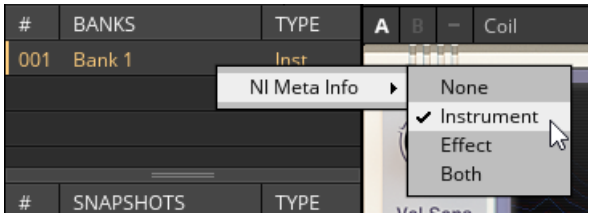


3. Click on the relevant tag for the Snapshot.
4. Save the Ensemble file to make the changes permanent.

Tagging a Snapshot Bank

Tagging a Snapshot Bank will tag all Snapshots contained in that Bank with the same setting.

1. Right-click on the Bank name in the browser.
2. Navigate to *NI Meta Info* to display the available tags.



3. Select the relevant tag for the Snapshot.
4. Save the Ensemble file to make the changes permanent.

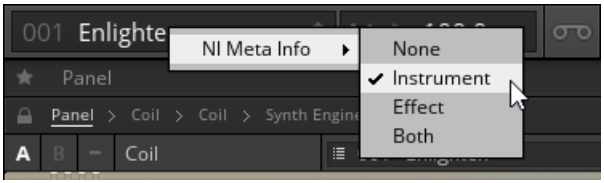
Single Preset Files

If you save a Preset file in REAKTOR - by using the *Save Preset As...* command in the File menu - the saved file will take the tag information from the last selected Snapshot.

If the Snapshot was saved without a tag, you can tag it by using the Snapshot menu in REAKTOR's toolbar:

1. Right-click on the Snapshot menu.

2. Select the relevant tag for the Snapshot.

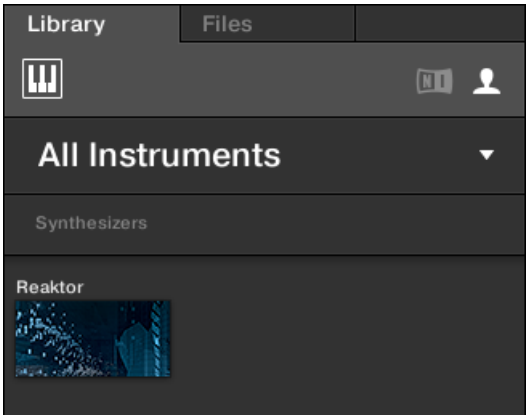


3. Re-save the preset file.

Tagging a preset using this method will not change the tag of the original Snapshot.

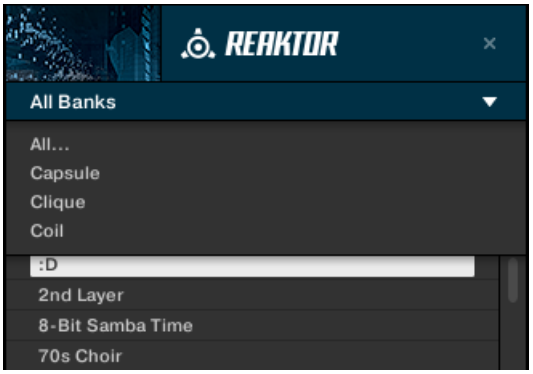
12.2 Defining the Sub-Bank Name

Any Ensemble created in REAKTOR containing tagged Snapshots will be located in the REAKTOR bank when it is added to the KOMPLETE KONTROL library (see section ↑12.6, Adding Ensembles to the Library ↑12.6, Adding Ensembles to the Library for a walkthrough of how to do this).



The User Tab of the Komplete Kontrol Browser

In order to make locating the sounds in the KOMPLETE KONTROL Browser easier, you can define a Sub-Bank name.



Displaying the Sub-Banks

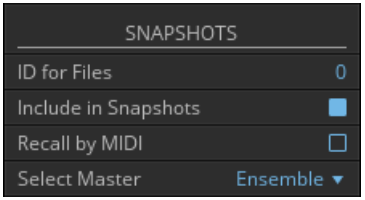
The Sub-Bank name is automatically taken from the name of the Ensemble's Snapshot Master.

Changing the Snapshot Master

By default, the Snapshot Master is the Ensemble itself. However it is possible to change it to any Instrument in the Ensemble.

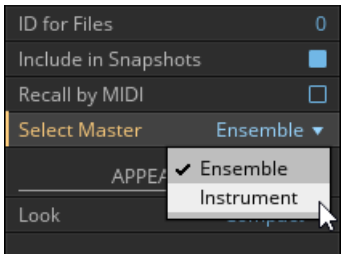
To change the Snapshot Master:

1. Select the Ensemble.
2. Open the Ensemble Properties
3. Select the FUNCTION tab.
4. Find the section called SNAPSHOTS at the bottom of the Properties.



5. In this section, click on the Select Master menu.

6. Select the Instrument you wish to use as the Snapshot Master from this menu and click on it.



→ The selected Instrument is now the Snapshot Master for the Ensemble.

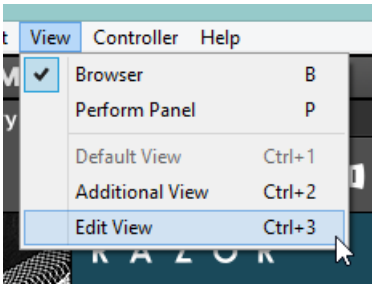
12.3 Panelsets

Even if no action is taken, your REAKTOR Ensemble's interface will appear in the KOMPLETE KONTROL interface when a sound from it is loaded. The appearance will be the same as the last saved state.

However, it is possible to make some optimizations to make using your Ensemble in KOMPLETE KONTROL a better experience.

12.3.1 Default and Additional Views

KOMPLETE KONTROL has three ways of viewing a REAKTOR Ensemble's interface:



The 3 Views in the View Menu

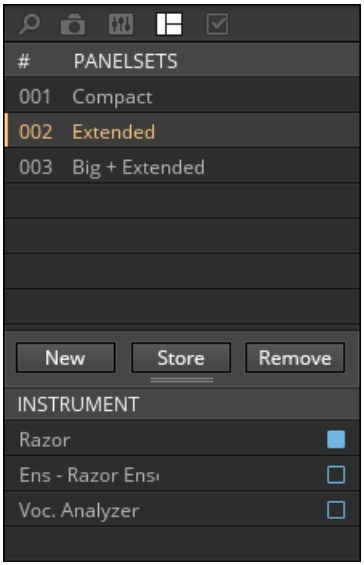
- *Default View*: The first view displayed when you load the Ensemble.

- *Additional View*: An optional second view.
- *Edit View*: Displays the Ensemble in the full REAKTOR interface.

The first two views are taken automatically from the Ensemble's Panelsets.

In order for KOMPLETE KONTROL to recognize a Default and Additional view, you need to create two Panelsets:

- The first Panelset will be the first view displayed - the Default View.
- The second Panelset will be the alternate view - the Additional View.



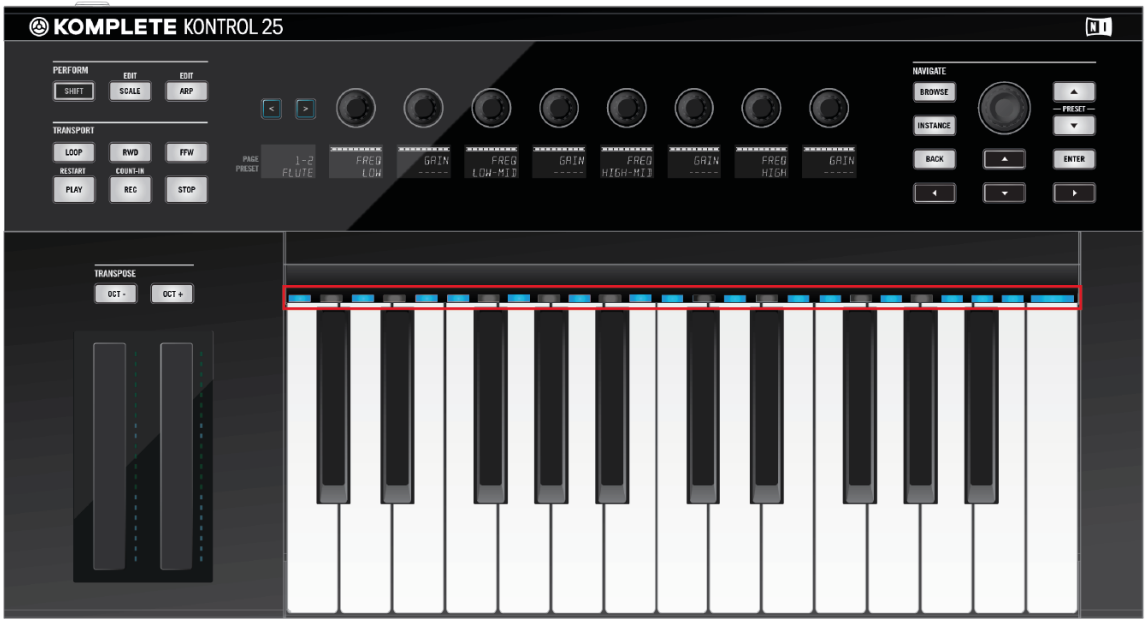
The Panelsets for RAZOR

Any further Panelsets will be ignored by KOMPLETE KONTROL. They will not interfere with the KOMPLETE KONTROL integration, but they will not be available either.

12.4 Hardware Control

The Hardware Control Module (listed in the menus as HW Control) allows greater integration with NATIVE INSTRUMENTS hardware products. The module uses an open design so that it can be updated to support any new hardware NATIVE INSTRUMENTS releases in the future.

At the time of writing, this Module allows custom control over certain features of the KOMPLETE KONTROL S-SERIES hardware, namely the Light Guide system.

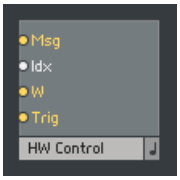


The Light Guide on the KOMPLETE KONTROL S-SERIES keyboard

REAKTOR must be used as a plug-in in the MASCHINE or KOMPLETE KONTROL software in order for the Hardware Control messages to be sent. Hardware Control messages are not sent when REAKTOR is used in stand-alone mode.

The Hardware Control module has four inputs:

- Msg (Message Type): Selects the message type to be sent by the module.
- Idx (Index): Sets the index of the array into which data is to be written.
- W (Write): Writes a value into the currently selected array index.
- Trig (Trigger): Triggers the contents of the module's array.



The Hardware Control Module

To use it correctly, you must send the messages in the following order:

1. Select the Message Type you want to use via the Msg port.
2. Use the Idx and W ports to fill the array with data.
3. Once the necessary data has been written into the array, send an event to the Trig input to send the data to the KOMPLETE KONTROL software.

→ KOMPLETE KONTROL will take the data and apply the necessary changes to the KOMPLETE KONTROL S-SERIES hardware.

12.4.1 Message Types

Message Type	Msg Input Value
Set single note attributes	10
Set 128 note colors	11
Set note pressed support	40

Set Single Note Attributes

For this message type, the first index of the array (index 0) sets the MIDI note number, and the following indices are used to set various attributes for that note:

Array[0]	Array[1]	Array[2]	Array[3]
MIDI Note Number	Note Color	Pressed State (0 or 1)	Note Type (0 or 1)
0 ... 127	0 ... 17	0: Unpressed 1: Pressed	0: Default 1: Control

The Note Type tells KOMPLETE KONTROL whether this is a standard playable note, or whether it is a control note used for special functions in instruments. Control notes will be excluded from any performance functions, such as the arpeggiator.

Sending any value outside of the value ranges specified above will cause the message to be ignored, and the relative setting will remain unchanged.

The pressed state will only be sent if Note Pressed Support is set to be active.

Set 128 Note Colors

This message type allows you to send the note colors for all MIDI notes in a single array. With this message type, the array index represents the MIDI note, and the value of that index sets the note color.

Array[0]	Array[1]	...	Array[126]	Array[127]
MIDI Note 0 Color (0...17)	MIDI Note 1 Color (0...17)	...	MIDI Note 126 Color (0...17)	MIDI Note 127 Color (0...17)

Set Note Pressed Support

A value of 40 at the Msg input sets the module to send the Note Pressed Support state. A single

yes/no (0/1) value is needed, and is sent in array index 0:

- : No - REAKTOR will not send Note Pressed messages; this is to be handled by the host.
- : Yes - REAKTOR will send Note Pressed messages, taking control of this from the host.

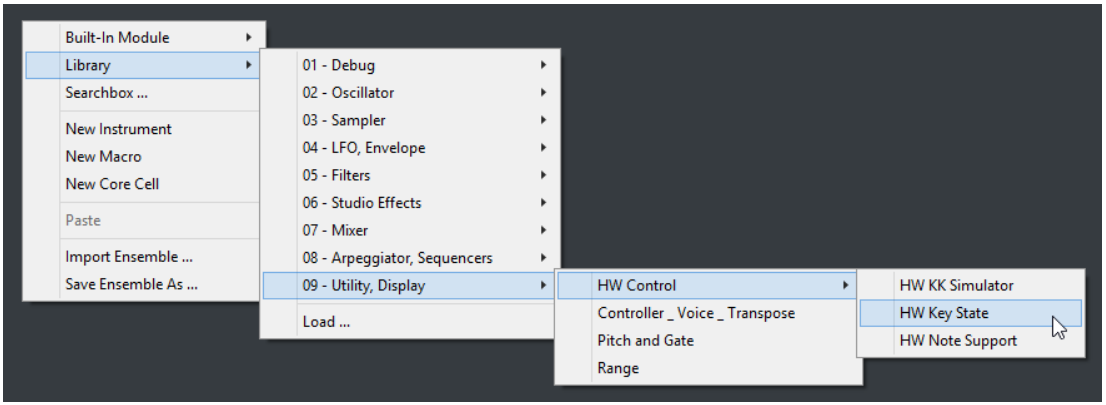
12.4.2 Available Note Colors

Value	Color
0	Red
1	Orange
2	Light Orange
3	Warm Yellow
4	Yellow
5	Lime
6	Green
7	Mint
8	Cyan
9	Turquoise
10	Blue
11	Plum
12	Violet
13	Purple
14	Magenta
15	Fuchsia
16	Default
17	Inactive

12.4.3 Related Macros

The Library contains a number of tools designed to help with support the Light Guide system, even if you do not have any KOMPLETE KONTROL S-SERIES hardware to test your Ensembles.

► These tools are found in the *09 - Utility, Display -> HW Control* submenu.



For debugging, the *HW KK Simulator* Instrument takes the same messages given to the HW Control Module, and uses them to display a simulation of how the KOMPLETE KONTROL S-SERIES hardware would interpret them.

12.5 Native Map

The Native Map technology used in KOMPLETE KONTROL scans the automatable controls in an Ensemble and maps them to the knobs on the KOMPLETE KONTROL or MASCHINE hardware.

As such, it is best to prepare an Ensemble's automation data to use intelligible names, and to be ordered in such a way as the most important controls appear first.

The controls are mapped according to their automation IDs, and the names published are taken from the automation name.

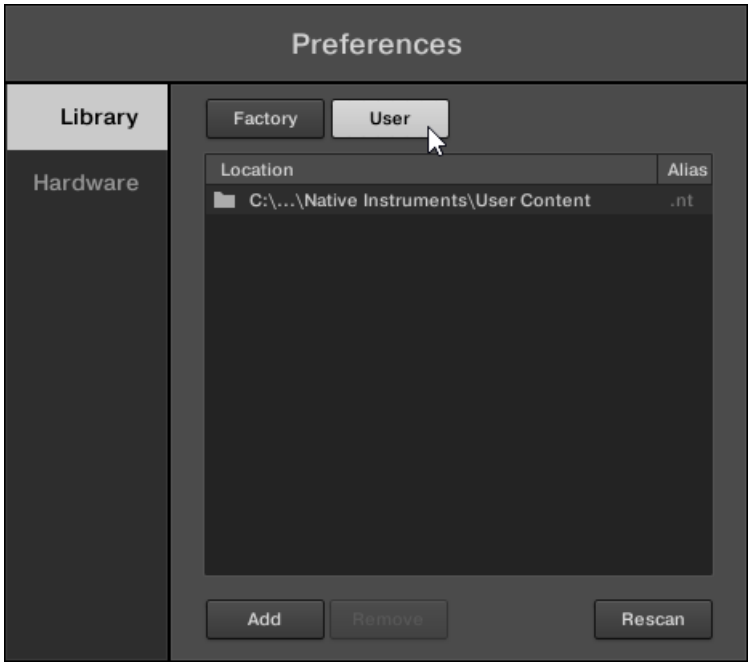
Section ↑11, Automation ↑11, Automation gives all the information you will need regarding editing automation properties in your Ensemble.

12.6 Adding Ensembles to the Library

All NATIVE INSTRUMENTS products are automatically integrated into the KOMPLETE KONTROL Browser during installation.

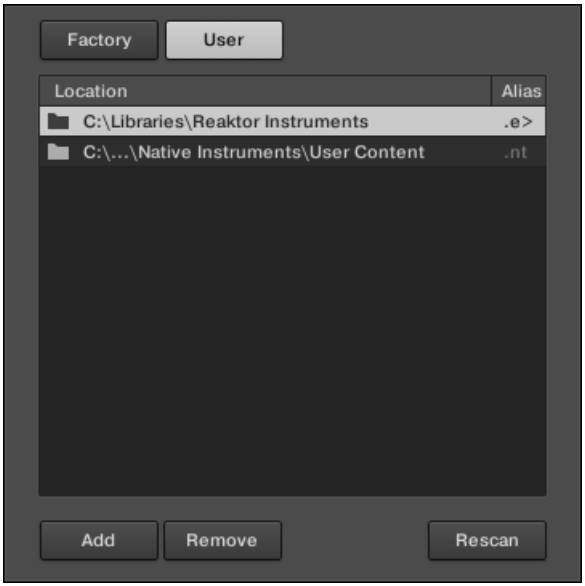
Since any Ensemble you create lacks an installation, it will not automatically appear in the Browser. However, it is possible to tell KOMPLETE KONTROL to scan a certain folder for content:

1. Open the Preferences window by selecting *Preferences...* from the Edit menu.
2. In the Library tab (which should be selected by default), click on the User button.



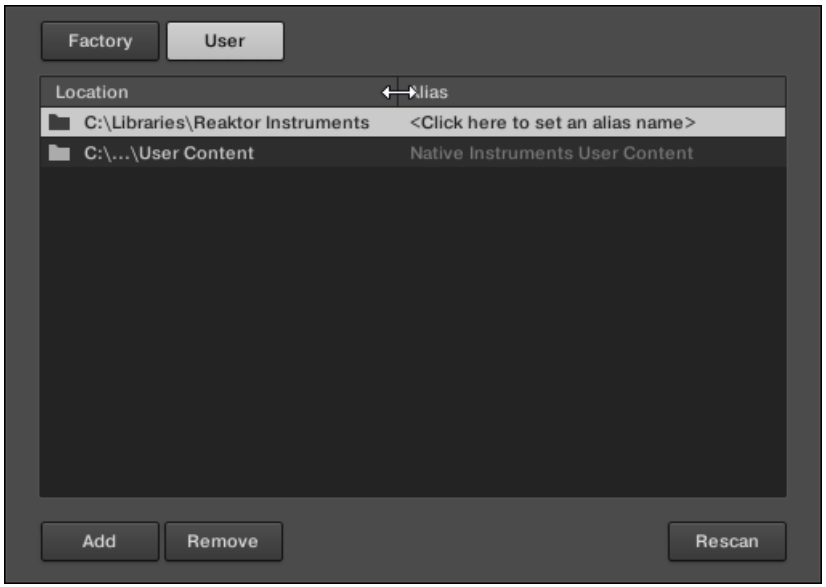
3. Click on the Add button.
4. A window will open, from which you can browse for a folder to add to the KOMPLETE KONTROL User Content.
5. When you have located the folder you wish to add, select it and click OK.

→ The selected folder will now appear in the Location table.



With the folder added for KOMPLETE KONTROL to scan, we can change the Alias of this folder to give it a name that is easier to identify.

1. If necessary, you can resize the Preferences window and the Location table columns to get a better view of the content.



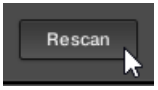
2. Click on the entry in the Alias column of the Location table.
3. Type the name you want to give this location.

4. When you are finished click away or press [Enter] to apply the changes.

→ The Location will now have a custom Alias.

With the folder added and named, there is only one final step left: KOMPLETE KONTROL needs to scan the folder for appropriately tagged Ensembles.

► Start the scan by clicking on the Rescan button.



→ After the scan has finished, the contents of the newly added folder(s) will be added to the KOMPLETE KONTROL browser.

If there are a large number of folders and files to scan, this process could take some time.

13 Internal Connection Protocol

The Internal Connection protocol (IC protocol, in short) is a feature that lets you send and receive Events to and from different parts of the structure without the need of wires. It also allows you to directly connect panel elements to each other, or to other IC Modules.

13.1 Connecting Two Panel Elements via the IC Protocol

First of all, you will need to create two controls:

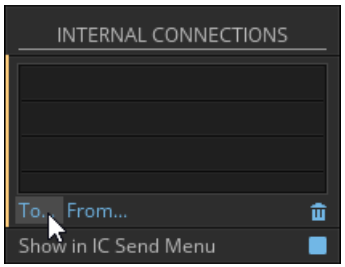
1. Create a new Ensemble.
2. Unlock the Panel.
3. Right-click the Panel area and choose the *Built-In Module* > *Knob* menu entry.
4. Repeat the previous step to create a second *Knob*.
5. Move the Knobs so that they do not overlap.
6. Double-click on each knob's label and rename the knobs to “Knob A” and “Knob B”.

7. Lock the Instrument Panels by pressing the Panel Lock button.

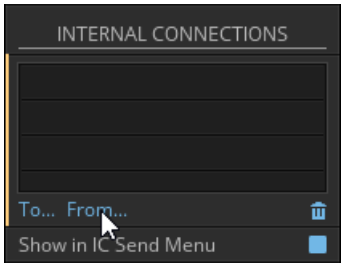
With the knobs in place, you can now use Internal Connections to control Knob B with Knob A:

1. Select *Knob A* to view its properties and navigate to the Connect tab.
2. At the bottom of the Connect tab is an area called INTERNAL CONNECTIONS.

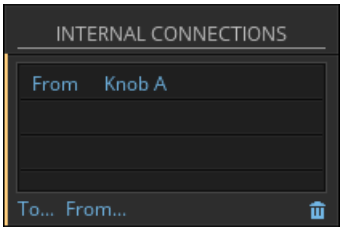
3. In this area is a button labelled To..., click on this to activate *Knob A* as a potential IC Send Module.



4. Go to the Connect tab of *Knob B* and find the INTERNAL CONNECTIONS area.
5. Click on the From... button to activate *Knob B* as a potential IC Receiver.



→ Since *Knob A* was already activated as a potential IC Send, a "wireless" connection is made between *Knob A* and *Knob B*.



If you now move *Knob A*, *Knob B* will mirror the value of *Knob A*. However, if you move *Knob B*, *Knob A* will remain as it was. Note that this holds true even if *Knob B* has a different value range than *Knob A*.

You can also set up an additional Internal Connection with *Knob B* as the sender and *Knob A* as the receiver. REAKTOR handles this in such a way as it will not cause any infinite feedback loops.

13.2 IC Send/Receive Modules

As well as being able to connect controls to each other, REAKTOR also includes *IC Send* and *IC Receive* modules for other uses.

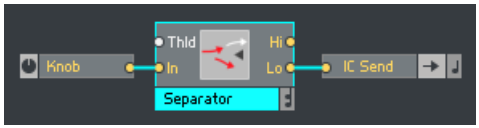
The IC Send and IC Receive Modules

To give an example of how this works, this tutorial will show you how to create a knob that deactivates a button whenever it is set to zero.

1. Create a new Ensemble.
2. Enter the Ensemble structure and create a *Knob*, a *Button*, and an *IC Send* Module.

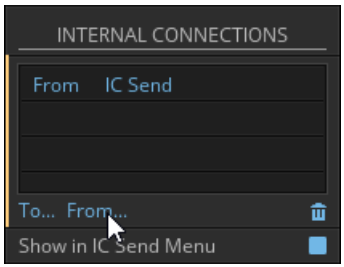
You will now need a way of sending a zero event only when the knob is at zero.

1. Create a *Separator* Module (*Built-In-Module* > *Event Processing*) to split the events from the *Knob* depending on their value.
2. Connect the output of the *Knob* to the In port of the *Separator*, and then connect the Lo out port of the *Separator* to the *IC Send* Module.

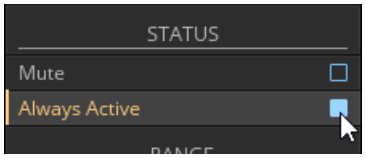


3. The Thld (threshold) port of the *Separator* does not need a *Constant* attached, since it defaults to zero, which is the value needed in this example.

4. You can connect the *IC Send* module to the *Button* in the same way as described in the previous section.



5. In order for everything to work as desired, you will need to activate the Always Active option in the Function tab of the IC Send properties.



→ When you return to the Panel, you will be able to active the *Button* by clicking on it, and then deactivate it by turning the *Knob* to 0.

On the panel you will also notice that the *IC Send* has a Panel element. This control allows you to make or break Internal Connections with any IC enabled Module from the Panel.

The settings made with this control are not saved in the Snapshot, but are saved in the Ensemble. As such, this control is generally reserved for helping with prototyping, and is often hidden when releasing an Ensemble to the public.

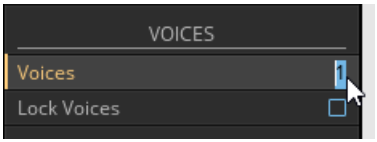
14 Optimization Techniques

14.1 Reduce the Number of Voices

Although a single oscillator with 8 voices will not be as heavy on the CPU as 8 monophonic oscillators, the general principal still applies: More voices leads to a greater load on the CPU.

If the instrument you are creating does not need a high level of polyphony, try setting the number of voices to just the number you need.

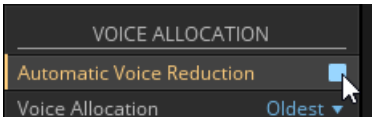
► Reduce the number of Voices in the Ensemble Properties.



14.1.1 Automatic Voice Reduction

Automatic Voice Reduction reduces the number of voices in an Ensemble until the CPU level is below the Max CPU value (which is defined in the Preferences window).

► Click on the Automatic Voice Reduction option in the Ensemble Properties to activate Automatic Voice Reduction.



→ REAKTOR will reduce the number of Voices until the CPU level drops below the Maximum threshold. It will continue to do this until the Automatic Voice Reduction option is unchecked.

14.1.2 Use Monophonic Processing

You should only use polyphonic Modules when it is a necessity. If you are building an FX unit, or a mono-synth, you do not need 16 voices all doing the same thing. And even for a poly-synth, you may only need the oscillators, envelopes and filters to be polyphonic; any LFOs or effects could be processed as mono signals.

See section ↑2.4, Mono and Poly Modes ↑2.4, Mono and Poly Modes for more information about polyphony.

14.2 Use Events Rather than Audio signals

When processing at the audio rate, calculations are made constantly at the sampling rate. Events are only processed when a value is triggered, and are also processed at a slower rate. If the process you are performing does not directly interact with the audio stream, consider processing with events.

14.3 Eliminate Superfluous Events

In REAKTOR it is sometimes possible to generate Events even if the value is not changing, or to generate multiple Events from a single process.

14.3.1 Filtering Duplicate Values

If you see an area in which this could occur, try using a *Step Filter* Module.

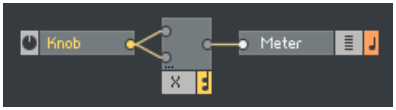
The Step Filter Module

This Module blocks all Events which carry a value that is within a certain tolerance interval from the value carried by the last Event at its input. The tolerance interval is set at the Tol input port. With the value 0 at the Tol input port, the Step Filter Module blocks all Events that carry the same value as the previous Event.

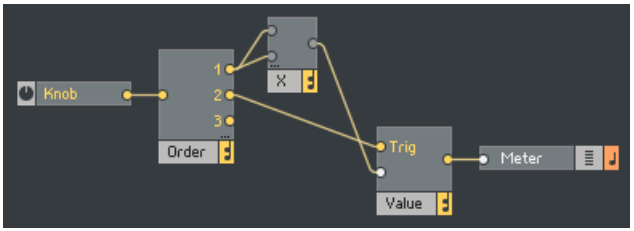
14.3.2 Triggering Values

Another way to filter unwanted Events is to use a combination of the *Order* Module and the *Value* Module.

In the following Structure, moving the *Knob* will produce two Events at the *Multiply* Module's output.



By adding an *Order* Module and a *Value* Module as illustrated below, you can effectively block the output of the *Multiply* Module until it has finished its calculations.

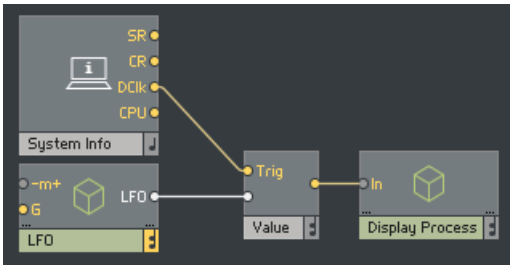


14.4 Use the Display Clock

If your ensemble has a complex display element that updates frequently, it can be beneficial to use the Display Clock.

REAKTOR's GUI processes function at around 25 frames per second (though this value can fluctuate depending on the instrument that is loaded). This means that, at most, the panel updates itself 25 times in a second.

The *System Info* Module includes an output called DClk (Display Clock) that sends an Event before each display update. You can use this clock to trigger any GUI processes in your structure, thereby only calculating them when necessary.



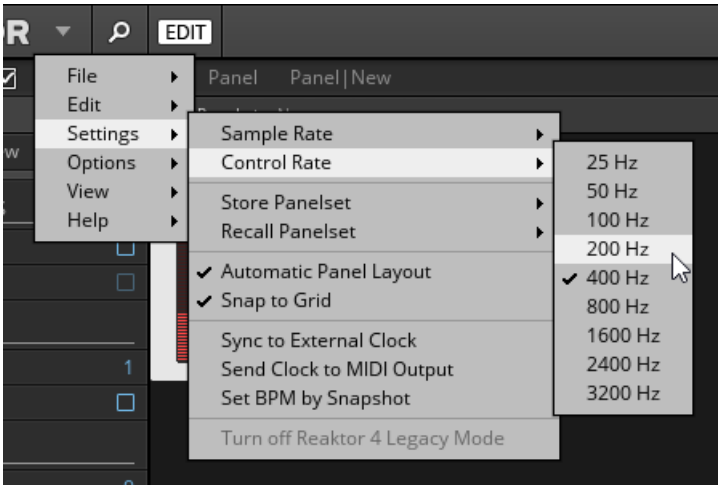
An Example of Using the Display Clock

In the example above, the LFO is driving a display process, but this process does not need to happen at the Audio Rate at which the LFO runs. The Display Clock is used to effectively down-sample the LFO value at the Display Rate, making the display process more efficient.

14.5 Reduce the Control Rate

The Control Rate is the rate at which Events are processed. By default this is set to 400Hz (400 times a second). Reducing the Control Rate means that Event signals will be processed more slowly, reducing the strain on your CPU.

► You can reduce the *Control Rate* from the Settings Menu:



14.6 Use Multipliers Instead of Dividers

Division is a more expensive process for a CPU to handle than a multiplication. For example, it would be a lot more efficient to multiply by 0.5 rather than divide by 2, even though the result is the same.

14.7 Avoid unnecessary use of Send/Receive Modules

Send and *Receive* Modules can be used to make wireless connections in your structure. This can be appealing for many uses, including improving the readability of your structure.

The Send and Receive Modules

However, overusing these Modules can cause your Ensemble to have a long loading time. They also make it difficult to debug your structure, so use them where necessary, but try not to use them frivolously.

14.8 Use the Modulo Module instead of Wrap options

Some Modules that use indices (like the *Multi Display* or the *Event Table*) contain an option in their properties to wrap indices that are out of bounds. So if you had an *Event Table* with a table of 5 elements, and you sent an index value of 6 to the RX input, the Module would wrap this value around to become a 1.

In some cases, like a step sequencer, this behavior is desired. Unfortunately, this process is CPU Intensive and it is much more efficient to use the *Modulo* Module instead.

The *Event Table* and the *Audio Table* both have the wrap option selected by default.

15 An Introduction to Core

Core Cells are in some ways like a hybrid between a Module and a Macro.

A Core Cell

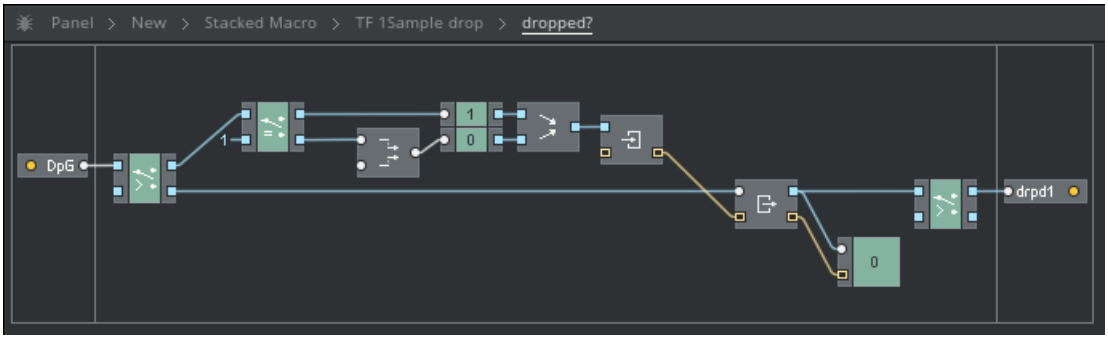
Core Cells can be identified by the icon of a cube made of dots.

The Primary level of REAKTOR treats Core Cells like Modules, but you can enter the Structure of a Core Cell like a Macro.

Although Primary and Core share the basic REAKTOR building paradigm of connecting blocks with wires to create new things, Core uses different signal types and has its own set of Modules that make it more optimized for creating low level DSP functions.

Another way to think about it is like this:

- The Primary level of REAKTOR is where you can connect an oscillator Module to a filter Module.
- The Core level is a programming environment where you would create your own oscillator and filter algorithms.



The Structure of a Core Cell

Even though Core is a deeper programming environment, Primary is still necessary for building user interfaces and receiving MIDI signals. In other words, Core requires the Primary level in order to communicate with the "outside world".

To learn how to program in Core, read the REAKTOR 6 Building in Core document.

Learning Core is not mandatory for building in REAKTOR, there are a number of Core Cells and Core Macros in the Factory Library for you to use. However, if you want to dive deeper into building and customizing REAKTOR instruments and effects, learning Core can be a very useful tool.

16 Table Framework

The tutorial in section ↑8, Drag and Drop Sampler introduces the basics of the Table Framework in a practical application. This section is a more technical reference, giving you information regarding the fundamental functionality of the Framework.

This section will also cover aspects of the Core level of REAKTOR, and thus it is advantageous to have experience with Core, or to read the REAKTOR 6 Building in Core document, before proceeding with this section.

16.1 Tables and Samples

A Table is a two-dimensional array of numbers which, on a technical level, is the same as a stereo audio file. Digital audio is essentially a sequence of numbers, each representing a sampled amplitude value. Taken out of this audio context, each sample is like an element in an array and the amplitude values are just numbers in the array.

The Table Framework allows you to share the data from an audio sample throughout an Ensemble Structure with ease. It also allows samples to be read as data in Core Cells, which enables the possibility of building sample-based DSP in Core.

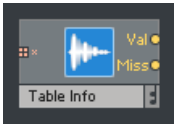
Currently the Table Framework is designed primarily for the sharing of audio data, and thus only accepts wav or aiff files; however, there are plans to expand the Table Framework in the future to make it more flexible.

16.2 Table Reference Wires

As suggested by its name, the Table Reference wire does not send the contents of a Table along it; rather, it passes a unique reference to a Table's memory location. This makes it unlike the other wires in REAKTOR, which transmit values. Table Reference wires simply join Modules and Core Cells so that they share the same Table data.

16.3 Table Reference Ports

Table Reference Ports can be identified by their orange square appearance.



The Table Info Module has a Table Reference input

For Built-in Modules, these ports are always labelled with a *.

Certain other Modules can accept Table Reference Connections:

- Merge
- Order
- Routers (all types)
- Value
- Switch
- To/From Voice

Ports that can accept Table Reference connections can be identified by their dark and hollow appearance.

The Order Module can accept Table Reference connections

16.4 The Table List Module

The Table List Module

This Module is the main Table manager and lets you maintain and select from a list of currently active Tables. It is the only Table Framework Module with Snapshot support and a Panel element.

The Panel view of the Table List is the same as the List Module, and contains a list of the currently active Tables.

The Table List Panel Element

16.4.1 Adding a Table to the Table List

There are two methods for adding a Table to the Table List:

- A Table Reference Event at the *Ins input will insert the referred Table at the index set at the Idx input and pushes any existing tables down the list.
- A Table Reference Event at the *Wr input will overwrite any Table at the index set at the Idx input. If the index is greater than the highest index currently used, then the new Table will be added to the end of the list.

16.4.2 Removing a Table from the Table List

To remove a Table from the Table List:

1. Send a value to the Idx input to select a Table index.
2. Send an event to the Del input to delete the Table at the selected index.

⇨ Upon deletion, the N and, if necessary, the Idx output values are updated.

If the currently selected Table was deleted, then the * Table Reference output is set to Null. In this case the Idx output is set to 1.

16.4.3 Selecting a Table Reference to Output

There are two methods for selecting a Table Reference:

From the Structure

1. Send a value to the Idx input to select a Table index.
2. Send an event to the Sel input to trigger the selection.

⇨ The Idx output will update and the selected Table Reference will be sent from the * out port.

When a selection is made from the Structure, the *Table List*'s Panel element will be updated to show which Table has been selected.

Idx in port values should be kept between zero and max index. Anything outside this range is ignored.

From the Panel

The Panel view of the *Table List* can be used to select a table.

► Click on a Table Reference in the *Table List* to select it.

1.

⇨ The Idx output will update and the selected Table Reference will be sent from the * out port.

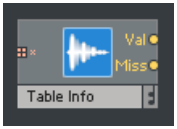
16.5 The Sample Map Module

The Sample Map Module

The *Sample Map* Module allows you to send sample data from a Sample Map into a Core Cell or Table Framework Module as a Table.

- The sample is selected using the pitch (P) and velocity (Vel) inputs.
- The sample is then sent to the output as a Table when an Event is sent to the Trig input.

16.6 The Table Info Module



The Table Info Module

The *Table Info* Module outputs a specific piece of information (meta-info) about the current table, such as Sample Rate, Table Size, Loop Points, etc.

- The table meta-info to be read is selected in the Module Properties.
- If the meta-info is not present in the selected table, then the Miss output sends a value of 1.
- If the meta-info is present, then the value is sent from the Val output.

16.7 The Table Location

The location of a Table depends on its origin:

- When using the *Sample Map* Module, the location of the Table is the same as that of the sample. So if the sample is embedded, then the Table will also be embedded in the Ensemble; if the sample is not embedded, then the Table maintains a reference to the location on the disk.
- A Table that originated from an embedded sample in the *Sample Map* Module remains embedded when it is stored in a *Table List* Module.
- A sample that was dragged onto the *Mouse Area* Module and then stored in a *Table List* maintains a reference to the location on the disk, like a non-embedded sample.

16.8 Drag and Drop

The *Mouse Area* Module supports the dragging and dropping of Table References.

16.8.1 The Drag and Drop Process

To give the builder full control over which samples are accepted and when, the following sequence happens when a user drags a sample or a group of samples over the *Mouse Area*:

- When the user drags a sample or a group of samples over the *Mouse Area* this is a drop request and the DrpG out port is set to 2.
- The *Drp out port iterates through the Table Reference of all samples being dragged.
- When the Table Reference iteration is finished, the DrpG out port is set to 0 and the *Drp output is set to Null.
- If DpA is set to 1, then the samples may also be dropped by the user by releasing the mouse button.
- When the user releases the mouse button, the DrpG out port is set to 1 (indicating drop receive).

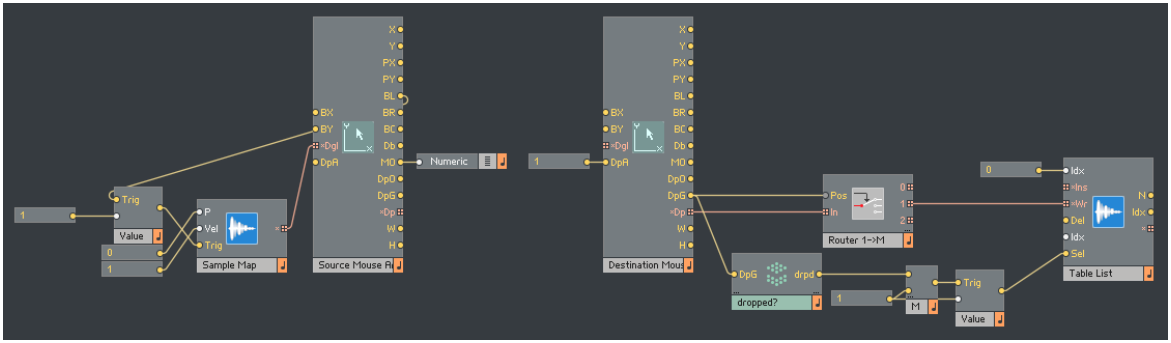
- The *Drp out port iterates through the Table Reference of all samples being dropped.
- When the Table Reference iteration is finished, the DrpG out port is set to 0 and the *Drp output is set to Null.

16.8.2 Dragging a Table Reference from a Mouse Area

You can also drag a Table Reference out of a *Mouse Area* and into another one.

When a drag event happens on the Mouse Area, the *DgI in port listens for any incoming Table References and these are added to the dragged object.

Example: Dragging a Table Reference from one Mouse Area to Another



An Example Structure Illustrating Drag and Drop between Mouse Area Modules.

In the above example, when the user clicks to start dragging from the Source *Mouse Area*, the *DgI in port is fed with a Table Reference to the sample at note 0 in the *Sample Map*.

The Table Reference can then be dragged and dropped into the Destination *Mouse Area*, where it is subsequently stored in a *Table List* Module.

The Example Structure's Panel View.

16.9 Table References in Core

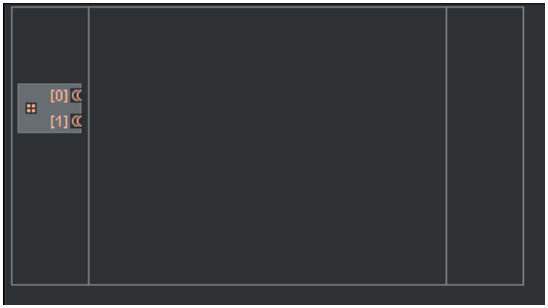
16.9.1 TableRef Input

A Table Reference can be passed into a Core Cell by using the *TableRef* input type.

► Insert a *TableRef* input by right-clicking on the input area to the left and selecting *New -> TableRef* from the menu.

1.

⇨ A *TableRef* input will be created.



By default, the *TableRef* input will have 2 outputs, one for each channel of a stereo sample.

- Output [0] references the data in the left channel.

- Output [1] references the data in the right channel.
- If the audio sample is mono, then both outputs reference the same mono data.

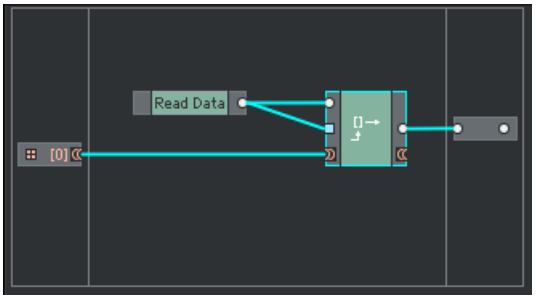
The number of outputs from the *TableRef* can be set in its Properties:

1. Select the *TableRef* input and open its Properties.
2. Navigate to the Function tab.
3. Use the Output Count parameter in the FUNCTION section to set the number of outputs.
4. Currently the output count can only be 2 or 1, so set the Output Count parameter to 1 to see how the *TableRef* input changes.

⇨ The *TableRef* input will be reduced to 1 output.

16.9.2 Accessing Table References

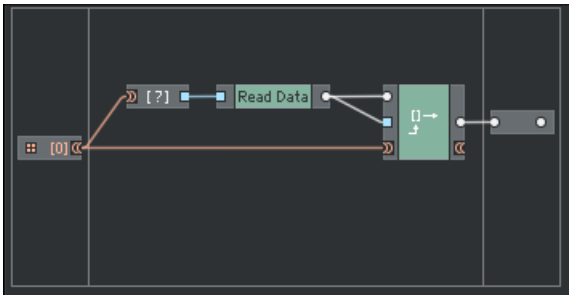
In Core, a Table Reference is accessed and treated like a normal Core Array or Table. The data in the Table can be accessed using the *Index* and *Read* Modules or the *Read []* Macro.



An example of a Core Structure that reads the data from a Table Reference

At the time of writing Tables are read only; data cannot be written into a Table Reference.

You can access the size of the Table using the *Size []* Module.



Reading the Table size using the Size [] Module

The *Size []* Module will also output an Event any time the Table Reference is updated from the Primary level. So the *Size []* Module can be used to trigger any processes that need to happen after a new Table is loaded.

16.9.3 Deleting Tables

When a Table is deleted from the *Table List* Module and the Table Reference becomes Null, the *Size []* Module in Core will not send out a new Event. This means that deleting a Table will not be registered in Core in the same way as changing a Table. Therefore, it is important to build your own structure in Core to deal with the deletion of Tables.

Although precautions have been put in place so that REAKTOR should not crash, reading data from a deleted table is a behavior that is undefined and should be avoided.

17 Compiled Core Cell Code Cache

When opening an Ensemble with large Core Cell structures in REAKTOR, a cache file of the compiled Core Cell code is saved on your hard drive. The next time you load the Ensemble, REAKTOR will access this cache file and skip the compiling stage, which significantly speeds up the loading process.

- The Compiled Core Cell Code is only used when REAKTOR is in Play Mode.
- Switching to Edit Mode when using Ensembles with large Core Cell structures causes a re-load of the Core Cells, which can cause REAKTOR to pause momentarily.

17.1 Default Cache Location

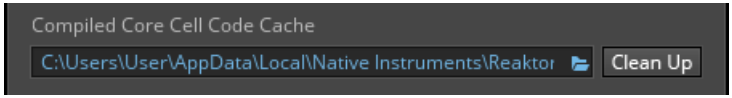
The default location of the Compiled Core Cell Code Cache is:

- Windows: C:\Users\<your user name>\AppData\Local\Native Instruments\Reaktor6\Cache \
- OS X: Macintosh HD/Users/<your user name>/Library/Application Support/Native Instruments/Reaktor6/Cache/

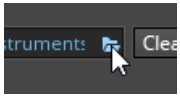
17.1.1 Changing the Compiled Core Cell Code Cache Location

To change the Compiled Core Cell Code Cache location:

1. Open the Preferences window by clicking on *Preferences...* in the File menu or by pressing [Ctrl] + [,] on Windows or [Cmd] + [,] on OS X.
2. Navigate to the Directories tab. At the bottom of the Directories tab you will see an area labeled Compiled Core Cell Code Cache.



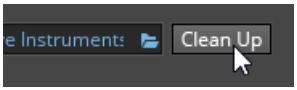
3. To change the Cache location, either click on the browser button on the left to locate the target directory on your system, or type in the location manually.



4. Click Close to close the window.
5. If your operating system prompts you, allow REAKTOR to make changes to your hard drive.

17.2 Purging the Compiled Core Cell Code Cache

▶ To delete all unused or corrupt files from the Cache directory, click on the Clean Up button under Compiled Core Cell Code Cache in the Directories tab of the Preferences.



The following files will be deleted:

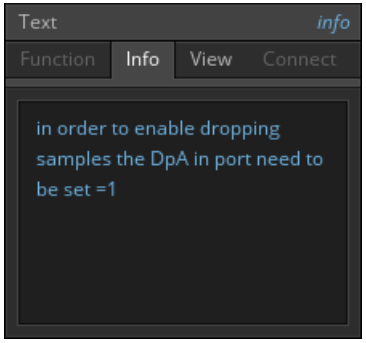
- Any Compiled Core Cell Code file for which the parent Ensemble can no longer be found
- Any corrupt Compiled Core Cell Code files

You can remove all of the files in the Cache manually, but REAKTOR will re-create the Compiled Core Cell Code files the next time you load Ensembles containing large Core Cell structures in Play mode.

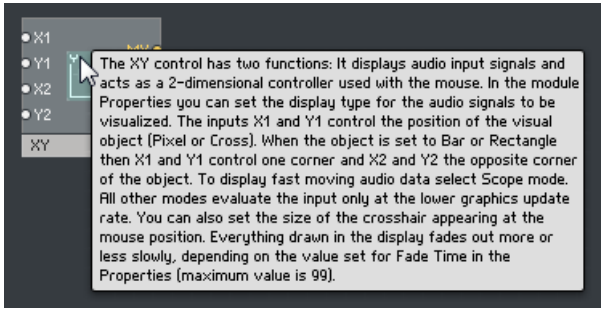
18 Module Reference

The following is a reference of all of the Primary level Modules; it will give you an overview of everything available.

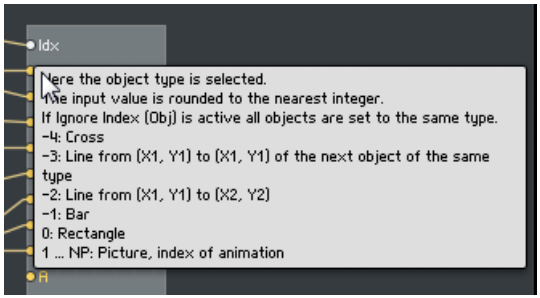
- For more information on each individual Module, you can reference the Module information from the Info tab of the Module Properties.



- You can also access this information by activating Info Hints and hovering your mouse over the Module.



- You can also get information on the Module Ports by placing the mouse over the Port.



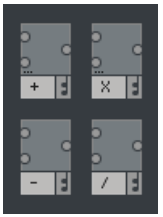
18.1 Math

18.1.1 Constant

The Constant module holds a single value, which it outputs at initialization.

The value of the Constant module can be set in the module properties or in the structure by double clicking on the module.

18.1.2 Basic Modules



The basic math Modules cover the fundamental mathematical functions:

- Add

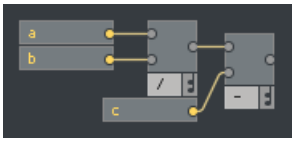
- Multiply
- Subtract
- Divide, x/y

These Modules all accept two inputs to produce a single result based on their functions. This result will be output whenever an event enters either the top or bottom input.

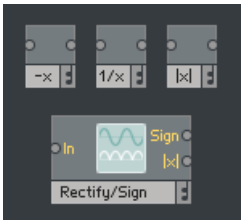
The top input will be the first number of the function and the bottom input will be the second. So, in REAKTOR format, the following formula:

```
(a / b) - c
```

Would be created like this:



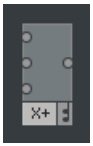
18.1.3 Modifiers



These Modules all take a single input and perform a function to produce a modified value.

- Invert, -x: Inverts the input (the equivalent of multiplying it by -1).
- Reciprocal, 1/x: The output of this module is the result of dividing 1 by the input value.
- Rectify, |x|: Can also be referred to as an 'absolute' function. The input has its sign removed, so that it is always a positive number.
- Rectify/Sign: The same process as the Rectify module, but with an additional output port which outputs a value (-1 or 1) depending on whether or not the input value is negative.

18.1.4 Mult/Add, a*b+c



Outputs the result of the formula:

```
(a * b) + c
```

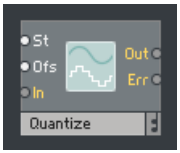
The inputs from top to bottom correspond with the variables in the formula from left to right. So the top input is `a` and the bottom is `c`.

18.1.5 Modulo

This Module takes two input values and outputs two values:

- The Div port outputs the truncated division of the inputs, so input values of A=5 and B=2 would produce an output of 2.
- The Mod port outputs the remainder of the division, so the input values of A=5 and B=2 would produce 1.

18.1.6 Quantize



Takes the input value and quantizes it according to the settings.

The module outputs the quantized value to the Out port, and outputs the difference between the original input value and the output value from the Err (error) port.

The settings can be set from the St and Ofs inputs:

- St (step size): sets the size of the steps to which the input should be quantized. So with an input of 3.2 and a St setting of 1, the Out value would be 3.
- Ofs (offset): applies an offset to the quantized value. So if the quantized value is 3, and the Ofs is set to 0.1, the Out value will change to 3.1.

This module can be used to create a simple bit-crusher effect.

18.1.7 Comparison

There are two comparison Modules that apply logic operations to two numeric inputs. If the operation is true, the output port sends a value of 1, if it is false it outputs a 0.

The modules are:

- Compare: Outputs two integer values based on testing the > (greater than) and <= (less than or equal) logic operations.
- Compare Equal: Outputs the results of 4 logic operations:
  - > (greater than)
  - < (less than)
  - = (equal to)

- != (not equal to)

As with all the mathematical modules, the top input is placed to the left of the logic operation, and the bottom input is placed to the right. Thus input A is always compared against input B in the following manner:

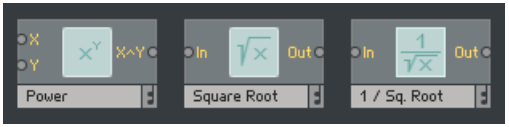
```
A > B; A<=B; etc…
```

18.1.8 Logarithms (Converters)

The logarithmic Modules in REAKTOR are primarily used to convert from one unit to another. There are 4 different modules for performing different conversions:

- Expon. (A): Converts level values in decibels (dB) to amplitude factor values. An input of -6 will produce an output of 0.5
- Expon. (F): Converts MIDI pitch values to frequency values in hertz (Hz). An input of 69 (A3) will produce an output around 440Hz.
- Log. (A): Converts amplitude factors to dB. An input of 1 will produce an output of 0.
- Log. (F): Converts frequency values (in Hz) to a MIDI pitch number. An input of 220 will produce an output of 57 (A2)

18.1.9 Power and Roots

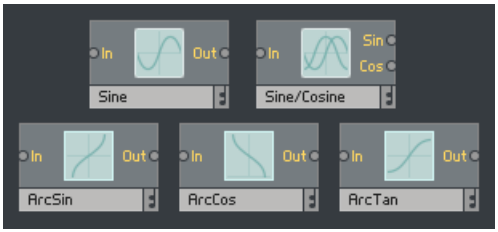


There are three mathematical Modules relating to powers and square roots.

- Power: Outputs the result of the formula `X^Y`.

- Square Root: Calculates the square root of the input value.
- 1 / Sq. Root: Calculates the reciprocal square root. The equivalent of dividing 1 by the output of the Square Root Module.

18.1.10 Trigonometric Functions



These Modules all take a single input and perform a trigonometric function to produce an output value.

The following Modules and functions are available:

- Sine
- Sine/Cosine

Inverse Trigonometric Functions

- ArcSin
- ArcCos
- ArcTan

The inverse trigonometric Modules have their inputs and/or outputs limited by necessity.

- ArcSin and ArcCos clip their inputs when they exceed the range -1 to +1
- ArcTan clips its output to the range -0.25 to +0.25

18.2 MIDI In

18.2.1 Pitch

There are two MIDI In Modules that relate to pitch:

- Note Pitch: Outputs the MIDI note value of any note message received by the Ensemble.
- PitchBend: Outputs the value of the PitchBend controller. The range of the output is set in the Properties.

18.2.2 Gate and Velocity

REAKTOR includes several Modules relating to Velocity and Gate.

Gate signals are used to communicate whenever a MIDI key is pressed and released. Velocity is the value of the strength at which these actions happen.

Gate

- Gate: Outputs the value of the note on velocity when a key is pressed, and a value of 0 when it is released.
- ST Gate (Single Trig. Gate): A monophonic variant of the Gate Module. This Module outputs a single value with the first note on and ignores subsequent note on messages. It then outputs a 0 with a note off and ignores any subsequent note off messages. Essentially this Module can be used to see whether or not the keyboard in use.
- Sel. Gate (Sel. Note Gate): Sends the gate signal from a single note, which is selected in the Module Properties.

Velocity

- On. Vel (On Velocity): Outputs Note On Velocity values, ignoring Note Off messages.
- Off Vel. (Off Velocity): Outputs Note Off Velocities, ignoring Note On messages.

Most MIDI Keyboards do not send Note Off Velocities, as such the Off Vel. Module is not guaranteed to output values.

18.2.3 Controllers

These Modules can be used to access MIDI controllers (CC) as well as aftertouch. They output a value when they receive their relative MIDI message.

- Controller: Outputs a MIDI controller value. The MIDI CC number can be selected in the Module Properties.
- Chan. AT (Ch. Aftertouch): Outputs the channel aftertouch value.
- Poly AT (Poly Aftertouch): Outputs the polyphonic aftertouch values.
- SelPolyAT (Sel. Poly Aftertouch): Outputs the polyphonic aftertouch of a single key (selected from the Module Properties).

18.2.4 Transport and Clock

There are four Modules that relate to the transport and clock (i.e. playback).

Transport

- The Start/Stop Module sends a gate signal that is set to 1 when the transport starts playback and 0 when it is stopped.
- Song Position outputs the current song position in 1/96 resolution. It has two Outputs
  - 96: An Event signal output. Since events can have their timing controlled, this output can be useful when building sequencers.
  - 96a: An Audio signal output, for sample-accurate timing.

Clock

- Clock (1/96 Clock): Outputs an event on every 1/96th note (i.e. 24 times per beat).
- Sync Pls. (Sync Clock): A MIDI synchronized clock. Its rate, length and values can all be set in the Module Properties.

18.2.5 Program Change

This Module outputs the value of MIDI Program Change messages.

18.2.6 Channel Message

The Channel Message Module outputs all MIDI messages that relate to MIDI channels:

- Note On
- Note Off
- Controllers (including Pitchbend and Aftertouch)
- Program Changes

The outputs send the MIDI message information:

- St (Status): The type of MIDI message.
- Ch (Channel): The MIDI channel.
- Nr (Number): For Note On messages, this would be the note number; for controllers, this is the CC number; and so on.
- Val (Value): The value of the message.

18.3 MIDI Out

18.3.1 Note Pitch and Gate

This Module is used to send MIDI Note Messages to other MIDI devices. The MIDI message will be sent when an event arrives at the G input.

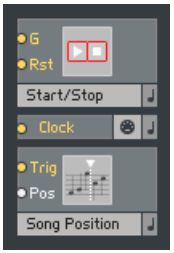
- P: Sets the MIDI Note number.
- G: Sets the MIDI Note On velocity. A value of 0 will send a Note Off message.

18.3.2 Controllers

These Modules can be used to send various MIDI Controller messages.

- Pitchbend: Sends Pitchbend messages.
- Controller: Sends MIDI Controller messages. The Controller Number is set in the Module Properties.
- Chan. AT (Ch. Aftertouch): Sends Channel Aftertouch messages.
- Poly AT (Poly Aftertouch): Sends Poly Aftertouch messages. The P input is used to set the Note Number, in the same manner as the Note Pitch and Gate Module.
- SelPolyAT (Sel. Poly Aftertouch): Sends a Poly Aftertouch message for a single note. The Note Number is set in the Module Properties.

18.3.3 Transport and Clock



These Modules can be used to send MIDI Transport and Clock messages.

- Start/Stop: Sends Transport Start/Continue/Stop Messages.
- Clock (1/96 Clock): An Event at the input sends a 1/96 clock MIDI message.
- Song Position: Sends a 1/96 resolution Song Position value when an Event with a positive value arrives at the Trig input.

18.3.4 Channel Message

The Channel Message Module can send any MIDI message that relates to MIDI channels:

- Note On
- Note Off
- Controllers (including Pitchbend and Aftertouch)
- Program Changes

The inputs set the MIDI message information:

- St (Status): The type of MIDI message.
- Ch (Channel): The MIDI channel.

- Nr (Number): For Note On messages, this would be the note number; for controllers, this is the CC number; and so on.
- Val (Value): The value of the message.

An Event at the St (Status) input triggers the MIDI message, so be sure to set this value last.

18.4 OSC

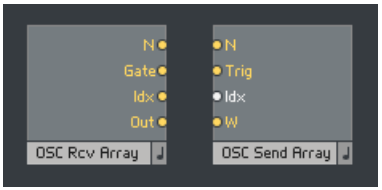
18.4.1 Send/Receive

These Modules can be used to Send and Receive OSC messages.

- The Address Pattern is set in the Connect Tab of the Module Properties.
- If the OSC message is an array, these Modules can have multiple inputs/outputs for each of the indices of the array.

When using array messages it might be better to use the OSC Array Modules.

18.4.2 Arrays



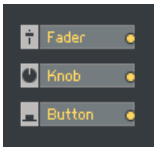
These Modules are used to send/receive OSC arrays. Rather than each index having an individual output, these Modules use the principals of iteration to send arrays of data.

- The Address Pattern is set in the Connect Tab of the Module Properties.

The tutorial ↑7, Additive Synthesizer ↑7, Additive Synthesizer covers the principals of how iteration works in REAKTOR.

18.5 Panel

18.5.1 Basic Controls



There are three basic Panel Controls. They are designed to include common functions found on GUI elements of music instruments (like MIDI Learn and Automation).

The three controls are:

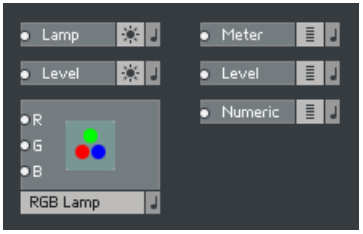
- Fader
- Knob

- Button

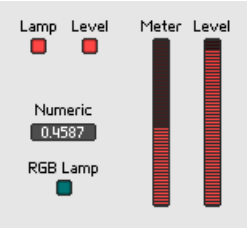
The Basic Controls on the Panel in their default states.

- All of these Modules can be given custom graphics.
- The Fader can have its orientation changed in its Module Properties.

18.5.2 Basic Displays



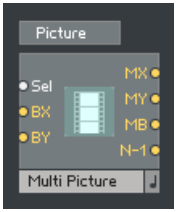
- Lamp: A small colored box, the brightness of which is controlled by the input value.
- Level Lamp: Like the Lamp, but the input uses the dB scale and is smoothed to respond better to audio signals.
- RGB Lamp: A Lamp with three inputs for setting the Red, Blue, and Green color values of the Lamp.
- Meter: Displays the input value as a bar.
- Level Meter: Like the Meter, but the input uses the dB scale and is smoothed to respond better to audio signals.
- Numeric Readout: Displays the input value.



The Basic Display Modules on the Panel in their default states.

The colors of the Lamps and Meters are set in the Module Properties.

18.5.3 Pictures



The Picture Modules are used to display images on the Panel.

- Picture: Displays a single static image.
- Multi Picture: Can be used to read an animation-style image file. It also has outputs for mouse interaction and thus can be used as a custom control.

18.5.4 Text

The Text Modules are used to display text on the Panel.

- Text: Displays static text.
- Multi Text: Contains a list of text options; the displayed is selected from the input.

The text displayed by these Modules is set in the Info tab of the Module Properties.

These Modules have several font options, which are set in the Module Properties. There are 9 available fonts, the color and size of which are fully customizable.

The Available Fonts

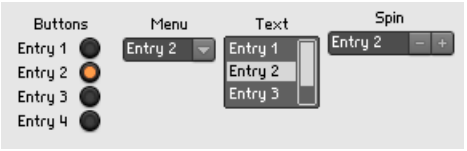
18.5.5 List

The List Module offers the user a selection of entries that they can choose from.

The entries and their values are set in the Module Properties.

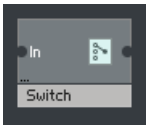
This Module can have four different appearances:

- Radio Buttons
- Dropdown Menu
- Text List
- Spin



The Different List Appearance Options

18.5.6 Switch



In the Structure, the Switch functions like an M -> 1 Router. It is used to select between multiple inputs and then routes the selected input to the output.

The main benefit of using a Switch is that it deactivates any Modules attached to the unselected inputs. Thus the Switch can be used as a CPU saving feature.

The other benefit is that the Switch already includes a Panel element, in the form of a Radio Button List.

18.5.7 Receive

The Receive Module can connect to any Send Module and can be used to create wireless connections.

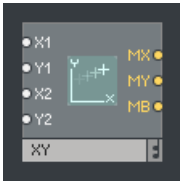
On the Panel, the Receive Module is presented like a List.

This Module is the same as the Receive Module located in the Terminal section of the Built-in Modules.

See section ↑18.17.2, Send/Receive ↑18.17.2, Send/Receive for information on the Terminal Modules.

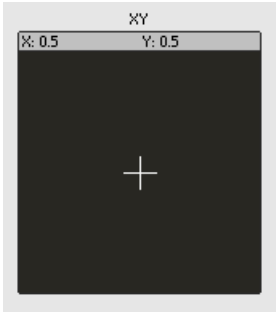
Note that Send/Receive Modules should be used in moderation due to their potential performance overhead. See section ↑14.7, Avoid unnecessary use of Send/Receive Modules ↑14.7, Avoid unnecessary use of Send/Receive Modules for more details.

18.5.8 XY



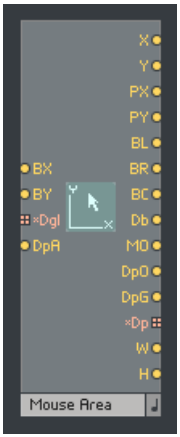
The XY Module is an area that outputs the mouse X and Y positions while the mouse button is pressed.

This Module also has a display element: it can accept up to two coordinates and draw an object in that space. This feature can even be used by audio signals as a scope.



The XY Module on the Panel

18.5.9 Mouse Area



This Module sends detailed information about the state of the mouse.

As well as X, Y coordinates and button states, the Mouse Area can also access Double-clicking and the mouse-over state.

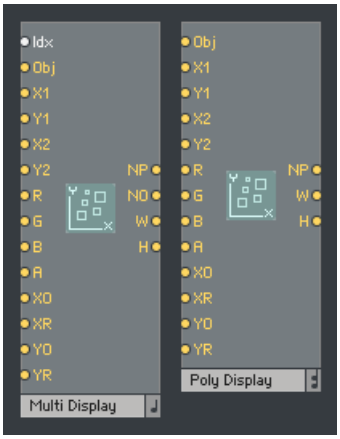
All of these features make the Mouse Area an important asset when creating custom interface elements.

The Mouse Area is used to create a custom control in the tutorial in section ↑6, Advanced Step Sequencer ↑6, Advanced Step Sequencer.

The Mouse Area can also be used in conjunction with the Table Framework to accept audio files for drag and drop.

The Drag and Drop feature of the Mouse Area is examined in section ↑8, Drag and Drop Sampler ↑8, Drag and Drop Sampler.

18.5.10 Advanced Displays



The Multi Display and the Poly Display and be used to create various numbers of objects.

These objects can be simple shapes (like lines and rectangles) or they can be custom images.

- The Multi Display can have the number of objects defined in its Properties.
- The Poly Display creates an object for each polyphonic voice in the Instrument.

The tutorial in section ↑6, Advanced Step Sequencer ↑6, Advanced Step Sequencer shows how to use a Multi Display.

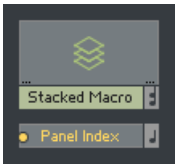
18.5.11 Scope

The Scope Module can be used to create oscilloscopes.

It samples the signal at the In input and then draws a trace of the input over time when a positive Event arrives at the Trig input.

The ranges and offsets of the time axis and Y axis can be set using the inputs.

18.5.12 Stacked Macro



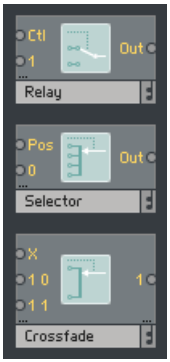
Stacked Macros can be used to create dynamic control panels.

- The Stacked Macro can contain any number of Macros, but it will only display the Panel of one of them at any one time.
- The displayed Macro is selected using the Panel Index, which must be placed inside the Stacked Macro.

Section ↑10.8, Stacked Macros ↑10.8, Stacked Macros describes the use of Stacked Macros in more detail.

18.6 Signal Path

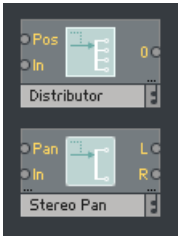
18.6.1 Selectors



There are three Signal Path Modules that can be used to select between multiple signals to output a single signal.

- Relay: A switch style selector.
- Selector: Can be used to blend between different inputs.
- Crossfade: Like the Selector, but it pairs the signals and crossfades them as a stereo pair.

18.6.2 Distributors



There are two distribution Modules. They take an input signal and route it to multiple outputs.

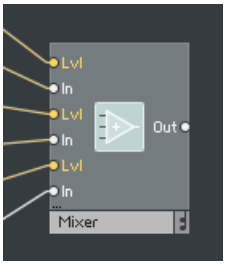
- Distributor: This Module is like the Selector in reverse. It takes an input and routes it to different outputs.
- Stereo Pan: A variation of the Distributor that takes an input and blends it between two outputs (one for each of the stereo channels).

18.6.3 Mixers

These Modules act like audio mixing amplifiers.

- Mixer: Takes an input signal (In) and alters its level using the decibel scale (Lvl).
- Stereo Mixer: An expansion of the Mixer Module, with two outputs (one for each of the stereo channels) and a Pan input to blend the input between these outputs.

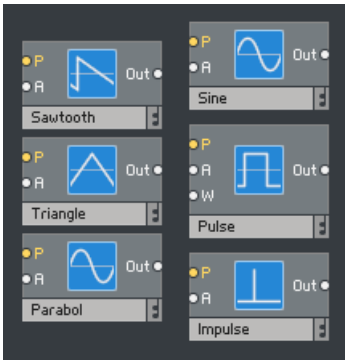
You can add multiple inputs to these Modules, meaning that a single Mixer Module can be used to mix together several signals into one.



The Mixer with multiple inputs

18.7 Oscillators

18.7.1 Basic Audio Oscillators



The basic audio oscillators produce an audio rate waveform based on the two input parameters:

- P - MIDI Pitch
- A - Amplitude

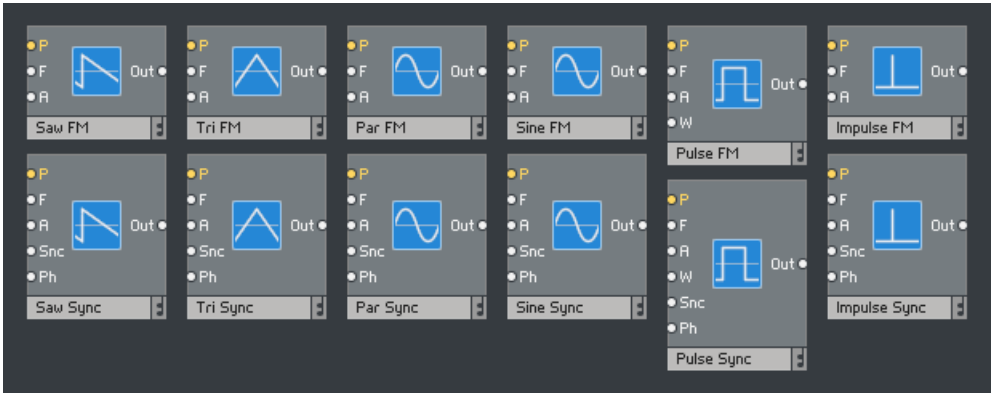
The basic oscillator modules are available in the following waveforms:

- Sawtooth
- Triangle
- Parabol
- Sine
- Pulse
- Impulse

The Pulse waveform has an additional input (W) that controls the width of the pulse cycle.

FM and Sync variants

All of the basic audio oscillators are all also available as FM and Sync oscillators.



The FM oscillators add an additional input, which allow them to be used for audio rate Frequency Modulation (FM):

- F - Linear frequency control, which is added to the frequency of the P input.

The Sync oscillators add two inputs to the FM oscillator:

- Snc - when the input rises above zero, the oscillator re-starts at the phase set by the Ph input.
- Ph - sets the re-starting phase point.

The FM and Sync variants will have a heavier impact on the CPU, so only use them when you need the additional functionality.

18.7.2 Variants

Some of the Basic Audio Oscillators also feature unique Variants. These offer extra control inputs that affect the shape of the waveform in some way.

Saw

- Saw Pulse: Provides control over the steepness of the saw slope.
- Bi-Saw: Provides control over the width of the saw.

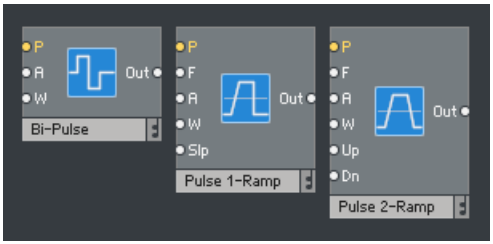
Triangle/Parabol

- Tri/Par Symm: Includes two outputs for Triangle and Parabol waveforms. The Symmetry of the waveforms can be controlled via the W (width) port.

This Module gives you the ability to morph between a Triangle wave and a Saw wave.

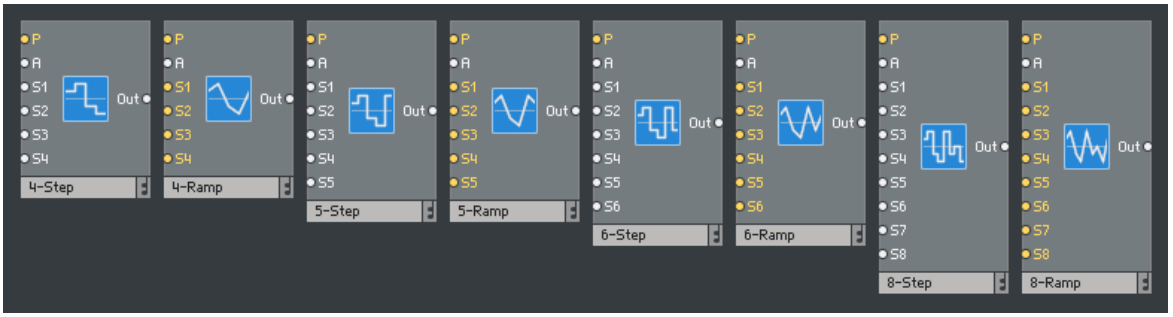
- Par PWM: A Parabol waveform that can have its width controlled. The resulting shapes are different from the Par output of the Tri/Par Symm Module.

Pulse



- Bi-Pulse: The Width (W) input controls the width of the whole pulse waveform, rather than one side of it.
- Pulse 1-Ramp: Provides control over the slope steepness of the rising edge of the waveform.
- Pulse 2-Ramp: Provides control over the slope steepness for both the rising and falling edges of the waveform.

18.7.3 Multi-step



The Multi-Step Oscillators enables you to draw your own waveforms by entering values for multiple steps in the waveform. The oscillator reads through these values like a table.

They come in two main types with varying numbers of steps:

- **Step**: Values change instantaneously, producing a blocky waveform.
  - 4-Step
  - 5-Step
  - 6-Step
  - 8-Step
- **Ramp**: Values change gradually, producing a smoother waveform.
  - 4-Ramp
  - 5-Ramp
  - 6-Ramp

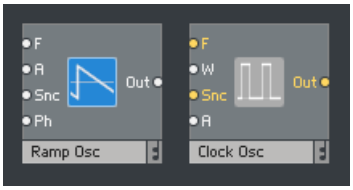
- 8-Ramp

18.7.4 Noise

There are three Modules that use random numbers to produce a signal:

- Noise: Randomly outputs a value of -0.5*A or 0.5*A at the Sample Rate. Filter Modules can be used to change the color of the noise.
- Random: Produces a random number at a rate defined by the P input, and in a range defined by the A input.
- Geiger: Outputs clicks (a value of 1 followed instantly by a value of 0) at definably random intervals.
  - The rate of the clicks is controlled by the P input and the randomization is controlled by the Rnd input.
  - The Geiger Module also has an Event output, which generates Events at the same random interval. For example, it can be used to trigger Envelopes at random intervals.

18.7.5 Utility



The Utility Oscillators are not designed to be used as Audio signals, but are tools that behave like oscillators.

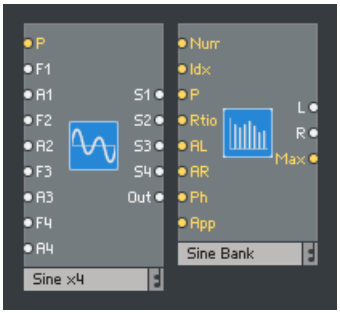
- The Ramp Osc ramps from 0 to the value defined at input A over the period defined by the F input.

The Ramp Osc is a useful driver for wavetables oscillators.

- The Clock Osc produces an Event of value A followed by an Event with a value of 0 at a rate defined at the F input.

The Clock Osc is used in the tutorial in section ↑5, Basic Step Sequencer ↑5, Basic Step Sequencer

18.7.6 Additive



Additive synthesis uses large numbers of sine waves to create complex sounds.

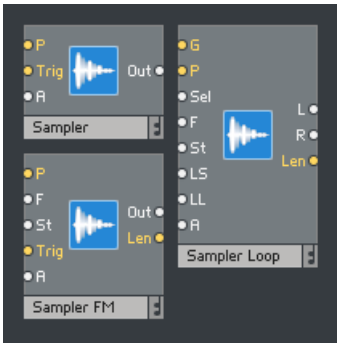
REAKTOR has two oscillators designed to use this technique:

- The Sine x4 Module essentially stacks four Sine Oscillator Modules inside one Module.
- The Sine Bank is optimized to produce large numbers of sine waves. Therefore, rather than having individual inputs for the parameters of each of the sine waves, the parameters are set as values in an array.

The Sine Bank is one of the main subjects in the tutorial in section ↑7, Additive Synthesizer ↑7, Additive Synthesizer.

18.8 Samplers

18.8.1 Basic

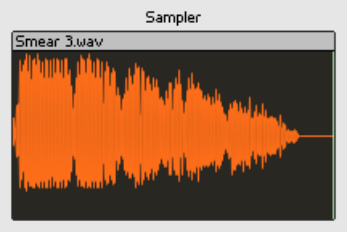


The Basic Samplers play back audio samples from a Sample Map.

- Sampler: Plays back the samples as they are mapped.
- Sampler FM: Provides the ability to modulate the frequency of the sample playback, as well as the option to define the sample start offset.
- Sampler Loop: Like the Sampler FM Module, but with inputs for settings the loop start and end points for the sample.

Sample Maps and the Sample Map Editor are described in more detail in the REAKTOR 6 Diving Deeper document.

Sampler Modules all have a Panel element, which is a display of the current sample waveform, as well as overlays for the loop points and the playback head position.



The Sampler Module Panel Element

18.8.2 Granular

The Granular Samplers cut the samples into small sections (called grains) and process each grain separately. This can produce choppy effects, or be used to individually control pitch and time.

- Resynth: The standard granular sampler, with control over the grain settings and loop points.
- Pitch Former: Like the Resynth, but with a Formant Shift (FS) input.

- Grain Cloud: Creates a number of grains (i.e. a cloud), with control over the envelope of the grains, and independent control over the distance between the grains and their length. Several of the parameters have Jitter inputs that control a randomization of their respective partner.

18.8.3 Beat Loop

Beat Loop is a granular sampler that aims to keep the tempo of the sample locked to the tempo defined by the Clk (1/96 clock) input.

This Module requires that the sample is an exact number of bars, preferably an even number like 1, 2, 4, 8, 16, etc...

18.8.4 Lookup

The Sample Lookup Module allows you to scrub through the waveform of the sample by manually setting the playhead position using the Pos input.

Unlike the other Sampler Modules, this Module does not use the Sample Map Editor. It can only hold one sample at a time and that sample is set in the Module Properties.

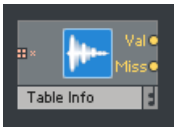
18.9 Table Framework

18.9.1 Table List

This Module is the main Table manager and lets you maintain the list of active Tables, reorder and select between them.

The Table List is the only table reference module with Snapshot support.

18.9.2 Table Info



The Table Info outputs a specific piece of information (meta-info) about the current table, such as Sample Rate, Table Size, Loop Points, etc.

- The table meta-info to be read is selected in the Module Properties.
- If the meta-info is not present in the selected table, then the Miss output sends a value of 1.
- If the meta-info is present, then the value is sent from the Val output.

18.9.3 Sample Map

This Module takes sample information from the Sample Map and outputs it as a table reference. It similar in function to the Sampler Module (see section ↑18.8, Samplers ↑18.8, Samplers), only it outputs the selected sample as a table reference rather than as an audio signal.

- The sample is selected using the pitch (P) and velocity (Vel) inputs.
- The sample is then sent to the output as a table when an event is sent to the Trig input.

18.10 Sequencer

18.10.1 Basic

The Sequencer Modules allow you to create sequences of values, which are read at a rate defined by the Clk input.

- Every time a rising value passes 0 at the Clk input, the sequence progresses to the next step. When the sequence passes the last step, it wraps around again.
- You can reset the sequence position using the Rst input.

There are 4 Sequencer Modules, each defined by their length:

- 6-Step
- 8-Step
- 12-Step
- 16-Step

18.10.2 Multiplex 16

The Multiplex 16 Module is more like a value selector than the Sequencer Modules.

- You can set the length of the sequence using the Len input and then set the sequence position using the Pos input.
- A value is read every time the Pos input receives an Event, so the Pos input also acts as the clock input.

18.11 LFO, Envelope

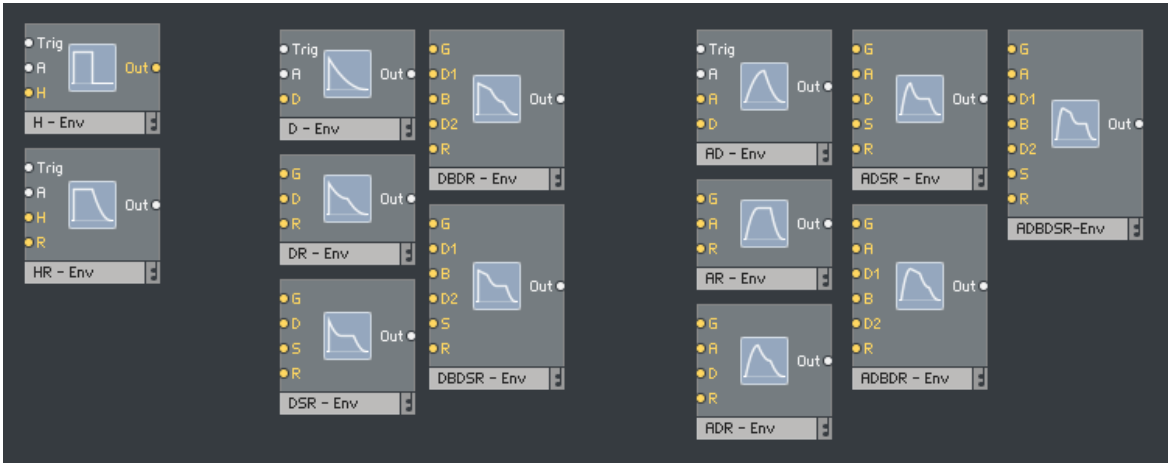
18.11.1 LFO

LFOs are Low Frequency Oscillators, i.e. Oscillators that are designed to run at a frequency lower than the range of human hearing.

The LFO Modules are Oscillators that output Events rather than Audio signals.

- LFO: A standard LFO with three different waveforms available from individual outputs.
- Slow Random: Produces a random value at the Control Rate. The values are smoothed by an internal filter, the cutoff of which is defined at the F input.

18.11.2 Envelopes



There are 13 different Envelope Modules, each with a different set of parameters that affect the envelope shape and behaviors in different ways. They all use the following abbreviations to show how they function:

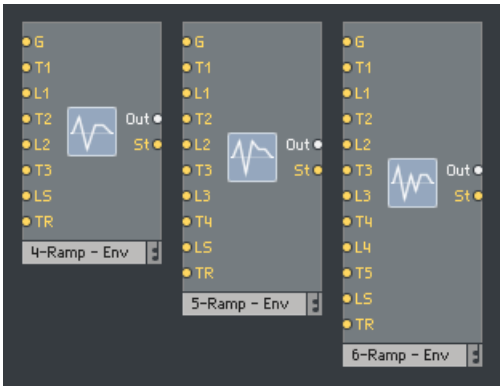
- A: Attack - A fade-in at the start of the envelope.
- D: Decay - A gradual reduction in level.
- H: Hold - Keeps a value level for a set amount of time.
- B: Break - A Break-point level in the envelope. Usually this is a level between two decay phases.
- S: Sustain - The level at which the envelope remains for as long as the gate is active.
- R: Release - The fade-out time after the gate is deactivated.

The available Envelopes mix and match these functions.

The most common Envelope type for synthesizer is the ADSR Envelope.

- With the exception of the HR Envelope, all of the Envelopes with a release phase (R) use a Gate (G) input as this tells the envelope whether or not the note is being held. The positive Gate value also sets the peak value of the Envelope.
- The Other Envelopes use an A input to set the peak value of the Envelope, and a Trig input to trigger the start of the envelope.

18.11.3 Multi-ramp



The Multi-ramp Envelopes allow you to define the shape of the Envelope.

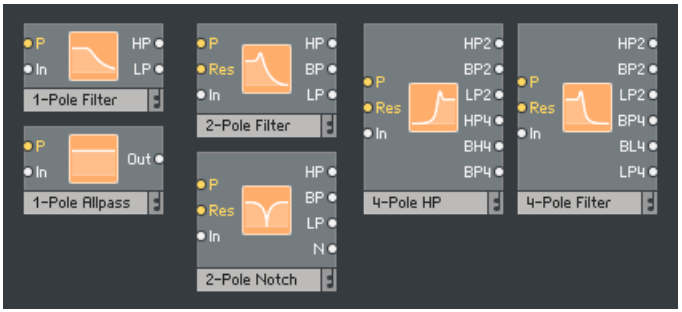
- Each point is defined by the time it takes to get there (T) and its level (L).
- These Envelopes also have a sustain level (LS) and a release time (TR).

There are three Multi-ramp Envelopes, defined by the number of ramps available (note that the release phase counts as a ramp):

- 4-Ramp
- 5-Ramp
- 6-Ramp

18.12 Filters

18.12.1 Basic



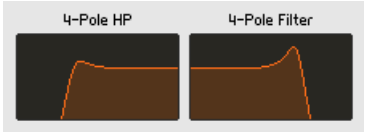
The Basic Filter Modules are designed to filter audio signals.

There are three steepness levels available:

- -Pole (6dB/Octave). The 1-Pole filters do not have resonance options.
- -Pole (12dB/Octave)
- -Pole (24dB/Octave)

As the number of poles increases, so do the number of available filter outputs. Note that the 4-Pole filters both contain the options available in the 2-Pole Filter.

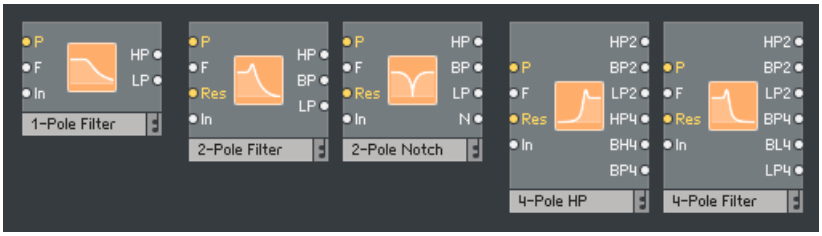
The Filters also have a panel element that can be activated from the View tap of the Module Properties.



The Panel Display of the Filter Modules

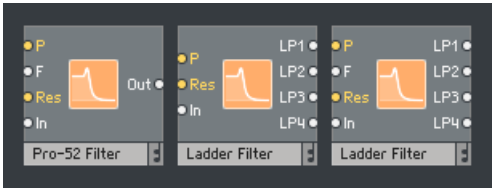
Note that the plot shown is that of the primary filter type. So the 4-Pole HP Filter will always show a high-pass filter plot, even if you are using the low-pass or band-pass outputs.

FM



The Basic Filters also have FM variants, which can have their cutoff frequency modulated at the audio rate.

18.12.2 Modeled



There are three Filter Modules that are based on hardware filters:

- Pro-52: This is an emulation of the Prophet 5 filter that was used in the now discontinued Pro-52 software.
- Ladder: An emulation of the classic Ladder Filter design. It has four outputs for different numbers of poles (i.e. the filter steepness).
- Ladder FM: Similar to the Ladder Module, but with an F input for frequency modulation.

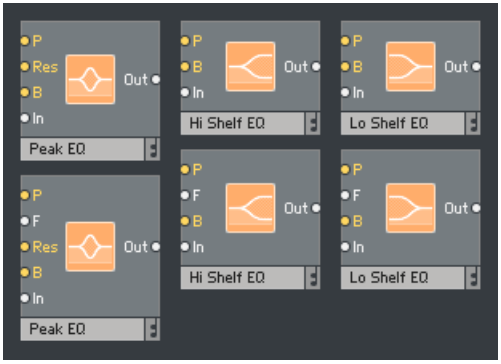
18.12.3 Modal Bank

The Modal Bank creates a large number of resonant band-pass filters. When these filters are excited by an input signal, they produce a tone. This process is called modal synthesis and is generally considered to be good for physical modeling (although it can take some fine-tuning to get the right results.

The Modal Bank functions similarly to the Sine Bank, in that the parameters of the individual band-pass filters are set in an array.

The tutorial in section ↑7, Additive Synthesizer ↑7, Additive Synthesizer goes into more detail about using the Sine Bank. The same principals apply to the Modal Bank.

18.12.4 Equalizers



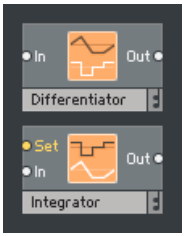
Equalizers (or EQ for short) are used to alter the levels of a select band of frequencies.

There are three EQ types available:

- Peak EQ
- Hi Shelf EQ
- Lo Shelf EQ

Each type comes with and FM variant.

18.12.5 Differentiator, Integrator



- Differentiator: A signal slope detector. Calculates and outputs how fast the input signal is rising or falling.
- Integrator: A ramp generator. Creates a ramp with a slope defined by the value at the input.

18.13 Delay

18.13.1 Basic

The Delay Modules take an input signal and delay it in time.

There are three Basic Delay Modules:

- Unit Delay (T): Delays the input signal by one sample. This Module is useful in feedback paths.

If you do not insert a Unit Delay in your feedback paths, REAKTOR will do it automatically. The location of the delay is marked by a Z.

- Single Delay: Delays the input signal by the time specified at the Dly Port.

The Single Delay is the main Module in the tutorial in section ↑4, Echo Effect Macro ↑4, Echo Effect Macro.

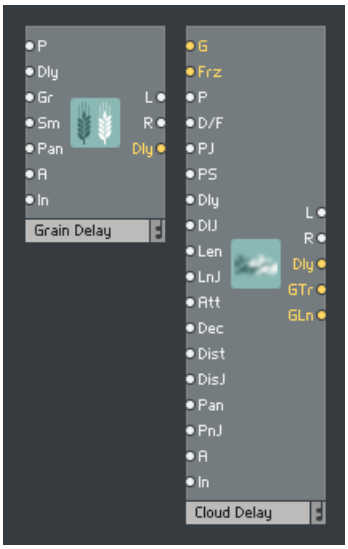
- Multi-Tap Delay: Can create up to 8 different delayed signals from a single input.

18.13.2 Diffuse

The Diffuser Delay Module is an all-pass filter containing a delay line with feedback. Its function is to smear (decorrelate) the input signal without emphasizing any particular frequency component. The delay time can be continuously modulated by an Audio signal.

These features make the Diffuser Delay a good component for a reverb effect.

18.13.3 Granular



The Granular Delay Modules cut up the input signal into small sections (grains). Essentially these Modules are effect counterparts to the Granular Sampler Modules.

- Grain Delay: This module allows you to modulate the delay time without affecting the pitch, and vice versa.
- Grain Cloud Delay: Gives you much more control over the parameters of the grains. This Module is an effect version of the Grain Cloud from the Samplers.

18.14 Audio Modifier

18.14.1 Saturators and Distortions

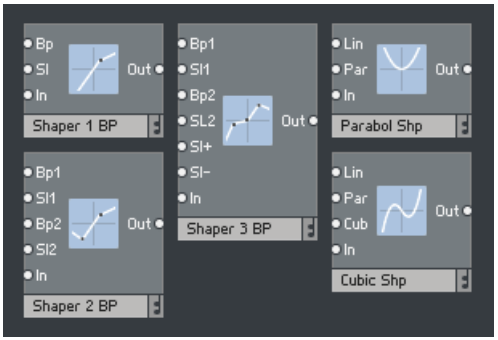
These Modules distort the input signal.

There are seven Modules available:

- Saturator: A soft saturation Module. The saturation amount depends on the level of the input.
- Saturator 2: A more advanced version of the Saturator. Gives you access to additional parameters that affect the character of the saturation.
- Clipper: Clips the input when it goes beyond certain threshold levels, which are defined at the Max and Min inputs.

- Mod. Clipper: A variation on the Clipper, optimized for modulation of the clipping threshold.
- Mirror 1: Rather than clipping the signal, this Module "reflects" any signal that passes above the threshold level.
- Mirror 2: Similar to the Mirror 1 Module, but is able to reflect signals at both an upper limit and a lower limit.
- Chopper: When the value at the M Port is positive, the signal at the In Port is multiplied by the value at the X Port. When the M value is negative, the In signal is left unchanged.

18.14.2 Shapers



The Shaper Modules also distort the input signal, but use waveshaping techniques.

There are three Break-Point (BP) based Shapers:

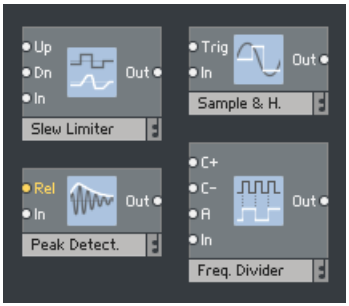
- Shaper 1 BP: When the input rises above the break-point (Bp), its slope is adjusted by the Sl value.
- Shaper 2 BP: Like the Shaper 1BP, but with an additional lower break-point (Bp2), below which the slope is affected by the Sl2 value.
- Shaper 3 BP: An expansion on the Shaper 2BP Module, with control of the slope of the signal between zero and Bp1, and control over the slope between zero and Bp2.

There are two mathematical Shapers:

- Parabol Shp (Shaper Parabolic): A Shaper with a parabolic curve. It allows you to mix the undistorted signal (Lin) and the parabolic signal (Par).

- Cubic Shp (Shaper Cubic): Adds cubic distortion to the Parabol Shp Module.

18.14.3 Modifiers

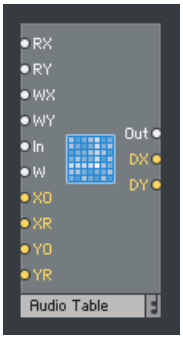


These Modules take the input signal and use it as the basis of a new output signal.

- Slew Limiter: Follows the input signal, but at a limited rate of change. The upward rate and downward rate can be set independently.
- Peak Detect. (Peak Detector): The input signal is rectified and then smoothed by an adjustable release time (Rel). The attack time is fixed at 0ms.
- Sample & H. (Sample + Hold): The input signal is sampled and held each time the Trig input rises above 0.
- Freq. Divider (Frequency Divider): The zero-crossing slopes of the input signal are counted and the output signal only changes when the counters reach the values set at the C+ and C- inputs.

The Frequency Divider Module is the same as the Frequency Divider found in the Event Processing section.

18.14.4 Audio Table



The Audio Table holds a two-dimensional table of values, which can be read and written at the audio rate.

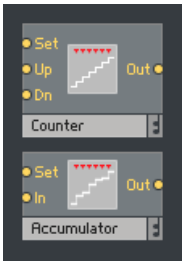
The Audio Table can be used to create a crude sampler or wavetable.

The Audio Table functionality is very similar to the Event Table, which is used in the tutorial in section ↑5, Basic Step Sequencer ↑5, Basic Step Sequencer.

The Audio Table is not part of the Table Framework.

18.15 Event Processing

18.15.1 Counters



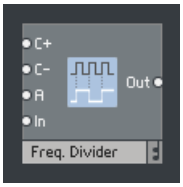
There are two Event Counters in REAKTOR:

- Counter: Adds or subtracts 1 from an internal value each time an Event arrives at Up or Dn (Down) inputs. The internal value can be set to a specific value by using the Set input.
- Accumulator: Adds the value of the input Event to an internal value. The internal value can be set to a specific value by using the Set input.

18.15.2 Random

The Random Module modifies the input Event value by a random deviation. The deviation amount is set at the Rng input.

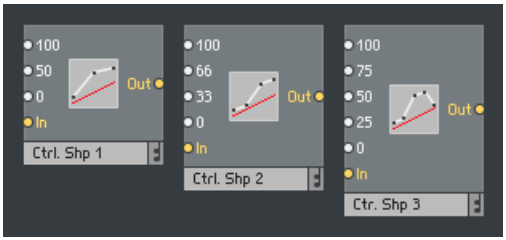
18.15.3 Frequency Divider



The zero-crossing slopes of the input signal are counted and the output signal only changes when the counters reach the values set at the C+ and C- inputs.

This Module is the same as the Frequency Divider found in the Audio Modifier section.

18.15.4 Control Shapers



The Control Shapers allow you to alter the slope or curve of Event signals. Unlike the Audio Shapers, these Modules remap specific values and thus alter the control curve.

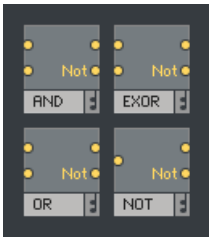
To define the slopes, you set the values of several points; Input values are changed to fit their respective location on the lines these points draw.

Note that the remap values as they are written at the Ports are presented as percentages, but the input requires a range of 0 to 1. So the 50 Port defines the value remap of a 0.5 input and the 100 Port defines the remap of a 1 input.

- Ctrl. Shp 1 (Ctrl. Shaper 1 BP): Allows you to alter the minimum point, a midpoint value, and the maximum point, creating two different slopes.
- Ctrl. Shp 2 (Ctrl. Shaper 2 BP): Allows you to alter the minimum point, the maximum point, and two midpoints (33 and 66), creating three different slopes.
- Ctrl. Shp 3 (Ctrl. Shaper 3 BP): Allows you to alter the minimum point, the maximum point, and three midpoints (25, 50, and 75), creating four different slopes.

Control Shapers can be useful for changing the scaling of a Knob or Fader.

18.15.5 Logic



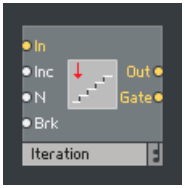
The Logic Modules are used to perform logic (Boolean) processes.

- When an input is greater than 0 it is considered to be **True**.
- When an input is less than or equal to 0 it is considered to be **False**.

The available Modules and processes are:

- AND
- OR
- EXOR
- NOT

18.15.6 Iteration



The Iteration Module is REAKTOR's equivalent of a loop.

- The input Event is set to the output along with N additional Events. The value of each Event increments upon the previous Event by the amount set at the Inc input.
- The iteration process can be set to a limited speed in the Module Properties.

- The iteration process can be stopped by sending a positive Event to the Brk input.

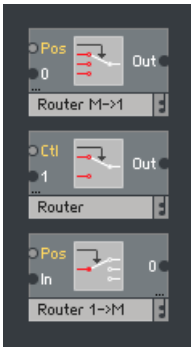
The Iteration Module is one of the main topics covered in the tutorial in section ↑7, Additive Synthesizer ↑7, Additive Synthesizer.

18.15.7 Signal Flow

These Modules control the flow of Event signals.

- Order: An Event at the input is set to the outputs in a defined order.
- Separator: Separates an Event to one of the two outputs depending on whether it is above or below the Threshold (Thld) value.
- Value: An Event at the Trig input will send an Event with the value of the signal at the lower input. Note that this Module can also be used to trigger the transmission of Table References.
- StpFlt (Step Filter): An incoming Event is only passed to the output if its larger or smaller than the previous value by an amount defined at the Tol input.
- M (Merge): This Module is used to merge multiple signals into one. Whichever input received an Event last will have its value sent to the output. If multiple Events arrived simultaneously, the lowest input will be used.

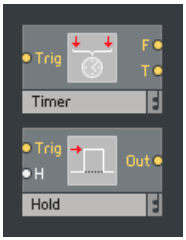
18.15.8 Routers



These Modules are used to route Events to/from different parts of the Structure.

- Router M->1: A "many to one" router. The Pos input is used to select which input signal is sent to the output.
- Router: This Module is a precursor to the Router M->1 Module. It can only accept 2 signals, and switches to the 1 signal when the Ctrl input is >=1
- Router 1->M: A "one to many" router. The Pos input is used to select to which output the input signal will be sent.

18.15.9 Timing

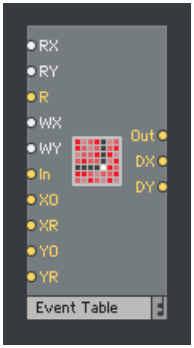


These Modules can be used to measure or manipulate the timing of Events.

- Timer: The T Port outputs the number of milliseconds between an Event at the Trig input and the previous Event at that input. The F Port outputs the frequency of Events (by calculating 1/T).
- Hold: When an Event arrives at the Trig input, it is held at the output for an amount of time defined at the H input. Once the hold time ends, the output returns to zero.

The timing of the Hold Module is course, as it uses the Control Rate clock for better CPU performance. For finer control over the Hold time, use the H Envelope from the LFO, Envelope section, as this uses the Sample Rate clock.

18.15.10 Event Table

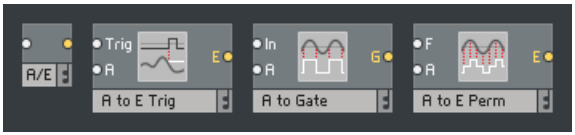


The Event Table holds a two-dimensional table of values.

The tutorial in section ↑5, Basic Step Sequencer ↑5, Basic Step Sequencer makes extensive use of the Event Table.

18.16 Auxiliary

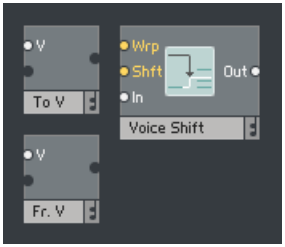
18.16.1 Audio to Event



These Modules are all designed to convert Audio signals into Event Signals.

- A/E (A to E): Simply resamples the incoming Audio signal at the Control Rate.
- A to E Trig: The A value is sampled when the Trig input has a rising edge.
- A to Gate: Converts the Audio signal into a Gate. A rising edge opens the gate, and a falling edge closes it. The on value of the gate is set at the A input.
- A to E Perm: Resamples the value A at a rate set at the F input.

18.16.2 Voice

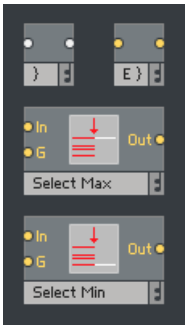


These Modules provide you with the ability to alter or interact with the polyphonic voices.

- To V (To Voice): Sends a monophonic signal to a specific voice in a polyphonic signal. The voice number is specified at the V input.

- Fr. V (From Voice): Outputs a monophonic signal by selecting the signal from a single polyphonic voice. The voice number is specified at the V input.
- Voice Shift: Can be used to shift a signal from one voice to another.

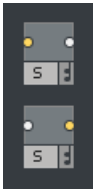
18.16.3 Voice Combiners



Voice Combiners are used to convert polyphonic signals into monophonic signals.

- } (Audio Voice Combiner): Sums the voices of a polyphonic audio signal into a monophonic audio signal.
- E } (Event V.C. All): Merges the voices of a polyphonic Event signal. The value of the last Event to arrive at the input will be the value at the output.
- Select Max (Event V.C. Max): Selects the voice with the current highest value to output as a monophonic signal.
- Select Min (Event V.C. Min): Selects the voice with the current lowest value to output as a monophonic signal.

18.16.4 Smoothers



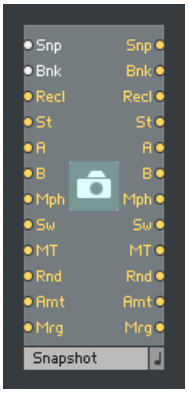
Smoothers slow the rate of change in values, effectively smoothing transitions.

The smoothing time is set in the Module Properties.

There are two Smoothing Modules:

- Audio Smoother: Takes an Event input and creates smoothed transitions at the Audio Sample Rate. This Module is better for higher quality smoothing.
- Event Smoother: Creates smoothed transitions at the Control Rate. This Module is better for performance.

18.16.5 Snapshot



The Snapshot Module has inputs and outputs that relate to almost all of the controls in the Snapshot tab of the Sidepane.

Not only does this Module output the current Bank and Snapshot numbers, but it also sends Events on Snapshot recall (Recl) and storage (St), which can be useful when initializing the instrument after a Snapshot loads.

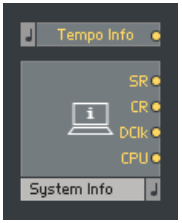
18.16.6 Snap Values

These Modules hold values that are stored and recalled with Snapshots.

- Snap Value: Stores and recalls a single value with the Snapshot.
- Snap Value [] (Snap Value Array): Stores and Recalls an array of values with the Snapshot.

The Snap Value Array is used in the tutorial in section ↑6, Advanced Step Sequencer ↑6, Advanced Step Sequencer.

18.16.7 System



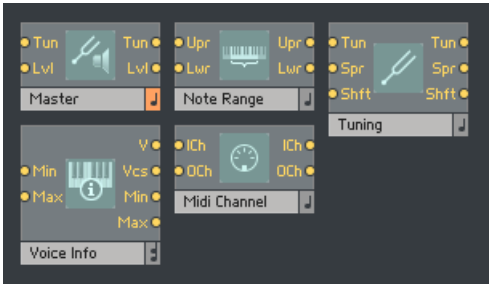
The System Modules output information about REAKTOR.

- Tempo Info: A source for the current tempo in Beats per Second.
- System Info: Contains several outputs relating to the REAKTOR engine:
  - SR (Sample Rate): The audio engine Sample Rate.

- CR (Control Rate): The Control Rate for Event signals.
- DClk (Display Clock): Sends an Event before the interface refreshes, which should happen around 25 times per second.
- CPU: A measurement of the current CPU load.

The Display Clock is a useful tool when optimizing custom interfaces. See section ↑14.4, Use the Display Clock ↑14.4, Use the Display Clock for more information.

18.16.8 Instrument Properties



These Modules can be used to retrieve or define certain settings in the Instrument (or Ensemble) Properties.

The available Modules are:

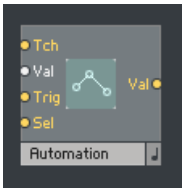
- Master: Accesses the main output volume and tuning.
- Note Range: Accesses the MIDI Note range.
- Midi Channel: Accesses the MIDI input and output channels.
- Tuning: Accesses the tuning properties.
- Voice Info: Accesses the polyphony settings, including an output (V) that sends the Voice ID on that Voice.

18.16.9 Spread

The Spread Module is only of use when the Ensemble it is inside is polyphonic.

This Module generates a different offset value for each voice, which can be used to create unison spread in other Modules.

18.16.10 Automation



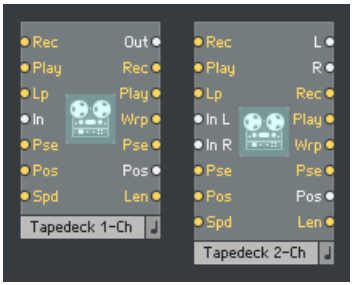
This Module is used to send and receive automation to/from a host program.

Using this Module is described in more detail in section ↑11.2, The Automation Module ↑11.2, The Automation Module.

18.16.11 Set Random

Sets the seed value for the random number generation within the Ensemble.

18.16.12 Tapedeck



The Tapedeck Modules essentially mimic the behavior of the Player and Recorder, but as Modules in the structure.

You can use them to record audio, save it, and play it back.

The Tapedeck is available in two versions:

- Tapedeck 1-Ch: A monophonic audio recorder.
- Tapedeck 2-Ch: A stereo audio recorder.

18.17 Terminal

18.17.1 Ports

Ports are an essential part of building in REAKTOR. They allow the contents of one level of the Structure to send values to the level above.

The final Ports are the Ensemble Ports, which send audio to the host software or computer soundcard.

18.17.2 Send/Receive

The Send and Receive Modules can be used to make wireless connection in the Ensemble.

The Receive Module listed here is the same as the Receive Module from the Panel section.

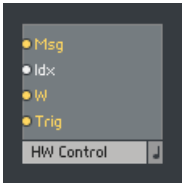
Caution is advised when using these Modules, as they can impact performance and loading times. See section ↑14.7, Avoid unnecessary use of Send/Receive Modules ↑14.7, Avoid unnecessary use of Send/Receive Modules for more details.

18.17.3 IC Connections

The IC Send and IC Receive Modules use the Internal Connection (IC) Protocol to wirelessly send signals.

The IC Protocol and its use is documented in section ↑13, Internal Connection Protocol ↑13, Internal Connection Protocol.

18.17.4 Hardware Control



The Hardware Control Module (HW Control) is used to communicate with NATIVE INSTRUMENTS hardware.

The use of the Hardware Control Module is documented in section ↑12.4, Hardware Control ↑12.4, Hardware Control.

Index

Numerics

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W