



REAKTOR 4

SUPPORT N.I PLEASE! THEY MAKE THE BEST SOFTWARE ON THE PLANET!

ENGLISH

OPERATION MANUAL



All content text has been **HYPERLINKED** for ease of use. You're welcome!

Table of Content

Introduction 1

- What is REAKTOR ? 1
- Loading older Ensembles 1
- The Browser window 2
- The Snapshot window 2
- Sample Map Editor 2
- Internal Communication 3
- New Modules 3

Product Authorization 4

- What is the Product Authorization? 4
- Conducting the Product Authorization 5
- Registration support 10

Installation under Windows 11

- System Requirements and Recommendations 11
- Software Installation 12
- DXi 2 Plug-in Setup 13

Installation under MacOS9 and MacOS X 14

- System Requirements and Recommendations 14

Audio Interfaces 16

- Stand-alone Application 16
- PlugIn 17

Reaktor as Plug-in 20

- VST 2.0 Plug-in 21
- DXi 2 Plug-in 25
- RTAS 26
- Audio Units 26

Reaktor Standalone Version 27

Open Sound Control (OSC) 30

- OSC system setup 31

First Steps in REAKTOR 35

- Opening and Playing Example Ensembles 35
- Your First DIY Synthesizer 43
- Your First Self-Made Structure 50

All content text has been **HYPERLINKED** for ease of use. You're welcome!

Basic Operation 55

Mouse 55

Context Menus 56

Key Commands 56

Windows 57

Menus 59

File Menu 59

Edit Menu 61

Insert Menu 63

Settings Menu 64

System Menu 66

Preferences 68

View 71

? Menu 77

Toolbar 78

The Browser window 80

Window access and organization 80

Browser window file types 81

Structure display 83

Wiring in the Browser window 85

Ensemble 87

Ensemble Structure 88

Ensemble window 89

Ensemble Properties 90

Instruments 97

What is an Instrument? 97

Creating Instruments 97

Ports 98

Context Menu 98

Instrument Header 99

Instrument Properties 100

Macros 111

What is a Macro? 111

Creating Macros 112

Ports 112

Context Menu 112

All content text has been **HYPERLINKED** for ease of use. You're welcome!

Structures 114

What is a Structure? 114

Modules 115

Sources 120

Switches 122

Terminals 123

Wires 124

Event Signals 127

Context Menu 128

Panel Editing 130

What is a Panel? 130

What are Controls? 130

Panel Controls 131

Connection properties of Panel Controls 133

Editing the Panels 135

Panel Operation 137

Mouse Control 137

Key Control 138

MIDI Control 138

MIDI Out 140

Snapshots 141

Sampling and Re-Synthesis 148

Sample Management 148

Sample-Maps 151

Sampler Map Editor 153

Akai Import 158

Table Modules 160

Properties 160

Context Menu 168

Advanced Operation 171

Loading older Ensembles

All content text has been [HYPERLINKED](#) for ease of use. You're welcome!

"Classic Modular" Macro Collection 173

Display 174
MIDI 175
Mixer/Amp 177
Oscillator 180
Sampler 181
Sequencer 183
LFO, Envelope 190
Filter 192
Delay 195
Audio Modifier 196
Event Processing 198

Appendix 199

Module Reference 201

Panel 202
MIDI In 220
MIDI Out 227
Math 231
Signal Path 242
Oscillator 246
Samplers 269
Sequencer 290
LFO, Envelope 294
Filter 312
Delay 328
Audio Modifier 335
Event Processing 346
Auxiliary 360
Terminal 376

1 Introduction

1.1. What is REAKTOR ?

REAKTOR turns your computer into a powerful synthesizer and audio processing system. With REAKTOR's completely modular structure you can build any synthesis and effects processing architecture that you can imagine. From relatively simple analog synths to large modular systems, from basic sample players to exotic granular models, from elementary delay lines to full-featured reverb units, your creativity will have virtually no limits.

If building your own instruments and effects is not your top priority, you'll still find plenty to do with REAKTOR. It comes packed with hundreds of instruments and effects of all descriptions. Want a simple FM synth? It's there. Want a sample player with independent control of time and pitch shifting? Load it up. Want a multi-effects box to munge your audio files? It's at your fingertips. And the best part of the REAKTOR library is that you can get right down to the business of making music.

Of course, if something in the library doesn't do exactly what you need, its structure is completely accessible for you to modify. Nothing is hidden or protected. And there's an active user community and online library with new instruments and effects being added all the time. In short, you can choose how to use REAKTOR—fire up a pre-built Ensemble today, add some snapshots and make some modifications tomorrow, build your own from the ground up the next day. Just get started.

Here is a brief overview of some things that are new and some that have changed:

1.2. Loading older Ensembles

Reaktor 4 no longer uses the USB key for copy protection, but Ensembles saved with previous versions of Reaktor will not open without the USB Key. If you have the USB key, simply open the Ensembles with the key installed then re-save them from Reaktor

4, and the key won't be necessary the next time you open them. If you are a new user and don't have the key, visit Native Instruments Web site for information about converting old Ensembles. A Web based service is provided for that purpose.

1.3. The Browser window

Although you can still use Reaktor's File menu for disk operations, the new Browser window greatly streamlines file management. It can be used for loading data of any type as well as analyzing any Ensemble's structure and input/output wiring.

1.4. The Snapshot window

Reaktor's Snapshot management has been greatly enhanced and streamlined as has the Snapshot window, which shares the space of the Browser window. One of the most interesting new features is the ability to morph between two snapshots with a transition time of up to 60 seconds. Another new feature is the ability to randomize snapshot settings, and a third is to have up to 16 banks of 128 snapshots each.

1.5. Sample Map Editor

Reaktor's sample key-mapping scheme and the Sample Map Editor have been completely revised and updated. The Sample Map Editor is now an independent window, much like but separate from the Properties window, that displays the sample map for whatever sampler module is selected. The sample map has both graphic and list modes. In graphic mode, samples can be dragged to the window and positioned graphically. Other new features include a pre-listen button, and a built-in loop editor.

1.6. Internal Communication

Internal communication between panel objects has been greatly expanded and simplified, as has OSC communication. Now any panel object can be designated as a source and destination of internal connection. Internal connections are set up in the Properties window. OSC communication between different, networked computers are set up in the OSC Settings window. Dedicated OSC Send and Receive modules make these messages available within the Reaktor structure.

1.7. New Modules

Reaktor 4 contains a number of new modules. The List, RGB Lamp, Multi-Picture, and Module-Text modules give added flexibility to Reaktor's user interface. A variety of new math modules both speed up and enhance Reaktor's math ability. The Grain Cloud Delay module brings all the features of the Grain Cloud sampler module to real-time audio processing. The event Iteration module makes it possible to program iterative processes without using and managing data-loops. Finally, the Snapshot module makes all the options available in the new Snapshot window usable from within Reaktor structures.

Using modules is also simplified by the addition of two new features. Hybrid modules automatically reconfigure themselves as either event or audio modules depending on their wiring. Dynamic modules sprout new inputs and outputs as needed, eliminating the need for similar functioning modules with a fixed number of inputs and outputs.

4 Installation under MacOS9 and MacOS X

4.1. System Requirements and Recommendations

To use the REAKTOR software, you need a computer with the following minimum specifications:

Hardware

- Apple PowerMac G3 500 MHz or faster
- 256 MB RAM
- Audio interface compatible with Sound Manager/ Core Audio
- OMS-compatible MIDI interface for connecting a MIDI keyboard or an external sequencer (only for the stand-alone version)
- 300 MB free space for the installation

The audio engine in REAKTOR has been designed to make optimum use of the available computing power in the CPU. The parallel data processing expansions and powerful FPU's integrated in modern CPUs are best suited to carrying out the complex computations of real-time synthesis. REAKTOR makes extensive use of these expansions in order to achieve optimum performance. As a minimum requirement, we recommend using a G4 733 MHz with 512 MB RAM.

Software

MacOS 9.2, OMS V 2.3.8 or MacOS 10.2.2

Installing the Reaktor Software

- Insert the Installation CD into the CD drive of your computer.
- Double-click the CD icon.
- Double-click the installation program **Install Reaktor** to start it.

- The start screen appears first. After clicking **Continue** and confirming the license agreement, a dialog opens where you can select the installation location and the destination folder.

The installation program suggests a path for the REAKTOR folder; if you do not select a different destination, the REAKTOR folder is created on the first hard disk. You can choose between **Easy Install**, where both the stand-alone and plug-in versions are installed, or **Custom Install**, where you can select which versions you want to install.

MacOS VST Plug-in Installation

- Insert the Installation CD into the CD drive and double-click the CD icon.
- Click the **Reaktor Install** application in the CD folder and select the **Custom** installation type.
- Select only **VST Plug-in** from the list of components to install.
- **MacOS 9:** You can now choose to automatically search for the VST plug-in folder or manually select the VST plug-in folder of your choice. Please select the option that best suits your installation requirements. **MacOS X:** REAKTOR is automatically installed in the VST folder of the MacOS X operating system.

5 Audio Interfaces

Audio interfaces allow REAKTOR to communicate with the audio hardware of your computer and other programs that you may have installed. This chapter contains detailed information on the various audio interfaces and how to use them. The features of the various interfaces are described together with their suitable applications.

Basically, there are two ways of using REAKTOR: as a “stand-alone” or as a “plug-in”. In following, the two versions are described together with their corresponding interfaces.

5.1. Stand-alone Application

This method allows you to use REAKTOR as a stand-alone program with any of the following the interfaces (drivers): MME, DirectSound, SoundManager, Core Audio or ASIO. In this case, your computer acts as a stand-alone instrument, similar to a hardware synthesizer with a MIDI port and analog inputs and outputs. The table contains an overview of which interfaces are suitable for stand-alone operation on the various computer platforms:

Interface/driver	Windows	MacOS	MacOSX
ASIO 2.0	●	●	
DirectSound	●		
MME	●		
SoundManager		●	
Core Audio			●

5.2. PlugIn

Used as a plug-in, REAKTOR is not a stand-alone program but rather a program “module” that can be integrated into a “host” program such as a sequencer. Plug-in mode allows you to integrate it seamlessly with the sequencer. Furthermore, it has many other uses as a plug-in:

- MIDI sequencing of the REAKTOR and audio mix-down of the MIDI tracks within a single program
- Comfortable automation of REAKTOR parameters in the sequencer
- Further processing of REAKTOR signals using additional plug-ins
- Sample-accurate timing with MIDI controllers (when used as VST 2.0 plug-in)
- Restoring of all plug-in settings when the host document (such as a songfile of the sequencer) is loaded
- Integration with other instruments into a “virtual studio”

This table provides you with an overview of which interfaces are supported by which host programs:

Interface/driver	Host Programs	Windows	Mac
VST 2.0 Plug-in	Cubase, Nuendo, Logic	●	●
Cakewalk DXi	Sonar	●	
RTAS	Pro Tools 5.x, LE, Free	●	●

Overview of Operating Systems and Plug-ins

The interfaces described below are effectively different ways in which REAKTOR can communicate with your sound card. The interfaces that are available on your computer depend on the sound card you are using as well as your computer platform (Windows or MacOS).

ASIO ("Audio Streaming Input Output") is an sound card driver architecture developed by Steinberg. ASIO is available for MacOS and Windows computers. It offers low latency and supports multi-channel audio cards. With its high performance and low latency, the ASIO driver interface is highly recommendable.

DirectSound is an interface developed by Microsoft and is a component of DirectX 5.0 or higher for Windows 98/ME/2000/XP. Whether or not DirectX works well depends on the sound card you are using. If the audio buffer size that you set is too small with DirectSound, glitches and clicks may occur in the audio.

MME is the standard "Wave" driver in Windows. Most sound cards support this interface and work with it quite well. However, MME is even less suitable than DirectSound for real-time applications. This is noticeable by a comparatively high latency.

DXI 2 is a plug-in interface for software synthesizers and instruments based on Microsoft DXi technology. Sonar from Cakewalk and Fruity Loops are the most well known host sequencers that support DXi.

SoundManager is the standard audio interface for the integrated audio hardware on Apple computers. The SoundManager audio interface has proven to be easy to use and reliable in real-time applications with an average latency of approximately 5 ms. With only the SoundManager interface, you can already play the REAKTOR well without need for an expensive sound card. Should you require even lower latencies or higher sound quality, then we recommend using an additional high-quality sound card. These sound cards also offer support for the SoundManager interface.

Core Audio is a new audio interface available with MacOS X that allows you to use external audio hardware as well as the integrated audio output of the Mac.

RTAS is based on an interface protocol from DigiDesign that allows you to use plug-ins with ProTools (or other software that is compatible with DigiDesign). RTAS plug-ins function independently from additional TDM hardware and are nonetheless able to offer the widest range of features. In this case, the host processor

alone performs all of the computations for the plug-in. At the time of completion of this user guide, the RTAS interface was not yet available. The most up-to-date information can be found in the Readme file supplied with your version of REAKTOR.

Audio Units can be used in MacOS X in a similar fashion to VST plug-ins. At the time of completion of this user guide, the Audio Units interface was not yet available. The most up-to-date information can be found in the Readme file supplied with your version of REAKTOR.

6 Reaktor as Plug-in

The plug-in version of REAKTOR looks a bit different from the standalone version, but you have still access to all main features unless it does not make sense for the plug-in operation.

A menu is available in the plug-in version as context menu. To call up the menu, right click (Mac: Ctrl + Click) within an empty space of the toolbar. You can hide the toolbar with the first entry in this context menu. Even if the toolbar is hidden, you can call up the menu anytime with a right click on an empty area of the Browser, Snapshot or Properties window.

If you hide the toolbar and close all other windows except for the Ensemble window and finally press the Resize button, your ensemble fits perfectly into the plug-in window.

The left part of the plug-in window can be toggled between three different views: Properties (F4), Browser (F5), Snapshots (F6). Alternatively you can hide this area by clicking on the Close button which shows a small cross.

If the toolbar is visible, the button on the right serves for switching to the **Browser** view. If it is hidden use the shortcut F5.



Browser button


The **Snapshot** view can be accessed by the Snapshot buttons in the Ensemble and Instrument Headers (F6).



Snapshot button

The **Properties** view can be accessed by double clicking any control within an Instrument panel (F6).

If you open the **Sample Map Editor** for a Sampler by double clicking a sampler waveform within an Instrument panel, it will be displayed at the bottom of the plug-in window.

 The **Resize** Button in the Toolbar causes a reinitialization of the plug-in window size to get an optimized display of all opened objects within the window.

Ensembles can be saved using the **Save** button in the Toolbar. You can save the Ensemble under a new name when you perform a **Ctrl** click on the **Save** button.

Automation: If your host supports plug-in automation, REAKTOR will pass the parameter names and value ranges of the controls used in the currently loaded ensemble to the host.

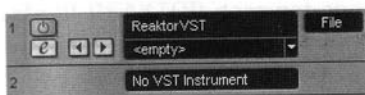
6.1. VST 2.0 Plug-in

In addition to the stand-alone version, REAKTOR can also be used as a VST plug-in. The advantages of the VST 2.0 format allow us to provide you with a powerful plug-in.

For more information on the VST 2.0 format, refer to the user guide provided with your VST host program.

Using the Reaktor Plug-in in Cubase SX

- Launch Cubase, go to the **Devices** menu option and select the **VST Instruments** menu option.
- A window showing the instrument rack appears. Click where it says **No VST Instrument** and choose REAKTOR from the available list of instrument plug-ins.



- The plug-in will now appear in your list and automatically be turned on. It will also create a set of audio channels in your VST mixer that will be used for mixdown within your project. This will allow you to mix, pan, and process REAKTOR's output just like any other existing audio track in your Cubase song.

- Click on the **Edit** button to call up the REAKTOR interface. Here you can control and edit all the features and functions that REAKTOR has to offer.
- Now go to the "Project" page and add a MIDI track (if you do not have one already created).



- Go to the **Output** parameter section for this MIDI Track and click on the field. This will create a list of available MIDI outputs to assign to this MIDI track. Choose **Reaktor VST** from the list.

Note: If REAKTOR does not appear in the list of available VST instruments inside your VST 2 host application, then it is not installed correctly. Please refer to the previous section on installing the plug-in for both Windows and Mac platforms for more assistance on setting this up.

After having loaded an Instrument from the REAKTOR library you should be able to trigger it via MIDI using a keyboard controller. REAKTOR's sound will generate through the VST mixer and directly to your sound card. If the plug-in does not receive MIDI or generate audio, then make sure to check the following areas:

- Make sure "MIDI thru" is enabled in Cubase.
- The MIDI channel of your MIDI track must correspond to the receive channel of the loaded REAKTOR instrument (normally channel 1).
- Make sure that you have properly configured your sound card for use with Cubase.

(please refer to your Cubase manual for more information)

Using the Reaktor Plug-in in Nuendo 1.5

- Launch an empty or current project in Nuendo.
- Click on the **Devices** menu and choose **VST instruments** from the menu options (or press F11 on your keyboard).

- A window showing the instrument rack appears. Click where it says **No VST Instrument** and choose **Reaktor** from the available list of installed plug-ins.



- The plug-in will now appear in your list and automatically be turned on. It will also create a set of audio channels in your VST mixer that will be used for mixdown within your project. This will allow you to mix, pan, and process REAKTOR's output just like any other existing audio track in your Nuendo project.
- Click on the **Edit** button to call up the REAKTOR interface. Here you can control and edit all the features and functions that REAKTOR has to offer.
- Now go to the "Project Editor" page and create a MIDI track (if you do not have one already created).
- Go to the **Output** parameter section for this MIDI Track and click on the field. This will create a list of available MIDI out ports to assign to this MIDI track. Choose **Reaktor (v1)** from the list. Also make sure you assign the MIDI input port to correspond to whatever MIDI controller you are using.



- Record enable the MIDI track.

Note: If REAKTOR does not appear in the list of available VST instruments inside your VST 2 host application, then it is not installed correctly. Please refer to the previous section on installing the plug-in for both Windows and Mac platforms for more assistance on setting this up.

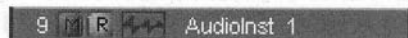
After having loaded an Instrument from the REAKTOR library you should be able to trigger it via MIDI using a keyboard controller. REAKTOR's sound will generate through the VST mixer and directly to your sound card. If the plug-in does not receive MIDI or generate audio, then make sure to check the following two areas:

- Make sure "MIDI thru" is enabled in Nuendo.
- The MIDI channel of your MIDI track must correspond to the receive channel of the loaded REAKTOR instrument (normally channel 1).
- Make sure that you have properly configured your sound card for use with Nuendo

(please refer to your Nuendo manual for more information).

Using the Reaktor Plug-in in Logic 5

- Launch Logic and create an audio instrument track or set an existing audio or MIDI track to an audio instrument track by clicking on it, holding down the mouse button and choose **Audio** ⇒ **Audio Instrument** ⇒ **AudiInst 1**.



- Double click the audio instrument track to open the environment window. Logic scrolls automatically to the first instrument bus in the Logic mixer.
- Choose the REAKTOR VST plug-in in the appropriate insert slot of the instrument mixer bus. Therefore click onto the insert slot, hold down the mouse button and choose **Stereo** ⇒ **VST** ⇒ **Reaktor**.



- The plug-in now appears in the instrument slot and is ready to use. The instrument mixer channel will allow you to mix, pan, and process REAKTOR's output just like any other existing audio track in Logic.

- Double click on the mixer's REAKTOR slot to call up the REAKTOR interface. Here you can control and edit all the features and functions that REAKTOR has to offer.

Note: If REAKTOR does not appear in the list of available VST instruments inside your VST 2 host application, then it is not installed correctly. Please refer to the previous section on installing the plug-in for both Windows and Mac platforms for more assistance on setting this up.

After having loaded an Instrument from the REAKTOR library you should be able to trigger it via MIDI using a keyboard controller. REAKTOR's sound will generate through the VST mixer and directly to your sound card. If the plug-in does not receive MIDI or generate audio, then make sure to check the following two areas:

- Make sure "MIDI thru" is enabled in Logic.
- The MIDI channel of your MIDI track must correspond to the receive channel of the loaded REAKTOR instrument (normally channel 1).
- Make sure that you have properly configured your sound card for use with Logic.

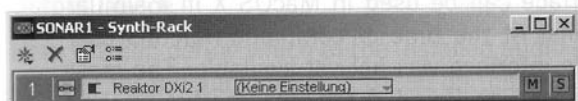
(please refer to your Logic manual for more information).

6.2. DXi 2 Plug-in

DXi 2 is a plug-in interface for software synthesizers and instruments based on Microsoft's DirectX technology.

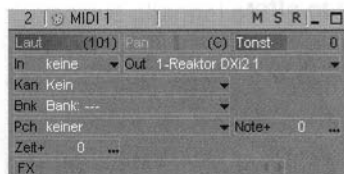
Using the Reaktor DXi 2 plug-in in Sonar

- Launch Sonar
- In the synth rack choose **Reaktor DXi 2**.



Loading the Reaktor DXi 2 plug-in in the synth rack

- Route a MIDI track to the DXi 2-Plug-in by selecting „REAKTOR“ in the Out drop down list.



Assign a MIDI track to the Reaktor-DXi-Plug-in

After having loaded an Instrument from the REAKTOR library you should be able to trigger it via MIDI using a keyboard controller. REAKTOR's sound will generate through the Sonar mixer and directly to your sound card. If the plug-in does not receive MIDI or generate audio, then make sure to check the following two areas:

- Make sure "MIDI thru" is enabled in Sonar.
- The MIDI channel of your MIDI track must correspond to the receive channel of the loaded REAKTOR instrument (normally channel 1).
- Make sure that you have properly configured your sound card for use with Sonar.

(please refer to your Sonar manual for more information).

6.3. RTAS

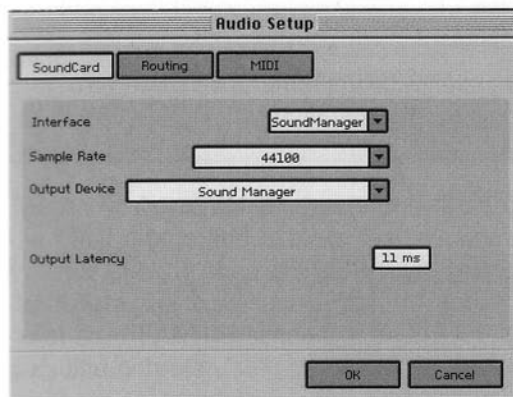
The RTAS format is an interface protocol for Mac OS that allows you to use plug-ins with ProTools independently from additional TDM hardware, while nonetheless offering the widest range of features. In this case, the host processor alone performs all of the computations for the plug-in.

6.4. Audio Units

The Audio Units interface can be used in MacOS X in a similar fashion to VST plug-ins. At the time of completion of this user guide, the Audio Units interface was not yet available. The most up-to-date information can be found in the Readme file on your installation CD.

7 Reaktor Standalone Version

The stand-alone version of REAKTOR allows you to use the application independently from other programs. In order to use the Standalone version you have to do the audio and MIDI settings first. You can call up the **Audio + MIDI Settings** setup dialog from the System menu of REAKTOR. For setting the standalone interfaces please choose **Audio + MIDI Settings...** from the **System** menu.



Audio + MIDI Setting dialog

Soundcard tab

Interface

All of the supported (and installed) audio interfaces are available in this drop-down list. Select the desired audio driver (MME, DirectSound, ASIO, SoundManager, Core Audio) from this list.

Sample Rate

Depending on the sound card and driver you are using, various sample rates are available. Set the desired sample rate here.

Output Device

Here you can define which of the installed audio interfaces should be used for the audio outputs based on the driver selected under **Interface**.

Input Device

Here you can define which of the installed audio interfaces should be used for the audio inputs based on the driver selected under **Interface**.

Note: With some interface types (e.g. ASIO or Core Audio), the **Input Device** setting is not available. Instead of that you can set the inputs for the chosen driver on the **Routing** tab.

Output Latency

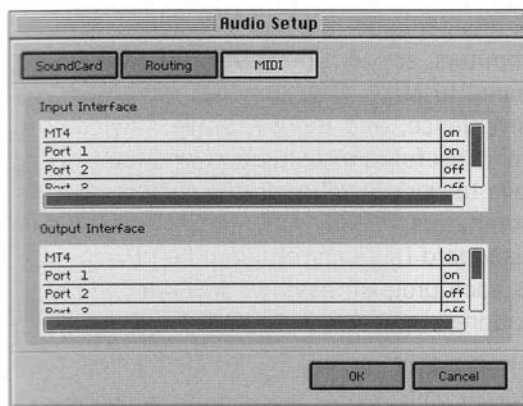
This box displays the output latency. With some drivers you also get a latency slider for setting an individual latency for REAKTOR.

Routing tab



If you are using a multi-channel sound card, REAKTOR also allows you to freely select which channels to use for the input and output signals. For example, you can set the REAKTOR output to channels 3 and 4 instead of channels 1 and 2.

MIDI tab



These two boxes (MIDI inputs and MIDI outputs) display all of the MIDI inputs and outputs that are correctly installed on your system. Click in the right column to “off” or “on” to activate or deactivate the corresponding MIDI input or output. From this point on, REAKTOR sends and receives MIDI on the activated inputs and outputs.

8 Open Sound Control (OSC)

OSC is an open, network-independent protocol developed for communication among computers, sound synthesizers, and other multimedia devices. Compared to MIDI, OSC provides increased reliability, greater user convenience, and more reactive musical control. Open Sound Control is useful in any situation where multiple music applications have to work together on the same computer or on networked computers. While MIDI only has the parameters defined in the standard (note on/off, pitch bend, control change, etc.), OSC lets each program have its own symbolic, hierarchical, and dynamic address space.

OSC can be used with any networking technology, including TCP/IP based LANs and the internet. OSC's time tags and bundles of messages provide for exact timing of musical results even if the network has latency and jitter. OSC supports a variety of argument types which will be successively integrated into future releases of REAKTOR.

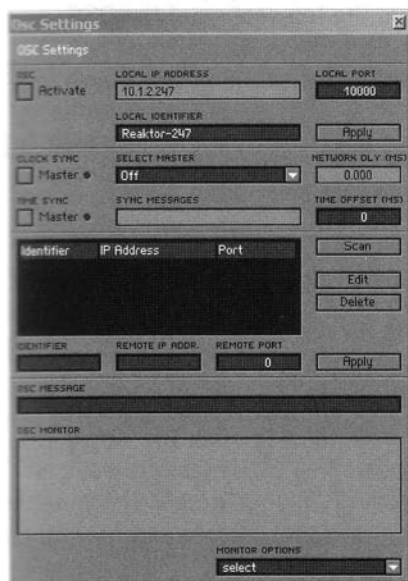
Application areas

The OSC implementation of REAKTOR allows an easy setup of

- Internet-based collaborative international music making
- Sound installations with dozens of computers in a single room coordinating with each other
- Coordinating synthesis between two (or more) computers to increase the total processing power
- Communication between music software applications within a single computer.

The OSC implementation in the current REAKTOR version only serves for transmitting event data between two or more REAKTOR computers, but not audio data. Additionally to the general REAKTOR requirements you will need an ethernet card to use OSC. Also the TCP/IP and UDP protocol stacks must be installed on your computer.

8.1. OSC system setup



OSC Settings window

Reaktor's Open Sound Control (OSC) settings are made using the **OSC Settings** window which you can open from the **System** menu. OSC provides communication between media devices and software such as Reaktor using a variety of network protocols, including TCP/IP and LANs.

Activating OSC

OSC communication can be enabled and disabled at will using the **Activate OSC** button at the top-left of the OSC Settings window. OSC communication is only possible when Reaktor's audio processing is active, and as a consequence, you need an audio card or built-in audio capability on your computer to activate OSC. The Activate OSC status is preserved between Reaktor sessions.

OSC Identification

In addition to the Activate OSC button, the top section of the OSC Settings window contains your Local IP Address, Local Identifier, and Local Port settings. The settings in this section are all preserved between Reaktor sessions.

- **Local IP Address:** This is the current IP address of your computer. It is recognized automatically and can not be edited.
- **Local Identifier:** This name will be used to identify you to other OSC clients. You can choose any name you like.
- **Local Port:** This is sub-network identifier by which other OSC clients recognize your system when they scan the network (see the Scan button below). Only certain ports are scanned, and you should use a number between 10,000 and 10,016.
- **Apply:** When you make changes, you need to click the Apply button to have them take effect.

OSC Synchronization

The second section of the OSC Settings window contains synchronization settings.

- **Clock Sync (Master):** Click this to have Reaktor send an OSC clock signal to other OSC clients. OSC clock works exactly like MIDI clock. Clock will be sent to all clients on the Member list (see below).
- **Time Sync (Master):** Time Sync is a control circuit system. Client constantly poll the master for the time stamp and compares the received time with its own and adjusts it if necessary.
- **Select Master:** When not operating in Clock Sync Master mode, use this menu to synchronize to an OSC master. Select Clock Sync to synchronize to Clock Sync signals. Select another OSC member to Time Sync with that client.
- **Sync LEDs:** There are small LEDs to the right of the Clock Sync and Time Sync checkboxes. These indicate when a synchronization signal is received or sent.
- **Sync Errors:** This field reports synchronization errors.

- **Time Offset (ms):** Adds the time offset to each OSC message sent to the clients. If you enter 1000 ms each message will be receive one second later by the client. This only applies if the participating clients are in Time Sync mode.

OSC Memebr list

This list contains all Reaktor OSC clients to whom a connection has been established.

You can edit and delete entries in this list. Therefore select an entry and press the **Edit** button. To apply any changes, click the **Apply** button.

To delete an OSC connection in the OSC member list, just select the entry and press the **Delete** button.

The **Scan** function is able to recognize OSC members within a sub-network automatically. This only works when the following conditions are accomplished:

- The client must be located within the same subnet.
- REAKTOR must be running on this computer (audio engine active).
- OSC has to be activated in the REAKTOR OSC Settings.
- In the REAKTOR OSC Settings a port address between 10000 and 10015 must have been entered.

Note: The **Scan** function only works in Windows and MacOS X, but not in MacOS 9.

If you want to connect two computers which are not located in the same subnet (for instance if you want to establish an OSC connection along the internet), you have to enter the **Identifier**, **IP address** and **Port number** of the other computer manually below the member list area and press the **Apply** button.

OSC Monitor

The bottom section of the OSC Settings window is for monitoring OSC activity.

- **OSC Message:** This field is for sending text messages to other OSC clients. It can be used to test OSC connections or as a chat box. First select a recipient in the Member list, type a message and finish the operation with the enter key. The message will then be sent to that client.
- **OSC Monitor:** The monitor displays all received OSC messages.
- **Monitor Options:** Here you can set certain functions for the monitor window.

OSC Synchronization

- The client must be located within the same subnet.
- REAKTOR must be running on the computer that is the source of the OSC signal.
- OSC must be selected in the REAKTOR OSC Settings field.
- In the REAKTOR OSC Settings field, you must select the correct IP address and Port number of the other computer manually. (Note: The OSC function only works in Windows and MacOSX, but not in MacOS).
- If you want to connect two computers which are not located in the same subnet for instance if you want to establish an OSC connection and the internet, you have to enter the identifier IP address and Port number of the other computer manually. (Note: The OSC function only works in Windows and MacOSX, but not in MacOS).
- Sync LEDs: There are two LEDs, one for the Sync function and one for the Sync Error function. They indicate when a sync signal is received or when a sync error occurs.
- Sync Errors: This field reports sync errors.

9 First Steps in REAKTOR

The purpose of this chapter is to make you familiar with the basics of the operation and the functionality of REAKTOR and how to program it.

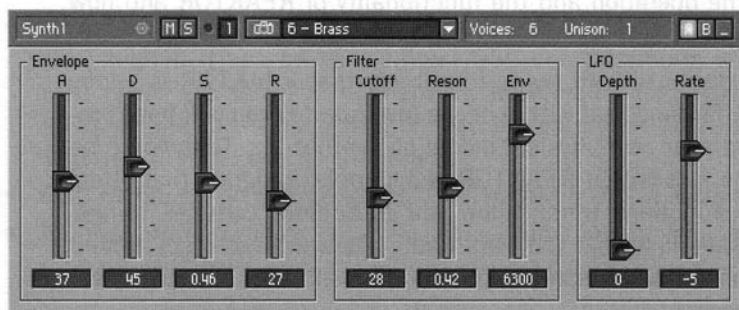
We will dispense with trying to tell you that REAKTOR is a very simple affair and that within only a few minutes you will have programmed your own physical modeling synthesizer. That would be a lie. The fact is that REAKTOR is a complex program that offers complex functions which allow you to achieve complex things. And if that's just what you want to do, you won't really get around an intensive initial learning phase. After all, real success never comes easy.

But don't worry. You can work with REAKTOR at a complex level, although you don't have to. As you will see in our first tour, it is possible to make music with the software using a number of different instruments, even without any knowledge of synthesis methods or processing structures. You simply help yourself to the provided library.

9.1. Opening and Playing Example Ensembles

First make sure that your MIDI controller instrument (master keyboard or MIDI workstation) is connected to one of the MIDI inputs of your computer. The input port should have been activated under **Input Interfaces** on the **MIDI** tab of the **Audio + MIDI Settings....** (For more information about activating MIDI-Ports, see section *Preferences* on page 68). The MIDI transmit channel on your controller should be set to 1.

Alternatively, you can just use the QWERTY keys on your computer keyboard to trigger notes (see section *Musical Note Keys* on page 199 for the key mapping).



Panel Window for Synth1

Synth1

Now open the folder **Tutorials** in the REAKTOR Library folder, select **Synth1.ens** and click on **Open**, or drag the Ensemble from the integrated Browser into the main REAKTOR window.

You should first take a quick look at the Toolbar. Is the Power button on and does the value field next to it show the current CPU usage? If it is, you can now play **Synth1**. If not then click the main power button to first turn on your new synth. Now you can let it rip! By the way, if the number display next to the main power button shows **Over** or a warning message pops up ("Processor Overload!") to inform you that audio processing has been turned off, then you will need to reduce the number of voices in the Instrument Header of the Synth 1 panel from **6** to a smaller number or choose a lower sample rate than the initial 44100 Hz in the toolbar. And if the **Out** level meter should ever light up red, this indicates that the soundcard is being overloaded – in which case you need to reduce the volume inside the REAKTOR software.

With every note you play, the MIDI LED in the Toolbar should light up red. The MIDI LED in the toolbar indicates that the system is receiving MIDI data, while the MIDI LED in the Instrument header informs you that the MIDI data is being correctly passed to the instrument.

Synth1 is a replica of a simple 6 voice analog synthesizer. It contains one sawtooth oscillator, a 24-dB lowpass filter, an LFO that affects the pitch and one ADSR envelope for both filter and amplitude.

In the window labeled **Synth1 – Panel** you can find the control elements that are available for changing the sound. From left to right, these are: **A**(ttack), **D**(ecay), **S**(ustain) and **R**(elease) for changing the shape of the envelope, **Cutoff** for controlling the filter cutoff (or corner) frequency, **Reson**(ance) for the amount of boost applied to the frequencies near this cutoff frequency, **Env** for setting how much the filter is affected (modulated) by the envelope, and finally **Depth** for setting the LFO intensity and **Rate** for the LFO speed.

If you have ever before had your hands on a synthesizer then dealing with this straightforward device should not be much of a challenge. On the other hand, if **Synth1** is your first synth, you now have the opportunity to experiment with the effect that these few but essential synthesis parameters have on the sound. And there's no reason to worry about getting too lost.

If during your experiments you should come across a sound that you particularly like, you can save it. For this you move your mouse pointer to the header of the Synth 1 panel, where there is a row of controls from which we now choose the camera icon. This one assists in creating so-called **Snapshots**, which correspond to the “patches” or “programs” in other programmable synthesizers. A window opens in which you can save the new snapshot. Therefore press the **Append** button and click on an empty slot of the snapshot list. You can rename the snapshot if you like. The snapshot will be stored when you press the **Enter** key. You can rename your snapshot later by double clicking the entry and typing in a new name.

Padecho

The next candidate we are going to have a closer look at is called **Padecho.ens**. You will find it in the same folder from where we previously pulled out **Synth1**.

Now you will just be asked whether you want to save the changes you have made in Synth1. You probably want to answer **No** here, unless you have just found a new sound that's worth keeping.

You see now three windows, of which one is labeled **Ensemble window**. Click on the Structure icon in the toolbar in order to open the ensemble structure window. The ensemble is the highest level in REAKTOR and the ensemble structure window is, in a manner of speaking, a bird's eye view of the complete working environment that is available to you. In this case it contains **Pad**, which is a synthesizer, and the stereo delay effect **Echo Stereo**. The output of the synthesizer is connected to the two inputs of the effects unit whose outputs are connected to the two inputs of the **Audio-Out** module.

You will find this **Audio-Out** module in every ensemble. It represents the software's connection to the rest of the world, which is normally the audio output of your soundcard, but can also be the Plug-In connection to another piece of software. Its counterpart is the **Audio-In** module which represents the audio inputs of your soundcard (or the Plugin connection) and which is also present in every ensemble. In this case it is muted, as you can see by the red cross which is displayed over the status LED in the **Audio-In** module, because the **Padecho** ensemble doesn't need any audio input.

A quick look at the **Padecho** ensemble already brings to light two essential features of REAKTOR. One is that an ensemble can consist of more than one instrument. The other that its generative power is not restricted to synthesizers, because **Echo Stereo** is obviously an effects unit.

You can switch between Ensemble structure and Ensemble window by double clicking the black background of a panel or dark grey background of a structure or by opening the context menu with the right mouse button (Mac: Ctrl + Click) and choosing "Ensemble Panel" or "Ensemble Structure" from the menu.

Naturally, you can play in the ensemble structure, as pressing a key on your MIDI instrument will show, but there are no control elements at your disposal which does detract from the entertainment value quite a bit.

The **Ensemble-Panel** gives access to two knobs, that is **Main** for controlling the master volume of the ensemble and **Tune** for setting the master tuning. Together with the panels for the Pad instrument and the Stereo Echo effect you see all control elements of the ensemble.

Before we allow you a few moments of musical activity with the **Padecho** ensemble, let's make a few short comments regarding its structure. The pad synth contains two oscillators that both generate a pulse-wave. The tuning of the second oscillator can be controlled relative to the first one coarse with the knob labeled **Interval** and fine with the **Fine** knob. The pulse width of both oscillators is set with **PWidth** and can also be modulated with the LFO. **LFO rate** sets the speed and **Depth** the amount of modulation. The controllers for the ADSR envelope, which again affects both filter and amplitude, as well as the knobs to the right for controlling the filter, correspond to those we got to know in **Synth1**.

The **Echo Stereo** consist of two delay lines, one of which processes the left and the other the right stereo channel. Their delay times can be controlled independently of each other using **Del L** and **Del R**, where a setting of zero means that there is no delay. The desired number of echo repeats is set with the knobs **F(eed)Back** and **Cross**, where **FBack** controls the amount of signal of a channel (L or R) going back into itself and **Cross** the amount going into the respective other channel. Finally, **Wet-Lvl** sets how much of the original signal goes through the delays, and thus it controls the strength of the effect.

FM Overdrive

The next example we want to present to you is called **Fm2opsov.ens**. You can also find this in the by now familiar **Tutorial** folder.

As a quick look at the **Ensemble** window shows, we again have a combination of two devices here: the synthesizer **FM 2 ops** followed by the distortion effect **Overdrive**.

The FM Overdrive synthesizer is an example of the flexibility of REAKTOR. In addition to subtractive synthesis, REAKTOR is capable of other types of synthesis. In this case, FM (frequency modulation), made popular by the Yamaha DX series of synthesizers, is used for tone generation.

In our example there are not 6 operators as in the DX7, or 4 like in the smaller DX models, but only 2 operators, so the whole structure remains quite clear. Both operators consist of an oscillator that generates a sine wave. One is the so-called carrier, responsible for generating the fundamental wave and thus setting the pitch of the sound. The other operator is the modulator that affects the frequency of the carrier and controls the timbre.

Play a few notes on your MIDI instrument. Not very exciting, right? Now slowly push the **FM** fader upwards and listen how the sound changes. A bell-like element starts to creep into the sound until it dominates it completely when the maximum position is reached. On a technical level, all we have done by sliding the **FM** knob up is to increase the level of the modulator and thereby determine how much it modulates the carrier's frequency.

In the next step we turn our attention to the **Interval** knob. The effect that this parameter has should quickly become quite clear. The knob placed next to it, **Detune**, allows you to make fine adjustments to the interval setting.

A very simple envelope is responsible for determining the sound's development over time. The carrier's envelope, which controls volume, has only the two parameters **D**(ecay) and **R**(elease). The envelope for the modulator is even simpler and has only the one knob for setting the decay, labeled **Mod-D**.

Armed with this knowledge you should not find it difficult to create your own sounds with this 2 operator FM synthesizer, and to do it with a sense of purpose.

Let's turn briefly to the **Overdrive**, the purpose of which is simply to furnish your FM sound creation with some amount of acoustic grit. The best thing is probably if you first try out the various snapshots before dedicating yourself to the following explanation of this device.

Drive sets the level of the signal that is sent to the distorting element and therefore controls the amount of dirt that is generated. With **Asym** it is possible to modify the overtone spectrum of the signal in such a way as to make it “warmer”, i.e., to make it sound as if the sound was generated using a valve (tube) circuit. The distortion circuit is followed by a filter with the parameters **Freq**(ency) for setting its cutoff frequency and **Emph**(asis) to emphasize this frequency. The setting of the **Volume** knob determines the output level of the sound signal.

16-Step Sequencer plus Bassline

The ensemble **Squnc16.ens**, with which we are now going to experiment, can also be found in the folder **Tutorial**.

A look at the **Ensemble** window tells us something about the construction of this setup: **Sequence16**, a 16-Step sequencer (another function that REAKTOR can provide) controls **Bassline-Puls**, a kind of 303 clone, whose signal reaches the audio output via the **Panner**.

The **Panel** window of the 16-Step sequencer is already open and the **Run** button seems to be smiling at us in such an inviting way, so we will just give it a push – and promptly the sequence of 16th notes starts bubbling away. You can change the pitch for every step with the **Pitch** faders in the top row, and the volume for each step with the **Lvl** (Level) faders in the row below. The tempo is adjustable using the **BPM** knob (next to the **Run** button) and the length of the notes can be manipulated with knob labeled **Length**.

Finally, the **Reset** button located underneath **Run** resets the sequence to step 1 every time it is pressed. If it is pressed while the sequencer is running, a nice shifted pattern can be generated. If it is pressed while the sequencer is stopped, this ensures that, on starting, the sequence will begin at the first step and not somewhere in the middle.

A click with the right mouse button (Windows) or **ctrl** key + mouse button (MacOS) on the instrument **Bassline-Puls** in the ensemble window and selection of **Panel** in the context menu give you access to the control elements of the synthesizer. This layout cor-

responds to what you may know from a 303; but even if you have never seen such a beast, with such a small number of controls it's unlikely that any confusion will arise. Just turn any knobs you want and listen to the result.

As you have probably already noticed, the sound in this ensemble is always moving back and forth between the left and right speakers. The **Panner** is responsible for this. Now open the panel (you know, context menu etc.). Very simple, isn't it? With **Amount** you set by how much the signal travels between the left and right channel and **Rate** is the speed of this movement. That is all there is to it.

Sample Loop Player

The final example for illustrating REAKTOR's capabilities is called **Wav-play.ens**. You find it, with the other REAKTOR examples, in the folder **Tutorial**.

This ensemble is made up of the units **Loop-Player** and **12-Band**. Before you can hear anything here, you first need to load a sample into the Loop-Player. Click the right mouse button (Windows) or **ctrl** key + mouse button (MacOS) on the sample slot displaying "untitled.wav" in the panel window and choose **Load Audio into Tapedeck...** in the context menu that appears. In the file selection dialog window select any of the WAV files that you may have on your hard drive and load it by clicking on **Open**. Presto, a click on the **Run** button and you hear the newly loaded sample as a loop.

The whole point of the **Wav-play** ensemble, however, is to be found in the **12-Band** effect. The panel looks very much like the classic control panel of a graphic equalizer – that is, faders which allow the level of various frequency bands to be controlled. What we have here is a filterbank which allows even more dramatic manipulation of the sound than an equalizer. The number above each fader tells you at how many cycles per second (Hertz, often abbreviated Hz) the band which the fader controls is located. Try out the effect of the different frequency bands on the sound, while the loop is running. You will notice that it's not just the timbre that changes, but that it is possible to nearly remove entire parts and so manipulate the musical character of the loop.

9.2. Your First DIY Synthesizer

As you may have noticed in the previous examples and by further rummaging through the library, REAKTOR offers a wealth of ready-made instruments, effects units and combinations. But the true thrill of REAKTOR is in the possibility of designing and constructing your own instruments. And as you will see, it isn't difficult if approached in the right way.

How about an analog synthesizer in the good old fashion? Let's do it using subtractive synthesis, where an oscillator first produces a signal rich in high frequency components, some of which are subsequently removed using a time variable filter. All right, here we go.

Preparation

To construct our synthesizer we will use a method that is very effective – the use of so-called **Macros**.

In REAKTOR terminology, macros are functional blocks with which the construction of complex structures becomes quite easy, and most importantly, everything remains clearly laid out. In REAKTOR you already have an extensive library of such macros at your disposal, and we will help ourselves to it.

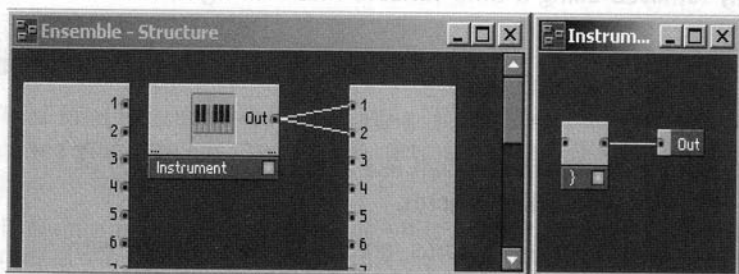
Initially, turn off the main switch in the **Toolbar**, so that you don't get startled when the half-finished construction suddenly starts making noises.

To begin with we will prepare the workspace in which to construct the synth. Please open the **File** menu at the top of the **Ensemble** window and choose the entry **New**. If you have made some changes in the current ensemble, you will first be asked whether you want to save these changes. After this you will see (in the **Ensemble** window) our old friends **Audio-Out** and **Audio-In**.

First we need a shell – a box so to speak – in which we will construct our synth. For this we take an empty instrument which we find in the library. Click with the right mouse button (Windows) or **ctrl** key + mouse button (MacOS) in the **Structure** window and in the context menu choose **Insert Instrument** ⇒ **_New - 2In2Out**.

The empty instrument we need appears in the structure. Double click the Instrument box in the structure and delete all terminals inside except for one **Out**-terminal and the **Voice Combiner** in front of it.

Now, in the Ensemble structure window, click on the **Out** port of the Instrument, move the pointer to the input **1** of the **Audio-Out** module and once more click the mouse button. Do you now see a connection between the two components? If not, try again. If so, we congratulate you on creating your first virtual Wire. In the same way connect the **Out** port of the Instrument to the **2** input of **Audio-Out** so that you can later hear the sound on both channels.



An empty instrument connected to Audio Out

Choice of Components

For our synth we need one or more oscillators whose signal should go through a filter. The volume of the oscillators should be controlled additionally by an envelope, and that's it. We will now assemble these components.

Click with the right mouse button (Windows) or **ctrl** key + mouse button (MacOS) on the **Instrument** box and then select **Show in Panel** to open the panel window. The panel obviously doesn't contain any controllers yet because we haven't built anything so far. In the same way open the **Structure** window contains an output terminal (**Out**) and a **Voice Combiner** (}). In the remaining space we load the components we have just talked about.

Click with the right mouse button (Windows) or **ctrl** key + mouse button (MacOS) in the **Structure** window and in the context menu choose **Insert Macro** ⇨ **Building Blocks** ⇨ **Oscillator** ⇨ **Waves** ⇨ **OSC (pls, saw, tri)**. In the **Structure** window you can now see the macro. A quick look in the **Instrument-Panel** window, which was empty until just a few moments ago, shows the first few controllers. We're making progress.

Before we go on we should make sure that the oscillator is working. As a precautionary measure, first open the **Ensemble-Panel** window (double click in the empty ensemble structure window) and set the **Level** fader of the master instrument to, let's say, **-10** to avoid any nasty surprises during the following audio test.

In the **Instrument-Structure** window, connect a wire from the **Out** port of the **Osc 3 Wave** macro to the input of the **Voice Combiner** module (marked with the symbol: **}**) by clicking first on one port to start the wire and then on the other port to finish connecting it. The **Voice Combiner** serves for converting a polyphonic signal into a monophonic. This conversion is especially important in front of an Instrument out-port, since Instrument ports are generally monophonic.

Let's add an **ADSR** volume envelope to the **Oscillator**-Macro by using the context menu again. Connect the lower Out Output of the **ADSR** macro to the A input of the **Osc 3 Wave** macro. Now we only need two more important MIDI modules to get a connection to an external MIDI input device. Insert the module **NotePitch (Insert Module ⇨ MIDI In ⇨ Note Pitch)** and connect it with the **P** input of the **Oscillator** macro. Finally we need a **Gate** module (**Insert Module ⇨ MIDI In ⇨ Gate**), which has to be connected with the **G** input of the **ADSR** macro. Press the power button in the Toolbar and you should hear a sound when you press some keys on your MIDI device. This is it, our oscillator in its raw form – not really beautiful but audibly functional.

Before loading the next component, the filter, first remove the wire you have just made between the **Oscillator** and the **Voice Combiner**. Simply click on the two ports again, as if connecting a second wire, and the connection is gone. Alternatively, just click on the wire (it changes color) and press the **Del** key (Delete on the Macintosh) – same result.

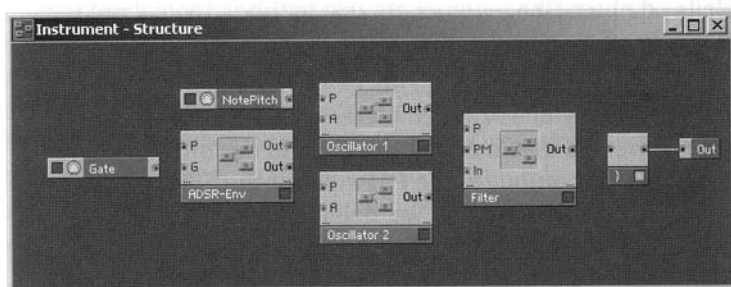
Now load a Filter macro in the same manner as previously the oscillator. You find it in the context menu under **Insert Macro** ⇒ **Building Blocks** ⇒ **Filter** ⇒ **Filter without FM** ⇒ **4 Pole Filter (BP, BLP, LP)**.

On considering these components you may question if a synth with only one oscillator is the be all and end all. After all, it's well known that two oscillators simply give a fatter sound. OK, so let's add another one: click the right mouse button (Windows) or **ctrl** key + mouse button (MacOS) on the **Osc 3 Wave** macro (careful, don't hit one of the ports), select **Copy** from the context menu, click the right mouse button (Windows) or **ctrl** key + mouse button (MacOS) on some patch of empty space in the Structure window, select **Paste**, done.

It is important to maintain a clean design – especially when dealing with a complex synthesizer. It's easy to create a chaotic layout, then spend hours searching for the cause of a problem. So let's clean up the structure window a bit. Move the 4-P Filter (BP, BLP, LP) macro a little way in front of the **Out** terminal and the two Oscillator macros we place neatly one above the other in front of the Filter macro.

For the next step we remove some of the confusion with the two oscillators, which at present are identical, even sharing the same name. Let's give them different labels. To accomplish this, click the right mouse button (Windows) or **ctrl** key + mouse button (MacOS) on the upper Oscillator macro and choose **Macro Properties** from the context menu. Now you see the field called

Label at the top left of the window that appears, where you can enter a new name, such as **Oscillator1**. Do the same for the other Oscillator below, but give it the label **Oscillator2**. You can also rename the macro **4 Pole Filter (BP, BLP, LP)** to **Filter**.



This is roughly how the structure window should look before wiring

Wiring

Now that we have all the components of our synth ready, we can start wiring them up. Make a connection from **Out** of **Oscillator1** to **In** of the **Filter** and then, because we want the signals of both oscillators to enjoy treatment by the filter, from **Out** of **Oscillator2** to **In** of the **Filter**. We clearly have a problem here, because whatever we do there is only ever one wire. Of course that's not really surprising, because you can't put two jack plugs in the same socket either. The solution to this problem is to be found with a little bit of thinking. What we are looking for is a component that can simply combine the signals from the two Oscillators and pass the sum on to the **In** of the **Filter**. And this component, a mixer for audio signals, is of course available in REAKTOR.

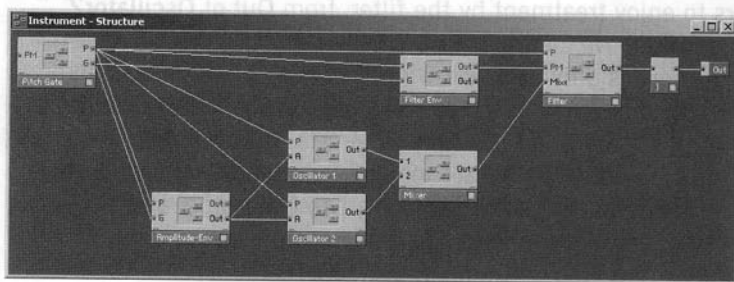
To insert the mixer, click the right mouse button (Windows) or **ctrl** key + mouse button (⌘ MacOS) on an empty area in the **Structure** window and in the context menu choose the macro **Amp (Mono, Pan Mod)** under **Insert Macro** ⇒ **Building Blocks** ⇒ **Amplifier** ⇒ **Mono**.

Now place the mixer between the two Oscillators and the Filter (we want things to be neat), and connect the output of **Oscillator1** to the upper input of the mixer and the output of **Oscillator2** to the lower input. The rest is child's play: a wire from the output of

the mixer to **In** of the Filter, a wire from **Out** of the Filter to the input of the **Voice Combiner** – done. As soon as you complete the last connection, the status lamps of all modules should light up to indicate that we now have a functional structure.

At the end of this tutorial we want to exchange the modules **Pitch** and **Gate** against a macro which does the same but also has an integrated module for Pitchbending. An extra ADSR envelope for the Filter makes our synthesizer complete. Let's begin with copying the **ADSR** envelope macro and assigning it to the filter by connecting the upper **Out** output to the **PM** input of the **Filter** macro. Now we need the MIDI modules we deleted before. Insert the macro **Pitch + Gate** which contains these modules in its structure. Connect the **P** output of the **Pitch + Gate** macro to the **P** inputs of the two ADSR macros, the two oscillator macros and the **Filter** macro. The **G** output has to be connected to the **G** inputs of both ADSR macros.

Now the synthesizer can be played properly. Pitch is recognized and used correctly and even the use of the pitchbend wheel on the MIDI keyboard show the proper response because the **Pitch + Gate** macro is set up to handle those tasks as well.



The structure window after insertion of additional components and wiring

Arranging the Panel

Have a look now at the **Ensemble window**. You see a bunch of knobs that are wrapped up with frames to form groups. Each frame corresponds to one of the macros that we have inserted, so we know exactly which controller belongs to the **Filter**, which to **Oscillator2**, etc.

Now you can start polishing the panel design. For example, at the moment the controllers for **Oscillator2** are still overlaying those for **Oscillator1**. To change this, first make sure that the protection mode is deactivated by unpressing the **Protection** button in the toolbar (lock icon) and that you are in panel edit mode by clicking on the **Panel Edit** button in the Instrument header (In **Panel Edit** mode the screw icon will be replaced by a wrench icon), then click the left mouse button on the label **Oscillator2** at the top of the frame, keep it pressed and drag the **Oscillator2** block to some part of the window that you regard as suitable. You can do the same for all the other functional blocks.

Once you are happy with the layout you can freeze it in its current state to make sure that knobs or frames aren't ever moved inadvertently. You achieve this by simply clicking again on the wrench icon in the Instrument Header in order to activate the **Panel Lock** mode. Operating the panel is only possible in **Panel Lock** mode.

Saving

You probably want to save the synthesizer you have built so that it isn't lost. Right Click (MacOS: Ctrl + Click) on the Instrument Header of the panel or the Instrument module in the Structure to open the Instrument's context menu and choose **Save Instrument as...** With this function you do not save the whole ensemble (which you could do using the **File** menu) but only the instrument. This allows us to reuse the instrument in a future ensemble.

In the file dialog window that appears, choose a folder in which you like to keep your instrument, specify a file name and click on **Save**. When you are asked later whether you want to save the ensemble, you can say **No** because the only part of the ensemble that's worth keeping (the instrument) has already been saved separately.

Luxury

In case you start to feel like adding more features after some time of playing around with your new synthesizer, rest assured that REAKTOR isn't going to limit your urge for experimentation. Just take a look at the macros which are included in the demo. You will find a lot of possibilities to transform this simple synthesizer into a luxurious sound machine.

9.3. Your First Self-Made Structure

Our first synthesizer project was executed mainly using macros in which very many functions were pre-built. We would now like to introduce you to the art of constructing a synthesizer completely from scratch. This job is carried out at REAKTOR's lowest level, the **Structure**. Contrary to the recommendation we gave above, which was to always to separate larger functional units into macros, this synthesizer will be constructed entirely within the instrument. The main reason for this is the fact that our new device will be of a quite modest nature. It will consist of so few components that any further subdivision into macros would probably cause confusion rather than make things clearer.

Building the Basic Structure

Select the entry **New** from the **File** menu. The ensemble window now has just an **Audio-In** and an **Audio-Out** module. Again, take an empty instrument module from the library (click with the right mouse button (Windows) or **ctrl** key + mouse button (MacOS) in the ensemble window: **Insert Instrument** ⇨ **New - 2In2Out**). Delete all except for one **Out-Terminal** and **Voice Combiner**, and connect it to **Audio-Out 1** and **2**.

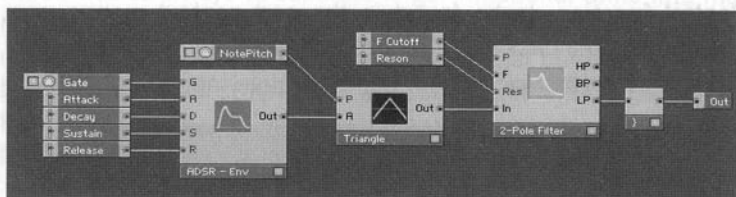
Open the **Structure** window by clicking with the right mouse button (Windows) or **ctrl** key + mouse button (MacOS) on the **Instrument** module and selecting **Structure** in the context menu. We will now implement the synthesizer circuitry in the **Structure** window.

You can already play this synthesizer, but you will very likely soon get fed up with the sound, because other than the volume envelope there's definitely nothing that can be adjusted. The whole thing becomes interesting when a filter comes into play.

Adding a Resonant Filter

To create our filter - a **Multi-2-PoleFM** filter to be precise - select **Insert Module** ⇒ **Filter** ⇒ **Multi 2-PoleFM** from the context menu of the structure window. Then connect **Out** of the **Triangle** module to **In** of the 2-pole filter module and the filter's output **LP** to the **Out** terminal.

Next, using the **Create Control** function, create the control elements for the filter parameters **F**(requency Cutoff) and **Res**(onance) to make the structure look something like the picture below.



Filter's Function

Play a few notes on your keyboard while at the same time changing the position of the knobs **FCutoff** and **Reson** in the **Panel** window.

You are using **FCutoff** to set the filter's cutoff frequency. When utilizing the output **LP** (low pass) of the filter module, as done here, all frequencies above the cutoff frequency are removed. By using the other outputs of the **Multi 2-PoleFM** filter module it can also be employed as a band pass (**BP**) or high pass (**HP**) filter.

Reson sets the resonance of the filter. The larger the resonance value is chosen, the more the frequencies "around the cutoff frequency" are boosted.

Adding Key Tracking

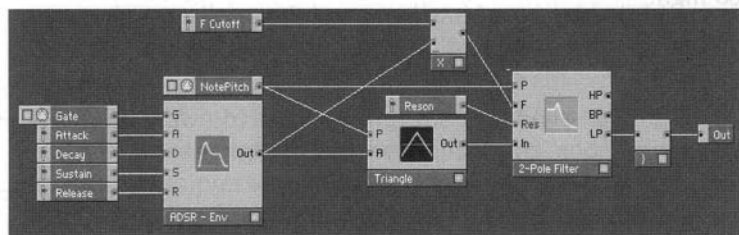
Now play some low notes and then some high notes on your MIDI instrument and you will notice that the high notes sound relatively dull. This is because the filter operates at a fixed cutoff frequency. This means that no matter what pitch you play, the filter always removes all frequencies above the fixed cutoff frequency. So if you play a note whose frequency is above this cutoff, almost nothing will be heard. We can change this by matching the filter frequency to the respective note pitch. Simply connect the **NotePitch** module with a second wire to the **P**(itch) input of the filter module.

The filter's circuit is designed to add the control signal at the **P** input to the frequency control signal at the **F** input. The sum of the two then determines the filter's cutoff frequency. If you play some high notes they will sound as you would expect.

Adding a Filter Envelope

Finally, we also want to control the filter cutoff frequency with an envelope. For simplicity's sake we will let the existing **ADSR** module take over this task, too. If you wanted to have a more sophisticated synth, you could add a separate envelope module just for the filter.

For the envelope to affect the filter cutoff we first need another component, a multiplier (**Insert Module** ⇒ **Math** ⇒ **Multiply**). Connect this module to **Out** of the **ADSR** module, to the controller **FCutoff** and to the input **F** of the filter module, as illustrated below.



Play a few bars and you will hear that the envelope now affects the filter's frequency. It works like this: The ADSR module outputs a control signal between 0 and 1. This signal is multiplied by the value of the controller **FCutoff**. When the envelope is at its maximum value (1), the filter's input **F** receives the value $1 \times \text{FCutoff} = \text{FCutoff}$. When the envelope reaches its minimum value, the signal at **F** is reduced to zero ($0 \times \text{FCutoff} = 0$).

The function that the controller **FCutoff** now performs is commonly known as "Envelope Modulation Depth". To take this into account in the display, open the controller's Properties dialog window by double-clicking on its module and change the label **F Cutoff** to **Env Mod**.

Variations

Here are some more suggestions for modifications you could make to the structure we have just built:

- Try out the HP and BP outputs of the 2-pole filter module
- Replace the 2-pole filter module with a 4-pole filter module.
- Try out different envelopes.
- Add an extra envelope for the filter module.

So you have other ideas? Go ahead and try them. Always remember that, in contrast to working with hardware components, there's never any danger of breaking anything in REAKTOR.

However, surprises regarding the generated sound will probably occur now and then, so to protect both your speakers and your ears you should not initially set the level of your external amplification too high.

So, get cracking!

10 Basic Operation

The REAKTOR user interface follows the conventions of the operating system, so it is easy to get used to the software for someone who has already worked with MacOS or Windows. Nevertheless we want to explain some particular characteristics and draw your attention to some features that may be new to you.

10.1. Mouse

Practically all functions in REAKTOR can be carried out using the mouse. The main operations that will be performed are the following:

- **Selection** of an object by clicking on it with the left mouse button. Selected objects are recognized by their label field which is colored red. If you want to select several objects, hold down the **Ctrl** key (Windows) or the **Shift** key (MacOS) on the computer keyboard while clicking on the desired objects one after the other. Alternatively you can click with the left mouse button on a blank part of the window and open a frame by dragging with the button pressed. All objects within the frame are selected.
- **Moving** an object is achieved by clicking the left mouse button on it and keeping the button pressed while moving the mouse pointer, and with it the object, to the desired location. To move several objects together, first select all the desired objects, and then move one of the objects as above. All of them will move together according to the mouse movement and all the wiring remains and follows like rubber bands. On releasing the mouse button the modules are aligned on a grid and then remain at the new position. The grid helps to ensure a tidy appearance.
- **Wires** are drawn by clicking with the left mouse button on the out-port whose signal is to be transmitted and keeping the mouse button pressed. Then you move the mouse pointer, and with it the wire, to the desired in-port of another module that is

to receive the signal. There you release the left mouse button - the connection is now established. The wiring operation can also be carried out in the other direction (in-port to out-port) - the result is the same.

- **Double-click** with the left mouse button on an object (or on the background of windows) and one of various actions is performed, depending on the object. The object's context menu shows which action it is that is executed on a double-click by listing the corresponding menu entry in bold type
- **Right mouse button** (Windows) or **ctrl** key + mouse button (⌘ MacOS) opens the context menu that belongs to the object (or window) on which the button was clicked. Context menus play a very important part in the REAKTOR operation which is why the next section is dedicated to explaining them in detail.

10.2. Context Menus

Context menus are lists of commands that are applicable to the object you are clicking on. So, if you want to perform an action on an object or you need information about it, click on it with the right mouse button (Windows) or **ctrl** key + mouse button (⌘ MacOS). A menu appears whose entries relate to the selected object. With a click of the left mouse button you activate the desired menu item. The menu disappears and the operation is carried out. For example, you can delete a module by selecting the entry **Delete** in its context menu.

10.3. Key Commands

A lot of functions in REAKTOR can be performed with keys or key combinations as well as with the mouse. Detailed lists of the key commands available in the different windows are given in chapter 16.4 of this guide.

10.4. Windows

There are two main kinds of windows - **Structure** and **Panel** – whose functions will be explained in detail in the respective chapters of this guide.

In general, the following can be said about the use of windows in REAKTOR:

- As many windows as you like can be open and displayed on the screen at any time.
- Opening windows and jumping to other windows is performed through either the **Window** menu, a context menu, a keyboard shortcut or a double click.
- **Structure** leads to the window that shows the internal wiring of the module.
- **Panel** opens the window with the control elements for the Instrument.
- **Show in Structure/Panel** shows the structure window that is above the current window in the structural hierarchy .
- All open windows are shown in a list at the bottom of the **View** menu. A window can be brought to the front by selecting its entry in the menu.
- You can move, size, minimize and close the windows just like you are used to from other programs. If a window is too small to display its entire contents, you will find the usual scrollbars at its right and bottom edges and you can use them to move around the window contents.

The following applies under Windows:

- All windows are contained in the main window. When the main window is minimized or covered by another application window, all the contained windows are affected.
- When a window is maximized, its border becomes the same as the border of the main window. Afterwards, the same applies to all the other windows until one of them is reset to a smaller size.
- When a window is minimized, it appears as a small box at the bottom edge of the main window.

You can step through the individual windows by pressing **Ctrl + Tab**.

There are two main kinds of windows: Structure and Panel. Whose functions will be explained in detail in the respective chapters of this guide.

In general, the following can be said about the use of windows in REAKTOR:

- Almost any window as you like can be open and displayed on the screen at any time and in any order.
- Opening windows and jumping to other windows is possible through either the Window menu or a context menu (right-click shortcut or double-click on the window title bar).
- Structure shows the window that shows the internal wiring of the module with a number of details of modules that are visible.
- Panel opens the window with the control elements for the instrument.

- Show in StructurePanel shows the structure window that shows the current window in the structure hierarchy.
- All open windows are shown as a list at the bottom of the Window menu. A window can be brought to the foreground by selecting it in the menu.
- You can move, maximize, minimize and close windows just like in other programs. If a window is too small, you can expand it to full screen. You can also use the keyboard shortcuts **Alt + F** to display the full screen and **Alt + W** to return to the normal size. At the right and bottom edges, you can use the mouse to move the window contents.

The following applies under Windows:

- All windows are contained in the main window. If a window is minimized or covered by another application window, all the contained windows are affected.
- When a window is minimized, its border becomes the same as the border of the main window. Afterwards, the same applies to all the other windows until one of them is reset to a smaller size.
- When a window is minimized, it appears as a small box at the bottom edge of the main window.

11 Menus

The commands for operating the REAKTOR system are to be found, in addition to the various context menus (see section *Context Menus* on page 56), in the menus in the menu bar of the main window. The program's global functions controlled from the menu are described below.

11.1. File Menu

New Ensemble

The menu item **New Ensemble** creates an empty **Ensemble**. It consists of only the **Audio-In** and the **Audio-Out** modules.

Open...

By selecting **Open...** you can load an ensemble from any data media connected to your computer. Ensembles have the file extension ***.ens**.

Save Ensemble

The menu item **Save Ensemble** stores the current ensemble together with all the contained instruments and their structures, panels and snapshots in an ***.ens** file.

Save Ensemble As...

The menu item **Save Ensemble As...** also stores the current ensemble together with all the contained instruments and their structures, panels and snapshots in an ***.ens** file. However, here you can specify a new filename and a new directory for the ensemble.

Import MIDI File...

There is an integrated MIDI File Player in REAKTOR allowing the import and playback of MIDI files in the Standard MIDI File format (SMF). Such MIDI files can be produced by nearly every sequencer program. Under Windows they have the file name extension .mid.

With the integrated MIDI File Player you can have REAKTOR play arrangements even without a separate sequencer. This option is especially interesting when using REAKTOR in live performances: A sequencer running in the background on the same computer can cause problems and makes the handling more complicated, since you have to load new files into the sequencer as well as into REAKTOR. On top of that, you have to keep switching between the two programs to have access to important parameters.

There is another advantage to using the integrated MIDI File Player versus using an external sequencer – you will have sample accurate timing. In other words: All notes in the MIDI file which begin at the same time are played by REAKTOR absolutely simultaneously, so the timing is perfectly tight. Nevertheless, it depends on the sequencer you have used for producing the MIDI file, what resolution and accuracy you will get for the timing of the notes.

The REAKTOR MIDI File Player can be load with a MIDI file either manually or automatically: You can find the entry **Open Midi File...** in the **File** menu, for opening a dialog to select a standard MIDI file. When opening and ensemble, REAKTOR loads a MIDI file automatically if it is in the same folder and has the same name as the ensemble (but with the extension .mid).

In the **Settings** menu there are three entries for navigating the MIDI File Player. When the item **Play MIDI File** is activated, the MIDI file is played back when you start the REAKTOR clock. The MIDI file will be played in an endless loop when **Loop MIDI File** is activated. You can use this for instance to keep repeating a pattern or sequence of patterns. Finally, the item **Ignore Tempo Change** disables the tempo changes contained in the MIDI file. When activated the tempo changes are ignored and the whole MIDI file is played back with the tempo set by the REAKTOR clock.

The transport functions of the MIDI File Player are controlled from REAKTOR's clock control functions.

- Pressing the **Start/Restart Clock** button starts the MIDI File playback from the beginning.
- Press the **Pause/Rewind Clock** once and playback of the MIDI File pauses, the **Start/Restart Clock** button jumps out and the player stops playing but maintains its current position.
- Press the **Pause/Rewind Clock** button once more and you reach the Stop mode: The button stays pressed down and the player returns to the start position of the MIDI file.

Recent Ensembles

With a simple mouse click, you can open one of the eight most recently used ensembles.

Exit

The menu item **Exit** closes the program and all its windows, including those in the task bar. Before closing, the software checks if any changes have been made since last saving and asks you what to do if it finds any.

11.2. Edit Menu

Undo

Selecting **Undo** from the menu or pressing **Ctrl + Z** (**⌘ + Z**) reverses the effect of the last editing operation carried out in any of the structures. The Undo function does not apply on panel control changes. Therefore please use the Compare function in the Snapshot window.

The number of steps through which operations can be reversed is determined by the settings in the Preferences. A smaller amount of steps can make sense when the computer memory becomes low.

Redo

Selecting **Redo** from the menu or pressing **Ctrl + Y** reverses the effect of the Undo operation. You can execute Redo as many times as you previously executed **Undo**, to return to the initial state.

Cut

Selecting **Cut** from the menu or pressing **Ctrl + X** (**⌘ + X**) removes the current selection. It is copied to the clipboard from where it can be inserted at another place, even in another window, using the **Paste** command (**Ctrl + V**).

Copy

Selecting **Copy** from the menu or pressing **Ctrl + C** (**⌘ + C**) marks the current selection for copying. A copy of the modules is stored in the clipboard from where it can be inserted at another place, even in another window, using the **Paste** command (**Ctrl + V**).

Paste

Selecting **Paste** from the menu or pressing **Ctrl + V** (**⌘ + V**) copies the current contents of the clipboard into the structure.

When using the keyboard shortcut **Ctrl + V** for pasting, you can specify a window and a point in the structure by clicking there with the left mouse button first.

Delete

The menu entry **Delete** removes the current selection. You can also delete the selected modules and wires with the **Del** key on the computer keyboard. The same function is also available as **Delete** in the context menu of the selected objects.

Select All

With the menu item **Select All** or the keyboard shortcut **Ctrl + A** (⌘ + A) you can mark the entire contents of the current window as selected. You can unselect individual items by clicking on them while holding down the **Ctrl** key.

Mute

The mute function turns off one or more selected modules. All instruments upstream of the selected one are also muted. The same function is available as **Mute** in the context menu of the selected instruments.

Mono

For most modules the operating mode can be switched between monophonic and polyphonic. Unless a module is really needed in poly mode it should definitely be used in mono mode. This is because the CPU load rises proportional to the number of voices. The same function is also available as **Mono** in the context menu of the selected modules.

11.3. Insert Menu

This menu is used to instruments, macros and elementary modules. It is equivalent to the insert menus in the context menu of the structure window.

11.4. Settings Menu

Sample Rate

This menu item sets the internal sample rate for audio signals. With higher sample rates you can achieve better sound quality, but the CPU load rises proportionately. If the internal sample rate is different from the soundcard's sample rate, the Audio In and Audio Out modules will do the conversion. The sample rate can also be adjusted using the selector in the Ensemble Toolbar.

Control Rate

A number of modules (**LFO**, **Slow Random**, **Event Hold**, **A-to-E**, **Event Smoother**) generate or process events at a constant rate. This rate is set here, and has a global effect. Since this sample rate is very low compared to the audio sample rate, these modules need very little CPU power.

Higher control rates give a better resolution in time, resulting in finer steps in the signal.

MIDI Learn

Activates the MIDI-Learn mode for the currently selected panel control. This mode is automatically deactivated after a MIDI-controller message is received. There is a corresponding button in the toolbar.

Set Protected/Set Unprotected

Activates the Protection mode. There is a corresponding button in the toolbar.

Automatic Layout

Activates the Automatic Layout style for all Instrument panels within the Ensemble window. Per default this option is switched on.

External Sync

Toggles between the internally generated clock and the external clock (received via MIDI) for all **Sync Clock** and **1/96 Clock** source modules. Control of the **Start/Stop** source by external MIDI-Start/Stop is also activated. When **External Sync** is activated, the tempo cannot be adjusted in the Master Clock Tempo field on the Toolbar. The internal clock will be adjusted according to the external clock.

MIDI Clock Out

If you enable this option, REAKTOR sends out MIDI clock on all MIDI out ports activated in the MIDI Settings of REAKTOR.

Clock Start

Starts the master clock which controls the Clock sources in the Instruments. This function works with both internal and external clock. It also sets the output of the **Start/Stop** sources to "start". In the Ensemble Toolbar there is a button for the same function.

Clock Stop

Stops the master clock which controls the **Sync Clock** and **1/96 Clock** sources. This function works with both internal and external clock. It sets the output of the Start/Stop sources to the on-value. In the Ensemble Toolbar there is a button for the same function.

Play MIDI File, Loop MIDI File, Ignore Tempo Change

These entries act on the MIDI File Player integrated in REAKTOR. For more information about this, please see section *Import MIDI File...* on page 60.

11.5. System Menu

The various items in the **System** menu are for controlling the audio and MIDI inputs and outputs, sample rate and CPU load of the REAKTOR system.

Run/Stop Audio

With this menu item the audio computations can be started (**Run Audio**) and stopped (**Stop Audio**). Effectively this is the main on/off switch for the REAKTOR software. The same function is delivered by a button in the toolbar.

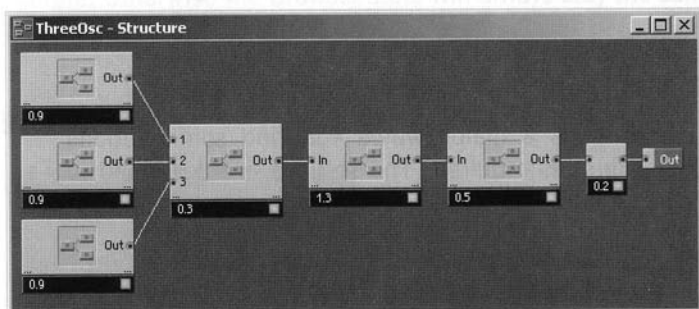
Measure CPU Usage

This menu item switches the modules to load measuring mode. The current processor load is measured for each module and displayed in black labels on the modules. This feature is useful to determine how much load the individual modules are causing. With this information it may be possible to optimize the structure to allow the generation of more voices.

Some modules do not have a number displayed on them, i.e. they keep their normal label. That's because these modules do not actually use up any CPU power for audio processing, either because they are not active or because they do purely event processing.

The displayed value may differ a little from the actual CPU load in normal operation.

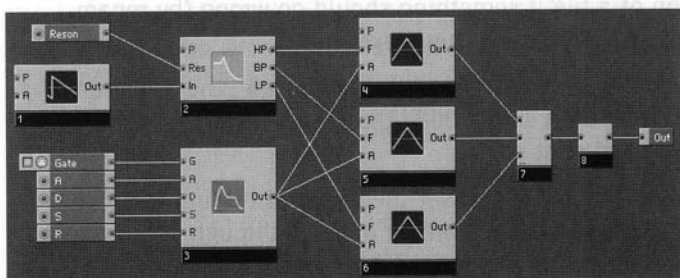
This mode is only available, when **Run Audio** is active. During load measuring the audio output is switched off. The same function is reached by the key combination **Ctrl + U** (**⌘ + U**).



CPU Usage Display

Show Sorting

This menu item switches all audio modules to sorting mode. In this mode the current position within the computing order will be calculated for each module which carries out audio signal processing. This position will be displayed in blue labels on the modules.



Show Sorting mode

Audio + MIDI Settings...

This menu item opens a dialog window for selecting your audio and MIDI interfaces. Detailed instructions are found in the installation chapter.

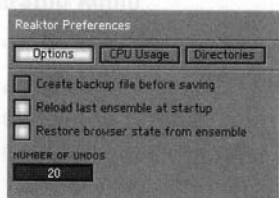
OSC Settings...

This menu item opens the OSC Settings dialog. Detailed instructions on how to use OSC are found in section 8 on page 30.

11.6. Preferences

This menu item opens a dialog where you find some options which you can adjust as required. The settings are described in the following section.

Options



If you enable **Create backup file before saving**, existing files will not be overwritten on **Save** or **Save As...** but will be kept with the filename extension .bak. In this way, you can always return to the original version of a file if something should go wrong (by renaming the extension .bak to the original extension).

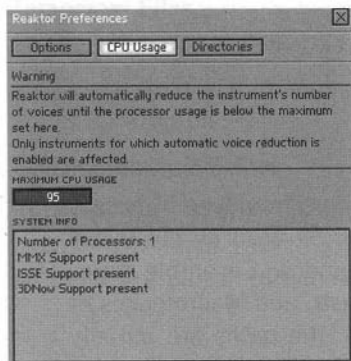
With **Reload last ensemble at start-up** you tell the program whether on starting up it should load the ensemble which was active when it was last shut down.

With **Number of Undos** you can set the depth of the undo buffer. This determines the maximum number of steps through which any changes can be reversed. The entry 0 deactivates the Undo function.

If you work on ensembles containing large audio files, you will need enough computer memory in order to use the undo function. If you come to the memory limit, it is recommended to reduce the Number of Undos or even set it to 0.

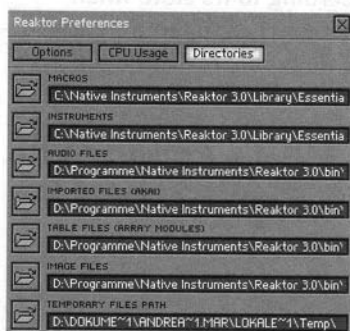
If you activate **Restore Browser State from Ensemble** the Browser position and size will be reconstructed according to its state when you saved the Ensemble the last time, whenever you open an ensemble. Otherwise the Browser state will always stay the same unless you manually change it.

CPU Usage



REAKTOR can automatically reduce the number of voices of polyphonic instruments when the total CPU load reaches a certain limit. In this way, the polyphony can be adjusted according to the available processing power. The upper limit for the CPU load is set here in the Properties under **Maximum processor usage in %**. REAKTOR changes the number of voices only for instruments in which **Automatic Voice Reduction** is activated in the Instrument Properties.

Directories



Macros and Instruments

Here you can set the paths to the two folders from which macros and instruments are loaded. The path will be used by the **Insert** menu, by context menus (e.g. Ensemble window, Ensemble structure, Instrument structure etc.) and the **Instr.** and **Macro** pages of the Browser. After installing the software, the paths are initially set pointing to the supplied library. If you want to build your own library, you can change these paths to point to the folders in which you keep your library.

Or you can add folders containing your own instruments and macros to the existing library and they will also appear in the REAKTOR software's menus. In this way you can customize your library to suit your own taste and needs.

The paths entered here will also be used by the **Inst** and **Macros** pages of the Browser.

Audio Files

This path will be used as default folder when you press the **Select File...** and **Save** buttons in the properties of the tapedeck modules.

Imported Files

This path will be used as default folder for storing converted Map-files using the Akai-Import function.

Table Files

This path will be used as default folder when you press the **Load** and **Save** buttons in the properties of one of the table modules. Table files have the file extension *.ntf and can be used in the Audio and Event Table modules.

Image Files

This path will be used as default folder when you press the **Open** button in the properties of the bitmap module.

Temporary Files

This path will be used for storing temporary files which are created by REAKTOR for opening an audio file in an external sample editor for example.

External Sample Editor

Enter the path of your favourite sample editor here. This entry is used for starting a sample editor by clicking the **Edit** button in the properties of all sampler modules.

11.7. View

Show/Hide Toolbox

Herewith you open and close the REAKTOR toolbar. Therefore you find more details in section 12 on page 78.

Show/Hide Playerbox



REAKTOR offers an audio file player for playing back a loaded audio at the audio input of REAKTOR. The tapedeck plays back directly from the computer's hard disk.

Important note: The Playerbox routes the audio signal directly to the first two Audio inputs of REAKTOR. Normally only effect ensembles or synths with an audio input use the REAKTOR inputs. When you playback an audio file in the Playerbox, external signals on the first two inputs of REAKTOR become muted.

The Playerbox contains the following navigation controls:

- **Load:** Use this button to load and playback an audio file from the hard disk.
- **File** dropdown menu: After having loaded an audio file, the folder from where you have loaded the file will be scanned for additional audio files. These will appear in the **File** dropdown menu for quick access.

Note: The samplerate of the resulting audio file will be the same as the rate you have set for your REAKTOR ensemble during the recording.

- **Play:** Start the playback of a loaded audio file.
- **Pause:** This button interrupts the playback of an audio file. Press the Pause button again to proceed the playback.
- **Stop:** This button stops the playback and sets the audio file playback back to the beginning.
- **Loop:** Enable the Loop function to playback an audio file in an endless loop.
- **Length** display: This display shows the length of the loaded audio file.
- **Played** display: This display indicates the current playback position.

Show/Hide Recorderbox

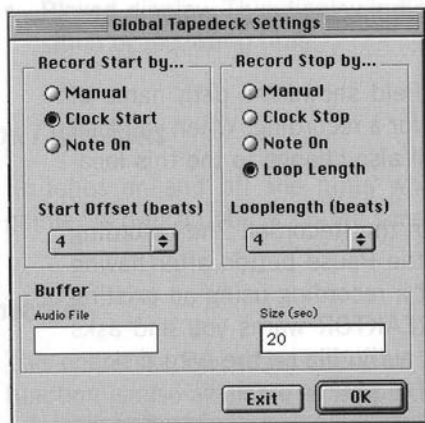


REAKTOR offers an audio recorder for recording the stereo sum signal of an ensemble easily. The tape deck records directly to the computer's hard disk as an audio file. You can also use the output recorder for loading and playing back an existing audio file from the hard disk.

Important note: The Recorderbox records the audio signal from the first two Audio outputs of REAKTOR. Since nearly all Ensembles are connected with these two outputs, you rarely will investigate difficulties with a recoding. If you playback a file with the integrated play functions, the first two outputs will be used for the playback of the Recorder. The signals of the ensemble which are normally routed to the output are muted.

The Recorder has its own toolbox and its own Settings dialog where you can make some important adjustments. The following controls are available in the Recorderbox:

- **Recorder Settings** (Hand with paper icon): This button opens a dialog where you can define events for starting and stopping the recorder.



Recorderbox Settings Dialog

You have the following options within the Recorder Settings:

- **Manual** (separately for start and stop): You start and stop the recorder using the transport buttons in the recorderbox.

- **Clock** (separately for start and stop): The recorder starts when the clock is switched on with the **Start/Restart Clock-switch** in the Recorderbox. The **Pause/Rewind Clock-switch** stops the recording.
- **Note On** (only used for starting a recording): The recorder is started by a MIDI Note On event.
- **Note Off** (only used for stop a recording): The recorder is stopped by a MIDI Note Off event.
- Under **Start Offset** you can define a time delay, measured in beats, for the start of the recording. This can be useful if you want to record an echo effect in an advanced phase for instance but you don't want the beginning. You can also define a length for the recording with the option **Loop Length**. In this menu item you can choose a number of beats as the duration of the recording.
- **Slave Player controls to Recorder** causes the sperate available Playerbox to react on the playback and recording controls of the Recorder box.

Additionally the Recorderbox contains the following navigation controls:

- **Load:** Use this button to select an audio file from the hard disk for playing it back.
- **Directory** display: This display field shows the path name of the current folder which is used for a recording. When you load an audio file, the path name will also change to the this location.
- **Record:** Press this button to arm the Recorder. The recording will start as soon as you press the Pause button after having armed the Recorder. If you start a recording using an existing audio file in the **File** display, REAKTOR warns you and asks wether you wish to overwrite the audio file on the hard disk.

Note: The samplerate of the resulting audio file will be the same as the rate you have set for your REAKTOR ensemble during the recording.

- **New:** Press this button to create a new audio file on your hard disk for a following recording. If you enter a new name in the **File** field after pressing this button, this name will be used for the new audio file. If you do not enter a new name, the old file name will be used, just with an added postfix.
- **File** field: Shows the name of the current or next recording. You can change the name of an existing audio file here. To get a new audio file with a new name, first press the New button and then rename the file.
- **Play:** Start the playback of a loaded or previously recorded audio file.
- **Pause:** This button interrupts the playback or recording of an audio file. Press the Pause button again to proceed the operation.
- **Stop:** This button stops the playback or recording and sets the audio file playback/recording back to the beginning.
- **Loop:** Enable the Loop function to playback an audio file in an endless loop.
- **Length** display: This display shows the length of the loaded audio file.
- **Played** display: This display indicates the current playback position or recording time.

Show / Hide Hints

Switches on and off the hints which appear when the mouse pointer is over an object on the screen.

Properties

This opens the **Properties** dialog for the selected object. The same function is also available as **Properties** in the context menu of the selected modules or panel elements. The **Properties** dialog always refers to the currently selected object and can stay open.

Show/Hide Browser

Opens and closes the Browser window.

Show/Hide Snapshots

Opens and closes the Snapshot window.

Show/Hide Map Editor

Opens and closes the Sample Map Editor window.

Close All Panels

Closes all panel windows.

Close All Structures

Closes all structure windows.

All Panels to Front

Brings all panel windows to the front.

List of Open Windows

By clicking on an entry in this list, the corresponding window will be restored or brought to the front.



plays and controls (from left to

plays and controls (from left to

- the About window.
- If you open an existing ensemble with all changes.
- Start and stop all audio processing function as the entry **Run/Stop** audio processing can be stopped no sounds are being played.
- The CPU time (in percent) used generates the sound. In case of **Over**. The value that can be operation is normally between well below 100%. The reason is to the soundcard, MIDI and display also take up some CPU time must be left over for the other programs to function. You computer by raising the number of the CPU overload message appears to display the level of the used to display the level of the
- REAKTOR displays REAKTOR's internal by selecting from the list. depends on the installed

- The **Stop** button stops the master clock, or the MIDI file player if it is playing back a MIDI file. Pushing the **Stop** button once puts playback in pause, pushing it a second time sets the MIDI file back to the beginning.
- The **Start** button starts the master clock, as well as playback of any imported MIDI file.
- The **Tempo** selector adjusts the rate of the internal master clock in beats per minute (BPM). A double click on the tempo value allows a numerical entry with the computer keyboard. The arrows next to the tempo display adjust the tempo in 1 BPM value increments.
- The **MIDI** lamp lights up red whenever REAKTOR receives a MIDI event from one of the installed MIDI in-ports.
- The **MIDI Learn** button allows you to easily assign a panel control to a MIDI-controller. You select the panel control, activate the **MIDI Learn** button, and then operate the MIDI Controller, e.g. the modulation wheel. In a control's **Properties** you can cancel this assignment by deselecting **Activate MIDI In**.
- The **Protection** button (lock icon) protects the ensemble against accidental editing. Activate the **Protection** mode has some consequences in certain parts of the REAKTOR user interface, e.g. you can not insert, move, delete, copy and paste modules, macros or instruments and snapshots can not be deleted or overwritten.
- The **Show Hints** button (icon with arrow and question mark) activates hints. This function provides information about panel controls. If **Show Hints** is active, you can also hold the mouse pointer above a wire in the structure in order to see the values the wire currently carries.
- The **Panel** button opens the Ensemble window or raises the window to the front.
- The **Structure** button opens the ensemble structure window or raises the window to the front.
- The **Browser** button opens the Browser window or raises the window to the front.

13 The Browser window

REAKTOR's Browser window can be used for loading any data type that is used in REAKTOR. That includes Ensembles, Instruments, Macros, Snapshots, Sampler Maps, Table files, audio files, MIDI files, and so on. You can also use it for analyzing the structure of REAKTOR Ensembles and for in-port and out-port wiring between Instruments and REAKTOR's Audio In and Audio Out modules.

13.1. Window access and organization

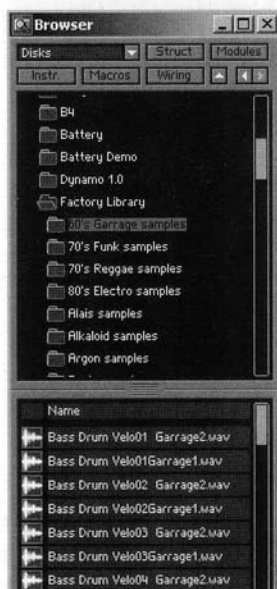
You can open the Browser window from the View menu or using the F5 function key. The window is organized in two panes. The upper pane is for navigation only-click elements there to view their contents or structure in the lower pane. The lower pane is for loading, displaying, and editing tasks.

The buttons along the top (Dropdown menu, **Struct**, **Modules**, **Inst**, **Macros**, and **Wiring**) control what type of data the Browser window displays. They work as follows:

- **Favorite menu:** Select a folder in the Favorites dropdown menu to show all files within this folder that REAKTOR is capable of loading. You can add a folder to the menu by choosing the entry **Add to Favorites**. You always return to your latest selected Favorite folder when none of the buttons above the Browser is selected. Press a selected button a second time to deselect it.
- **Inst:** Use this button to display the Instruments in the default file path you specify for Instruments in REAKTOR's Preferences.
- **Macros:** Use this button to display the Macros in the default file path you specify for Macros in REAKTOR's Preferences.
- **Modules:** Use this button to display all of REAKTOR's Modules, categorized by type.
- **Struct:** Use this button to display the Structure of the current Ensemble in tree format. (See below for further details).

- **Wiring:** Use this button to display all audio in-port and out-port wiring for the current Ensemble. (See below for further details).

13.2. Browser window file types



The Browser

You can use the Browser window to load the following types of data files (specified by file extension) by dragging and dropping to an appropriate location:

- **.wav:** Audio WAV files may be dragged to the Sample Map editor, to a Sampler Module's waveform display on the Control Panel, and to a Tape Deck Module's waveform display on the Control Panel.
- **.aif:** Audio AIF files may be dragged to the same locations as WAV files.
- **.ens:** Ensembles may be dragged anywhere in the REAKTOR window. You will be presented with a dialog allowing you to save the current Ensemble.

- **.ins:** Instrument files may be dragged to either the Ensemble Control Panel or the Ensemble structure window.
- **.mdl:** Macro files may be dragged to either an Instrument Control Panel or an Instrument structure window.
- **Modules:** Modules may be dragged into any Instrument structure window as well as into any Macro structure window.
- **.map:** REAKTOR Sample Map files may be dragged to the Sample Map editor or to a Sampler Module's waveform display on the Control Panel.
- **.mid:** Standard MIDI Files (SMFs) may be dragged anywhere in the REAKTOR window. They will be loaded in to REAKTOR's MIDI file player.
- **.ntf:** REAKTOR Table files may be dragged onto the Table display in any Control Panel.
- **.ssf:** REAKTOR Snapshot bank files may be dragged onto the Snapshot window.

Note: After having inserted an Instrument it is not automatically connected to the output so that you can not hear the signal it produces. This state will be indicated by a grey instead of white label in the Instrument panel header. You can press the Solo (**S**) button in the Instrument header for routing the Instrument directly to the REAKTOR output, independently from the Ensemble's internal structure. This way the Solo button of an Instrument is useable as prehear function. If you want to use an Instrument permanently in the Ensemble, you should connect it with the Output module using the **Wiring** page of the Browser window.

13.3. Structure display



The Browser window can display any REAKTOR structure in a tree format. You can use it to see the structural organization of the Ensemble, to open individual structure windows, and to drag Snapshots to the Snapshot window. The structure tree hierarchy is organized as follows:

Top level

The Ensemble is the top icon in the upper pane of the Browser window. It is displayed with the global Ensemble Snapshot folder, represented by a camera icon, directly below it. This folder only appears if at least one Ensemble Snapshot exists. Select the folder to drag Ensemble Snapshots of the first bank from the lower pane. If your Ensemble contains multiple Snapshot banks, you can select a bank using the Snapshot bank icons which appears in the upper Browser pane when you click on the main Snapshot icon.

Level 2

All Instruments of the Ensemble are displayed in the second level (Note that Instruments embedded in other Instruments or Macros do not appear at this level). The Instrument Snapshot folder appears directly below the Instrument and can be selected for dragging Instrument Snapshots of the first bank from the lower pane. If your Instrument contains multiple Snapshot banks, you can select a bank using the Snapshot bank icons which appears in the upper Browser pane when you click on the main Snapshot icon.

You might see an Instrument on this level named [Ensemble]. That is automatically created when loading earlier REAKTOR Ensembles into the current version, and it displays any structural elements other than Instruments that appear in the Ensemble structure. In later versions of REAKTOR, only Instruments can be placed in the Ensemble structure.

Level 3 and lower

Macros and Instruments inside other Instruments appear on these levels. How many levels there are depends on the depth of the particular Ensemble's structure. Instruments appearing on these levels have Snapshot icons that can be selected to display Snapshots in the lower pane for dragging to the Snapshot window.

13.4. Wiring in the Browser window



The Browser window's Wire display may be used for displaying and wiring Instrument in- and out-ports as well as the Ensembles Audio In and Audio Out Modules. It displays a list of all Instruments and is for wiring on that level only.

Important note: The ports displayed on the Wiring Browser page consists only of the audio ports existing in an Instrument. Event ports will not appear in the list and can not be connected this way. Therefore this feature is only useable for setting up audio connections between Instruments, e.g. routing a sound generator through an audio effect. It is not possible to connect a sequencer with a sound generator using event ports in order to send event information to the sound generator.

When you select an Instrument in the upper pane, you will see port icons which are representing the ports of the instrument. Inputs are displayed with an arrow coming from the left and pointing to the icon, outputs are displayed with arrows pointing to the right and out of the icon. You can select an individual port by clicking on a port icon.

When you select one of the Input or Output MIDI icons, the lower pane will display a list of all ports that can be wired to it (in-ports for an out-port and visa versa). That display has four columns labeled: Module, Port, #, and Wire. You can click on the labels to sort by that column.

- **Module:** The name of the instrument/module owning the port.
- **Port:** The name of the port.
- **#:** REAKTOR's internal numbering of the port.
- **Wire:** A checkbox for connecting or disconnecting a wire from the selected port to the indicated port.

14 Ensemble

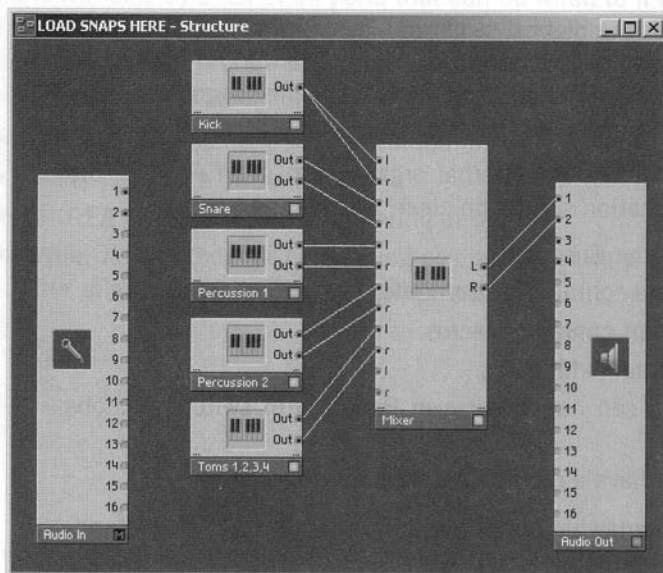
The ensemble is the highest structural level in the REAKTOR software. In an ensemble you store your complete work in its current state so that you can restore it later. As the name suggests, it is an assembly of different instruments.

When understanding the internal organization in REAKTOR, the levels of organization should be clear:

- Top level is the Ensemble.
- An Ensemble contains Instruments.
- An Instrument contains Macros.
- A Macro contains Modules.
- Instruments can have their own Panels with switches, knobs and faders.
- Macros can have a rectangular frame in the Panel

The above is somewhat simplified but sufficient for understanding the general principle.

14.1. Ensemble Structure



*Ensemble with sechs instruments: five sound sources and a mixer.
At left is the Audio-In module.*

The ensemble's structure window gives a bird's eye view of the entire environment, which consists of a number of **Instruments** as well as the audio inputs and outputs representing the sound-card or the mixer of your plug-in host.

Audio-In Module

The **Audio-In** module represents the audio input you have defined with **In Port** under **Audio + MIDI Settings...** in the **System** menu. This module is a fixed part of the ensemble window and cannot be removed from it.

The context menu of the **Audio-In** module contains two entries:

- **Mute** is used to disable the module.
- **Properties** opens the audio interface settings dialog window just like **System Audio + MIDI Settings...**

The **Audio-In** module offers 16 ports. If your audio interface does not support all 16 inputs, not available inputs will be marked with a green cross. You can connect wires to the green marked ports as well to construct an ensemble for a different than the current setup.

Audio-Out Module

The **Audio-Out** module represents the audio output you have defined under **Audio + MIDI Settings...** in the **System** menu. This module is a fixed part of the ensemble window and can not be removed from it.

The context menu of the **Audio-Out** module contains two entries:

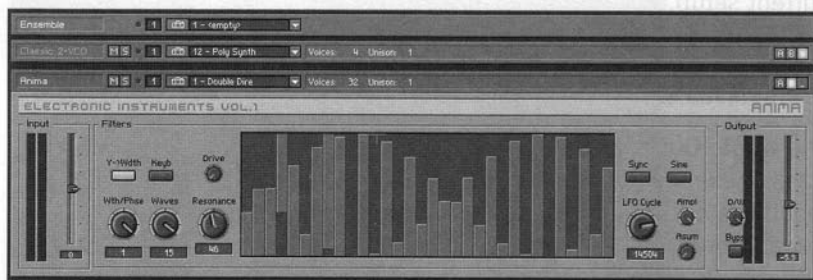
- **Mute** is used to disable the module.
- **Properties** opens the audio interface settings dialog window just like **System Audio + MIDI Settings....**

The **Audio-Out** module offers 16 ports. If your audio interface does not support all 16 outputs, not available outputs will be marked with a green cross. You can connect wires to the green marked ports as well to construct an ensemble for a different than the current setup.

14.2. Ensemble window

Since the ensemble can combine several instruments, its panel can display the controls of several instruments. For this purpose, it is divided into sub-panels. A beam with the label **Ensemble** is always visible. It contains a snapshot dropdown list for **Ensemble**

Snapshots which normally control the snapshots of all Instruments within the ensemble. For each instrument in the ensemble there is an additional sub-panel with the name of the instrument visible in the title bar.



Ensemble-panel with the sub-panels of instruments

You can use the switch **Visible** in the Properties (**Appearance** page) of the panel elements to determine whether you can see them in the Ensemble window. On the **Appearance** page of a panel element's Properties you can also choose the panel you want the control to be visible. You have three options here: A only, B only, A and B. The default setting for an instrument's control is that it is visible in both panels of the instrument.

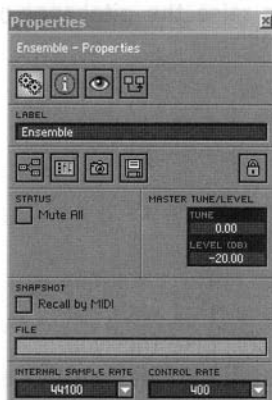
You can switch between both panels in the Instrument Headers of the Ensemble window. You can minimize an Instrument panel by clicking the **Minimize** button on the right of the Instrument Header. Clicking the **Minimize** button a second time will open the Instrument panel again.

14.3. Ensemble Properties

The Ensemble Properties can be edited in the non-modal dialog box which you open with a double click on the Ensemble name or an empty area in the Ensemble Header. You can also open the Ensemble Properties by right clicking (MacOS: Ctrl + Click) on the Ensemble Header and choosing the appropriate entry from the context menu.

The Ensemble Properties contain four pages represented by icons. These icons stand for **Function**, **Info**, **Appearance** and **Connection**. The box can stay open while you are working on your REAKTOR ensemble. When selecting an instrument, a macro or a module the content of the Properties window will change and display the parameters for the selected element.

Function page



Properties window of an ensemble (Function page)

- In the **Label** field, the ensemble can be given a name which appears on the ensemble header.
- The next row contains shortcut buttons for quick access to a few basic Ensemble functions: The **Structure** button (structure icon) opens the Ensemble Structure or raises the Structure to the foreground. The **Panel** button (panel icon) opens the Ensemble window or raises the panel to the foreground. The **Snapshot** button (camera icon) opens the Snapshot window for the Ensemble or raises the Snapshot window to the foreground. The **Save as** button (disk icon) opens the Save dialog in order to save the Ensemble as Ensemble file (file extension ***.ens**). Finally you find the **Protection**-function here represented by padlock icon. Most functions are also available in the REAKTOR toolbar.

Status

- With **Mute All** all instruments within the ensemble can be deactivated. The ensemble then uses no CPU power and all instruments are marked with a red cross over their status lamp in the **Ensemble** structure window.

Master Tune/Level

- You can use **Tune** to change the ensemble's pitch. The value is given in units of semitones. Positive values raise the pitch, negative values lower it.
- You can use **Level** to change the ensemble's global level. The value is given in decibel. Positive values raise the level, negative values lower it.

Snapshot

- When **Recall by MIDI** is activated, any received MIDI Program Change messages will cause the Ensemble snapshot with the appropriate number to be called up if it exists. Like this you can quickly recall an Ensemble snapshot in REAKTOR from your MIDI controller (master keyboard) by issuing a MIDI Program Change command with the appropriate number.

File

- The field labeled **File** shows the name of the file from which the ensemble was loaded.

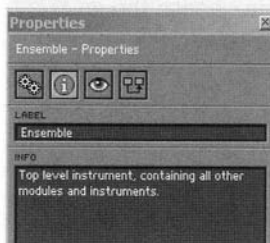
Internal Sample Rate

- The selector for the **Sample Rate** displays REAKTOR's internal sample rate, which can be changed by selecting from the list. Which sample rates are available depends on the installed soundcard.

Control Rate

- A number of modules (**LFO**, **Slow Random**, **Event Hold**, **A-to-E**, **Event Smoother**) generate or process events at a constant rate. This rate is set here, and has a global effect. Since this sample rate is very low compared to the audio sample rate, these modules need very little CPU power. Higher control rates give a better resolution in time, resulting in finer steps in the signal.

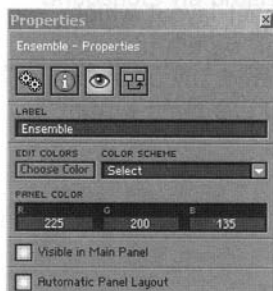
Info page



Properties window of an ensemble (Info page)

- Enter important information about your ensemble into the **Text field** of the Info page. This text is displayed whenever you hold your mouse above an empty space of the ensemble header within the Ensemble window (when the Info-Button in the Toolbar is pressed).

Appearance page



Properties window of an ensemble (Appearance page)

Edit Color

- **Choose Color:** Press this button in order to open the standard system's color palette to choose a color for the Ensemble header in the Ensemble window. You actually choose a color for the corresponding panel like in the Instrument properties, but since an Ensemble panel is not existing in REAKTOR, the Ensemble header uses the chosen color as base and generates it in a darker shade.
- **Color Scheme:** Choose **Save Custom** to store the current color settings. You can only save one Custom color scheme. To apply your custom color scheme to another Ensemble choose the entry **Set to Custom** there. The entry **Set to Default** sets the color scheme to the standard REAKTOR colors (grey panel with orange indicator colors).
- **Panel Color:** The panel color can be mixed from the three primary colors (RGB = red ,green, blue). Entering 0 for all primary colors creates black, entering 256 for all primary colors creates white.

Visible in Main Panel

- This setting is for showing and hiding the Ensemble header in the Ensemble window.

Automatic Panel Layout

- This setting is active by default. If you load old REAKTOR ensembles where you see a lot of empty Instrument panel space or if for some reason you want to have a panel layout which does not fit in the REAKTOR 4 panel concept, switch this option off. If **Automatic Panel Layout** is deactivated, the boundaries of a panel are determined by the control elements within the panel.

Connection page



Properties window of an ensemble (Connection page)

MIDI In

- It is possible to set an individual **MIDI In Device** for your ensemble. The MIDI device has to be activated in the **Audio + MIDI Settings** of REAKTOR in order to appear in the **MIDI In Device** drop-down menu..
- **Activate MIDI In** has to be active for receiving any external MIDI data in the ensemble (e.g. for controlling Ensemble Snapshots via program change). This setting has no impact on MIDI reception of Instruments.
- With **Receive Channel** you set the MIDI Channel used by the ensemble for MIDI input. The ensemble receives only MIDI data which is sent on the MIDI channel you enter in the box **Receive Channel**.

MIDI Out

- It is possible to set an individual **MIDI Out Device** for your Ensemble. The MIDI device has to be activated in the **Audio + MIDI Settings** of REAKTOR in order to appear in the **MIDI Out Device** drop-down menu..
- **Activate MIDI Out** has to be active for sending any MIDI data to an external MIDI device (e.g. master clock or program change messages).
- With **Send Channel** you set the MIDI Channel used by the ensemble for MIDI output. The ensemble sends MIDI data on the channel which is entered here.

Internal Connection

- Like with instruments you can also include an ensemble within an internal or OSC connection. The ensemble is handled like an instrument in this context. The internal/OSC connection can be used in an ensemble to control ensemble snapshots. More details can be found in the Instrument Properties section.

External Sync

- Toggles between the internally generated clock and the external clock (received via MIDI) for all **Sync Clock** and **1/96 Clock** source modules. Control of the **Start/Stop** source by MIDI-Start/Stop is also activated. When **External Sync** is activated, the tempo cannot be adjusted in the Master Clock Tempo field on the Toolbar.

15 Instruments

15.1. What is an Instrument?

An **Instrument** in REAKTOR is a module that has an internal structure, its own MIDI processing, a separate control panel and separate snapshots. Instrument modules can be recognized by their dark blue label and have an icon symbolizing a control panel (represented by two faders and two knobs).



Instrument Module

An instrument can contain other instruments and macros in such a way that they form a hierarchical structure. The highest level in this hierarchy, i.e. the instrument that contains everything else, is the ensemble.

15.2. Creating Instruments

Instruments are added into a structure (normally the ensemble) by loading them from the library which comes with the REAKTOR software. In the folder **Instruments** of the library there are numerous ready-made sound generators and effects. If you want to start developing a new instrument you first need to load an empty one from the library (**Instruments\New**).

When inserting an instrument you are in effect creating a local copy of the instrument from the file. This means that changes made later to the file do not touch the instrument once it has been inserted. Likewise, the file does not change when editing the instrument in REAKTOR. If you want to update the file you must write the instrument back to it using **Save As....**

15.3. Ports

There is no fixed arrangement of ports for instruments. The type and number of ports in an instrument is determined by the user by the insertion of so-called **Terminals** in the instrument structure (see section *Terminals* on page 123).

The connections into and out of an instrument through the terminals are always **monophonic**, they cannot be polyphonic. This is necessary because the number of voices inside the instrument will normally be different from that outside - only monophonic signals will work in any configuration. If the instrument is polyphonic, a **Voice Combiner** module needs to be inserted before the output port(s) to make sure that the signal leaving the instrument is monophonic.

15.4. Context Menu

The context menu of an instrument contains the following entries:

- **Mute** disables the instrument.
- **Cut** removes the instrument from its current position. It is copied to the clipboard from where it can be inserted at another place, even in another window, using the **Paste** command.
- **Copy** marks the instrument for copying. A copy of the instrument is stored in the clipboard from where it can be inserted at another place, even in another window, using the **Paste** command.
- **Delete** removes the instrument and everything it contains.
- **Save As...** allows an instrument to be saved. The file location as well as the filename can be specified. Instrument files are given the filename extension `.ism`.
- **Panel** opens the instrument's panel window. See section *Panel Editing* on page 130 for details about the panel.
- **Structure** opens the instrument's structure window to display its internal wiring. See section *Structures* on page 114 for details about creating and editing structures.
- **Properties** opens a window with various information about and settings for the instrument. Details are given in the following section.

15.5. Instrument Header

The **Instrument Header** is used to adjust and control an instrument.



The Instrument Header

The Instrument Header contains the following displays and controls (from left to right):

- The Instrument **Label** which you can enter in the Instrument Properties appears on the left side of the Header.
- Right hand of the label there is an icon displaying the state of the Instrument panel. If you see a screw icon the panel is in **Panel Lock** mode and elements within the panel can be edited but not moved. If you see a square icon the panel is in **Panel Edit** mode and elements within the panel can be moved but not edited. You can switch between both modes by clicking once on the icon or by choosing the appropriate entry in the Instrument panel's context menu.
- The **Mute** button mutes the selected instrument. All instruments upstream of the selected one are also muted.
- The **Solo** button directly connects the output of the selected instrument to the audio output. All instruments which lie upstream of the selected instrument in the structure remain active. All other instruments in the ensemble are muted. An example: A synthesizer signal is processed by a chorus effect. The chorus effect is switched to **Solo**. Then the synthesizer with the chorus effect is connected with the output, while all other instruments in the ensemble are muted.
- The **Instrument MIDI Activity** lamp shows the reception of MIDI events by the instrument.
- The **Channel** selector displays and changes the MIDI channel of the instrument. In order to set a different MIDI channel click on the value field and drag up or down while holding down the mouse button. For a numerical entry double click on the value and enter the channel with the computer keyboard. You can also adjust the MIDI channel in the Instrument Properties.

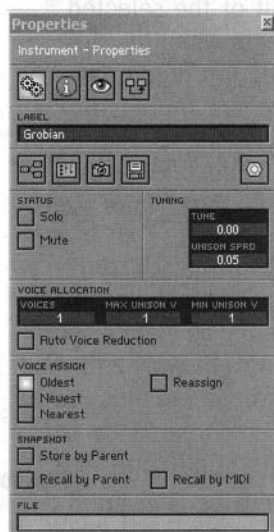
- With the **Snapshot** button (camera as icon) you open the window for snapshot management (see section *Snapshots* on page 141).
- With this **Snapshot selector** you can choose a snapshot to recall on the instrument.
- The **Voices** selector displays and changes the number of voices of the instrument. The number of voices can also be adjusted in the Instrument Properties.
- The **Unison** selector displays and changes the (maximum) number of unison voices per note. The unison effect is enabled by selecting a value greater than one. The unison voices are detuned by a value which is set with **Unison Spread** in the Instrument Properties. The minimum number of unison voices per note (**Min Unison Voices**) can be set there as well.
- The panel selectors **A** and **B** are for choosing one of two panels for the same Instrument which you can use for two alternative views. You can decide for each panel element in its properties (Appearance tab) whether the control should appear only in one of the two panels or in both.
- The **Minimize** button minimizes the Instrument panel so that only the Instrument Header remains visible. Click on the button again to unfold the Instrument panel again.

15.6. Instrument Properties

The Instrument Properties can be edited in the non-modal dialog box which you open with a double click on the instrument name or an empty area in the Instrument Header. You can also open the Instrument Properties by right clicking (MacOS: Ctrl + Click) on the Instrument Header or the Instrument module in the Ensemble Structure window and choosing the appropriate entry from the context menu.

The Instrument Properties contain four pages represented by icons. These icons stand for **Function**, **Info**, **Appearance** and **Connection**. The box can stay open while you are working on your REAKTOR ensemble. When selecting another instrument, a macro or a module the content of the Properties window will change and display the parameters for the selected element.

Function page



Properties window of an instrument (Function page)

- In the **Label** field, the instrument can be given a name which appears on the instrument module in the ensemble.
- The next row contains shortcut buttons for quick access to a few basic Instrument functions: The **Structure** button (structure icon) opens the Instrument Structure or raises the Structure to the foreground. The **Panel** button (panel icon) opens the Instrument Panel or raises the panel to the foreground. The **Snapshot** button (camera icon) opens the Snapshot window for the Instrument or raises the Snapshot window to the foreground. The **Save as** button (disk icon) opens the Save dialog in order to save the Instrument as Instrument file (file extension *.ism). Finally you find the **Panel Lock**-function here represented by the screw and wrench icons as you know it from the Instrument panel header (you find a detailed description in the Instrument header section).

Status

- The **Solo** button directly connects the output of the selected instrument to the audio output. All instruments which lie upstream of the selected instrument in the structure remain active. All other instruments in the ensemble are muted. An example: A synthesizer signal is processed by a chorus effect. The chorus effect is switched to **Solo**. Then the synthesizer with the chorus effect is connected with the output, while all other instruments in the ensemble are muted.
- With **Mute** the instrument can be deactivated. It then uses no CPU power and is marked with a red cross over its status lamp in the **Ensemble** structure window.

Tuning

- You can use **Tune** to change the instrument's pitch with respect to the tuning set in the ensemble. The value is given in units of semitones. Positive values raise the pitch, negative values lower it. A typical application would be to detune two instruments to get a fatter sound. A good value for this would be **Tune** = 0.05 (5 cents).
- The parameter **Unison Spread** determines the degree of detuning between the voices sounding in unison. The value is given in semitone steps. A typical value would be 0.05 (5 cents), which would mean that each of the voices playing the same note is detuned by 5 cents relative to the next voice.

Voice Allocation

- Each instrument has its own polyphonic voice allocation. **Voices** specifies how many voices of polyphony the instrument is to process at a time. This number applies to all the polyphonic modules in the instrument (i.e. all modules except those set to mono) and tells the modules how many parallel versions of the processing to carry out.

- **Unison mode** is activated if you set **Max Unison V** to a value greater than 1. This means that more than one voice plays the same note and the voices are detuned slightly to get a rich, fat sound. **Max Unison V** determines the number of voices that are assigned to a newly triggered note when enough voices are still available.
- **Min Unison V** is used to set the minimum number of voices assigned to a new note when not enough voices are available. It can never be greater than **Max Unison V**.
- If **Auto Voice Reduction** is activated, REAKTOR can automatically reduce the number of voices when the total CPU load (set in Preferences under **Processor Usage**) exceeds a certain limit. In this way, the polyphony can be adjusted according to the available processing power.

Voice Assign

- When the number of voices in an Instrument is not sufficient for playing all the pressed keys, REAKTOR needs to choose intelligently which voice to use for a newly pressed key. This voice is then taken away from a previously pressed key (voice stealing). The method used for voice assignment has three options: **Oldest**, **Newest** and **Nearest**. When **Oldest** is activated the voice which has already been held longest is stopped and assigned to the new note. This is the most common mode. With **Newest** it is the most recently played note which is taken away to play the new note. This can be useful for playing a melody over held notes. In **Nearest** mode, the voice playing the note which is closest in pitch to the new note is reused. This is good when using polyphonic portamento (glide).
- With **Reassign** you can set what happens when playing the same note several times. Either the voice already playing the note is reused or a second voice is used to play the same note again. **Reassign** mode is good for making efficient use of a limited number of voices, and it is also what you are used to from the piano.

Snapshot

- If **Store by Parent** is activated, everytime you make a snapshot for the instrument or ensemble that is above this instrument in the REAKTOR hierarchy, a snapshot with the same name will be created for this instrument at the same time. Like this, you do not need to make snapshots for every instrument in a complex ensemble first, before you can generate ensemble snapshots, instead you can immediately make ensemble snapshots in one take.
- When **Recall by Parent** is activated, the instrument will change its snapshot when the instrument above it in the hierarchy (usually the ensemble) changes its snapshot. Each snapshot of the parent is linked to one snapshot of the child instrument. The connection is made when a snapshot is stored in the parent, storing the number of the child's currently selected snapshot.
- When **Recall by MIDI** is activated, any received MIDI Program Change messages will cause the snapshot with the appropriate number to be called up if it exists. Like this you can quickly recall a snapshot in REAKTOR from your MIDI controller (master keyboard) by issuing a MIDI Program Change command with the appropriate number.
- With **All Controls Visible in Ensemble** you can put all controls of the instrument into the Ensemble window at one stroke.

File

- The field labeled **File** shows the name of the file from which the instrument was loaded.

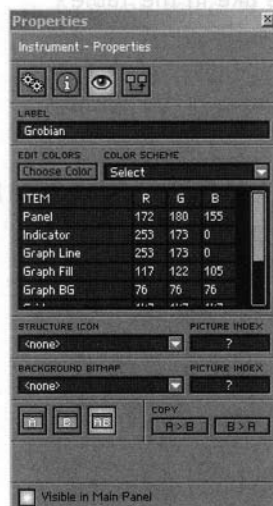
Info page



Properties window of an instrument (Info page)

- Enter important information about your instrument into the **Text field** in the Info tab. This text is displayed whenever you hold your mouse above an empty space of an instrument panel or the instrument symbol in the structure window (when the Info-Button in the Toolbar is pressed).

Appearance page



Properties window of an instrument (Appearance page)

Edit Color

- **Choose Color:** Press this button in order to open the standard system's color palette to choose a color for the selected entry in the item list below.
- **Color scheme:** Choose **Save Custom** to store the current color settings entered in the item list. You can only save one Custom color scheme for using it in your instruments. To apply your custom color scheme to another Instrument choose the entry **Set to Custom** there. The entry **Set to Default** sets the color scheme to the standard REAKTOR colors (grey panel with orange indicator colors).
- **Item list:** This area lists all items of the panel which can have customized colors. The color for each item can be mixed from the three primary colors (RGB = red, green, blue). Entering 0 for all primary colors creates black, entering 256 for all primary colors creates white. The following items are available:
- **Panel:** Color for the background of the panel if no background bitmap for the panel has been chosen.
- **Indicator:** Indicator color of panel controls like Fader, Knob and Buttons.
- **Graph Line:** This color is used for graph lines like in the Table module, the cursor color in the XY module or the outlines of the fill color in the filter and envelope curve displays.
- **Graph Fill:** This color is used for graph fills like in the Table module, the object color in the XY module or the fill color in the filter and envelope curve displays.
- **Graph BG:** This color defines the background color of the panel displays for some modules, e.g. Table, XY, Envelope, Filter etc..
- **Grid:** This color is used for the grid in the Table modules.
- **2D Table Min:** This color is used for the minimum value in the 2D view of a Table module.
- **2D Table Max:** This color is used for the maximum value in the 2D view of a Table module.
- **2D Table Default:** This color is used for the default value in the 2D view of a Table module.

Structure Icon

- **Structure Icon:** You can load your own bitmap for the instrument symbol in the structure window.
- **Picture Index:** If you use a bitmap containing multiple smaller pictures you can select the index for the picture cutout after having set the **Num Animations** in the **Picture Properties**.

Background Bitmap

- **Background Bitmap:** You can load your own bitmap for the Instrument panel. All controls and other bitmaps will overlay the background bitmap.
- **Picture Index:** If you use a bitmap containing multiple smaller pictures you can select the index for the picture cutout after having set the **Num Animations** in the **Picture Properties**.

Panel Controls

- **A, B, AB:** Select one of three options for applying the commands "All controls visible" and "All controls invisible" below. If you select A these commands will only be applied to Panel A, if you select B, only to Panel B, if you select AB, they will be applied to both, A and B.
- **Copy A > B/B > A:** Click on one of these buttons for copying the full content of the according panel to the other.

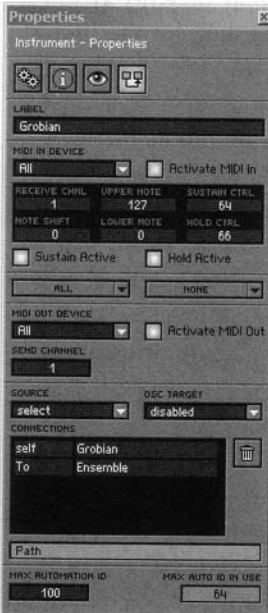
All Controls

- The **Visible** button allows to put all panel controls of the Instrument into the Ensemble window with one click. This function corresponds to the panel(s) which are selected with the panel selection buttons **A, B, AB**.
- The **Invisible** button allows to hide all panel controls of the Instrument in the Ensemble window with one click. This function corresponds to the panel(s) which are selected with the panel selection buttons **A, B, AB**.

Visible in Main Panel

- This setting is for showing and hiding the whole Instrument including the Instrument header in the Ensemble window. The panel selection buttons have no impact on this setting, since you apply it always to the complete Instrument.

Connection page



Properties window of an instrument (Connection Tab)

MIDI In

- It is possible to set an individual **MIDI In Device** for your Instrument. The MIDI device has to be activated in the **Audio + MIDI Settings** of REAKTOR in order to appear in the **MIDI In Device** drop-down menu..
- **Activate MIDI In** has to be active for receiving any external MIDI data in the Instrument.

- With **Receive Channel** you set the MIDI Channel used by the instrument for MIDI input. The instrument receives all MIDI data which is sent on the MIDI channel you enter in the box **Receive Channel**.
- With **Note Shift** all the received MIDI note information is transposed by the given number of semitones. For example, if you want to transpose the whole instrument down by one octave you have to enter the value -12 here.
- With **Upper Note Limit** and **Lower Note Limit** the range of MIDI note numbers that the instrument recognizes can be limited. Any note numbers outside the given range are ignored. This can be used for example to program a keyboard split.
- With **Sustain Control** the number of the MIDI controller (foot pedal) which works as sustain pedal (called "hold" or "damper pedal" by some MIDI equipment manufacturers, standard controller number 64) is set. As long as sustain is on (controller value 64 or more) any playing note will be held even after its key is released. This function is only used if the **Sustain Active** option underneath is ticked.
- With **Hold Control** the number of the MIDI controller (foot pedal) which works as hold pedal (called "sostenuto" by some MIDI equipment manufacturers, standard controller no: 66) is set. All notes that are already playing when hold is switched on will be held even after their key is released until hold is switched off. Keys that are pressed when hold is already on are not affected. This function is only used if the **Hold Active** option underneath is ticked.

All and None drop-down menu

- Choose an of the entry of the **All** drop-down menu to transfer the selected option to all controls of the Instrument. Choose an entry of the **None** drop-down menu to remove this option from all controls of the Instrument. Read more about the options appearing here in the Panel Controls chapter.

MIDI Out

- It is possible to set an individual **MIDI Out Device** for your Instrument. The MIDI device has to be activated in the **Audio + MIDI Settings** of REAKTOR in order to appear in the **MIDI Out Device** drop-down menu..
- **Activate MIDI Out** has to be active for sending any MIDI data to an external MIDI device.
- With **Send Channel** you set the MIDI Channel used by the instrument for MIDI output. The instrument sends MIDI data on the channel which is entered here.

Connection

- With **Connection** you can have an internal, wireless connection between different instruments in the REAKTOR ensemble. So for example, you can animate controls or snapshots of one instrument from another instrument. To do this, open the Instrument properties of the target Instrument and choose another Instrument in the **Source** drop-down menu. You can delete an existing Internal Connection via the **Delete** button showing a trash bin icon. The **OSC Target** drop-down is used to select another OSC computer for addressing MIDI data generated by the Instrument. Only computers which are present in the OSC member list of the **OSC Settings** dialog (System menu) are available here. The display field below the Connections list area displays the "path name" of the Instrument referring to the internal structure of the ensemble, e.g. if an Instrument called "Synth" is located in an ensemble called "Ensemble" the path will be indicated as "Ensemble/Synth".

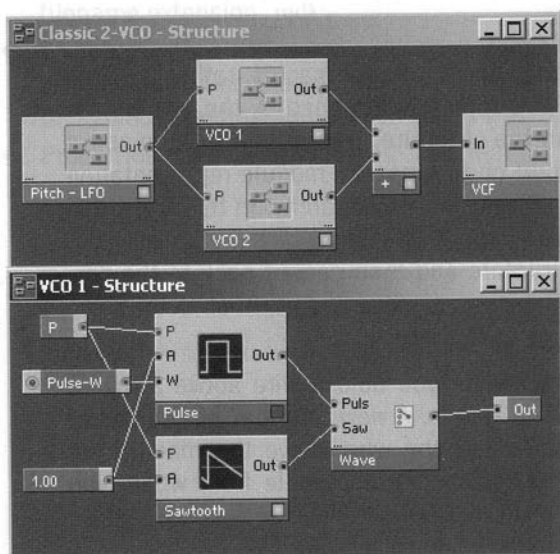
Important note: All Instruments used in an Internal or OSC Connection must be build in a certain way in order to work. While all sound generators you are able to play via MIDI will have the necessary modules for receiving MIDI notes, many integrated sequencers will not have the appropriate MIDI out modules included for sending MIDI notes. You should refer to the Instrument's documentation if existing whether it is capable of this feature or not.

16 Macros

16.1. What is a Macro?

Macros have an internal structure just like instruments, but unlike instruments they do not have their own management of MIDI data, no separate panel and no snapshots. Macros have a gray label and can be recognized by an icon representing a structure with 3 modules.

The main application for macros is the encapsulation of functional blocks to obtain a hierarchical and clearer layout of complex structures. Extensive structures should always be realized using macros. Macros are also a convenient way to build re-usable components.



Example for the integration of a macro into a structure

16.2. Creating Macros

Macros are added into a structure (often an instrument) by loading them from the library which comes with the REAKTOR software. In the folder **Macros** of the library there are numerous ready-made utility and specialized components. If you want to start developing a new macro you first need to load an empty one from the library (**Macros\New**).

When inserting a macro you are in effect creating a local copy of the macro from the file. This means that changes made later to the file do not touch the macro once it has been inserted. Likewise, the file does not change when editing the macro in REAKTOR. If you want to update the file you must write the macro back to it using **Save As....**

16.3. Ports

There is no fixed arrangement of ports for macros, rather the type and number of ports in a macro is determined by the user by the insertion of **Terminals** in the structure. A terminal in the macro's structure appears as a port on the macro's module representation at the next higher level (**Parent**) in the hierarchy. In this way wires connected to the macro's inputs feed it signals which are processed in its internal structure and then passed back to the parent structure through the macro's output ports. For details see section *Terminals* on page 123.

Ports can also be created when you hold a wire above a macro where you want the port to appear. A new terminal will be added automatically when you release the wire above the macro. The port will get the name of the macro or module from where you have dragged the wire.

16.4. Context Menu

The context menu of a macro contains eight entries:

- **Mute** disables the macro.

- **Mono** switches the macro to monophonic operation by switching all the modules inside to mono. You should always use this function unless the module has to work polyphonically because in mono mode there is significantly less load on the CPU
- **Cut** removes the macro from its current position. It is copied to the clipboard from where it can be inserted at another place, even in another window, using the **Paste** command.
- **Copy** marks the macro for copying. A copy of the macro is stored in the clipboard from where it can be inserted at another place, even in another window, using the **Paste** command.
- **Delete** removes the macro and everything it contains.
- **Save As...** allows saving the macro. The file location as well as the filename can be specified. Macro files are given the filename extension .mdl.
- **Structure** opens the macro's structure window to display its internal wiring. See the section *Structures* on page 114 for details about creating and editing structures.
- **Properties** opens a window with information about the macro.

Creating

To create a new macro use the context menu of the structure window. When building structures in REAKTOR, keeping a certain hierarchy of modules is highly recommended. REAKTOR does not force these on you but basically gives you complete freedom to do what you want. For example, it is not possible to create a structure with just elementary modules at the ensemble level. The highest level in REAKTOR - Only ensembles can be used at the ensemble level.

When creating complex devices, it is important to maintain a clear layout. The following recommendations will help maintain an appropriately clean design.

- At the ensemble level, you can only work with instruments, not with elementary modules. Even elements like mixers which you use to process signals from several instruments before audio output are available as separate instruments in the library.
- During the construction of instruments group as many functional blocks as possible in the form of macros. This has the advantage that identical elements such as oscillators or envelopes, which are often used more than once in the construction of synthesizers, need only be constructed once and can then be copied easily. Also, your structures will become very clear so that you will find it easier to track down any problems.

17.2. Modules

A module is the smallest hierarchical unit in REAKTOR. It is displayed as a graphical object. Each module is marked with a **label** and an **icon** (e.g. showing its waveform for oscillators).



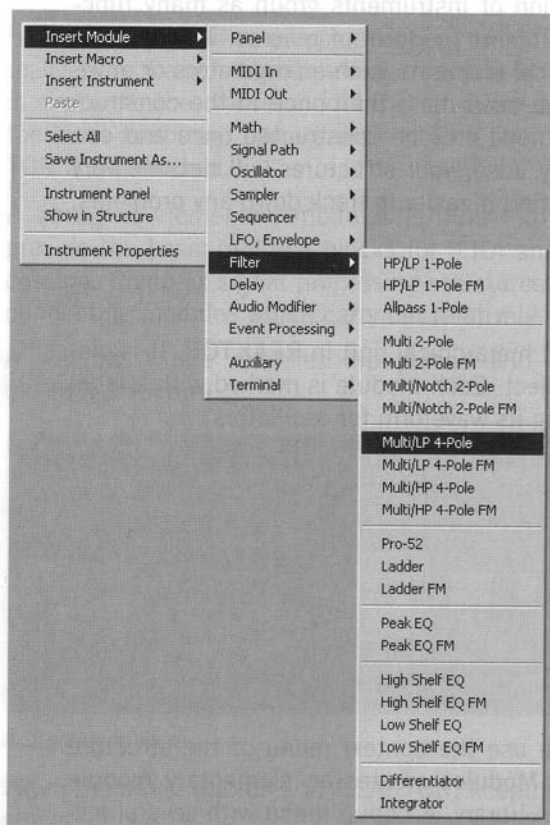
The Pulse FM oscillator module

Creating

To create a new module use the context menu of the structure window. The sub-menu **Modules** creates an elementary module from REAKTOR's built in library. A popup menu with several levels appears:

First you select the functional group (e.g. Filter) and then choose the actual module (e.g. Multi/LP 4-Pole). Detailed information about all of REAKTOR's modules can be found in the Module Reference.

The new module will be placed at the point in the window where you opened the context menu (with a right mouse-click (Windows) or **ctrl** + mouse-click (MacOS)), but then you can move it around like any other REAKTOR object.



Menu for inserting a new module

Ports

Each REAKTOR module contains one or more ports through which the module can be connected to other modules. The left side of the module contains the **In-Ports** and the right side holds the **Out-Ports**.

When any in-port is left unconnected, it always uses a zero signal. So, connecting no wire to an in-port has the same result as connecting a constant source with the value set to zero.

REAKTOR distinguishes between two kinds of information that can be understood or sent by a port, **audio** and **events**:

- **Audio** signals are comparable to sound signals and control voltages in the analog world. The processing of such a signal constitutes a permanent load on the CPU. Ports for audio signals are labeled with black characters. When wiring audio ports, note that an audio input can never process more than one signal. If more than one audio signal is to be fed to an audio input, they must first be mixed using an audio adder or a mixer module. If a connection is made to an audio in-port that already has a wire attached, the first wire will be deleted as the second one is connected.
- **Event** signals are control messages for changing a value. Typical sources for events are MIDI inputs and panel faders. Event processing allows complex manipulation of control messages without continuous calculations and thereby reduces the load on the CPU. Ports for event signals are labeled with red characters and marked with a small red dot. An event input can be fed from several event outputs, in which case the events from the different sources will be merged into one stream. Gate signals are a special case of Event signals. An event with a non-zero value turns on the gate. When it is followed with a zero-valued event, the gate is turned off again.

Some modules can be used either for audio or event signals. If you insert such a module (the **Add** module for instance), it will appear first as an event module. As soon as you connect an audio wire to one of its inputs it will convert to an audio module and the CPU usage becomes remarkable higher with each additional connection.

Each port has a **context menu** with the following entries:

- **Create Control** automatically creates a suitable panel controller for the port. (see the section *Panel Controls* on page 131 for details about working with controllers on the panel).

- **Wire/Unwire** creates a connection from this port to another port.
- **Mute** temporarily deactivates the port, i.e. sets its value to zero. Muted ports are marked with a small red cross.
- **Properties** opens a dialog window with information about the port.

Mono

A module can operate either in monophonic, i.e. single voice, mode or in polyphonic mode where processing is carried out for several voices in parallel. The number of voices of a polyphonic module is determined by the instrument to which the module belongs. Polyphonic modules can be identified by the yellow color of the status lamp at the bottom left corner of the module. Monophonic modules have an orange status lamp.

For most modules the operating mode can be changed using the entry **Mono** in the context menu or the **Mono** switch in the Module Properties. Unless a module is really needed in poly mode it should definitely be used in mono mode. The CPU load is proportional to the number of voices used.

Mute

Modules can be deactivated by selecting **Mute** in the context menu or in the Properties dialog of the module. Muted modules are recognized by a red cross over the status lamp.

A muted module no longer causes any computational load. If a module is not needed temporarily it should be deactivated (if it is never used it should be deleted).

Modules are automatically deactivated if their outputs are not connected, or are only connected to other deactivated modules. The status lamp of deactivated modules is unlit.

This feature is especially useful when using switches, because alternative branches of signal processing can be selected but only one causes CPU load. It works like this: Only one of the inputs of a switch is active at any one time – the switch position determines which input. The signals of all the modules connected to inactive inputs are therefore not needed. REAKTOR turns them off, so that they do not cause any unnecessary load on the CPU.

Cut, Copy & Paste

The context menu entry **Cut** removes the module from its current position. It is copied to the clipboard, where it can be inserted at another place, even in another window, using the **Paste** command.

Copy marks the module for copying. A copy of the module is stored in the clipboard from where it can be inserted at another place, even in another window, using the **Paste** command.

You can also use the key combinations **Ctrl (⌘) + X** (Cut), **Ctrl (⌘) + C** (Copy) and **Ctrl (⌘) + V** (Paste). When using the keyboard shortcut **Ctrl (⌘) + V** for pasting, you can specify a point in the structure by clicking there with the left mouse button first.

Delete

The context menu entry **Delete** removes the module and everything it contains. You can also delete a selected module(s) with the **Del** key on the computer keyboard.

Properties

A dialog window with information about the macro can be opened with the context menu entry **Properties**. For detailed information about all modules please see the Module Reference.

17.3. Sources

What are Sources?

In REAKTOR, **Source** is the name given to a module which outputs a control signal. There are three different kinds of sources:

- **Control-Sources** have a representation on the panel. The panel element is used to set the value of the control signal.
- **MIDI-Sources** convert MIDI data to control signals.
- **Constant-Sources** have a fixed value.

Control Sources

Fader, **Knob** and **Button** are examples of control sources. There are two ways to insert them into a structure:

- Choose the desired module from the context menu of the structure window (**Modules** ⇨ **Panel** ⇨ **Fader / Knob / Button**).
- In the context menu of a module in-port select **Create Control**. A control source is created and connected to the input. Type, label and settings of the control source are already configured to suit the input. In many cases you can save a lot of time by using this feature to add control sources.

Control sources and their respective panel elements can be controlled via MIDI in various ways. Please see the section *MIDI Control* on page 138 for details about this topic.

MIDI Sources

MIDI source modules are used for controlling the audio signal processing with MIDI events. For each type of MIDI event there exists a particular kind of source module. The output signal of such a source corresponds to the values transmitted by the particular MIDI events. The **OnVelocity** source for example outputs a control signal.

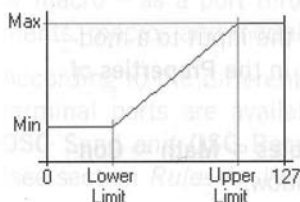
MIDI sources are created through the context menu of the structure window by choosing the desired kind of MIDI data in **Modules**

⇒ **MIDI In....**

Range of Values

For control sources and MIDI sources the range of the output control signal is scaled to the range between **Min** and **Max** (set in **Properties**) to achieve optimum control of the particular module parameter.

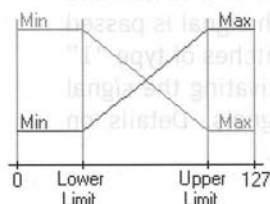
For MIDI sources the range can also be limited with the Properties **Lower Limit** and **Upper Limit**. The output value of the source is limited to **Min** for MIDI values below **Lower Limit** and to **Max** for MIDI values above **Upper Limit**. The range between the two limits is interpolated linearly between **Min** and **Max** as shown on the diagram:



Scaling and limiting

The values for **Lower Limit** and **Upper Limit** lie between 0 and 127 and the value for **Upper Limit** must be greater than that set for **Lower Limit**.

However, **Max** can be smaller than **Min** to achieve inverted operation. If opposite characteristics are set for two sources, a **crossfade** effect can be programmed.



Crossfade

A switch with adjustable threshold level can be emulated by setting **Lower Limit** and **Upper Limit** to neighboring MIDI values, e.g. 63 and 64. When the input value to such a source is below 64 **Min** is output, otherwise the output value is **Max**.

Stepsize

The range of values in source modules normally has a resolution of 128 steps. In many modules (particularly Fader and Knob) the parameter **Stepsize** can be used to reduce the resolution to fewer than 128 steps. You enter the step size by which the output value is to change, beginning at **Min**. For example you can set a pitch parameter to select only octaves by giving it a **Stepsize** value of 12.

Constant Sources

Constant sources are what you need to supply the input to a module with a fixed value. The desired value is set in the **Properties** of the **Constant** module with **Value**.

A constant source is created by selecting **Modules** ⇒ **Math** ⇒ **Constant** in the context menu of the structure window.

17.4. Switches

Switches are not sources because they do not generate any control signals. They are controls, however, because (like other control sources) they are represented on the panel by a control element.

Several modules can be connected to the inputs of a switch and the position of the switch then determines which signal is passed to the output of the switch. An exception are switches of type "1" which only toggle between activating and deactivating the signal path. They are simply on/off switches for signals. Details on switches can be found in the Module Reference.

The use of switches in a structure can also play a significant part in reducing the load on the processor. This is because modules or parts of the structure that are not connected to REAKTOR's audio outputs (or to an input of a Tapedeck module) do not add anything to the audio signal and are therefore automatically switched off. In this state they do not cause any CPU load. For example, you may use a switch to select one of several oscillators. Only the oscillator whose signal is being output will be active, while all the other oscillators are automatically deactivated.

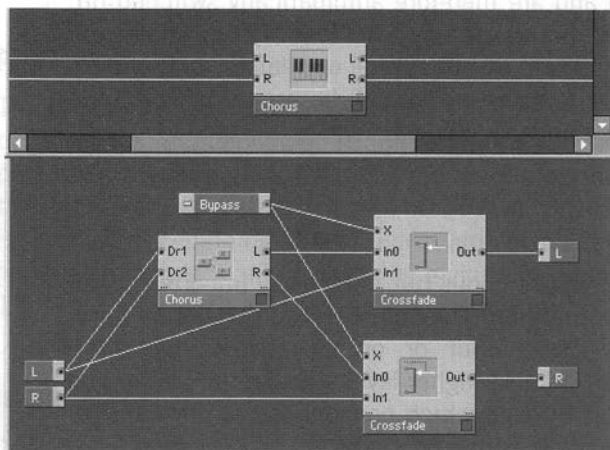
17.5. Terminals

Terminals are very inconspicuous but immensely important modules in REAKTOR structures. They are like the sockets on hardware instruments. Each input or output terminal within a structure appears at the next higher level – i.e. in the instrument or macro – as a port through which connections to other instruments, macros and modules can be made.

According to the different kinds of module ports, several types of terminal ports are available: **In Port**, **Out Port**, **Send**, **Receive**, **OSC Send** and **OSC Receive**. The normal rules for wiring apply (see section *Rules for Wiring* on page 125).

Terminals are created using the context menu of the structure window by selecting the desired kind of terminal under **Modules** ⇒ **Terminal** The **Label** of a terminal is initially just **In** or **Out**, but if you have several Ins or Outs you should give them meaningful names (like **L** and **R** in this picture) to prevent any possible confusion. You should also provide terminals with a description

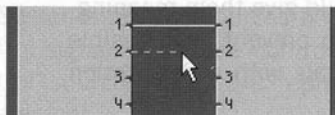
(info) in the **Properties**. The label appears as the port label in the representation at the next higher level (**Parent**), and the description is shown as the hint for the port when the mouse pointer rests on it (provided **Show Hints** is enabled).



Instrument with ports and its structure with terminals

17.6. Wires

The connection between the ports of two modules, shown as a line, is called a **wire**. Wires transport signals between the modules.



Connecting a wire

Creating

There are two ways to make a new wire:

- Click on one of the two ports to be connected with the left mouse button, move the mouse pointer to the other port while keeping the mouse button pressed and release the mouse button above the target port. A visible connection (that's the wire) appears between the ports and the effect on the sound resulting from the change to the structure can be heard immediately. The wiring operation is aborted if you click the left mouse button somewhere other than on a valid port.
- Click on the entry **Wire/Unwire** in the context menu of the port. Then move the mouse pointer to the other port and click the left mouse button on it. Again, the wiring operation is aborted if you click the left mouse button somewhere other than on a valid port, like an empty part of the window.

Deleting

There are two ways to delete a wire:

- Do the same as you would to create a new wire. When you repeat the wiring operation for an existing wire this removes the connection. (That's why the entry in the context menu is called **Wire/Unwire**)
- Select the wire you want to delete by clicking on it with the left mouse button or by opening a frame that covers the wire (selected wires are displayed colored). To delete it simply press the **Del** key on your computer keyboard.

Rules for Wiring

When wiring modules together there are some **general rules** to be observed:

- A wire can only connect an out-port to an in-port.
- An out-port can be connected to up to 40 in-ports.
- When no wire is connected to an in-port, it receives a zero signal.

In addition, the following **special rules** apply:

- An **event in-port** cannot process **audio** signals . If an event in-port is to be fed from an audio out-port, the signal must first be converted with an **A to E** module (see Module Reference).
- An **event in-port** can be fed from up to 40 **event out-ports**. If there is more than one connection, the event signals are merged so that it is always the last received event that determines the value.
- An **audio in-port** can only be fed from a single out-port.
- An **event out-port** can be connected to **audio in-ports** as well as **event in-ports**.
- When connecting a **mono** output to a **poly** input all voices receive the same signal. For pitch signals this means the voices in effect play in unison.
- A **poly** output cannot be connected to a **mono** input (a red cross appears on the in-port). A **Voice Combiner** module must be used for converting poly to mono.

Hints

While the mouse pointer rests on a wire (and if **Show Hints** is switched on), the value of the signal on the wire is shown as a hint.

For event signals, the value of the last event is shown (if the events come faster than the rate at which the display updates, some intermediate values may be missed).

For audio signals, a rough indication of minimum and maximum values, i.e. the range of the signal, is given. (Short peaks in the signal may be missed and thus do not show up in the display). If the range of the signal is changeable, you may need to move the mouse pointer away from the wire to close the hint, and then point on the wire once more to start measuring minimum and maximum values again.

For polyphonic signals, the values for all voices are displayed, one line of values for each voice. At the left, the MIDI note numbers which are playing on the respective voices are shown. If a voice is not playing any note, **Note: Off** is displayed along with the value of the signal on the wire. Voices with note off are always shown below the voices with note on.

17.7. Event Signals

An event has two properties: the time instance at which it occurs, and the value it carries, which is the new value when used as an audio signal.

Every event signal is also an audio signal, so it has a value for every sample. The difference is that this value is constant, until an event comes along to change the value. This means that every event output can also be used like an audio output, but the signal will be stepped, not smooth.

Some Modules (**A to E**, for example) only evaluate an audio signal connected to its input at the control rate.

Most Modules which operate on Events (**Add** used as event module, **Timer** or **Compare**, for example) work at the exact instance that an event arrives. The event timing is limited only by the audio sample rate, i.e. it's as accurate as anything. Other event modules (**A to E** or **LFO**, for example) operate only at the lower timing resolution determined by the Control Rate, say 200 times a second.

Order of Event-Processing

Most event processing modules, in response to an input event, generate an output event immediately. That is, an event travels through the chain of event modules to the end (possibly fanning out if there are several paths) before the next event travels the chain.

The method is "depth before breadth: an event propagates as deep as it can along one track before another wire fanning out from the same port is considered.

If one event is to go down more than one branch, and you need to have the branches executed in a defined order, you should use the **Order** module to fan out to the different paths.

Another important module in this context is the **Event Value** module. A complex event processing structure can be connected to its **Val** input, but it will only pass on this value as an event when a triggering event arrives at the **In** input. You can look at this as a Sample&Hold circuit triggered by an event. You can use the **Order** module to generate this triggering event and make sure that it occurs after other event processing has completed.

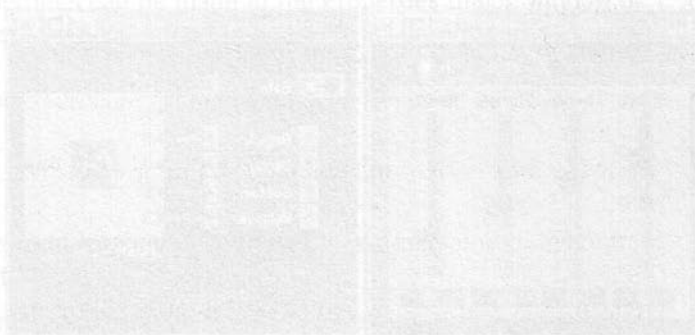
When different source modules produce events at the same moment in time, for example when they are initialised as the structure is turned on, they actually send events in the order in which the source modules were originally inserted into the structure. To have one module initialised after the others, simply cut it and paste it back into the structure.

17.8. Context Menu

The context menu of the structure window has nine entries:

- **List of Modules** is for inserting elementary modules into the structure.
- **Macros** is for inserting macros from the library into the structure.
- **Instrument** is for inserting instruments from the library into the structure.
- **Paste** inserts a previously cut or copied object into the structure at the point where the context menu was opened. When using the keyboard shortcut **Ctrl + V** for pasting, you can specify a point in the structure by clicking there with the left mouse button first.
- **Select All** selects all the objects in the structure.
- **Save As...** is for saving the structure to a file with a new name. Depending on the type of structure (instrument or macro) the correct filename extension will be appended.

- **Panel** opens the panel window which belongs to the structure. For macros, the panel command opens the panel window of the instrument or ensemble of which the macro is a part.
- **Parent** opens the structure of the hierarchical level above. For example, if you are in the structure of a macro which is part of an instrument, the parent command will open the instrument's structure.
- **Properties** opens a dialog window with settings and a description of the instrument or macro to which the current structure belongs. See the section *Instrument Properties* on page 100 for details about the Properties of instruments.



18 Panel Editing

18.1. What is a Panel?

A **Panel** is the user interface of an instrument. It corresponds to the front panel of a hardware synthesizer or effects unit, where the knobs and switches for operating the device are located. Instrument panels are displayed in the Ensemble window.

18.2. What are Controls?

All the control sources and switches that are part of a structure also appear in the panel window in the form of **controls**. Depending on the type of source module, they are represented as **faders**, **knobs**, **buttons**, **switches** etc.. Each is used for setting the output signal of the corresponding source module or (in the case of switches) for changing the signal flow.



On the left a panel with faders, on the right the structure with the corresponding control source modules

18.3. Panel Controls

Fader

Faders are linear sliding controllers in the panel, whose position determines the control value of the corresponding source module. The range of output values of the fader is set with the values **Min** and **Max** in the Module Properties. The resolution can be set with the property **Step** (when **Step** is zero there are 128 steps).

In addition to moving a Fader with the mouse, it can also be remote controlled with MIDI (see section *MIDI Control* on page 138).

A fader can be changed to a knob by selecting **Knob** under **Panel Appearance** in the Properties. With **Show Value** and **Show Label** the display fields for value and label in the panel representation can be enabled or disabled. The fader can also be displayed at half size by selecting **Small Design**.

All these settings can be made in the **Properties** dialog window of the fader control in the **Panel** window or in the **Properties** dialog window of the corresponding control source module in the **Structure** window.

Knob

Knobs work just like faders, only that they appear as rotary controllers in the panel.

A knob can be changed to a fader by selecting **Fader** under **Panel Appearance** in the Properties. With **Show Value** and **Show Label** the display fields for value and label in the panel representation can be enabled or disabled. The knob can also be displayed at half size by selecting **Small Design**.

Button

Buttons are switching controllers in the panel, whose position determines the control value of the corresponding source module. The output values of the button in the states **On** and **Off** are set with **On Value** and **Off Value** in the Module Properties.

Besides operating the Button with the mouse, it can also be remote controlled with MIDI (see section *MIDI Control* on page 138).

With **Show Value** and **Show Label** the display fields for value and label in the panel representation can be enabled or disabled. The button can also be displayed at half size by selecting **Small Design**.

All these settings can be made in the **Properties** of the button control in the **Panel** window or in the **Properties** of the corresponding control source module in the **Structure** window.

Switch

Switches are used to activate and deactivate different signal paths by choosing one of the switch inputs to connect to the output.

The **Labels** of the switch module's in-ports in the structure window are also used as labels for the buttons that make up the switch in the panel window. You should definitely give them meaningful names, otherwise you may quickly forget what it was that the particular switch turns on or off.

With **Show Label** the display field for the label in the panel representation can be enabled or disabled.

Context menu

The panel controls have the following context menu when you right click on it (Mac: Ctrl + Click):

- **MIDI Learn:** With this entry you enable the MIDI Learn function for the control.

- **Show in Structure:** Choose this entry in order to open the structure window in which the module corresponding to the panel control is located.
- **"Module name" Properties:** This entry opens the Properties for the control.

18.4. Connection properties of Panel Controls

Most panel controls have a Connection page (MIDI-connector icon) with the following sections and settings:

Receive External MIDI

Activate MIDI In: when activated, the panel control can be changed by incoming MIDI events. You can choose between MIDI Controller and Polyphonic Aftertouch messages, and you can specify the controller or Aftertouch-note number.

MIDI

Soft Takeover: when enabled, the control will not be affected until the incoming value passes the control's current value (either going up or going down). That prevents sudden jumps in the control value when the position of the software controller does not match that of the hardware controller, which can happen, for example, after an onscreen change or a snapshot recall.

Incremental: If active, incoming MIDI messages will be interpreted as coming from an incremental controller. Incremental controllers (often called endless rotaries or continuous controllers) are found on many MIDI control surfaces, including Native Instruments 4Control.

Panel to MIDI: If enabled, REAKTOR will send MIDI events whenever the panel control is changed with the mouse.

Remote to MIDI: if enabled, causes MIDI events to be output by REAKTOR when the control is changed by incoming MIDI events (cf. Activate MIDI In). When connecting to a sequencer, bear in mind that a feedback loop may result if the sequencer both sends MIDI data to REAKTOR and receives MIDI from it.

Controller, Poly Aftertouch and MIDI Note: determine whether the panel control receives and/or sends MIDI Controller, Polyphonic Aftertouch (Key Pressure) or MIDI note messages.

Cont/Note: sets the number of the MIDI controller or note that is assigned to the panel control.

Connection

The Connections section on the Connection page is available for **Fader, Knob, Button, Switch, XY, Lamp, Meter, Multi Picture** and **Multi Text** modules. Note that it is also available for all MIDI In and MIDI Out modules, thereby allowing wireless communication between different Instruments and Macros. This section controls internal, wireless communication of data within REAKTOR as well as the enabling of OSC connections between REAKTOR applications running on different computers linked by OSC. Two operations are required to create an internal connection:

- Select the panel control you want to use as control master and press the topmost button to the right of the Connections list.
- Select the panel control you want to use as control slave and press the middle button to the right of the Connections list.

You can make those selections in any order and one master can control several slaves, in which case the Connection list for the master becomes longer.

To make OSC connections use the two drop-down menus labeled **OSC Source** and **OSC Target**. The **OSC Source** drop-down menu displays controls on other REAKTOR computers, from which values have already been received over OSC. (If REAKTOR has not received any OSC data, the drop-down list will be empty.)

The **OSC Target** drop-down menu shows other OSC REAKTOR computers. This is the same list as in the **OSC Settings...** window of the REAKTOR **System** menu. Use the **OSC Target** drop-down menu to tell the OSC REAKTOR target computer to place this control in its **OSC Source** drop-down menu.

Any existing internal or OSC connection for a module will be listed as an entry in the Connections list. If the module is master of a connection, the entry will have the prefix "to", whereas if it is the slave, it will have the prefix "from". To delete an entry, select it and click the **Delete** button (trash can icon) to the right of the Connections list.

The two settings at the bottom of the Connection page determine whether the panel control should appear as a selectable parameter in the plug-in hosts parameter automation list (**Disable Automation**) and what position it should have in the list (**ID**). If you enter a number in the ID field that is used by another control in your ensemble, the ID will be swapped with the other control.

Make sure that the number entered in the **Max Automation ID** field in the Instrument properties is high enough to ensure that this parameter can be shown in the plug-in host's parameter automation list.

Two-dimension panel controls like **XY** and **Multi Picture** actually have two automation IDs. The second ID will automatically be set to one greater than the first ID and can not be edited. That ensures that these two parameters appear consecutively in your host software's parameter automation list.

18.5. Editing the Panels

Just like the modules in a structure, the controls in the panel can be edited with **Duplicate** and **Delete**. However, remember that these operations always have a direct effect on the respective structure. For example, if you delete a control from the panel you are also deleting the corresponding source module in the structure, because the two are inseparable. We that you carry out these operations only in the structure so you can keep control of the outcome.

Moving controls in the panel, on the other hand, has no effect on the structure. Simply click the left mouse button on the label of the control you want to move and drag it with held mouse button to the desired position. Once all controls have been arranged it is best to activate the **Panel Lock** function. When **Panel Lock** is enabled it is no longer possible to move and panel elements. The **Panel Lock** function is set using the context menu of the panel window or by clicking on the wrench icon in the Instrument Header so that the icon is replaced with a screw icon.

19 Panel Operation

19.1. Mouse Control

Fader

To change the value, click the left mouse button on the fader and, keeping the button pressed, move the mouse up or down until the desired position is reached.

Knob

To change the value, click the left mouse button on the knob and, keeping the button pressed, move the mouse up or down until the desired position is reached.

Button

There is a choice of three operating modes for buttons. Each mode can be chosen in the Properties dialog window:

- **Trigger:** Pressing the button generates an event with the **On Value**. Releasing the button does not generate an event.
- **Gate:** Pressing the button generates an event with the **On Value**. Releasing the button generates an event with the **Off Value**.
- **Toggle:** The button has two stable states. Pressing it once switches it on; an event with the **On Value** is generated. Pressing it again switches it off; an event with the **Off Value** is generated.

When connecting the button to an audio in-port, **Trigger** mode behaves the same as **Gate** mode, i.e. the signal returns to the **Off Value** when the button is released.

Switch

Pressing one of the buttons on a switch selects the corresponding input of the switch and connects it to the output. Only one of the buttons of a switch can be active at a time, and therefore only one of the inputs can be enabled at any one time.

19.2. Key Control

The currently selected fader, knob or switch can also be controlled with keys on the computer keyboard.

The position of **Faders** and **Knobs** changes like this:

Key: Value Change:

- \uparrow +Step
- \downarrow -Step
- **PgUp**+10 × Step
- **PgDn**-10 × Step

A **Switch** changes the active input on pressing the keys \uparrow/\downarrow .

19.3. MIDI Control

MIDI Data Types

If **Activate MIDI In** is activated in the **Properties** dialog window of a control element it can be remote controlled via MIDI. One of the following kinds of MIDI data can be used:

- MIDI Controller messages: The number of the MIDI controller assigned to the panel element is set with **Controller/Note No** in the Properties or using the **MIDI Learn** function.
- MIDI Poly-Aftertouch messages: The MIDI note number of the message is set with **Controller/Note No** in the Properties or using the **MIDI Learn** function.

Faders and **Knobs** move their position to follow the received MIDI data.

When operating **Buttons** by remote control, the Button turns on when the received MIDI Controller or Poly Aftertouch message has a value greater than 63. With Buttons it is also possible to use Note On/Off messages to control the button.

When operating **Switches** by remote control with MIDI Controller or Poly Aftertouch messages, the Switch changes to a position corresponding to the received value. For this the range of possible values (0 to 127) is divided into regions of equal size according to the number of inputs on the switch (e.g. with 4 inputs: 0 ... 31, 32 ... 63, 64 ... 95, 96 ... 127). The value 0 always selects the input at the bottom, 127 selects the input at the top.

MIDI Learn

The **MIDI Learn** function is switched on with the button on the Toolbar. It is identified by an icon representing a MIDI socket and the letter **L**. It is a very efficient tool for assigning MIDI messages to control elements on the panel.

Select the control element which is to be remote controlled, click on the **Learn** button and send the MIDI data that you want to use for controlling (by moving the hardware wheel, knob, fader, pedal or other controller). To MIDI-fy other controls just repeat this operation.

REAKTOR automatically detects whether the controller data comes from a standard MIDI controller or from an incremental controller. The panel element's **Incremental** mode switch is set accordingly. In the rare case that the **MIDI Learn** function chooses the wrong mode, simply repeat the operation or set the correct mode in the Properties of the panel element manually.

Incremental

You need to activate this entry in the **Properties** of controls or MIDI source modules if you want to control it using a MIDI Controller with endless knobs sending incremental values.

Soft Takeover

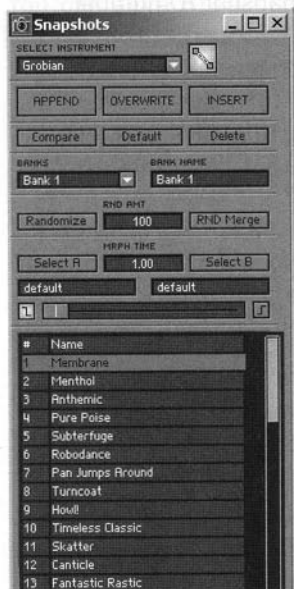
When a fader or knob is being operated by MIDI remote, it normally jumps straight to the received controller value. Such a jump can be quite noticeable in the sound, depending on the controlled parameter (e.g. amplifier level) and is often not desirable. Such jumps can be avoided by selecting **Soft Takeover** in the Properties of the relevant controls. The control will only move when the value received via MIDI reaches or goes past the current position.

19.4. MIDI Out

When **Panel to MIDI-Out** is activated in the Properties of a panel element, any movements on the panel will be translated to MIDI data which are output by REAKTOR. The same kind of MIDI events as are received by the controller (**Activate MIDI In**) are used to output the movements.

A second option **Remote to MIDI-Out** determines whether the events that are received over MIDI (**Activate MIDI In**) for the particular control element are also sent to the output. Take care, though, that when connecting to a sequencer which sends MIDI data to REAKTOR and at the same time receives MIDI from it. In this case, a feedback loop may result.

20 Snapshots



The Snapshots window

What are Snapshots?

Snapshots are REAKTOR's sound settings and correspond to the memories of conventional synthesizers, called "programs" or "patches". A snapshot records the current setting of all the instrument's panel control elements and MIDI controllers. By recalling a snapshot all the settings are restored to the saved state. Each instrument can store 16 banks of 128 snapshots each.

Snapshot IDs

Every REAKTOR panel control has a unique Snapshot ID number, which is visible and changeable in the control's Properties. Snapshot files associate values with these ID numbers. If you load a Snapshot into an Instrument that is different than the one from which it was saved, REAKTOR will set the Instrument's controls based on the IDs. That can produce unusual (but often interesting) results.

Recalling

Snapshots are recalled using the Snapshot drop-down menu at the top of the Instrument's control panel. There's also a Snapshot drop-down menu for Ensemble Snapshots at the top of the Snapshot's control panel. The snapshot that was last recalled is always shown in the header. You can also recall snapshots by moving around the list with the up and down cursor keys.

Snapshots can be recalled using MIDI Program Change messages. Make sure that **Snapshot Recall by MIDI** is activated in the instrument's **Properties**, otherwise MIDI program change messages have no effect. The number preceding the snapshot label, which indicates the snapshot's position in memory, is also the number of the MIDI Program Change that recalls the snapshot.

You can link snapshots in different layers of the hierarchy. When recalling a snapshot in the ensemble, for example, the instruments in the ensemble will also change snapshot if they have **Snapshot Recall by Parent** activated in their respective Properties. The snapshot that is recalled for the instrument reverts is the one that was active at the time the ensemble's snapshot was saved.

Keep in mind that you can also use REAKTOR's Snapshot module (see the Module Reference section) to automate Snapshot recall, store, randomize, and morph operations.

Storing

You can store Snapshots in a separate Snapshot window which you open with a click on the camera button on the left side of the Snapshot drop down list.

Below the Instrument selection menu in the Snapshot window you'll find six buttons for managing Snapshots: **Append**, **Overwrite**, **Insert**, **Compare**, **Default**, and **Delete**.

The **Append** button saves the current settings as a Snapshot at the end of the Snapshot list. If the current Snapshot bank is full, REAKTOR will search for the next bank with some empty space and append the Snapshot there. If necessary, a new bank will be created. If there are no free Snapshot slots in the Instrument, nothing will be done.

The **Overwrite** button replaces the currently selected Snapshot with the current settings.

The **Insert** button, inserts the current settings after the selected Snapshot and pushes all other Snapshots down one position. That operation continues through subsequent banks as necessary.

The **Compare** button is used to compare the current control panel settings with those stored in a special compare buffer. (See the section on Comparing, below.)

The **Default** button replaces the selected Snapshot with the default settings for each control as specified in the control Properties.

The **Delete** button deletes the selected Snapshot(s) and moves all subsequent Snapshots up in the list.

When you store a Snapshot, its entry in the Snapshot menu becomes selected and you can name it by immediately typing in the desired name. The default name is the name of the last selected Snapshot followed by a number. The number is incremented by subsequent store operations.

MIDI program change messages recall Snapshots in the order they appear in the Snapshot list. For example, program change 0 recalls the first Snapshot in the list, program change 1 recalls the second, and so on. You can rename any existing Snapshot by selecting it in the menu and typing in a new name.

Copying and renaming

To rename a Snapshot, simply resave it using the **Overwrite** button and type in a new name. To copy a Snapshot, resave it using either the **Append** or **Insert** button.

Comparing

The **Compare** button in the Snapshot window is used for comparing two groups of panel settings - the current panel settings and the settings stored in REAKTOR's compare buffer. When you press the **Compare** button, the current settings are swapped with the settings in the compare buffer. You can use either of the two groups as the basis for more editing, because after clicking the **Compare** button, changing any setting causes the current settings to be stored in the compare buffer. Also, when you recall a Snapshot the current settings are first stored in the compare buffer. That allows you to compare the current settings with any Snapshot as well as to recover your settings if you accidentally recall a Snapshot. If that's a bit confusing, remember these four things:

- To save the current settings in the Compare buffer, click the Compare Button twice, then continue editing.
- After recalling a Snapshot, you can return to the previous settings by clicking the Compare Buttons once.
- To compare two Snapshots, recall the first then recall the second. The Compare button will now switch between them.
- After using the Compare button, changing any control panel element will cause the settings before the change to be put into the compare buffer.

Snap Isolate

When recalling a snapshot, the controls normally jump to the position stored in the snapshot. If for some reason you want to prevent this for certain panel elements or MIDI controllers, you can do so by activating **Snap Isolate** in the **Properties** dialog window. The relevant controls are then immune to snapshot recall.

Bank Menu

The Bank menu has two parts. The upper part is for selecting the active bank and it shows all banks that have so far been initialized. The lower part is for managing banks. To activate an existing bank, select it from the upper part of the menu. The Snapshot menu will change to show the Snapshots in that bank.

You can rename any bank using the Bank Name window to the right of the Bank menu. Simply double-click in the window and type in a new name.

The lower part of the Bank menu has five options: New, Sort, Init, Clone, Save, Load, and Delete. New creates a new bank at the end of the bank list. You can have up to 16 banks. Sort sorts the Snapshots by number and deletes any empty Snapshots in the process. Clone duplicates the bank in a new bank at the end of the bank list. Delete deletes the selected bank.

Save exports a Snapshot file of the bank to your hard drive in REAKTOR's *.ssf format. Load loads a Snapshot file in REAKTOR's *.ssf format from your hard drive. You will be given the option to overwrite or append to the current Bank. Loading and saving are handy when you want to create more than 16 banks of presets, say in different categories. If you load Snapshots into a different Instrument or Ensemble than created them, they will not work as intended (See the section on Snapshot ID).

Randomizing

REAKTOR offers very sophisticated randomizing options. The Randomize button will randomize all controls in a Snapshot except those that are isolated (see the Random Isolate section). The Random Amount field controls how much of the range of the control is allowed in randomization. 100% allows the full range of the control whereas 0% results in no randomization. The randomization takes place "around" the control's current setting.

The Random Merge (RND Merge) works in conjunction with the A and B Snapshot selections used for the morphing function, which is described below. It creates a "child" preset by selecting a value for each control that is between the setting for that control in the A and B Snapshot. If Random Amount is on 0% when you use this function, the new value is located exactly between the values of the selected snapshots. If it is on 100% the complete intersection is used.

Random Isolate

If for some reason you want to prevent a specific control from being randomized when the Randomize button is pressed, you can do so by activating Random Isolate in the Properties dialog window. The relevant controls are then immune to randomization.

Snapshot Morphing

REAKTOR will morph all control settings between the values in two Snapshots over a period of time of up to 60 seconds. Set the morphing time in seconds in the MRPH Time window. Click the Select A button to select the first Snapshot and click the Select B button to select the second Snapshot. You can use the slider below to achieve any position between the two Snapshots.

If there is a higher value entered in the MRPH Time field, the hearable and visible morph can lag in relation to the position of the morph slider. If you enter 60 seconds as morph time for example and move the slider very fast to the opposite side, the morph will last 60 seconds anyhow.

The two buttons at the left and right side of the morph slider determine, of which snapshot the position of all buttons and switches is used, since buttons and switches will not be affected by the morph. If the left button is pressed, the button state of Snapshot A is used, if the right is pressed, the state of Snapshot B is used.

As Instrument designer bear in mind that you can use the Snapshot Module (see Module Reference section) to morph between Snapshots from within REAKTOR Instruments.


21 Sampling and Re-Synthesis

Since REAKTOR is a modular system, it does not require a fixed structure for organizing samples.

21.1. Sample Management

Sound Files and Samples

REAKTOR takes its samples, i.e. the audio material to be processed in the Sampler modules, from WAV or AIFF formatted mono or stereo sound files, which can be at any sample rate. Besides the sound waveforms that are contained in the sound file, REAKTOR also reads any loop and keyboard allocation information (if available) from the sound file.

Before REAKTOR can use a sound file it needs to be loaded into the computer's RAM. Regardless of the data format (bit depth) of the audio material in the sound file REAKTOR uses 32-bit format for the internal representation of the sample. This is done for reasons of efficiency. One minute of stereo sound in CD quality uses 20 MB RAM. If virtual RAM is being used (the default in Windows, but not in  MacOS), then a great deal more main memory can be allocated than is actually available. For this reason error messages are not displayed during the loading of large sound files even when the free RAM is already exhausted.

Instead you will get an error message later: REAKTOR will inform you that the CPU is overloaded and processing will stop. This error message is displayed because the software cannot access the sample data on the hard drive fast enough (after first having tried to access it in RAM and not having found it there). Frequently the message is preceded by an unintentional granular sound effect caused by the more or less regular interruptions of the audio data stream. This situation, particularly undesirable during live performances, can only be avoided through the addition of more main memory.

Multiple Use of Identical Samples

If one particular sample is used by several sampler modules in an ensemble, all the modules will access the same representation of the sound file that is stored in RAM. This means you can load the same sample in as many sampler modules as you like without increasing the demand on RAM by any appreciable amount. The number of voices used in a sampler module is irrelevant as far as memory requirements are concerned.

REAKTOR identifies a sample using the path of the sound file from which the sample was loaded. The path comprises the directory in which the sound file is located and the sound file's name. When a sound file is loaded, REAKTOR searches through the list of samples that are already loaded, looking for a sample that has the same path and name. If this sample is already present, it will simply be "reused".

Missing Samples

REAKTOR saves only the paths of the samples contained in a macro / instrument / ensemble. If the sound files were deleted, renamed or moved after the macro / instrument / ensemble was saved, it is likely that the instruments cannot be entirely recreated.

In this situation you will get an error message after having opened the macro / instrument / ensemble. The "missing" samples are labeled in the Sample Map Editor's list view as **File missing, and data missing**. Using the **Replace** button in the Sample Map Editor, the File-Open dialog window is activated. This function allows you to localize or replace missing sound files.

Storing samples in the Module

The loss of samples can be avoided if the **Store Map with Ensemble** option in the sample module's properties dialog is activated, which can be accessed via the sampler module's Properties dialog window. Sampler modules using this option will save a copy of the

sample file with the module file. In most cases the increase in the file size of macro / instrument / ensemble files will be considerable, but your work can be reconstructed regardless of the circumstances.

Whenever a module containing embedded samples is loaded it will not try to load the samples from hard disk but will load the embedded samples directly.

Sample-Analysis

Some modules (**Sample Resynth**, **Sample Pitch Former**, **Beatloop**) perform real-time re-synthesis. For this purpose, after loading, the sample will be analyzed. This process takes approximately as long as the duration of the sample. To avoid repeating analysis REAKTOR tries to save the analysis data with the sound file. This of course is only possible if the sound file is not write-protected. Also keep in mind that sound files from CD-ROMs should be copied to hard disk. If a sound file has been changed it is possible to re-analyze it.

Sample-Editors

REAKTOR does not have its own sample editor, since there are already many sample editors available on the market. In the preferences menu, the edit button can be set to launch your preferred sample editor (**System ⇨ Preferences ⇨ Directories: External Sample Editor**).

Naturally sound files changed in the sample editor have to be **saved** before they can be re-loaded into REAKTOR. Use **Reload** in the Sample Map Editor and in the context menu of the sample module.

Note: Keep in mind that some sample editors will ignore loop information present in sound files. In this case the loop regions will be lost.

21.2. Sample-Maps

One of the ways of using sample maps is to better simulate acoustic instruments, usually this can be achieved by using multiple samples over a keyboard range. Stretching a sample over fewer keys will improve the naturalness of an acoustic sound. Another use of sample-maps is to trigger multiple samples with one key. Following are explanations for both uses:

Multi-Sampling

The repetitive nature of samples makes it difficult to simulate acoustic instruments. With REAKTOR it is possible to vary the sample each time it is triggered. Usually when a sample is transposed over multiple keys, the further it is transposed from the root key of the original sample the less natural it sounds. This change occurs because the spectral aspects of transposed samples do not change according to the natural changes occurring in acoustic instruments. Samplers try to overcome this limitation by using multi-sampling techniques, when each sample is only transposed a small amount from its root key.

Assuming a sample-map contains multiple samples of the original sound at least three parameters have to be set to control each sample in the map. You can set these parameters grafically in the grafical map window, or numerical in the list editor.

- The **Root Key** sets pitch of the untransposed sample.
- **L** (left split) sets the starting point of the key zone.
- **R** (right split) sets the end point of the key zone.

The optimum result is achieved when the Root Key is kept in the middle of the Split zone and the transposition is kept to a minimum.

Every sampler module has a **P**(itch) input, which is used to select a sample from the map and to control the tuning of the sample. Usually this input is connected to a **Note Pitch** module, which sends the current MIDI note. When a sample is triggered in multi-sample mode, the value received at the **P**-input selects the appropriate sample and tuning.

The sample module also has a **Sel**(ect) input. If connected it overrides the sample selection of the **P** input, which then only sets the tuning. This can result in interesting sound variations.

Sample maps in REAKTOR can also be used to trigger multiple samples with only one key. This is a technique used to further mirror an instrument's dynamics. Velocity is used to achieve this via MIDI. Usually an instrument is sampled playing the same note with different dynamics. Then the dynamically different samples can be triggered using different velocities, resulting in a more expressive representation of the original sound.

Drum-Maps

As in Multi-Sampling, the "Drum-Map" uses the **Root Key** and left and right splits to determine the position on the samples in the map. Sometimes in drum sounds the **Root Key** of the original sample is not used or gets overly transposed. With Drum-Maps, it is possible to use any area of the key range – therefore not having the actual **Root Key** in the range.

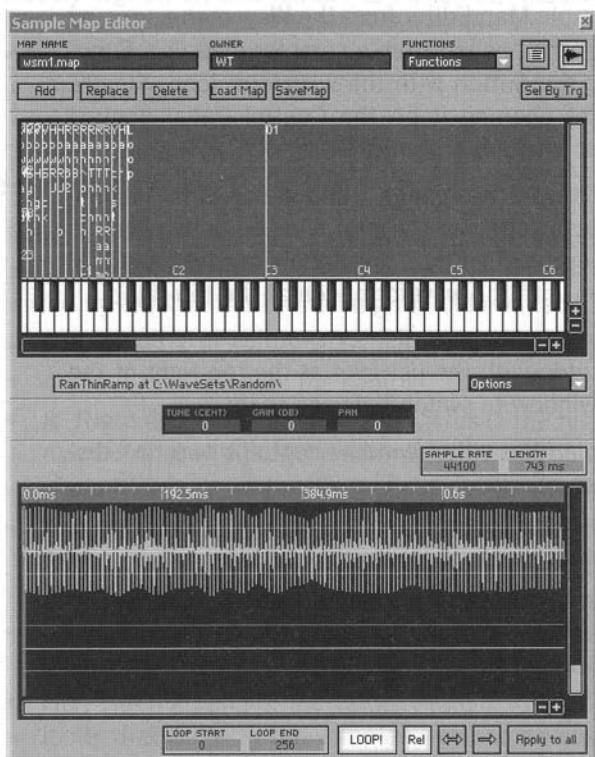
As described above the **Sel**-input is used to override the sample selection of the **P**-input, which then only sets the tuning. The **Sel**-input can be used to achieve interesting sound variations. For example by connecting the output of a **Gate** module to the **Sel** input, the velocity information is then used to select samples from the map. This way it is possible to trigger many different samples using only one key's velocity information.

Saving Maps

You can save a Map to disk, separate from the Instrument or Ensemble that uses it. Under Windows it will be stored with the file-name extension **.map**. The map file can contain all the sample data that is used by the map, or it can be just a small file with references to the sample files.

The loss of samples can be avoided if the **Store Sample with Ensemble** option is activated, which can be accessed via the sampler module's properties window.

21.3. Sampler Map Editor



Sample Map Editor

REAKTOR's **Sample Map Editor** is for loading, saving, keyboard mapping, and looping audio files used in REAKTOR's various sampler modules. Like the Properties window, the Sample Map Editor can be open at all times and will change to reflect the contents of the currently selected sampler module. If no sampler module is selected, the Sample Map Editor will be empty.

Window access and organisation

You can open the Sample Map Editor from the **View** menu or using the **F7** key. It can also be opened from a sampler module's **Function** properties using the button with the waveform icon. Finally, there is an option for opening it on the Control Panel context menu for any sampler module.

The Sample Map Editor has two panes. The top one is for loading and arranging samples in the sample map and the bottom one is for editing sample loop points. The loop window can be folded up by clicking the button with the waveform icon at the top-right.

The sample mapping window can display samples in graphic or list form. Use the button with the list icon at the top-right of the Sample Map Editor window to switch between them.

The top of the Sample Map editor window contains two text displays and a drop-down menu:

- **Map Name:** The name of the sample map file on disk. This may be edited.
- **Owner:** The name of the sampler Module whose map is displayed.
- **Functions:** Remap to single key and Set transpose to zero (see below for details).

Sample access buttons

There are six buttons below the text displays for loading and saving samples and sample maps:

- **Add:** Adds an individual sample to the sample map.
- **Replace:** Replaces the selected sample in the sample map.
- **Delete:** Deletes the selected sample in the sample map. This does not affect the sample on your hard disk.
- **Load Map:** Loads a sample map file.
- **Save Map:** Saves the current sample map file for use in other sampler modules.
- **Import:** Opens the Akai Import window for importing sample maps from Akai CD-ROMs (S 1000-3000 format).

To the far right of the sample access buttons is a button labeled **Set By Trg**. When this feature is turned on, incoming MIDI notes will cause the sample selection to change accordingly. Note that this option is only available if the sampler module has been wired correctly.

The sample list display

In list mode, the sample map display (the upper pane in the Sample Map editor window) contains nine columns of data. It can be sorted by any column by clicking the column label.

- **L:** Left end of the sample zone. This is the lowest MIDI note number that will play the sample. This can be edited.
- **R:** Right end of the sample zone. This is the highest MIDI note number that will play the sample. This can be edited.
- **Name:** The name of the sample. This can't be edited.
- **Trp:** The sample transpose value. This can be edited and the Root value is automatically adjusted (see Set Transpose to Zero).
- **LowV:** The low end of the velocity range. This is the lowest velocity that will play the sample. This can be edited.
- **HiV:** The high end of the velocity range. This is the highest velocity that will play the sample. This can be edited.
- **Root:** This is the MIDI note number that will play the sample at its original pitch. This can't be edited in the list display.
- **Status:** The status of the sample. Choices are: **data present:File exists**, **data present:File missing**, and **data missing:File missing**. This can't be edited.
- **Location:** The location on your hard drive where the file is located. This can't be edited.

The sample graphic display

In graphic mode, the sample map display (the upper pane in the Sample Map editor window) contains two dimensional regions representing each sample in the map and a piano-keyboard graphic along the bottom. The position as well as the horizontal

and vertical size determine which MIDI note events cause the sample to be played. The vertical position determines the velocity range and the horizontal position determines the pitch range. You can move the regions around with the mouse when it shows a four-direction cursor and you can move any edge when the mouse shows a two-direction cursor.

You can select multiple regions for editing, in which case all cursor activity will apply to relatively all region.

Regions can be made to overlap, but this only is possible for easier positioning a region. It is not possible to playback two samples at the same time. Therefore overlaps should always be avoided.

Remap to single key

Remap to single key causes each sample in the map (regardless of which samples are selected) to be mapped to one key starting at the leftmost key used in the current sample map. The order of samples in the original map is preserved.

Set transpose to zero

Set transpose to zero resets the transpose amount of each sample to zero and changes the root key accordingly. (The root key is always the left-most key minus the transpose value.) Setting transpose to zero has the effect of making the leftmost key in each region play the sample at its natural pitch. You can best see the effect of using it in the List view.

Name, prelisten, and options

Below the sample map display is a text box for displaying the name and location of the selected sample, a button to its left for playing the sample once through. If you click again on this button before the sample playback has ended, the playback will be interrupted. There is an options drop-down menu to its right with the following entries:

- When the **Show Sample Names** option is active, the names of the samples are displayed in the graphic sample map display.
- When **Ignore Root Key when Loading** is active, the root key information saved in the sample file header is ignored.
- When **Single Key Mode** is active (meaningful in list display mode only), changing the left-split note will automatically change the right-split note to the same value. This is an editing convenience.

Tune, gain, pan, sample rate, and length

These date fields change the tuning in cents, the gain in decibels, and the pan position for the selected sample.

The **Sample Rate** and **Length** displays show the sampling rate for the selected sample and its length in milliseconds. These values can not be edited.

Loop editor

The **Loop editor** (bottom window) allows you to set and edit loop points for individual samples. The brown area in the sample waveform display indicates the current loop area. You change the loop endpoints by dragging on the ends of the loop area display. You can also move the whole loop in either direction by dragging in the middle of the display. The loop start and end points are displayed numerically in samples below the graphic display, but they can not be edited numerically.

The five buttons below the loop display control the looping properties for the selected sample:

LOOP!: Turns looping on and off.

Rel: Turns on loop-in-release mode. When loop-in-release is on, samples continue looping even after the gate signal has been turned off.

<->: Alternating Loop. This causes the sample loop to reverse at each end. The loop plays forward then backward then forward, and so on. That often produces a smoother, less noticeable loop.

->: Backward playback. This causes the entire sample to play backward rather than forward.

Apply to All: Causes the current loop settings to apply to all samples in the map.

21.4. Akai Import

Akai import is possible from the sample modules context menus and the Sample Map Editor. Select the entry **Akai Import** to open an import window which shows the content of an Akai formatted CD-ROM inserted into your CD drive. If the content of the CD doesn't appear in the browser window press the **Reload CD** button.

The browser window displays all partitions, volumes, programs and samples, lets you explore the content of the Akai-CD and allows converting selected Akai programs and samples to REAKTOR Map files (file extension *.map). Converted files will be stored into the folder you have specified as **Imported Files (Akai)** path in the REAKTOR preferences.

When loading or converting to a Map file, REAKTOR will keep the following sample information:

- Sample data
- Loop points
- Root key
- Pan

The following information will not be kept at this time:

- Filter settings
- Envelope settings
- Velocity splits
- Gain

It is also possible to load Akai samples and programs directly into a sample module by pressing the button **Load Prg as Map**. In this case you can save the samples as Map file using the **Save Map** button in the Sample Map Editor or save them with the ensemble by ticking the checkbox **Store Map with Ensemble** in the properties.

Ticking the checkbox **Mute Audioengine during conversion** causes a faster import, but REAKTOR will stop any audio output while loading.



22.1 Properties

The Event Table and Audio Table modules have an extensive set of properties which are identical for both kinds of module. Some of the general properties which are available in many other modules are not covered again here.

Function page

- **Interpolation:** See how slider set to one and played samples newly clip. you select set to data with profile and read. The set set to play values. Only the integer part of values arriving at the RX and RY inputs are used. The result is a stepped output signal even when the input position changes smoothly. The display looks stepped like a bar chart in the ID-modes.
- **X:** Interpolation between values is only used on the X-axis. The full precision of fractional values at the RX input is used and the compute smooth transitions when reading between table cells.
- **Y:** Interpolation between values is only used on the Y-axis. The full precision of fractional values at the RY input is used to compute smooth transitions when reading between table cells.

The with a file of results in more than one step and the Save button. The New button creates a new, empty table. The Audio Table and Event Table modules can also be loaded into the main workspace.

22 Table Modules

The Table modules allow handling events and audio data in a very flexible way. The Table modules can be used to design Oscillators, LFOs or Waveshapers by drawing your own waveforms with the mouse. Or you can crossfade between wavetables and create envelopes with curve shapes drawn by hand or with countless breakpoints. The Event Table can be used as a sequencer for outputting gate and pitch values or for controlling any parameter in a REAKTOR ensemble.

22.1. Properties

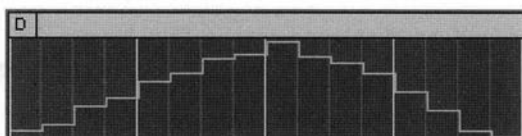
The Event Table and Audio Table modules have an extensive Properties dialog which is identical for both kinds of module. Some general properties which are available in many other modules are not covered again here.

Function page

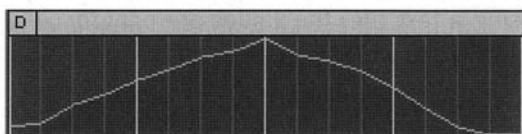
Interpolation

- **None:** No interpolation happens when reading between cell values. Only the integer part of values arriving at the **RX** and **RY** inputs are used. The result is a stepped output signal even when the input position changes smoothly. The display looks stepped like a bar chart in the 1D-modes.
- **X:** Interpolation between values is only used on the X-axis. The full precision of fractional values at the **RX** input is used to compute smooth transitions when reading between table cells.
- **Y:** Interpolation between values is only used on the Y-axis. The full precision of fractional values at the **RY** input is used to compute smooth transitions when reading between table cells.

- **XY:** Interpolation between values is used in both the X and Y-axis. The full precision of fractional values at the **RX** and **RY** inputs is used to compute smooth transitions when reading between table cells.



No Interpolation



X-Interpolation

When X interpolation is active, the table diagram shows the interpolation by displaying a smooth curve. Interpolation is never displayed in 2D-mode.

Clip/Wrap XY

- **Clip:** When reading beyond the end of the table you get the value of the last cell. Reading before the start of the table you get the value of the first cell.
- **Wrap:** When reading beyond the end or start of the table it continues at the other end of the table as if it was arranged as a circular loop.

Backup Data with Module

Activate this option to save the data in the table within the ensemble, instrument or macro file.

File

The data in the table can be read from or stored to a file with the **Load** and **Save** buttons. The **New** button creates a new, empty table. The Audio Table and Event Table modules can read the following file formats:

- Table Files (*.ntf)
- Audio Samples (*.wav or *.aif)
- Plain Text (*.txt) containing numbers separated by spaces (Text files are treated as one row of data, so Y-Size is always 1.)

The name of a loaded file will be displayed in the File Name field.

It is possible to use a text editor for creating a Table file. Just enter values for the X axis in a row with spaces between the values. Save the file with the file extension *.txt. It is not possible to create values for the Y-axis using a text file, the values are only for Y=0.

You can save the data from a table in a file for reusing it in other Table modules. If the same file is loaded into more than one Table module in the same ensemble, the data in this file will be shared between these modules. Modifying the table content in one module affects all other modules. If all modules display the content of the same table cells, any modification of the values is visible in realtime in the panel graphs of all table modules.

The **Clients** field shows the number of Table modules in the ensemble which share the same Table file.

X Size/Y Size

With the **Set**-Button you can define the size of the table storage. The first field is for the number of cells on the X-axis (the width of the rows), the second for the number of cells on the Y-axis (height of the columns). Changes to the numbers of cells will not become valid until pressing the **Apply** button.

Note: If you reduce the number of cells, and the cells being removed contain data, then this data will be deleted together with the removed cells.

Value

Min, Max, Stepsize and **Num Steps** work just like with Knobs and Faders.

The **Default** value is used to initialize cells when creating or enlarging a table or when cutting a selection from it. The **Default** value is also important in the display because it appears as a distinct color (normally black). **Default** is commonly set to 0.

Display Units

When editing the table in Draw mode, the current value is displayed in the graph's status bar in a format according to the **Display Units** setting.

- **Numeric:** Standard format for numbers in any range.
- **MIDI Note:** The value is rounded to the nearest integer and displayed as the equivalent MIDI Note number. For example, 60 is C3 and 58 is A#2.
- **% (percent):** The range 0...1 is displayed as 0...100%. For example, 0.5 becomes 50% and 2 becomes 200%.

X Units

Sets the units which are used to measure the horizontal cell position in the table. The following units are available:

- **Index:** This is the default unit. The cells are numbered with integer values (0, 1, 2 ... n).
- **[0... 1]:** The first cell has the position 0, the last the position 1. The position of the cells inbetween are computed by their relative location in the table as a fractional value between 0 and 1.
- **milliseconds:** The position of a cell will be computed as time in milliseconds (ms), depending on the sample rate entered in the **Samples/Sec** field below. This unit is especially interesting in the Audio Table module for moving inside an audio sample which was recorded in real time.
- **Tempo Ticks:** The position of a cell will be computed in ticks of a tempo clock. This option is especially useful in the Audio Table module with a rhythmic piece of audio loaded of which you know the BPM (Beats per Minute) tempo. The **Ticks/Beat** field defines how many ticks make up a beat (usually 24).

When **X Units** is set to **milliseconds** you can adjust the sample rate of the data:

- **Samples/Sec:** How many cells correspond to one second of time. When loading a Wav or Aiff file its sample rate automatically appears here.

When **X Units** is set to **Tempo Ticks** you can also adjust the following values:

- **Samples/Tick:** How many cells in a subdivision of a beat.
- **BPM:** Tempo in Beats per Minute.
- **Ticks/Beat:** The subdivision of a beat. For REAKTOR's Master Clock this is 24.
- **Number of Beats:** The length of the data measured in beats (usually 4 or 8 for a smoothly looped audio beat).

When changing one of these values, the others are automatically recomputed to match the length of the data and sample rate.

Y Units

Sets the units which are used to measure the vertical cell position in the table. The following units are available:

- **Index:** This is the default unit. The rows are numbered with integer values (0, 1, 2 ... n).
- **[0... 1]:** The first row has the position 0, the last the position 1. The position of the rows inbetween are computed by their relative location in the table as a fractional value between 0 and 1.

Appearance page

Visible in View A and B

The settings in this area are global module settings and apply always to both panel views, A and B.

- **Picture:** Tick this option to see the table display in the panel.

- **H(orizontal) Scroll Bar:** Tick this option to see a scrollbar under the table graph.
- **V(ertical) Scroll Bar:** Tick this option to see a scrollbar to the right side of the table graph.

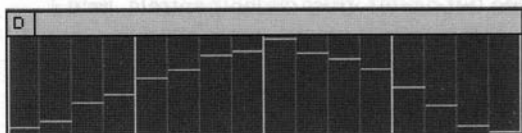
Size X/Size Y

- Enter here the size of the Table display within the Panel. The display size is entered in pixel.

Graph Format

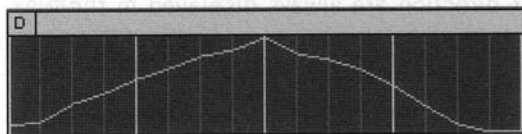
This dropdown list selects how the values are displayed in the Table module's panel graph. You can choose between four modes:

- **Pixel:** Values are drawn with a horizontal line. If you set no interpolation for the X axis the lines look like small faders.



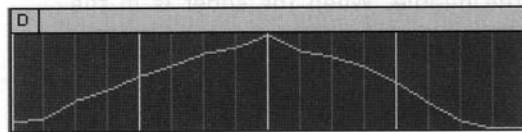
Pixel

- **Line:** Values are drawn with a horizontal line and vertical lines are also drawn to connect the values.



Line

- **Bar:** Values are drawn with a horizontal line, vertical lines are drawn to connect and the area underneath is filled with color.



Bar

- **2D Color:** This mode is for viewing more than one row at a time. This is what you need to see all of a 2-dimensional table ($Y\text{-Size} > 1$). The value of each cell is displayed as the color of the corresponding rectangle. The rows are numbered from top to bottom, the columns always from left to right.
- **2D Curve:** Similar to the setting **2D Color**, but in contrast to this you can edit multiple rows at the same time by drawing their curve shapes. While the **2D Color** setting offers the bird's eye view on the Table data, **2D Curve** allows the front view.
- **Solid:** Like the setting **Bar**, but without the outline.

View Parameters

- **X Auto Fit:** When this option is ticked all cells in the X direction are always displayed in the graph. The number of displayed cells is the same as the **X Size** in the **Table** tab.
- **X Alignment:** When **Auto Fit** is off this slider controls how smaller regions of the data are selected. When the slider is in the left position the cell selected with the **XO** input appears at the left edge of the display. With the slider centered the selected cell appears in the middle. When the slider is in the right position the last cell visible at the right edge of the display is the one before **XO**.
- **Y Auto Fit:** When this option is ticked and 2D mode is selected, all cells in the Y direction are always displayed in the graph. The number of displayed cells is the same as the **Y Size** in the **Table** tab. In display modes other than 2D, **Auto Fit** has no effect.
- **Y Alignment:** When **Auto Fit** is off this slider controls how smaller regions of the data are selected. When the slider is in the left position the cell selected with the **YO** input appears at the top edge of the display. With the slider centered the selected cell appears in the middle. When the slider is in the right position the last cell visible at the bottom edge of the display is the one before **YO**.
- **Value Auto Fit:** When this option is ticked the display's value range in Pixel, Line and Bar mode is the same as the range set with **Min** and **Max** on the Table tab. In 2D mode **Auto Fit** has no effect.

Grid

The settings are the same for X, Y and Value Grid:

- **Enable Grid:** When these boxes are ticked vertical lines (**X Grid**) or horizontal lines (**Y Grid** in 2D display mode or **Value Grid** in the Pixel, Line or Bar display mode) will be displayed in the panel graph.
- **Grid Step:** The value entered here relates to units set for X and Y on the **Table** tab. It sets the spacing of the grid. If you want to set the basic resolution of the grid to one unit enter 1 here. Enter 2 to have every second unit as a grid step, or 0.5 for two grid steps per unit.
- **Size 1 ... Size 4:** There are four different sizes available for the grid lines: 1 is the finest and 4 is the thickest line size. The number you enter in these boxes defines how often a line with a certain thickness is drawn (number of Grid Steps per line). Enter 1 for a line at every Grid Step, or 2 for a line at every other Grid Step, for example. Enter 0 if you do not want to use a size at all.

Visible

The panel display options in this area can be set for panel A or B only, or for both panels depending on what panel button is activated.

- **Label:** Tick this checkbox for displaying the module name in the panel.
- **Visible:** The graphic area is displayed on the panel when this option is enabled. It is used for viewing and editing the data.
- **Value:** The status bar at the top of the Table graph is visible when this option is selected. It shows information relating to the mouse position in the graph, such as the X and Y position and the value in the table at the position. The units of the displayed values depend on the selected units in the Properties under the **Table** tab.
- **Small Label/Value:** Tick this checkbox for setting a smaller display size for the module label and value status bar in the panel.

22.2. Context Menu

When you click the right mouse button (Mac: **⌘**+mouse button) on a table graph, a special context menu for editing in the graph becomes visible. Once you get more familiar with the editing options we recommend using keyboard shortcuts which are available for most of the operations since you can navigate and operate much faster like this. Keep in mind that the shortcuts only apply to the data in the graphs when the **Panel Lock** mode for the Instrument is active.

Draw/Select/Control Mode

- **Table Draw Mode:** This mode allows you to enter values with the mouse. You can draw curves or modify single values by moving the mouse up and down. In 2D-mode you set the value with which you draw with the menu option **Set 2D Draw Value...** or you pick an existing value with Ctrl-mouse click on a cell in the graph.
- **Table Select Mode:** In Select mode, instead of drawing with the mouse to change the values, you use the mouse to select a region for modifying later. The selected data can then be worked on with the editing functions.
- **Table Control Mode:** In this mode Table data can not be change in any way. You can neither enter new data nor can you modify existing data via the panel.

File

- **Load Data into Table, Save Table Data:** These menu options are shortcuts for the **Load** and **Save** buttons in the **File** section of the **Table** tab in the Properties. Please see page 161 for more details on Table files.
- **Save Table Data as...:** Save a Table file under a new name.
- **Reload Table Data:** If you have modified a table file outside REAKTOR you can load the new version with this menu entry.

Show

- **Show All:** Zooms out fully so that the whole table is visible in the graph.
- **Show Selection:** Zooms the display so that the current selection completely fills the display.
- **Next Y (Page Down):** In Pixel, Line and Bar mode the next higher row of table cells is displayed. In 2D mode the graph is scrolled vertically one row up.
- **Previous Y (Page Up):** In Pixel, Line and Bar mode the next lower row of table cells is displayed. In 2D mode the graph is scrolled vertically one row down.

Graph

- **Pixel Mode**
- **Line Mode**
- **Bar Mode**
- **2D Color Mode**
- **2D Curve Mode**
- **Solid Mode**

It is possible to change the display mode of the table graph directly via the context menu without overwriting the Properties settings. Read more about the four graph display modes above in the description of the **Appearance** tab in the Properties.

View

- **Show Read Position:** When this option is selected, a vertical line at the current read position is displayed.
- **Show Write Position:** When this option is selected, a vertical line at the current write position is displayed.
- **Show Horizontal Position Line:** When this option is selected, a position ruler above the Table graph is displayed.
- **Show Horizontal Scroll Bar:** Select this option to see a scrollbar under the table graph.

- **Show Vertical Scroll Bar:** Select this option to see a scrollbar to the right side of the table graph.

Select

- **Select All (Ctrl+A):** Selects all currently visible data.
- **Select X All:** Selects all currently visible data in the selected rows.
- **Select Y All:** Selects all currently visible data in the selected columns.
- **Snap Selection to Grid:** With this option enabled, the edges of any selection are always adjusted to lie on the Grid, which may be much coarser than the cell size, depending on the settings in the Properties on the **Grid** tab. When the Grid's smallest line size is not visible because the display is zoomed out to display a lot of data, then the selection snaps to the smallest Grid size that is still visible.

Process

- **Mirror X:** Swaps the data between left and right using a vertical symmetry axis in the middle of the selection.
- **Mirror Y:** In 2D mode, swaps the data between top and bottom using a horizontal symmetry axis in the middle of the selection.
- **raubt eine numerische Eingabe zur Anwendung verschiedener Rechenoperationen auf ausgewählte Daten.**
- **Rotate/Add/Scale...:** Allows numerical entry for applying several mathematical operations on selected data. **Add Value...:** Allows numerical entry for adding or subtracting from the values of selected data. **Rotation:** Rotates the selection by the given amount. Cells which are moved out of the selected area on one side reappear on the other side in a circular motion. **Scale Value...:** Allows numerical entry for scaling the values of selected data. (1=100%, 0.5=50%, 2=200% ...).

- **Trim Selection:** When you apply this function to a selection of Table cells, all table cells which are not located within the selection will be cropped. The number of X and Y cells in the Table will be updated after trimming.
- **Delete Rows:** All rows which are located within a selection will be deleted. The number of Y cells in the Table will be updated after applying this command.
- **Insert Rows:** The number of rows you have defined with a selection will be added in the Table. The number of Y cells in the Table will be updated after applying this command.
- **Quantize Value to Step Size:** When this option is selected, values drawn with the mouse snap to the step size set in the table module Properties. This is the normal behaviour. Unselect this option to draw at a finer resolution despite the set step size.
- **Set 2D Draw Value...:** You can enter a value which is used when you draw values in 2D mode. You can also pick a value from the View by pressing Ctrl+mouse button in draw mode.

Cut, Copy, Paste

- **Copy (CTRL+C):** Copies the selected data into the copy buffer.
- **Cut (CTRL+X):** Cuts the selected data and puts it in the copy buffer.
- **Paste (CTRL+V):** Pastes data from the copy buffer into the current selection.

22.3. Advanced Operation

Draw / Select Mode

The current editing mode is displayed in a square at the top left in the status bar as a **D** or **S** to indicate Draw or Select Mode. You can toggle the mode by clicking on the square.

The Tab key also toggles the editing mode.

Rotate

By holding the Shift key and dragging with the mouse you can move all cells in the selected area left and right, or in all directions in 2D-mode. When everything is selected, you can drag around the whole graph.

Add

By holding the Ctrl key and dragging the mouse up or down you modify the value of everything in the selected area. An amount is added to or subtracted from all the values, depending on how far you drag the mouse.

Panel Lock Mode

There is a connection between the **Panel Lock** mode of the Instrument and the Table modules. When the **Panel Lock** mode is activated, panel operations with keyboard shortcuts like copy (Ctrl+C) and paste (CTRL+V) apply to the selected data inside the table graph instead of the module itself. So when the panel is locked you are moving the data in the module instead of moving the module's representation on the panel.

23 "Classic Modular" Macro Collection

The Classic Modular macro collection is a collection of high-level Reaktor building blocks which simulate classic analog modular systems. In addition to impressive sound quality and flexibility that rivals the originals, the Classic Modular macros have a number of advantages over their analog counterparts. The Classic Modular macros include sampling, granular sampling, and sophisticated sequencers, for instance, in addition to a generous selection of oscillators, filters, and modulators.

Several instruments in the Reaktor Library were created out of the Classic Modular macros: The Green Matrix synthesizer, the Blue Matrix sequenced synth, the Analogic Filter Box were built in parts out of elementary Classic Modular macros. Take some time to listen to those instruments in order to get a feel for what the Classic Modular collection can do.

These macros have two main properties:

1. Standardized signal range from -1 to 1 for all in and outputs. That means that all output signals can be connected to any input without down or up scaling. This is the main difference to the modules and other macros of REAKTOR in which the value range can vary depending on the unit of the parameter like semitones, decibel (dB) or milliseconds. For instance the pitch (P) input of an oscillator module expects values in the range 0 to 127, the amplitude (A) input expects values between 0 and 1.

Even the pitch (P) signals in this collection are in the range 0 to 1 to follow this standard. To convert these signals to the range 0 to 127 please use the "0-1 to 0-127 Range Converter" macro in the "Event Processing" folder.

The only exception is the position (Pos) signal of the sequencer macros. Since its integer value addresses steps in a sequence it doesn't make sense to limit its range to 0-1.

2. For each important parameter of a macro there is a "manual" control (usually a knob) to control this parameter directly and one or two modulation amount controls. For each modulation amount control there is a corresponding input with a similar name to connect modulation sources like LFOs and envelopes to. The in-

coming modulation signals are scaled by the modulation amount controls and are added to the value of the manual control. If more modulation inputs are needed, it's possible to connect a modulation mixer to the modulation input. The corresponding modulation amount control should be turned to maximum since the actual modulation amount is set with the mixer.

There are event and audio type modulation inputs. Most modulation sources like LFOs and envelopes offer both types. If other audio signals should be connected to event inputs they have to be converted to event signals first with an "A to E" module (can be found in the "Auxiliary" folder of REAKTOR's modules).

Some file names contain "Event" or "Audio". These macros have the purpose to scale, mix, invert and switch between modulation signals like LFOs and envelopes and are available in event and audio versions. Some names contain "Stereo" or "Mono". These macros have the purpose to mix, amplify, filter or distort 'normal' audio signals (in contrast to modulation signals) so they are available in stereo and mono versions.

The "Classic Modular" macros can be used to build polyphonic structures. That is one big advantage to analog modular systems. These were normally monophonic. Since the output of an instrument is monophonic, polyphonic signals within the instrument have to be converted to monophonic signals with a voice combiner module (can be found in the "Auxiliary" folder of REAKTOR's modules). Please note that all display modules like meters, number displays, oscilloscopes, lamps etc in the "Classic Modular" macros are connected only to the last played voice!

23.1. Display

Number Display

Shows the current value of the input signal. If it changes too fast, it is recommended to turn on the built in peak detector with the "Peak" switch. Then the overall amplitude envelope is displayed.

Simple Scope

Displays the curve of the incoming signal just like an analog oscilloscope.

XY Scope

Oscilloscope in XY mode. The incoming "X" signal determines the horizontal position of the displayed dot. The incoming "Y" signal the vertical position.

23.2. MIDI

Controller

Receives the following MIDI controller messages and passes them on to its outputs: pitchbend, modulation wheel, aftertouch, volume.

Notes - Monophonic

Receives MIDI note messages and passes their pitch, gate and note on velocity values to the corresponding outputs in monophonic manner, means that only one note is present at a time. This note is sent to all voices simultaneously. The key range can be set on the panel and is independent of the key range set in the properties of the instrument, which is the key range of the "Notes - Polyphonic" macro. It is possible to have several "Notes - Monophonic" macros with different key ranges.

Notes - Polyphonic

Receives MIDI note messages and passes their pitch, gate and note on velocity values to the corresponding outputs in polyphonic manner, means that the notes are assigned to different voices. The key range can be set in the properties of the instrument. All "Notes - Polyphonic" macros share this key range.

Selective Gates - MIDI Keyboard

The gate signal of 12 continuous keys starting with the one defined by the "Lower" knob are received and passed to the 12 corresponding outputs of the macro in a monophonic manner. The gate signals are sent to all voices of the instrument simultaneously. The selected key range is independent of the key range set in the properties of the instrument.

Useful applications are to trigger different envelopes with each key or to start/stop sequencers with pressing/releasing a certain key, while the other keys outside this key range are used to play notes.

Selective Gates - QWERTY - Lower Keys

If an instrument is selected the keys from the computer keyboard are sending MIDI notes to the instrument like an external MIDI keyboard is doing. This feature is called "QWERTY", just like the first 6 keys on the English keyboard. This macro receives the notes of the lower key range starting with the "Z" key and ends with the "M" key and passing only their gate signals to the corresponding output of the macro in a monophonic manner. Which means that the gate signals are sent to all voices of the instrument simultaneously. There is an output for each key. The key range of the QWERTY keys is independent of the key range set in the properties of the instrument.

Useful applications are to trigger different envelopes with each key or to start/stop sequencers with pressing/releasing a certain key, while the other keys outside this key range are used to play notes.

Selective Gates - QWERTY - Upper Keys

The same as the "Selective Gates - QWERTY - Lower Keys" macro but the key range starts with the "Q" key and ends with the "P" key.

23.3. Mixer/Amp

About amplifiers:

Signal generators like oscillators and samplers already have built-in amplifiers (A input). If other signals have to be amplified or attenuated the amplifier macros should be used.

To amplify normal, non-modulation audio signals there are two possible ways depending on the used modulation signal. Because of the exponential way the ear perceives amplitude changes, either the modulation signal changing the amplitude or the amplifier itself should have exponential characteristics. For example the ADSR envelope has exponential curves for the decay and release phase so it can be connected to linear amplifiers like the built in ones in oscillators and samplers. The modulation signals from the modulation sequencer have a linear characteristic and should be connected to an exponential amplifier.

To amplify modulation signals, it's best to use linear amplifiers.

Amp - Exponential

Amplifier with exponential characteristic. The amplitude is set in decibels (dB).

Amp - Linear

Amplifier with linear characteristic. The amplitude can be set in the range from 0 to 1.

Crossfade

Crossfading module. The output signal is mixed from the both input signals.

Inverter

Inverts the polarity of the incoming signal. Two modes are available: In the unipolar (Uni) mode the signal is mirrored round 0.5. This is the right mode to invert non-velocity sensitive gate signals. In bipolar (Bi) mode the signal is multiplied by minus 1. This is the right choice for inverting LFOs and other bipolar modulation sources.

Master Volume

Master level control with built-in limiter to prevent the signal from clipping. Should be the last macro in the instrument. The input signals must be monophonic. Please insert voice combiners (can be found in Modules/Auxiliary) to convert polyphonic signals into monophonic ones.

Mixer - Simple

Simple 4-channel mixer for audio signals, which aren't used for modulation. To mix modulation signals, please use the Modulation Mixer.

Mixer - Studio

8-channel mixer with one post fader effect send, one pre fader effect send, panning, and an on/off button for each channel. The input signals must be monophonic. Please insert voice combiners (can be found in Modules/Auxiliary) to convert polyphonic signals into monophonic ones.

Modulation Mixer

3-channel mixer for modulation signals with switches for inverting the incoming signal. If a higher resolution at small values of the knobs is needed, the characteristic of the knobs can be switched from linear to exponential with the "exp" button.

Since the actual modulation amount is set with this mixer, the modulation amount control of the macro that the mixer is connected to should be set to maximum.

Modulation Matrix - Mixer

8x8 mixer matrix for modulation signals. The columns correspond to the inputs, the rows correspond to the outputs of the matrix. Each row defines a mix of the modulation signals present at the columns. The mix is passed to the corresponding output. If a higher resolution at small values of the knobs is needed, the characteristic of the knobs can be switched from linear to exponential with the "exp" buttons.

Since the actual modulation amount is set with this matrix mixer the modulation amount control of the macro the mixer is connected to should be set to maximum.

Modulation Matrix - Switch

An 8x8 switch matrix for audio modulation signals. The rows correspond to the inputs, the columns correspond to the outputs of the matrix. One modulation signal out of eight can be selected. This signal is passed to the corresponding output without scaling.

Panner

The input signal can be positioned between the two outputs.

Scanner

The 8 inputs are scanned in dependence of the scan position. If a scan position is set between two inputs the output signal is obtained by crossfading between these two.

23.4. Oscillator

Geiger - Counter

Generates impulses or pulses at random intervals, much like a Geiger counter radiation particle detector. The average rate and randomness of the impulses/pulses can be set.

Noise

Noise generator offering four different noise types:

White noise: All frequencies have the same amplitude.

Pink noise: High frequencies are damped with 3 dB per octave. A filter sweep with a bandpass filter results in a signal with a constant amplitude.

Coloured (filtered) noise: The colour can be set with the "Colour"-knob.

808: Noise source used in the legendary TR-808 drum machine to synthesize the hihats and cymbals. Consists of 6 detuned pulse oscillators.

Oscillator - Symmetry

Oscillator with symmetry/pulse width modulation for all available waveforms: bipolar ramp pulse (similar to sawtooth), bipolar pulse, normal pulse, triangle/sawtooth and parabolic (similar to sine wave). Please note that the two symmetry/pulse width modulation inputs have different characteristics: The first has a linear characteristic, which sounds good for LFOs. The second has an exponential characteristic, which sounds good for envelopes.

Oscillator - Sync

Versatile oscillator with hard and soft synchronization, phase modulation and frequency modulation. Available waveforms are sawtooth, pulse, triangle, sine and impulse. On hard synchronization the oscillator restarts at the set phase. On soft synchronization the oscillator plays back its waveform in reverse direction. This results in a more gentle "sync" sound.

Random

Random level sample + hold generator. Random numbers are generated in the set rate and held until the next number is generated. If the "Rmp" switch is turned on connecting ramps are generated between successive numbers.

23.5. Sampler

Samplers play back samples, which are included in the sample map. A sample map editor can be found on the "gearwheel" page of the samplers' properties. The properties can be opened with a double click on the sample display. The "Select" knob of the samplers selects a sample of the sample map.

For playing back beat loop samples the "Classic Sampler" or the "Beat Loop" sampler should be used. The advantage of the "Beat loop" sampler is that the tempo and the pitch of the playback can be set independently. If this isn't wanted the "Classic Sampler" should be used. Like all main parameters the loop length of these samplers can be modulated. To assure that the loop length is always multiples of a suitable musical length like 1/4 notes, one bar etc the loop length can be quantized to the step size set with the "LL Q" knob. A step size of zero turns off the quantization. Similar quantization features are available for the sample start position, loop start position and position offset (only for the "Beat Loop" Sampler). Please note that for the "Classic Sampler" the loop playback has to be turned on in the sampler's properties while the "Beat Loop" sampler is always in loop mode.

Beat Loop

Sampler specialized in playing back beat loop samples.

Synchronises any beat-loop sample, regardless of the original tempo, to a clock source that is connected to the "Clk" input of the macro. The playback pitch of the sample is independent of the tempo. The sample is played back in a permanent loop.

Beside the sample selection and playback pitch following parameters can be set and modulated: start position, loop start position, loop length and position offset. The unit of the corresponding controls is set with the "Unit" knob in 16th notes.

The macro has two position outputs to drive sequencers. The position events at the "Pos" output are generated at the beginning of the 16th notes and should be used to drive sequencers, which do not modulate the parameters of the "Beat Loop" sampler itself. The position events at the "Pos*" output are generated at the beginning of each grain, which is a 32nd note before the next 16th note. Since all of the parameters of the "Beat Loop" macro except the start position are sampled and held in that moment, this output should be connected to sequencers which modulate one of these parameters.

Classic Sampler

The "Classic Sampler" plays back samples the "old fashioned" way, linking playback pitch and speed. The Classic Sampler features frequency modulation and the possibility to reverse the playback direction with modulation signals. In the sample-map editor, the loop feature can be activated for each sample individually.

The start position, loop start position and loop length are set in 1/128 of the sample length. That means that if the sample is one bar long then multiples of 8 would address 16th notes positions.

Resynth

Sample resynthesizer with independent control over pitch and playback speed, which is specialized in playing back samples without beats. The resynth sampler cuts the sample in small pieces called "grains". During playback these grains are played one after the other. The "granularity (Granu)" knob determines the numbers of grains in a certain time. To reduce glitches the grains can overlap each other. The fade time between two overlapping grains can be set with the "Smooth" knob.

The start position, loop start position and loop length are set in 1/128 of the sample length. That means that if the sample is one bar long then multiples of 8 would address 16th notes positions.

23.6. Sequencer

This folder contains a selection of macros to build powerful sequencers with. There are three classes of macros: clock generators, clock modifiers and the actual sequencers, which are driven, by the clock. The clock generator creates events in a certain rate. With each event the value is incremented by 1. This value corresponds to a certain position within a sequence stored in a sequencer. That is why these events are called "position events".

The clock modifiers manipulate the stream of position events and can be inserted between the clock and the sequencers.

Global Clock

Sends out the position events generated by the global clock. This clock can be started/stopped by the corresponding buttons in the toolbar of Reaktor. The time resolution of the events can be set on the panel. Another output signal is the clock gate. It is zero when the clock is stopped and one when the clock is running, i.e. after Start or Continue. Some sequencers need this signal to get initialised, the note sequencer needs it to prevent note hanger on sequencer stop.

Position Delay

Position event delay, which can be modulated to achieve a shuffled time resolution.

Position Looper

Loops the incoming position events. Unlike the built in loop function of the sequencer macros, the loop start and length can be modulated by a modulation signal. Also a different mode is implemented called "Freerun". In this mode the looper only jumps back to the loop start if it reaches the loop end. In the "Hardsync" mode the looper folds every incoming position event in the loop range so the looper would jump into the new loop immediately once the loop start is shifted by X steps. In "Freerun" mode it would take X steps to get into the new loop.

Position Offset

Adds an offset to the incoming position events to shift the read out position within a sequence.

Sequencer - 1x Notes, 4x Mod, 8x Trigger

A combination of a note sequencer, a 4 track modulation sequencer and a 8 track trigger sequencer sharing the same sequence number and global controls like the loop bar, edit bar, view bar and the global functions like copy, paste, clear and record start/stop. More information on the sequencers can be found in the explanations of the "Sequencer - Note", "Sequencer - Modulation 4x" and "Sequencer - Trigger 8x" macro. Note that the Blue Matrix synth in the Reaktor Library is driven by this sequencer. Please see the Blue Matrix documentation for a detailed overview of the sequencer in action.

Sequencer - Classic Step

A classic step sequencer with 16 steps. Unlike the other sequencers in this collection, the classic step sequencer is built with faders, with one fader for each step. This sequencer is the one to choose if the values of the steps should be controlled remotely via a midi fader box or other midi controllers.

Sequencer - Modulation 4x

Sequencer for playing back 4 parallel modulation signals. These signals can be used to modulate any synthesis parameters, such as oscillators and filters, just as LFOs and envelopes do.

A click on the "View" button toggles between the "all" and "solo" view. In "all" view all channels are visible at once. In "solo" view only one channel is visible, displayed with a higher vertical resolution. The "Select" bar (the second vertical bar from the left) selects the channel in "solo" view.

The "Seq" knob selects one of 128 different sequences so each snapshot can have its unique sequence number. The length of a sequence is 768 steps which equals 8 bars in a 96th notes resolution or 48 bars in a 16th note resolution. The numbers of sequences and the length can be set in the properties of the sequence display. The variables can be found on the "gearwheel" page of the properties. The "X" variable is the length and "Y" is the number of sequences. Since the last 8 (4×2) sequences are used as copy/paste and undo buffers the number of sequences must be at least 12 (4×3).

Please note that the sequencer doesn't have an edit buffer. All changes are stored immediately. If you want to create different variations of a sequence, you have to make a copy of the sequence using the copy and paste buttons before editing, or else the original sequence will be altered.

On the "Eye" page of the properties the time grid of the sequencer can be adjusted to the time base of the incoming position events ("Pos"-In) from the connected clock. If they have a 96th notes resolution the "Grid step" should be set to 6, if they have a 16th notes resolution it should be set to 1. The time grid will be visible in the sequencer, depending on the grid step value.

There are three horizontal bars: the edit bar, the loop bar and the view bar. To alter the size of these bars, their left or right ends can be clicked and dragged. Clicking and dragging in the middle functions like a scroll bar. A click beside the bars will cause them to scroll one length to the left or to the right. The edit bar determines the sequence range the following functions can be applied on: copy/paste/cut/insert, randomization (Rand), quantization (Quant), ramp, clear, and recording (Rec). The loop bar determines the sequence range, which is looped during play-back. The view bar determines the visible range of the sequence. A quantization can be set for both the edit and loop bars with the "Bar /" control.

Information on the functions can be found in the mouse-over hints of the function buttons. Please note that they are applied to all visible channels.

The modulation signals connected to the "Mod" inputs of the macro can be recorded. The "recE" buttons of the channels has to be turned on to enable the recording. The recording starts when the "Rec 1/0" is pressed. Please note that the recording is bound to the range defined by the horizontal edit bar. If the "1 shot" button is on, the actual recording will start once the first modulation event is received. Recording will stop after the locator has passed once through the edit range.

Sequencer - Note

The Note Sequencer is used to sequence notes with a standard piano-roll style editor. The sequencer consists of two data fields. The upper is for the actual notes, displaying the notes in piano-roll style: The horizontal direction represents the time, the vertical the pitch of the notes. A note starts when the pitch graph ex-

ceeds an adjustable threshold and ends when it falls below the threshold. The threshold is set with the second vertical bar of the upper data field. The lower data field is for generating re-trigger events in the notes set in the upper data field and for defining the velocity of the notes.

The "Seq" knob selects one of 128 different sequences so each snapshot can have its unique sequence number. The length of a sequence is 768 steps which equals 8 bars in a 96th notes resolution or 48 bars in a 16th note resolution. The numbers of sequences and the length can be set in the properties of the data fields and must be the same for both. The variables can be found on the "gearwheel" page of the properties. The "X" variable is the length and "Y" is the number of sequences. Since the last two sequences are used as copy/paste and undo buffers the number of sequences must be at least 3.

Please note that the sequencer doesn't have an edit buffer. All changes are stored immediately. If you want to create different variations of a sequence, you have to make a copy of the sequence using the copy and paste buttons before editing, or else the original sequence will be altered.

On the "Eye" page of the properties the time grid of the sequencer can be adjusted to the time base of the incoming position events ("Pos"-In) from the connected clock. If they have a 96th notes resolution the "Grid step" should be set to 6, if they have a 16th notes resolution it should be set to 1. The time grid will be visible in the sequencer, depending on the grid step value.

There are three horizontal bars: the edit bar, the loop bar and the view bar. To alter the size of these bars, their left or right ends can be clicked and dragged. Clicking and dragging in the middle functions like a scroll bar. A click beside the bars will cause them to scroll one length to the left or to the right. The edit bar determines the sequence range the following functions can be applied on: copy/paste/cut/insert, randomization (Rand), quantization (Quant), clear and recording (Rec). The loop bar deter-

mines the sequence range, which is scanned and looped during playback. The view bar determines the visible range of the sequence. A quantization can be set for both the edit and loop bars with the "Bar /" control.

Information on the functions can be found in the mouse-over hints of the function buttons.

The pitch and gate signals connected to the "P" and "G" inputs of the macro can be recorded. To do so the "recE" button has to be turned on to enable the recording. The recording starts when "Rec 1/0" is pressed. Please note that the recording is bound to the range defined by the horizontal edit bar. If the "1 shot" button is on the actual recording will start with the advent of the first note to be recorded and will stop after the locator has gone thru the edit bar region once.

Sequencer - Simple Modulation

A simple modulation sequencer. Similar to the "Sequencer - Modulation 4x" sequencer but with only one channel and less features.

Sequencer - Trigger 8x

Sequencer for playing back 8 parallel trigger channels. These triggers can be used to trigger envelopes, samplers, drum synthesizers etc.

A click on the "View" button toggles between the "all" and "solo" view. In "all" view all channels are visible at once. In "solo" view only one channel is visible, displayed with a higher vertical resolution. The "Select" bar (the second vertical bar from the left) selects the channel in "solo" view.

The "Seq" knob selects one of 128 different sequences so each snapshot can have its unique sequence number. The length of a sequence is 768 steps which equals 8 bars in a 96th notes resolution or 48 bars in a 16th note resolution. The numbers of sequences and the length can be set in the properties of the sequence display. The variables can be found on the "gearwheel"

page of the properties. The "X" variable is the length and "Y" is the number of sequences. Since the last 16 (8×2) sequences are used as copy/paste and undo buffers the number of sequences must be at least 24 (8×3).

Please note that the sequencer doesn't have an edit buffer. All changes are stored immediately. So if you want to do versions of a sequence you have to make a copy of the sequence using the copy and paste buttons before editing otherwise the "original" sequence is altered.

On the "Eye" page of the properties the time grid of the sequencer can be adjusted to the time base of the incoming position events ("Pos"-In) from the connected clock. If they have a 96th notes resolution the "Grid step" should be set to 6, if they have a 16th notes resolution it should be set to 1. Then an appropriate time grid is generated.

There are three horizontal bars: the edit bar, the loop bar and the view bar. To alter the size of these bars their left or right ends have to be clicked and dragged. To move them the middle of the bars have to be clicked and dragged. A click beside the bars let them jump one length to the left or to the right. The edit bar determines the sequence range the following functions can be applied on: copy/paste/cut/insert, randomization (Rand), quantization (Quant), clear and recording (Rec). The loop bar determines the sequence range, which is scanned and looped during playback. The view bar determines the visible range of the sequence. For the edit bar and the loop bar a quantization can be set with the "Bar /" control.

Information on the functions can be found in the mouse over hints of the function buttons. Please note that they are applied to all visible channels at once.

The trigger signals connected to the "Trig" inputs of the macro can be recorded. To do so the "recE" buttons of the channels has to be turned on to enable the recording. The recording starts when the "Rec 1/0" is pressed. Please note that the recording is

bound to the range defined by the horizontal edit bar. If the "1 shot" button is on the actual recording will start once the first trigger event is received. Recording will stop after the locator has passed once through the edit region.

23.7. LFO, Envelope

Envelope - ADSR

Envelope generator with the classic attack-decay-sustain-release characteristic.

Envelope - Decay

Envelope generator with the decay characteristic.

Envelope - One-Ramp

Envelope generator which generates a ramp between the set start and end point within an adjustable time. The shape of the ramp can be switched from exponential to linear with the "lin" button. The values of the start and end point are sampled and held in the moment the envelope is triggered if the "s/h" buttons are turned on.

Envelope Follower

The envelope follower generates an envelope in dependence on the amplitude of the input signal.

In "Peak" mode the output signal follows the amplitude peaks of the input signal. The input signal is rectified and smoothed by an adjustable release time. The attack time is zero.

In "Roots means square (Rms)" mode, the output signal follows the loudness of the input signal. That means for instance that short impulsive signals will not enter the output signal so much as in "Peak" mode since the shorter a signal gets, the smaller is its loudness. Technically the RMS value is the square root of the average of the squared values over the specified time interval.

LFO

Low frequency oscillator providing the following waveforms: slow random, sine, triangle, and pulse. The "Wave" control scans through them; intermediate positions between two successive waveforms can be set. The symmetry or pulse width of the waveforms is set with the "Width" control. A rising sawtooth is a triangle wave with the "Width" set to 1. For a falling sawtooth the "Width" control must be set to minus 1.

There are three different modes for setting the speed of the LFO, which is set with the "Unit" switch in the upper left corner of the macro. In "P" mode the "Speed" control changes the pitch of the LFO in semitones. In "bpm" mode the speed is set relative to the beats per minute (bpm) of the global clock of REAKTOR. In "pos" mode the speed is set relative to the frequency of the incoming events at the "Pos" input of the macro. The last mode should be selected if the LFO should sync to a clock, which is independent from the global clock. Then the position events of this independent clock must be connected to the "pos" input of the macro. The "Div" control divides the speed of the global clock and the measured speed of the incoming events at the "pos" input of the macro. E.g. in "bpm" mode the "Div" control sets the unit of the "Speed" control in 1/96 notes. 6 corresponds to 16th notes, 12 to 8th notes, 24 to quarter notes etc. In the "Pos" mode this correspondence applies if the incoming position events have a 1/96 notes resolution.

The LFO can be synced to an external signal or if the "Unit" switch is set to "bpm" or "pos" to the song position. When the synchronization occurs the LFO restarts at the phase set with the "Phase" control.

Sample and Hold

With incoming events at the "TE" input or at clock edges present at the "C" input, the current value of the incoming signal is sampled and held until the next sample is taken. It's possible to adjust the conditions for sampling from the panel.

Triggered Random

Random number generator that transmits a random number each time a clock edge in the input signal is detected. The "Edge" switch defines on which occasion a random number is triggered. If set to "+" the trigger occurs when the input signal rises above zero. If set to "-" the trigger occurs when the input signal falls under zero. If set to "both" the trigger occurs in both cases.

23.8. Filter

3 Band Filter

Versatile 3 band equalizer. Each band can be muted to achieve different filter curves including highpass, bandpass, lowpass and notch.

This filter can be used to simulate the "kill" filters of disc jockey mixers.

Bandsplit

Splits the incoming signal in high, mid, and low frequency bands for further processing. Mixing these bands would result in the unfiltered signal without any coloration.

Comb

Comb filter to produce flanger and chorus effects. A comb filter mixes the input signal with the delayed input signal, resulting in a frequency spectrum that resembles a comb with multiple sharp peaks and valleys. At multiples of the set filter frequency ($1/\text{delaytime}$), there are resonant peaks while at multiples of .5, 1.5, 2.5 etc of the filter frequency, the sound is cancelled out.

The amount of feedback (filter resonance) can be set with the corresponding knob. The "GainC" knob determines how much an increase of the feedback decreases the output level of the filter, which is especially helpful if the feedback is modulated. The "Ex" switch activates the external feedback path. The delayed signal is sent out at the "Send" output of the macro so it can be processed by filters and other signal modifiers. The processed signal should be connected to the "Ret" (Return) input of the macro. To avoid a constantly increasing volume in the feedback path the processing filters and signal modifiers should not amplify the signal beyond 1. For filtering the "Multimode - Accurate" macro of the classic modular library should be used (the "GainC" Knob should be set to 1).

Ladder Lowpass

Lowpass filter modelled on the classic circuit patented by Bob Moog with smooth saturated resonance, self-oscillation at high resonance settings and frequency modulation (FM).

The filter calculates four different lowpass filters: a 1 pole, a 2 pole, a 3 pole and 4 pole filter. With each pole the damping of frequencies greater than the cut-off frequency increases with 6 dB per octave. For instance the 4-pole filter has a damping of 24 dB per octave. The "Poles" knob "scans" thru the output signals of the four filters. If the scanning position is in between two filter signals the output signal is obtained by crossfading between these two.

Multimode - Accurate

CPU friendly multimode filter with an accurate frequency response. This means that there are nearly no unwanted boosts or attenuations of frequencies. The "GainC" knob determines how much an increase of the resonance decreases the output level of the filter, which is especially helpful if the resonance is modulated. With the "GainC" knob set to 1 the amplification for all frequencies is smaller or equal 1 regardless of the resonance boost. This setting is recommended if the filter is inserted in a feedback path of a delay line (the "delay" and the "comb" macro allow external processing of the feedback signal) to prevent the level within the feedback loop from getting louder and louder.

The following filter types are available:

- 6 dB/oct low/high pass filter
- 12 dB/oct low/band/high pass filter
- 24 dB/oct low/band/high pass filter

Multimode - Resonance Limiter

Multimode filter with a built-in resonance limiter. The "limit" control sets the threshold of this limiter in dB. If the bandpass filter signal exceeds this level the resonance for all filter types is reduced. This prevents loud filter responses at prominent frequency components of the filtered signal. The "F foll" control determines how much the threshold is decreased when the cut-off frequency is increased.

The "GainC" knob determines how much an increase of the feedback decreases the output level of the filter, which is especially helpful if the feedback is modulated.

The following filter types are available:

- 12 dB/oct low/band/high pass filter
- 24 dB/oct low/band/high pass filter

23.9. Delay

Delay

Delays the incoming signal by the time set with the "Delay" knob.

There are three "Unit" controls, which define the step size of the "Delay" knob. One of these controls is selected with the "Mode" switch. In "ms" mode the step size of the "Delay" knob is set in milliseconds. In "bpm" mode the delay time is set relative to the beats per minute (bpm) of the global clock of REAKTOR. The corresponding "Unit" control selects the note length. For instance "1/16" equals a 16th note. In "pos" mode the delay time is set relative to the measured time between incoming events at the "Pos" input of the macro. The step size of the "Delay" knob is the product of the measured time and the value of the "Unit" control. For instance, if the incoming position events have a 1/96 note resolution, the "Unit" control should be set to 6 to achieve a 16th notes step size. The "pos" mode should be selected if the delays should sync to a clock, which is independent from the global clock. Then the position events of this independent clock must be connected to the "pos" input of the macro.

The delay time can be modulated. The "MQ" control sets the quantization step size for the modulation signal in numbers of units set by the "unit" control.

The amount of feedback can be set with the corresponding knob. The "Ex" switch activates the external feedback path. The delayed signal is send out at the "Send" output of the macro so it can be processed by filters and other signal modifiers. The processed signal should be connected to the "Ret" (Return) input of the macro. To avoid a constantly increasing volume in the feedback path the processing filters and signal modifiers should not amplify the signal beyond 1. For filtering the "Multimode - Accurate" macro of the classic modular library should be used (the "GainC" Knob should be set to 1).

The "Dry/Wet" knob defines the mix of the dry and the wet (delayed) signal.

23.10. Audio Modifier

All audio modifiers which distort the incoming signal like the "clipper" or "saturator" have a built in input amplifier to set the level of the incoming signal to zero dB. This is the case if the corresponding meter just doesn't turn orange. It is recommended to do so since the value ranges of the control of the modifier are optimised for this signal volume.

Another common control is the gain correction "GainC" knob. It has the same functionality for all modifiers and is explained here for the "clipper". If it is set to 0 the gain correction is turned off. That means that lowering the clipping level lowers also the output level. This is the right setting if the modifier is inserted in the feedback path of a delay since then the signal isn't amplified. If set to 1 this loss in the volume is compensated. That means if the input signal has a level of 0 dB the output signal will also have a level of 0 dB independent of the set clipping level. This function is very helpful if the clipping level is modulated. Please note that this function only works properly if the incoming signal is levelled to 0 dB.

A lot of the modifiers have a symmetry ("Sym") knob. This controls how much the positive and negative part of the signal is distorted differently. If it is zero the distortion is the same for both sides.

Clipper

Clips the input signal at a controllable level.

Quantizer

Quantises the amplitude of the incoming signal. The signal is distorted into a step waveform. Can be used to simulate low bit resolutions of vintage samplers.

Ringmodulator

Signal modifier for amplitude and ring modulation. The "Mod Depth" control determines how much the amplitude of the input signal is modulated by the modulation signal. To achieve ring modulation this control has to be set to 1. Then the input signal is multiplied with the modulation signal.

Saturator

Soft saturating overdrive modifier to simulate tube distortion and tape saturation effects.

Slew Limiter

Slew rate limiter and smoothing filter.

The output signal follows the input signal with a limited rate. Different rate limits can be set for the rising signal and falling signal. This macro can be used to smooth modulation signals in general and to realise portamento (smooth gliding from one note pitch to the next).

Waveshaper

Signal modifier that shapes the input signal using 2 adjustable breakpoints.

Wrapper

Wraps or folds the incoming signal around an adjustable limit. Very powerful if used to shape oscillator signals. The results are similar to "Sync" or "Pulsewidth-modulation" sounds.

23.11. Event Processing

0-1 to 0-127 Range Converter

Multiplies the incoming signal with 127. Should be used if a modulation signal should be connected to a "P" (Pitch) input of a module or macro, which expects a value between 0 and 127. Can also be used to transform a modulation signal into position events so it can address positions in a step sequencer.

The output can be quantised to integer steps.

Quantizer

Distorts the incoming signal into a step waveform by rounding its values to the nearest multiple of the set step size.

Randomizer

Randomises the incoming events, which means that a random number in a definable value range is added.

24 Appendix

Musical Note Keys

S	D	G	H	J	2	3	5	6	7
49	51	54	56	58	61	63	66	68	70
Z	X	C	V	B	N	M	Q	W	E
48	50	52	53	55	57	59	60	62	64
							R	T	Y
							65	67	69
								U	I
								71	72

- Holding **⇧ Shift** raises all notes by two octaves (+24).
- Holding **Ctrl + ⇧ Shift** lowers all notes by two octaves (-24).

This reference table of modules is located in REACTOR and provides a quick description for each module. The description includes properties, settings, and so on. It is a description of all available modules.

All Reactor objects (Modules, Instruments, Effects, and Ensembles) have a Label field in their Properties. This label appears on the top-left corner in the structure Browser. This label for modules describes the module's function. Review modules with care, especially if you want Reactor to be able to understand your creation's structure.

Hybrid modules

A lot of modules in REACTOR are hybrid modules. After inserting such a module it appears as an input processing module per default indicated by red input labels. A green label on a port is indicating that you can connect modules as well as a normal value with this port. As soon as you connect a module to one of its inputs

25. Module Reference

Modules are the most elementary components of a REAKTOR ensemble. This section is primarily intended for REAKTOR users who wish to create their own ensembles from scratch. If you do not wish or feel experienced enough to create your own ensembles from scratch, REAKTOR offers alternative ways of accessing its potential:

- If you are not interested at all in building ensembles on your own, work on the ensemble level.
- If you want to assemble your favourite instruments in one ensemble and use them together, work on the instrument level.
- If you want to build your own Instruments by making use of prebuilt building blocks, work on the macro level.
- If you want to have control over each single parameter of your ensemble, work on module level.

This reference lists all modules available in REAKTOR and provides a basic description for each module. The description includes properties settings if available as well as a description of all input and output ports.

All Reaktor objects (Modules, Instruments, Macros, and Ensembles) have a Label field in their Properties. This label appears on the object's icon in the structure. By default, this label for modules describes the module's function. Rename modules with care, especially if you want other REAKTOR users to be able to understand your creation's structure.

Hybrid modules

A lot of modules in REAKTOR are **hybrid** modules. After inserting such a module it appears as an event processing module per default indicated by red port labels. A green dot on a port is indicating that you can connect audio as well as event cables with this port. As soon as you connect an audio cable to one of its inputs

the module is converted to an audio processing module indicated by black dots and labels on the ports. If you connect an event cable the port gets a red dot. A hybrid module has the following behaviour:

- If you mix audio and event cables at the inputs or if you solely connect audio cables, the module always works with audio rate.
- If you solely connect event cables to the inputs, it works with event rate.
- If the module's output is connected to an event input of another module, it becomes automatically an event processing module, and it is not possible anymore to add audio cables to the module's inputs.
- If the module's output is connected to an audio input, the module can work either as audio or event module, depending on the cables connected to the module's inputs.
- If the module already works with audio rate due to the input connections, you can not connect the output to an event input of another module.

Dynamic ports management

Where it makes sense, the modules offer a **dynamic** in- and/or out-port management. You can add inputs to the module by dropping additional cables above the module apart the area of an existing port while holding down the **Ctrl**-key. You can see where the new port will be added while holding the mouse pointer above the target module. The vertical position of the mouse pointer determines where the port is added so that it is possible to add a port between existing ports.

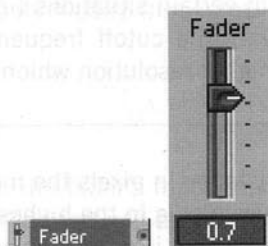
Panel

Panel modules provide onscreen controls for various Reaktor processes. They can be independently set to appear on the A and B Control Panels and they can be arranged differently on those panels. Some panel modules are for display purposes only. Those include lamps, meters, and scopes for displaying Reaktor proc-

esses as well as graphics and text modules for ornamentation. The remaining modules generate or route data for Reaktor processing. Those include faders, knobs, buttons, switches, menus, and an XY controller (which can be used for both display and control).

Fader

Panel



With the slider control in the panel you adjust the value which is output (as event and audio) by the corresponding module in the structure. The signal is mono – when connected to a poly input all voices receive the same value.

Properties - Function page

Range: The output value corresponds to the position of the knob which refers to a range between the limits **Min** and **Max** set in the properties dialog window. With the parameter **Stepsize**, the display and output values can be quantized to coarser steps, e.g. to display fewer decimal digits. Alternatively it is possible to use the **Num Steps** field to set the step resolution of the fader. The values of both fields, **Stepsize** and **Num Steps**, are dependent on each other. Changing the value in one of the fields causes an automatic value change in the other. While you enter the value range per step in the **Stepsize** field, the **Num Steps** field represents the total number of steps across the whole value range of the fader.

The highest possible resolution for a fader is 127.000 steps which is available if you enter 0 in the **Stepsize** field or 127.000 in the **Num Steps** field.

Note: You are able to enter a step resolution in the **Num Steps** field which is higher than the standardized MIDI resolution of 128. Be aware that Reaktor can use a higher resolution internally, but it can only exchange MIDI data with external hard- or software within the MIDI range of 128. Under certain circumstances this might effect the functionality or the sound of an ensemble when used in different production environments. Normally this is not a problem since the MIDI resolution is high enough for controlling parameters. Nevertheless in certain situations (using a filter with high resonance while moving the cutoff frequency for example) you might wish to have a higher resolution which then is possible inside Reaktor as well.

Mouse Res specifies the distance in pixels the mouse pointer has to cover to get from the lowest value to the highest. If you enter a value in the **Mouse Res** field which is double as high as the one in the **Pixel in Y** field inside the **Appearance** tab, you will have to draw the mouse twice the way of the fader size to change from the lowest to the highest value.

If **Num Steps** is bigger than **Mouse Res**, not every step is reached by moving the mouse. On the other side, if **Mouse Res** is bigger than **Num Steps** the number of pixels per step can be set to a higher value than 1.

The **Default** value is used whenever an initialization of the control happens. The value will be used in the following situations:

- Pressing the **Default** button in the snapshot window will reset all controls in the ensemble/instrument.
- If "default" appears in one of the morph targets in the snapshot window, you can morph between one snapshot and the default values of all controls..
- If you add a control to an Instrument which already contains a snapshot list, the default value is used for the new control until you overwrite the snapshot with a new value for that control or you activate **Snap Isolate**.

If the **Snap Isolate** switch is activated, the fader will not respond to snapshot recall.

Activating **Random Isolate** avoids that the fader reacts on executing the snapshot randomization function.

ID for Snapshot Files: The ID number is used for the snapshot management of REAKTOR. The numbers are entered automatically whenever you insert a panel controller (fader, knob, button, switch....). If you change this ID (if you want to import snapshots from another instrument with similar controls for instance) already existing snapshots doesn't recognize the panel element anymore and ignores it. Only modify this number if you know exactly what you do.

Properties - Info page

An info text to explain the fader's function can be entered under **Info** in the properties. The text will appear as a hint (Tooltip) while the mouse pointer rests on the fader in the panel, if hints are switched on.

Properties - Appearance page

The label you enter here will only be shown above the fader in the panel if **Label** is activated in the **Visible** section in the properties. Likewise, the currently set value will only be shown as a number below the fader if **Value** is activated. It is also possible to hide the fader bitmap itself, so that you only see the value box. In this case you can still use the fader in the panel by clicking and dragging up and down onto the value box.

Faders can appear **vertical** or **horizontal** in the panel. Tick the appropriate radio button in the **Form** section to achieve the desired look.

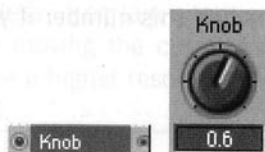
The length of the fader can be set freely in the **Pixel in Y** field in the **Size** section. You can also define the width of the fader with the three radio buttons **Big**, **Medium** and **Small**.

Properties - Connection page

The Connection page of the Properties for Panel Controls is described in the section *Connection properties of Panel Controls* on page 133.

Knob

Panel



Just like the fader but with a different appearance in the panel.

Button

Panel



With the push button control in the panel you select the value which is output (as event and audio) by the corresponding module in the structure. The output values in the on and off state are set with **On Value** and **Off Value** in the button's properties. The signal is mono – when connected to a poly input all voices receive the same value.

Properties - Function page

Range: The values which are sent by the Button when it is pressed and when it is released can be defined in the fields **On Value** and **Off Value**.

Mode: There is a choice of three operating modes: **Trigger**, **Gate** and **Toggle**. In trigger mode, events are only generated when the button is pressed, nothing happens when it is released. In gate mode, an event with value zero is sent when the button is released. In toggle mode, the button changes state every time it is pressed.

Default = On: Sets the default value which is used at several actions in REAKTOR to On.

Activating **Snap Isolate** avoids that the button reacts on snapshot recalls.

Activating **Random Isolate** avoids that the button reacts on executing the snapshot randomization function.

ID for Snapshot Files: See fader description.

Properties - Info page

An info text to explain the button's function can be entered under **Info** in the properties. The text will appear as a hint (Tooltip) while the mouse pointer rests on the button in the panel, if hints are switched on.

Properties - Appearance page

The label will only be shown above the button in the panel if **Label** is activated in the properties. Likewise, the currently set value will only be shown as a number below the button if **Label** is activated.

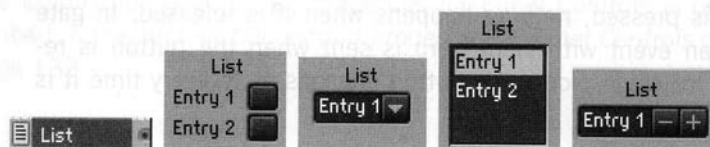
The button can be displayed in three sizes: **Small**, **Medium** and **Big**.

Properties - Connection page

The Connection page of the Properties for Panel Controls is described in the section *Connection properties of Panel Controls* on page 133.

List

Panel



The List module is for constructing onscreen lists, drop-down menus, button selectors, and scrolling text displays.

Properties - Function page

The **Function** Properties contains a list into which you can **Append**, **Insert**, and **Delete** individual items. You can also specify the number of items directly using a numerical (**NUM Entries**). The items in the list will be automatically displayed on the control panel in the selected display format. For each item, you can type in a value to be sent to the module output when that item is selected.

The Function page also contains a value generator. With this little tool you can generate values for multiple entries in the entries list. Example: you want to have an increment of 4 instead of 1 for each next entry. Therefore enter "4" as **Stepsize** in the Value Generator and press the **Apply** button. Now the values in the **Value** column of the entry list are modified according to the settings in the Value Generator.

The **Mouse Resolution** only applies to the panel control if you choose the **Spin** style on the **Appearance** page, where you can click on the control entry and drag up or down to change the entry.

Properties - Appearance page

The panel representation of this module changes depending on the chosen style on the properties' **Appearance** page. The following styles are available:

- **Button:** Each module in-port creates a button. All buttons will be arrayed vertically in the instrument panel. The currently activated button will be displayed in the Indicator color of the Instrument.
- **Menu:** Each module in-port creates a new entry in a drop down list.
- **Text Panel:** Each module in-port creates a new entry in a list which displays multiple entries at the same time. If you have created more entries than fit into the text panel display specified by the **Size X** and **Size Y** fields on the Appearance tab, you will get scrollbars in the panel.
- **Spin:** Each module in-port creates a new entry in a list. You can switch through the list using a + and a - button right hand of the list entry panel display.

The **Size X** and **Size Y** fields are controlling the display size of the control element in the panel.

Ports

- **Out:** Event output for the value associated with the selected item.

Switch

Panel



Switches are used for changing the signal flow. They do not contain any signal processing of their own but establish a switchable connection between other modules. The output is connected to the selected input by pushing the corresponding button or by selecting a list entry. All the other unselected inputs are disconnected.

Modules whose outputs are disconnected, through the action of a switch or otherwise, are automatically deactivated to reduce unnecessary load to the CPU. A module's status lamp remains dark while the module is deactivated.

This module is a hybrid module, so it can process event signals as well as audio signals, depending on its connections, and it contains a dynamic management of its in-ports.

Properties - Function page

Enable Switch Off: If this option is enabled, the switch can have a state where no input is selected. If your display is set to button style on the **Appearance** page of the Properties, it is possible to switch off all buttons by clicking a second time on the currently selected button. Using any other of the display styles you get an additional entry called "Off".

Min Num Port Groups: You can set the minimum number of module ports here regardless of the number of cables you have connected to the module.

The **Mouse Resolution** only applies to the panel control if you choose the **Spin** style on the **Appearance** page, where you can click on the control entry and drag up or down to change the entry.

Properties - Appearance page

The label will only be shown above the button in the panel if **Label** is activated on the **Appearance** page of the properties.

The panel control can be displayed in three sizes: **Small**, **Medium** and **Big**.

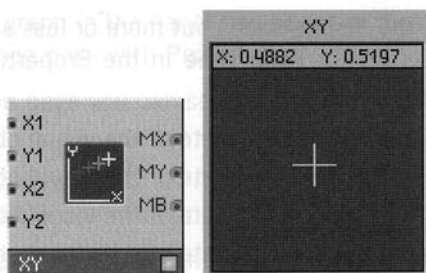
If you use a switch with two in-ports, only one button (the upper button) is shown when **1 Toggle Button** is selected: When the button is on, input 1 is connected, when the button is off, input 2 is connected.

The panel representation of this module changes depending on the chosen style on the properties' **Appearance** page. The following styles are available:

- **Button:** Each module in-port creates a button. All buttons will be arrayed vertically in the instrument panel. The currently activated button will be displayed in the Indicator color of the Instrument.
- **Menu:** Each module in-port creates a new entry in a drop down list.
- **Text Panel:** Each module in-port creates a new entry in a list which displays multiple entries at the same time. If you have created more entries than fit into the text panel display specified by the **Size X** and **Size Y** fields on the Appearance tab, you will get scrollbars in the panel.
- **Spin:** Each module in-port creates a new entry in a list. You can switch through the list using a + and a - button right hand of the list entry panel display.

XY

Panel



The XY control field has two functions: It displays audio input signals and acts as a 2-dimensional controller used with the mouse.

Properties - Function page

The **Always Active** option enables the ability of the module to activate a signal branch connected to one of the in-ports of the module.

In **Incremental Mouse Mode** the control reacts similar to a knob. In this mode you can click anywhere within the panel display of the control and move the mouse without resetting the value to the position of the mouse pointer directly. Instead, the value will follow the mouse pointer with a fixed offset.

Properties - Appearance page

In the module Properties you can set the display type for the audio signals to be visualized. The inputs **X1** and **Y1** control the position of the visual object (**Pixel** or **Cross**). When the object is set to **Bar** or **Rectangle** then **X1** and **Y1** control one corner and **X2** and **Y2** the opposite corner of the object.

To display fast moving audio data select **Scope** mode. All other modes evaluate the input only at the lower graphics update rate.

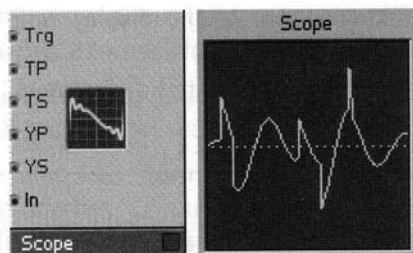
You can also set the size of the crosshair appearing at the mouse position.

Everything drawn in the display fades out more or less slowly, depending on the value set for **Fade Time** in the Properties (maximum value is 99).

- **X1**: Audio input for the X1 coordinate of the visual object.
- **Y1**: Audio input for the Y1 coordinate of the visual object.
- **X2**: Audio input for the X2 coordinate of the visual object.
- **Y2**: Audio input for the Y2 coordinate of the visual object.
- **MX**: Event output for the X position of the mouse cursor when the mouse button is pressed on the display.
- **MY**: Event output for the Y position of the mouse cursor when the mouse button is pressed on the display.
- **MB**: Left mouse button state (1=pressed, 0=released)

Scope

Panel



Oscilloscope for displaying a time-varying signal.

Every time an Event is received at the Trigger input (**Trg**), the module starts recording the audio signal at the input (**In**) and displays it on the panel. When no trigger events are received, the display continues showing the stored signal and it can be shown at varying scales and positions by adjusting the relevant control inputs.

The size of the graph in the panel can be set in the module's properties **Appearance** page with **Pixel in X** and **Pixel in Y**.

On the **Function** page you can set the buffer used by the scope in ms.

You would typically connect an **A to E Trig** module to the **Trg** input for triggering the Scope from an audio signal.

- **Trg**: Mono event input for the trigger signal that synchronises the trace.
- **TP**: Mono event input for controlling the offset in time (Time Position) of the trace in milliseconds. The trace starts at the left edge of the display **TP** ms after the trigger event.
- **TS**: Mono event input for controlling the time scale of the displayed trace. From left to right edge, the display shows **TS** ms of the signal.
- **YP**: Mono event input for controlling the amplitude offset (Y Position) of the trace. **YP** = -1 corresponds to the bottom edge of the display, +1 is the top edge.

- **YS:** Mono event input for controlling the amplitude scaling (Y Scale) of the trace. The difference between the signal values at the top and at the bottom of the display is 2 **YS**. When **YP** = 0, the display shows values between +**YS** and -**YS**.
- **In:** Mono audio input for the signal to be displayed.

Lamp

Panel



Indicator lamp for a monophonic signal.

The lamp in the panel lights up as long as the input signal (sampled at 25 Hz) is within the range set with **Min** and **Max** on the properties' function tab, i.e. the lamp is on when the signal's value is larger than **Min** and less than or equal to **Max**.

If the **Continuous** mode is checked in the properties, the lamp color will fade in and out between the **Min** and **Max** values.

The color of the lamp (red, green, blue, yellow or indicator) can be chosen in the properties. If you check **Indicator Color**, the lamp will use the indicator color chosen in the instrument properties.

It is also possible to define custom colors for the On and Off position of the lamp using the buttons **Set On Color** and **Set Off Color**.

If you uncheck the **Has Frame** option, you are able to place lamps pixel accurate side by side in the panel without any gaps between them.

The label will only be shown above the lamp in the panel if **Label** is activated in the properties.

Level Lamp

Panel



Indicator lamp for a monophonic signal with logarithmic settings.

The lamp in the panel lights as long as the input level is within the range set with **Min** and **Max** (in dB) in the properties, i.e. the lamp is on when the signal's value is larger than **Min** and less than or equal to **Max**.

If the **Continuous** mode is checked in the properties, the lamp color will fade in and out between the **Min** and **Max** values.

The color of the lamp (red, green, blue, yellow or indicator) can be chosen in the properties. If you check **Indicator Color**, the lamp will use the indicator color chosen in the instrument properties.

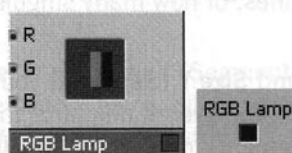
It is also possible to define custom colors for the On and Off position of the lamp using the buttons **Set On Color** and **Set Off Color**.

If you uncheck the **Has Frame** option, you are able to place lamps pixel accurate side by side in the panel without any gaps between them.

The label will only be shown above the lamp in the panel if **Label** is activated in the properties.

RGB Lamp

Panel

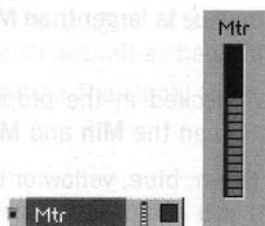


The RGB Lamp module is a resizable, colored display. Its color is controlled by the values appearing at its three color inputs. Its horizontal and vertical size can be set in pixels in the **Appearance** Properties.

- **R:** Audio input for the intensity of the red component. Range 0 to 1.
- **G:** Audio input for the intensity of the green component. Range 0 to 1.
- **B:** Audio input for the intensity of the blue component. Range 0 to 1.

Meter

Panel



Value indicator for a monophonic signal.

The value of arriving signal is sampled (at 25 Hz) and displayed on a linear scale. The displayed range is set with **Min** and **Max** in the properties.

The color of the meter (red, green, blue, yellow or indicator) can be chosen in the properties. If you check **Indicator Color**, the lamp will use the indicator color chosen in the instrument properties.

It is also possible to define custom colors for the upper and lower part of the meter using the buttons **Set On Color** and **Set Off Color**.

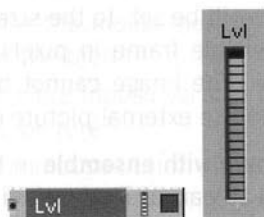
Number Of Segments defines, of how many singular elements the meter is composed.

Under **Size X (Segment)** and **Size Y (Segment)** you can define the size of one meter element. The overall height of the meter will be the result of **Size Y (Segment)** multiplied by **Number of Segments**.

The label will only be shown above the meter in the panel if **Label** is activated in the properties.

Level Meter

Panel



Level indicator for a monophonic audio signal.

The amplitude of the connected audio signal is displayed on a logarithmic scale. The displayed range is set in dB with **Min** and **Max** in the properties.

The color of the meter (red, green, blue, yellow or indicator) can be chosen in the properties. If you check **Indicator Color**, the lamp will use the indicator color chosen in the instrument properties.

It is also possible to define custom colors for the upper and lower part of the meter using the buttons **Set On Color** and **Set Off Color**.

Number Of Segments defines, of how many singular elements the meter is composed.

Under **Size X (Segment)** and **Size Y (Segment)** you can define the size of one meter element. The overall height of the meter will be the result of **Size Y (Segment)** multiplied by **Number of Segments**.

The label will only be shown above the level meter in the panel if **Label** is activated in the properties.

Picture

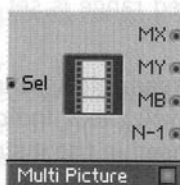
Panel

Allows loading a panel decoration bitmap from a picture file in TGA format (File extension *.tga). The bitmap module has no in- or outputs. The display size will be set to the size of the image. It is possible to modify the visible frame in pixels under Appearance/Size in the properties. The image cannot be resized inside Reaktor, for this you need to use external picture editing software.

Select the option **Save bitmap with ensemble** in the Properties to have the image data stored as part of Reaktor's file.

Multi Picture

Panel



The Multi Picture module is a 2-dimensional controller (like the XY module) that reports the mouse position and mouse button status at its outputs. It supports multi-frame animation when pictures are set up that way in the Picture Properties window by indicating the number of frames (animations) and their orientation (horizontal or vertical).

24-bit BMP and 32-bit Targa (uncompressed) formatted graphics may be used in Reaktor in several places: as Instrument and Macro control panel backgrounds, as Instrument and macro structure icons, and in the Picture and Multi Picture control panel modules. The advantage of Targa is that it supports an alpha channel, which can be used as a mask for the visible portion of the graphic-the unmasked portion will then be transparent. That's useful for round knobs on a square background, for example.

You can load pictures using the drop-down Select-Picture menu in the Properties of any object that can display pictures. Opening a picture automatically brings up the Picture Properties window where you make all relevant picture settings. All pictures are automatically shared and available to all appropriate modules.

- **Sel:** Audio input for selecting the frame by number. Range 0 to number of the last frame.
- **MX:** Event output for the mouse horizontal (X) position when the mouse is within the picture.
- **MY:** Event output for the mouse vertical (Y) position when the mouse is within the picture.
- **MB:** Event output for mouse button status (0 when up, 1 when down).
- **N - 1:** Event output for the number of animations you have entered in the Picture Properties -1.

Text

Panel


 A small rectangular icon with a dark background and the word "Text" in white.

The Text module does not process any signals, its purpose is only to add text to the structure. For example it can be used for noting the author and creation date of an instrument and to explain how it works.

Multi Text

Panel


 A small rectangular icon with a dark background, the text "Multi Text", and a small square icon to the right.

The Multi Text module provides a changeable text display for control panels. Any number of text items can be added in the module properties, where items can also be edited or deleted.

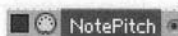
In: Audio input for the number of the text item to be displayed.

MIDI In

MIDI In modules are for routing MIDI messages in to Reaktor from external MIDI devices. There are separate modules for various MIDI data types including notes, Velocity, pitchbend, mono and poly aftertouch, controllers, program changes, and MIDI clock. Reaktor also generates separate gate messages for MIDI note events and there are modules for that. Some MIDI In Modules can also be used in an internal or OSC connection.

Note Pitch

MIDI In



Polyphonic event source for the pitch of MIDI Note On events.

A Note On event sets the output to a value determined by the key number (note pitch). The range of the output value is set with **Min** and **Max** in the properties dialog window. The resolution is 128 steps. A Note Off event has no effect here.

When using the default range of **Min** = 0 to **Max** = 127 and controlling an oscillator's **P**-input (for logarithmic pitch control) you will play in the common equal tempered tuning. One unit corresponds to one semitone and middle C is at 60. By setting **Min** and **Max** to different values, pitch can be skewed or scaled, e.g. for playing in quarter tones.

Pitchbend

MIDI In



Monophonic event source for MIDI Pitchbend (pitchwheel change) events. The resolution is 16384 steps. When the pitch bend controller is in its neutral position, the output value is always 0. The range of the output value for upward or downward pitchbend can be set independently in the properties dialog window. For pitchbend down the range is set with **Min** and for pitchbend up with **Max**.

When controlling an oscillator's **P**-input (for logarithmic pitch control), e.g. after adding to a note pitch value, and with a range of **Min** = -1 to **Max** = 1 you can change the pitch up or down one semitone. A range of -12 to 12 means pitchbend within ± 12 semitones which is ± 1 octave.

Gate

MIDI In



Polyphonic event source for MIDI Note On and Note Off events.

A Note On event sets the output to a value determined by the key velocity. The range of the output value is set with **Min** and **Max** in the properties dialog window. The resolution is 128 steps. A Note Off event sets the output to zero. If velocity sensitivity is to be disabled, **Min** and **Max** should be set to the same value.

Single Trig. Gate

MIDI In



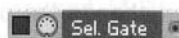
Monophonic event source for MIDI Note On and Note Off events with single trigger characteristic.

A Note On event sets the output to a value determined by the key velocity. The range of the output value is set with **Min** and **Max** in the properties dialog window. The resolution is 128 steps. A Note Off event sets the output to zero. If velocity sensitivity is to be disabled, **Min** and **Max** should be set to the same value.

Only the first note produces an event and thus triggers (or retriggers) a connected envelope. A new note that is started while the previous one is still held (legato play) does not generate an event and therefore does not retrigger any envelope.

Sel. Note Gate

MIDI In

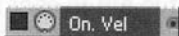


Monophonic event source for selected Note On and Note Off events.

A Note On event which has the selected note number sets the output to a value determined by the key velocity. The range of the output value is set with **Min** and **Max** in the properties dialog window. The resolution is 128 steps. A Note Off event which has the selected note number sets the output to zero. If velocity sensitivity is to be disabled, **Min** and **Max** should be set to the same value.

On Velocity

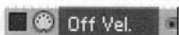
MIDI In



Polyphonic event source for velocity of MIDI Note On events. A Note On event sets the output to a value determined by the key velocity. The range of the output value is set with **Min** and **Max** in the properties dialog window. The resolution is 128 steps.

Off Velocity

MIDI In



Polyphonic event source for the velocity of MIDI Note Off events. A Note Off event sets the output to a value determined by the key release velocity. The range of the output value is set with **Min** and **Max** in the properties dialog window. The resolution is 128 steps.

There are very few keyboards that generate a value other than zero for this event.

Controller

MIDI In



Monophonic event source for MIDI Controller events. An event sets the output to a value determined by the position of the controller (e.g. modulation wheel). The number of the controller is set in the properties dialog window with **Controller No** and the range of the output value is set with **Min** and **Max**. The resolution is 128 steps.

The current position of all MIDI controllers (and faders) of an instrument can be stored in a snapshot from where it can be recalled at a later time. If the **Snap Isolate** switch is activated, the controller module's output value will not respond to snapshot recall.

The output of a controller is monophonic and can be used as an event or audio signal.

The numbers of some standard MIDI controllers are:

- 1 Modulation Wheel
- 2 Breath Controller
- 7 Volume
- 10 Panpot
- 64 Sustain Switch
- 65 Portamento Switch
- 66 Hold Switch (Sostenuto)

See your keyboard's MIDI implementation chart to find out which controllers it can transmit.

Ch. Aftertouch

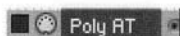
MIDI In



Monophonic event source for MIDI Channel Aftertouch events. An event sets the output to a value determined by the pressure on all keys. The range of the output value is set with **Min** and **Max** in the properties dialog window. The resolution is 128 steps.

Poly Aftertouch

MIDI In



Polyphonic event source for MIDI Poly Aftertouch events. An event sets the output to a value determined by the pressure on the key. The value is only set for the particular voice in the REAKTOR instrument which is playing the note associated with the key. The range of the output value is set with **Min** and **Max** in the properties dialog window. The resolution is 128 steps.

There are very few keyboards that generate poly aftertouch events.

Sel. Poly AT

MIDI In



Monophonic event source for selected MIDI Poly Aftertouch events.

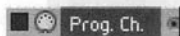
An event which has the selected note number sets the output to a value determined by the pressure on the key. The number of the key (note number) is set in the properties dialog window with **Controller No** and the range of the output value is set with **Min** and **Max**. The resolution is 128 steps.

There are very few keyboards that generate poly aftertouch events.

The current value can be stored (together with the MIDI controllers and faders) in a snapshot from where it can be recalled at a later time. If the **Snap Isolate** switch is activated, the module's output value will not change on snapshot recall.

Program Change

MIDI In



Monophonic event source for MIDI Program Change events. A MIDI event sets the module's output to the value determined by the transmitted program number. The range of the output value is set with **Min** and **Max** in the properties dialog window. The resolution is 128 steps.

When using this module, you probably want to turn off **Prog. Change Enable** in the Instrument Properties so that the MIDI Program Change Events do not also recall snapshots.

Start/Stop

MIDI In



Source for start and stop events for the synchronization with external MIDI devices or the internal master clock.

The output of the **Start/Stop** module is a monophonic gate signal. Its value can be set with **Output Value** in the properties. The signal jumps to the set value when the start button in the toolbar is pressed or when a MIDI Start event is received, as appropriate. The signal jumps back to zero when the stop button is pressed or when a MIDI Stop event is received.

The module is typically connected to the reset input of sequencers and event dividers which are synchronized with the MIDI Clock to force a synchronous start.

1/96 Clock

MIDI In



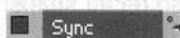
Source of a clock signal which corresponds to the external MIDI Clock or the internal master clock.

For each 96th note an event is sent at the output. The value can be set as **Output Value** in the properties.

In contrast to the **Sync Clock** source, the events of the **1/96 Clock** have to be processed by an **Event Freq. Divider** (see page 357). This has the advantage that you can experiment with arbitrary division factors.

Sync Clock

MIDI In



Source for a clock signal which is derived from an external MIDI Clock or the internal master clock.

The output of the **Synch Clock** module is a monophonic gate signal. The value can be set with **Output Value** in the properties. At each beat, the gate jumps to the respective value and back to zero after a certain time interval. The module is activated and deactivated by start-stop events.

The rate of the clock signal (quarter, eighth, sixteenth notes, etc.) can be adjusted in properties as **Rate**.

The duration of the gate signal (eighth, sixteenth note, etc.) can be adjusted in properties as **Duration**.

Song Pos

MIDI In



Source for the Song Position, counted in 96th notes from the song start. Typically connect this to input (A) of the Modulo module and a Constant of 6 to the (B) input, to get a 16th note count at the (Div) output.

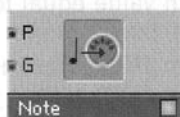
- **96**: Event output for the integer count of 96th notes (that's 24 ppq). Use the Modulo module to get counts of other denominations.
- **96a**: Audio output for the sample-accurate fractional song position measured in 96th notes (24 per quarter note).

MIDI Out

Reaktor can generate as well as process MIDI messages. MIDI Out modules are for sending those to external MIDI devices. There are modules here corresponding to each of the MIDI In modules. Some MIDI Out Modules can also be used in an internal or OSC connection.

Note Pitch/Gate

MIDI Out

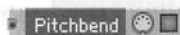


Converts a monophonic or polyphonic event signal to MIDI Note events. Each event at the gate input **G** generates a MIDI message at the MIDI port which REAKTOR uses for output. The value of the event specifies the velocity. An event value of 1 produces a velocity of 127. An event with value zero produces a Note Off event.

The pitch of the Note On and Note Off events is the current value at the pitch input **P** in the range set with **Min** and **Max**. An event with value equal to **Min** sets the MIDI Note Pitch to 0, an event with value equal to **Max** sets the MIDI Note Pitch to 127.

Pitchbend

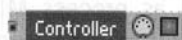
MIDI Out



Converts a monophonic event signal to MIDI Pitchbend events. Each event generates a MIDI message at the MIDI port which REAKTOR uses for output. The range of the input value is set with **Min** and **Max**. The output resolution is 16384 steps. An event with value equal to **Min** sets the MIDI Pitchbend value to -8192, an event with value equal to **Max** sets the MIDI Pitchbend value to +8191.

Controller

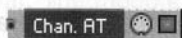
MIDI Out



Converts a monophonic event signal to MIDI Controller events. Each event generates a MIDI message at the MIDI port which REAKTOR uses for output. The number of the controller is set in the properties dialog window with **Controller No** and the range of the input value is set with **Min** and **Max**. The output resolution is 128 steps. An event with value equal to **Min** sets the MIDI Controller value to 0, an event with value equal to **Max** sets the MIDI Controller value to 127.

Ch. Aftertouch

MIDI Out



Converts a monophonic event signal to MIDI Channel Aftertouch events. Each event generates a MIDI message at the MIDI port which REAKTOR uses for output. The range of the input value is set with **Min** and **Max**. The output resolution is 128 steps. An event with value equal to **Min** sets the MIDI Aftertouch value to 0, an event with value equal to **Max** sets the MIDI Aftertouch value to 127.

Poly Aftertouch

MIDI Out

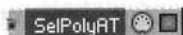


The Aftertouch (**AT**) events are converted to MIDI Poly Aftertouch events. The momentary value at the Pitch (**P**) input is converted to the note number. Values between **Min** and **Max** result in note numbers between 0 and 127. Aftertouch values between 0 and 1 are converted to values between 0 and 127.

- **P:** Input for the Pitch converted to the MIDI note number. Values between **Min** and **Max** result in note numbers between 0 and 127.
- **AT:** Input for the Aftertouch signal. Values between 0 and 1 are converted to MIDI aftertouch values between 0 and 127.

Sel. Poly AT

MIDI Out

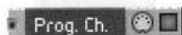


Converts a monophonic event signal to MIDI Poly Aftertouch events. Each event generates a MIDI message at the MIDI port which REAKTOR uses for output. The number of the key for which to set the pressure (note number) is set in the properties dialog window with **Note No.** and the range of the input value is set with **Min** and **Max**. The output resolution is 128 steps. An event with value equal to **Min** sets the aftertouch value to 0, an event with value equal to **Max** sets the aftertouch value to 127.

There are very few MIDI devices which handle poly aftertouch events.

Program Change

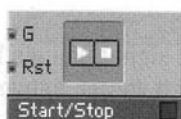
MIDI Out



Converts a monophonic event signal to MIDI Program Change events. Each event generates a MIDI message at the MIDI port which REAKTOR uses for output. The range of the input value is set with **Min** and **Max**. The output resolution is 128 steps. An event with value equal to **Min** sets the MIDI Program Change value to 0, an event with value equal to **Max** sets the MIDI Program Change value to 127.

Start/Stop

MIDI Out



Input events create Start/Continue/Stop events at the MIDI output.

- **G:** A positive event sends a Start or a Continue message. Negative or zero events send a Stop message.
- **Rst:** After a positive event at this Reset input, the next positive event at the Gate sends a Start (at zero) message.

1/96 Clock

MIDI Out



Input events create (1/96th) clock events at the MIDI output.

- **In:** An event with a positive value creates a MIDI clock event.

Song Pos

MIDI Out



The value at the **Pos** input is sent with an event at the **Trig** input as MIDI song position.

- **Trg:** Every event with a positive value creates a MIDI song position event.
- **Pos:** The value at this input (as a multiple of 1/96th notes) is taken with a Trigger event and sent as MIDI song position.

Math

You figure out the equation and Reaktor does the math. There are modules here for common operations such as addition, subtraction, multiplication, and division as well as for less-familiar mathematical functions such as absolute value, arc-tangent, and reciprocal square root. There are also modules here for exponential and logarithmic scale conversion to match Reaktor's module-input scaling. For example, some Reaktor modules have linear frequency inputs (measured in Hertz) while others have exponential frequency inputs (measured in semitones and adjusted for MIDI note numbering). There are modules here for converting between those two formats.

All modules in this section are hybrid modules, meaning they can be used as either audio or event modules depending on their inputs. Many of the modules (**Add** and **Mult**, for example) also have dynamic inputs as indicated by three small dots at the bottom-left of the module icon. When a wire is dragged to an empty part of the in-port region (the left edge of the module icon) of a dynamic-input module while holding down the **Ctrl** key, a new input is created automatically.

Constant

Math



Monophonic source for a constant value. The value is set with **Constant Value** in the properties dialog window. The module only sends an event once, that is for initialization when it is activated.

Add

Math

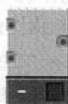


Adder for two or more audio or event signals. The output signal is the sum of the input signals (**Out** = **In1** + **In2** + **In3** etc.).

Can also be used as a simple multi-channel mixer where the level of all channels is fixed at 0 dB.

Substract

Math



Subtractor for two audio or event signals. The output signal is the subtraction of the second input signal from the first (**Out** = **In1** - **In2**).

Invert, -X

Math



Inverter for an audio or event signal. The output signal is the inverse of the input signal (**Out** = **-In**).

Simply inverting a sound has no audible effect, except when combining in some way with the uninverted sound. But inverting control signals generally has a very obvious effect.

Multiply

Math



Multiplier for two or more audio or event signals. The output signal is the product of the input signals (**Out** = **In1** **In2** * **In3** etc).

The typical application is as an amplifier controlled by a signal (corresponds to VCA in analog synthesizers) when an audio signal is fed to one input and an amplification factor to the other.

When a zero is connected to one of the inputs the output value is always zero.

Can also be used to compute the square of a signal when the same signal is fed to two inputs (**Out** = $\text{In} \cdot \text{In} = \text{In}^2$).

When two different sounds are connected to the inputs, the output is the ring modulation of the two sounds.

a * b + c

Math



Combined multiplier-adder: Output is the result of $(A * B) + C$.

Reciprocal 1/x

Math



The output is one divided by the input.

Be careful with small input values (near zero) because the output values become very large. If the input signal is exactly zero the output is also set to zero.

Processing sound signals does not normally yield anything useful. The module is rather meant for processing control signals (which don't come near zero).

Divide x/y

Math



The output is the upper input divided by the lower input.

Be careful with small values (near zero) at the lower input because the output values become very large. If the input signal is exactly zero the output is also set to zero.

Dividing by sound signals does not normally yield anything useful.

Whenever possible use a multiplier instead of a divider for audio signals because the CPU load will be significantly less. For example, rather than dividing by a constant or an event signal, invert the event signal ($1/x$) and multiply audio with the result.

Modulo x % y

Math

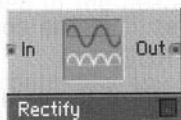


Modulo and Div. Computes the integer division of the two input values and also the remainder.

- **A:** Hybrid input for signal **A** to be divided by **B**. Typ. Range: [0 ... 100].
- **B:** Hybrid input for signal **B** used for dividing **A**. **B** is normally an integer, but doesn't have to be. Typ. Range: [1 ... 100].
- **Div:** Hybrid output for the integer division of **A** and **B**. This is the largest integer smaller than or equal to A/B . Typ. Range: [0 ... 100].
- **Mod:** Hybrid output for the modulo of **A** and **B**. This is the remainder of the integer division of **A** and **B**. Range: [0 ... B].

Rectifier

Math

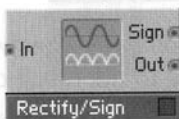


The input signal is rectified, i.e. negative values are inverted and become positive.

- **In:** Hybrid input for signal to be rectified. Negative values become positive with equal magnitude.
- **Out:** Hybrid output for the rectified signal.

Rect./Sign

Math



The input signal is rectified, i.e. negative values are inverted and become positive. The sign of the input signal is available at the **Sign** output.

- **In:** Hybrid input for signal to be rectified and analyzed for the sign.
- **Sign:** Hybrid output for the sign of the input signal. 1 for positive signals, -1 for negative signals.
- **Out:** Hybrid output for the rectified signal. Always positive.

Compare

Math

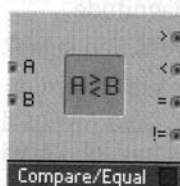


Comparing Logic Function. Compares the two input values and sets the two outputs according to the result of the comparison.

- **A:** Hybrid input for the first of two values to be compared.
- **B:** Hybrid input for the second of two values to be compared.
- **>:** Hybrid output for the result of the comparison "**A** greater than **B**". (0 = FALSE, 1 = TRUE)
- **<:** Hybrid output for the result of the comparison "**A** less than **B**". (0 = FALSE, 1 = TRUE)

Compare/Equal

Math

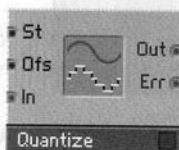


Comparing Logic Function. Compares the two input values and sets the four outputs according to the result of the comparison.

- **A:** Hybrid input for the first of two values to be compared.
- **B:** Hybrid input for the second of two values to be compared.
- **>:** Hybrid output for the result of the comparison "**A** greater than **B**". (0 = FALSE, 1 = TRUE)
- **<:** Hybrid output for the result of the comparison "**A** less than **B**". (0 = FALSE, 1 = TRUE)
- **=:** Hybrid output for the result of the comparison "**A** equal to **B**". (0 = FALSE, 1 = TRUE)
- **!=:** Hybrid output for the result of the comparison "**A** not equal to **B**". (0 = FALSE, 1 = TRUE)

Quantize

Math



Quantizer for audio and event signals, with adjustable step size.

The input signal is rounded to the nearest quantization step before being output.

- **St:** Hybrid input for controlling the size of the quantization step. When set to zero, the signal is not quantized.
- **In:** Hybrid input for the signal to be quantized.
- **Out:** Hybrid output for the quantized signal.
- **Err:** Hybrid output for the quantization error that is generated by rounding: **Err = Out - In**.

Expon. (A)

Math



Exponentiator for converting logarithmic level values in dB to linear amplitude values.

- **Lvl:** Hybrid input for logarithmic level values in dB to be converted to linear amplitude values. Typ. range: [-50 ... 10].
- **A:** Hybrid output for a linear amplitude control signal.

Expon. (F)

Math

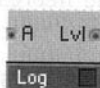


Exponentiator for converting logarithmic pitch values in semitones to linear frequency values in Hz.

- **P:** Hybrid input for logarithmic pitch values in semitones to be converted to linear frequency values in Hz. Typ. range: [0 ... 127].
- **F:** Hybrid output for a frequency control signal in Hz.

Log (A)

Math

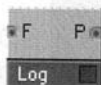


Logarithm for converting linear amplitude values to logarithmic level values in dB. Also for driving the logarithmic time inputs of envelopes etc.

- **A:** Hybrid input for linear amplitude value to be converted to logarithmic level value in dB. Typ. range: [0 ... 1000].
- **Lvl:** Hybrid output for level in dB. Typ. range: [-60 ... 0].

Log (F)

Math

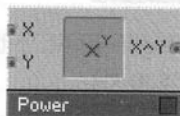


Logarithm for converting linear frequency values in Hz to logarithmic pitch values in semitones.

- **F:** Hybrid input for linear frequency value in Hz to be converted to logarithmic pitch value in semitones. Typ. range: [0 ... 5000].
- **P:** Hybrid output for pitch in semitones. Typ. range: [0 ... 100].

Power x y

Math



Power Function. The output delivers the result of X to the power of Y (usually denoted X^Y or $X^{\wedge}Y$).

X: Hybrid input for the basis.

Y: Hybrid input for the exponent.

X^Y: Hybrid output for the result of the calculation.

Square Root

Math

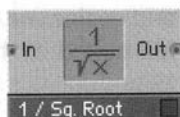


Computes the square root of the input values.

- **In:** Hybrid input for the argument to the square root function. For negative input values, the output is zero. Typ. Range: [0 ... 100].
- **Out:** Hybrid output for the square root of the input value. Typ. Range: [0 ... 10].

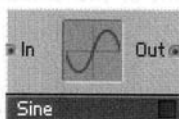
1 / Square Root

Math



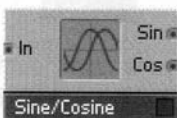
The reciprocal square root module computes the reciprocal of the square root of the input. It is more efficient than using the Square Root module followed by the Reciprocal module. For negative inputs the output is zero.

- **In:** Hybrid input for the argument.
- **Out:** Hybrid output for the value.



The Sine module calculates the trigonometric sine function. Both the input and output are scaled to a range of -1 to 1. To calculate an input based on degrees, first divide by 360. To calculate an input based on radians, divide by $2 * \pi$ (approximately 6.283). The output ranges from 1 (for input .25) to -1 (input .75).

- **In:** Hybrid input for the argument.
- **Out:** Hybrid output for the value.



For each input event, the sine and cosine are computed. An input value of 1.0 corresponds to a full period of the sin and cos functions (i.e. 360 degrees).

- **In:** Hybrid input for the argument to the sine function. A value of 1.0 corresponds to one period of the sine function (360 degrees). Typ. Range: [-1 ... 1].
- **Sin:** Hybrid output for the sine of the input value. Range: [-1 ... 1].
- **Cos:** Hybrid output for the cosine of the input value. Range: [-1 ... 1].

Arcsin**Math**

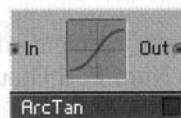
The ArcSin module calculates the inverse sine function. (The arc sine of x is the number between 0 and 1 whose sine is x .) Since the sine function's output range is -1 to 1, the arc sine function is only valid for inputs within that range. For arguments < -1 the ArcCos returns -0.25 and for arguments > 1 , it returns 0.25.

- **In:** Hybrid input for the argument.
- **Out:** Hybrid output for the arc sine of the input.

Arccos**Math**

The ArcCos module calculates the inverse cosine function. (The arc cosine of x is the number between 0 and 1 whose cosine is x .) Since the cosine function's output range is -1 to 1, the arc cosine function is only valid for inputs within that range. For arguments < -1 the ArcCos returns 0.5 and for arguments > 1 , it returns 0.

- **In:** Hybrid input for the argument.
- **Out:** Hybrid output for the arc cosine of the input.

Arctan**Math**

The ArcTan module calculates the inverse tangent function. (The tangent is the sine divided by the cosine.) The output of ArcTan module ranges from -0.25 to 0.25, which corresponds to -90 to 90 degrees.

- **In:** Hybrid input for the argument.
- **Out:** Hybrid output for the arc tangent of the input.

Signal Path

Signal Path modules allow both control and audio data to be flexibly routed in Reaktor structures. These include mixers, input and output selectors, remote-control switches (called Relays), crossfaders, and panners.

Selector/Scanner

Signal Path



Selector/Scanner. The inputs are scanned by sweeping the value at the **Pos** input. When **Pos** is an integer, you get just one input signal, otherwise a mix of two inputs. The output signal is obtained by crossfading between the two inputs whose index is closest to the Value at the **Pos** input. The crossfading is done according to crossfading mode specified in the Properties.

When **Wrap** mode is selected in the Properties, **Pos** wraps around so that Max+1 is the same as 0, Max+2 is 1 etc.

The Selector/Scanner makes nice effects when the inputs are fed from the Multitap Delay and the scan position is controlled by a Ramp Oscillator.

The module has a dynamic in-port management. The number of in-ports can be defined with **Min Num Port Groups** on the **Function** page of the Properties.

- **Pos:** Hybrid input for selecting the input(s) to scan. **Pos** = 0 selects **In0**, **Pos** = 1 selects **In1**, **Pos** = 0.5 gives a mix of **In0** and **In1**. Typ. Range: [0 ... Max]
- **In 0...Max:** Hybrid inputs for the signals to be scanned.
- **Out:** Output for the scanned signal.

Relay 1,2

Signal Path



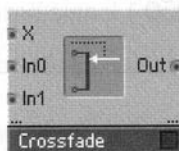
Relay. The upper input is connected to the output if **Ctl** > 0. Otherwise the lower input is connected to the output. If the lower input is not present a zero signal is produced at the output.

The module can have one or two in-ports. The number of in-ports can be defined with **Min Num Port Groups** on the **Function** page of the Properties.

- **Ctl:** Control input for selecting a signal input.
- **In:** Signals input(s).
- **Out:** Output for the selected signal.

Crossfade

Signal Path



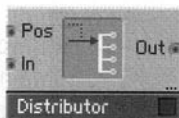
Crossfade module for two signals. The output signal is mixed from the both input signals **In0** and **In1**. The mix ratio is controlled by the **X** input.

The module has a dynamic port management. The number of in-ports pairs (**In0** and **In1**) and out-ports can be defined with **Min Num Port Groups** on the **Function** page of the Properties.

- **X**: Hybrid control input for the mixing ratio. Value between 0 (**Out** = **In1**) and 1 (**Out** = **In2**).
- **In1**, **In2**: Hybrid inputs for the two signals to be mixed.
- **Out**: Hybrid output for the mixed signal (**Out** = (1 - X) **In1** + X **In2**).

Distributor/Panner

Signal Path



Distributor/Panner. The signal at the input will be connected/panned to the output(s) selected by the Pos input. When Pos is an integer, the signal is connected to just one input signal, otherwise you get a 'panning' between two inputs. The panning is done according to panning mode specified in the Properties. When Wrap mode is selected in the Properties, Pos wraps around so that Max+1 is the same as 0, Max+2 is 1 etc..

The module has a dynamic out-port management. The number of out-ports can be defined with **Min Num Port Groups** on the **Function** page of the Properties.

- **Pos**: Input for selecting the thru output.
- **In**: Signal input.
- **Out**: Output for the input signal.

Stereo Pan

Signal Path



Left-right panner. By changing the signal level at the two outputs the input signal is positioned in the stereo field. The sum of the **L** and **R** output values is always exactly twice the input value, i.e. the input is split across the two outputs at a variable ratio.

The module has a dynamic port management. The number of in-ports and out-port pairs (**L** and **R**) can be defined with **Min Num Port Groups** on the **Function** page of the Properties.

- **Pan**: Control input for the left-right position. Value -1 = left, 0 = center, 1 = right.
- **In**: Hybrid signal input for the signal to be positioned in stereo.
- **L**: Hybrid signal output for the left channel signal.
- **R**: Hybrid signal output for the right channel signal.

Amp/Mixer

Signal Path



Amp/Mixer for an adjustable number of input signals. The input signals are amplified (or attenuated) by their respective amounts at the **Lvl**-inputs (in dB) and then summed to form the output.

The module has a dynamic in-port management. The number of in-port pairs (**Lvl** and **In**) can be defined with **Min Num Port Groups** on the **Function** page of the Properties.

- **Lvl**: Logarithmic control input for controlling the gain, value in dB.
- **In**: Audio input for the signal to be amplified.
- **Out**: Output for the mixed signal ($\text{Out} = \text{In1} \cdot 10^{\text{Lvl1}/20} + \text{In2} \cdot 10^{\text{Lvl2}/20}$ etc.).

Stereo Amp/Mixer

Signal Path



Amplifier for an adjustable number of audio signals with integrated left-right panner. The input signals are amplified (or attenuated) by the set amount at the **Lvl** inputs (in dB). By changing the signal level at the two outputs the input signals are positioned in the stereo field.

The module has a dynamic in-port management. The number of in-port groups (**Lvl**, **Pan** and **In**) can be defined with **Min Num Port Groups** on the **Function** page of the Properties.

- **Lvl**: Logarithmic input for controlling the gain, value in dB. When the value is negative, the output signal is smaller than the input signal.
- **Pan**: Control input for the left-right position. Value -1 = left, 0 = center, 1 = right.
- **In**: Audio signal input for the signal to be amplified and positioned in stereo.
- **L**: Audio signal output for the amplified left channel signal.
- **R**: Audio signal output for the amplified right channel signal.

Oscillator

Reaktor uses the term oscillator for a broad range of signal generators. In fact, everything signal generating process that doesn't involve playing samples is called an oscillator. That includes the usual, single-cycle waveform generators (sine, pulse, sawtooth, and so on) as well as impulse, step, noise, and table-driven generators.

All of REAKTOR's oscillators can run at any frequency, from 0 Hz (standstill) through the entire audio range right up to the limit set by the sample rate. Like this they are all equally suited for use as LFOs or for producing sound waves. When using an oscillator as an LFO to modulate another module's input which is of the type that only accepts events (e.g. P as opposed to F) you need to insert an A to E (perm) module for conversion.

Sawtooth

Oscillator



Oscillator for sawtooth waveform with logarithmic pitch control and linear amplitude modulation.

- **P:** Logarithmic event input for controlling the pitch (oscillator frequency). Value in semitones ($69 = 440 \text{ Hz}$).
- **A:** Audio input for controlling the amplitude. The output signal moves between $+A$ and $-A$.
- **Out:** Audio signal output for the sawtooth waveform.

Saw FM

Oscillator

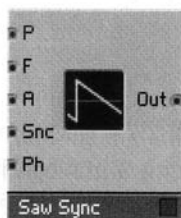


Oscillator for sawtooth waveform with logarithmic pitch control, linear frequency modulation (FM) and linear amplitude modulation.

- **P:** Logarithmic event input for controlling the pitch (oscillator frequency). Value in semitones ($69 = 440 \text{ Hz}$).
- **F:** Audio input for linear frequency modulation. Value in Hz. **P** and **F** together determine the oscillator frequency.
- **A:** Audio input for controlling the amplitude. The output signal moves between $+A$ and $-A$.
- **Out:** Audio signal output for the sawtooth waveform.

Saw Sync

Oscillator



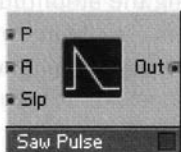
Oscillator for sawtooth waveform with synchronization, logarithmic pitch control, linear frequency modulation (FM) and linear amplitude modulation.

Whenever the synchronization signal leaves zero to positive values (rising edge) the phase of the oscillator is reset to a position specified by the phase input.

- **P:** Logarithmic event input for controlling the pitch (oscillator frequency). Value in semitones ($69 = 440$ Hz).
- **F:** Audio input for linear frequency modulation. Value in Hz. **P** and **F** together determine the oscillator frequency.
- **A:** Audio input for controlling the amplitude. The output signal moves between $+A$ and $-A$.
- **Snc:** Audio input for controlling synchronization of the waveform. A rising edge restarts the oscillator.
- **Ph:** Input for specifying the phase (position in the waveform) to which the oscillator is reset when synchronization occurs.
- **Out:** Audio signal output for the sawtooth waveform.

Saw Pulse

Oscillator

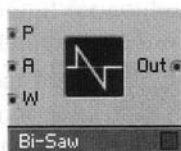


Oscillator for variable sawtooth/needle-pulse waveform with logarithmic pitch control and linear amplitude modulation. The slope of the ramp is variable and with it the shape of the waveform can be changed from a normal sawtooth to a short triangular pulse.

- **P:** Logarithmic event input for controlling the pitch (oscillator frequency). Value in semitones ($69 = 440$ Hz).
- **A:** Audio input for controlling the amplitude. The output signal moves between $+A$ and $-A$.
- **Slp:** Audio input for controlling the slope of the waveform. A value of 0 produces a normal sawtooth, a larger value shortens the ramp to a short spike (needle).
- **Out:** Audio signal output for the sawtooth/pulse waveform.

Bi-Saw

Oscillator



Oscillator for bipolar sawtooth waveform with zero-phase. With logarithmic pitch control, pulse width modulation (PWM) and linear amplitude modulation.

- **P:** Logarithmic event input for controlling the pitch (oscillator frequency). Value in semitones ($69 = 440$ Hz).
- **A:** Audio input for controlling the amplitude. The output signal moves between $+A$ and $-A$.
- **W:** Audio input for controlling the pulse width (PWM). Range of values is 0 to about 6. Normal sawtooth-wave at $W = 0$, larger W results in a shorter pulse and a longer zero-phase.
- **Out:** Audio signal output for the bipolar sawtooth waveform with zero-phase.

Triangle

Oscillator



Oscillator for symmetric triangle waveform with logarithmic pitch control and linear amplitude modulation.

- **P**: Logarithmic event input for controlling the pitch (oscillator frequency). Value in semitones ($69 = 440 \text{ Hz}$).
- **A**: Audio input for controlling the amplitude. The output signal moves between $+A$ and $-A$.
- **Out**: Audio signal output for the triangle waveform.

Tri FM

Oscillator

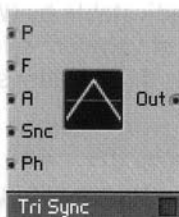


Oscillator for symmetric triangle waveform with logarithmic pitch control, linear frequency modulation (FM) and linear amplitude modulation.

- **P**: Logarithmic event input for controlling the pitch (oscillator frequency). Value in semitones ($69 = 440 \text{ Hz}$).
- **F**: Audio input for linear frequency modulation. Value in Hz. **P** and **F** together determine the oscillator frequency.
- **A**: Audio input for controlling the amplitude. The output signal moves between $+A$ and $-A$.
- **Out**: Audio signal output for the triangle waveform.

Tri Sync

Oscillator



Oscillator for symmetric triangle waveform with synchronization, logarithmic pitch control, linear frequency modulation (FM) and linear amplitude modulation.

Whenever the synchronization signal leaves zero to positive values (rising edge) the phase of the oscillator is reset to a position specified by the phase input.

- **P:** Logarithmic event input for controlling the pitch (oscillator frequency). Value in semitones ($69 = 440 \text{ Hz}$).
- **F:** Audio input for linear frequency modulation. Value in Hz. **P** and **F** together determine the oscillator frequency.
- **A:** Audio input for controlling the amplitude. The output signal moves between $+A$ and $-A$.
- **Snc:** Audio input for controlling synchronization of the waveform. A rising edge restarts the oscillator.
- **Ph:** Input for specifying the phase (position in the waveform) to which the oscillator is reset when synchronization occurs.
- **Out:** Audio signal output for the triangle waveform.

Tri/Par Symm

Oscillator



Oscillator for variable triangle and parabolic (near to sine) signals, with logarithmic pitch (P) control and linear amplitude (A) modulation. The symmetry is adjustable by (W). Setting the symmetry (with W) allows a range of waveforms from symmetric with few harmonics to asymmetric with a broader spectrum.

- **P:** Logarithmic event input for controlling the pitch (oscillator frequency). Value in semitones ($69 = 440$ Hz).
- **A:** Audio input for controlling the amplitude. The output signal moves between $+A$ and $-A$.
- **W:** Event input for controlling the symmetry of the waveform. A value of -1 produces a rising-ramp sawtooth, 0 produces a symmetrical triangle and $+1$ a falling-ramp sawtooth.
- **Tri:** Audio signal output for the triangle signal.
- **Par:** Audio signal output for the parabolic signal.

Parabol

Oscillator

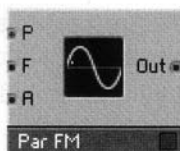


Oscillator for parabolic waveform with logarithmic pitch control and linear amplitude modulation. The waveform consists of two halves that are each a section of a parabola. The oscillator sounds like a sine wave with some added odd numbered overtones at very low level. In many cases it can be used as replacement for a sine generator with less computational load.

- **P:** Logarithmic event input for controlling the pitch (oscillator frequency). Value in semitones ($69 = 440$ Hz).
- **A:** Audio input for controlling the amplitude. The output signal moves between $+A$ and $-A$.
- **Out:** Audio signal output for the parabolic waveform.

Par FM

Oscillator

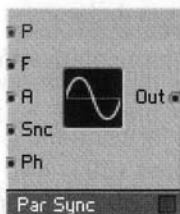


Oscillator for parabolic waveform with logarithmic pitch control, linear frequency modulation (FM) and linear amplitude modulation. The oscillator sounds like a sine wave with some added odd numbered overtones at very low level. In many cases it can be used as replacement for a sine generator with less computational load.

- **P:** Logarithmic event input for controlling the pitch (oscillator frequency). Value in semitones ($69 = 440 \text{ Hz}$).
- **F:** Audio input for linear frequency modulation. Value in Hz. **P** and **F** together determinethe oscillator frequency.
- **A:** Audio input for controlling the amplitude. The output signal moves between $+A$ and $-A$.
- **Out:** Audio signal output for the parabolic waveform.

Par Sync

Oscillator



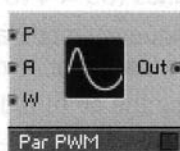
Oscillator for parabolic waveform with synchronization, logarithmic pitch control, linear frequency modulation (FM) and linear amplitude modulation.

Whenever the synchronization signal leaves zero to positive values (rising edge) the phase of the oscillator is reset to a position specified by the phase input.

- **P:** Logarithmic event input for controlling the pitch (oscillator frequency). Value in semitones ($69 = 440 \text{ Hz}$).
- **F:** Audio input for linear frequency modulation. Value in Hz. **P** and **F** together determine the oscillator frequency.
- **A:** Audio input for controlling the amplitude. The output signal moves between $+A$ and $-A$.
- **Snc:** Audio input for controlling synchronization of the waveform. A rising edge restarts the oscillator.
- **Ph:** Input for specifying the phase (position in the waveform) to which the oscillator is reset when synchronization occurs.
- **Out:** Audio signal output for the parabolic waveform.

Par PWM

Oscillator



Oscillator for variable parabolic waveform with logarithmic pitch control and linear amplitude modulation. The ratio between the length of the upper and lower parts of the curve can be controlled, and with it the waveform can be changed from a normal symmetric parabolic wave to a simple parabola.

This variable waveform is also particularly effective when used as an LFO.

- **P:** Logarithmic event input for controlling the pitch (oscillator frequency). Value in semitones ($69 = 440 \text{ Hz}$).
- **A:** Audio input for controlling the amplitude. The output signal moves between $+A$ and $-A$.
- **W:** Event input for controlling the symmetry of the waveform. A value of -1 produces parabolas open downward, 0 a normal symmetric parabolic wave and $+1$ parabolas open upward.
- **Out:** Audio signal output for the variable parabolic waveform.

Sine

Oscillator



Oscillator for pure sine waveform with logarithmic pitch control and linear amplitude modulation.

If the sine tone does not need to be completely pure, the parabolic oscillator makes a good replacement with less computational load.

- **P:** Logarithmic event input for controlling the pitch (oscillator frequency). Value in semitones ($69 = 440$ Hz).
- **A:** Audio input for controlling the amplitude. The output signal moves between $+A$ and $-A$.
- **Out:** Audio signal output for the sine waveform.

Sine FM

Oscillator



Oscillator for pure sine waveform with logarithmic pitch control, linear frequency modulation (FM) and linear amplitude modulation.

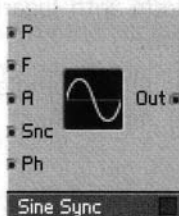
If the sine tone does not need to be completely pure, the parabolic oscillator makes a good replacement with less computational load.

- **P:** Logarithmic event input for controlling the pitch (oscillator frequency). Value in semitones ($69 = 440$ Hz).
- **F:** Audio input for linear frequency modulation. Value in Hz. **P** and **F** together determine the oscillator frequency.

- **A:** Audio input for controlling the amplitude. The output signal moves between $+A$ and $-A$.
- **Out:** Audio signal output for the sine waveform.

Sine Sync

Oscillator



Oscillator for pure sine waveform with synchronization, logarithmic pitch control, linear frequency modulation (FM) and linear amplitude modulation.

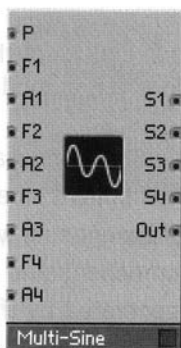
Whenever the synchronization signal leaves zero to positive values (rising edge) the phase of the oscillator is reset to a position specified by the phase input.

If the sine tone does not need to be completely pure, the parabolic oscillator makes a good replacement with less computational load.

- **P:** Logarithmic event input for controlling the pitch (oscillator frequency). Value in semitones ($69 = 440$ Hz).
- **F:** Audio input for linear frequency modulation. Value in Hz. **P** and **F** together determine the oscillator frequency.
- **A:** Audio input for controlling the amplitude. The output signal moves between $+A$ and $-A$.
- **Snc:** Audio input for controlling synchronization of the waveform. A rising edge restarts the oscillator.
- **Ph:** Input for specifying the phase (position in the waveform) to which the oscillator is reset when synchronization occurs.
Ph = 0: Phase = 0 deg (middle of rising slope), **Ph = 0.5:** Phase = 180 deg (middle of falling slope), **Ph = 1:** Phase = 360 deg (same as 0 deg).
- **Out:** Audio signal output for the sine waveform.

Multi-Sine

Oscillator



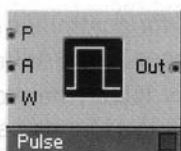
Oscillator for Additive Synthesis. A waveform is built up in its harmonics by layering individual sine waves. For each component, the amplitude **A** and the frequency multiple **F** can be set.

- **P**: Logarithmic event input for controlling the pitch (oscillator frequency). Value in semitones (69 = 440 Hz).
- **F1**: Input for the frequency factor (harmonic number) of the first sine wave. Scale: multiple of fundamental frequency. Typ. Range: [0 ... 20].
- **A1**: Input for linear amplitude control of the first sine wave. Can also be used for tremolo and ring modulation. Typ. Range: [0 ... 1].
- **F2**: Input for the frequency factor (harmonic number) of the second sine wave. Scale: multiple of fundamental frequency. Typ. Range: [0 ... 20].
- **A2**: Input for linear amplitude control of the second sine wave. Can also be used for tremolo and ring modulation. Typ. Range: [0 ... 1].
- **F3**: Input for the frequency factor (harmonic number) of the third sine wave. Scale: multiple of fundamental frequency. Typ. Range: [0 ... 20].
- **A3**: Input for linear amplitude control of the third sine wave. Can also be used for tremolo and ring modulation. Typ. Range: [0 ... 1].

- **F4:** Input for the frequency factor (harmonic number) of the fourth sine wave. Scale: multiple of fundamental frequency. Typ. Range: [0 ... 20].
- **A4:** Input for linear amplitude control of the fourth sine wave. Can also be used for tremolo and ring modulation. Typ. Range: [0 ... 1].
- **S1:** Output for the first component sine wave.
- **S2:** Output for the second component sine wave.
- **S3:** Output for the third component sine wave.
- **S4:** Output for the fourth component sine wave.
- **Out:** Output for the signal generated by the oscillator by adding all the sine waves.

Pulse

Oscillator

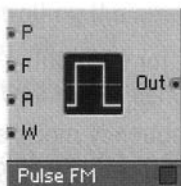


Oscillator for pulse wave with logarithmic pitch control, pulse width modulation (PWM) and linear amplitude modulation.

- **P:** Logarithmic event input for controlling the pitch (oscillator frequency). Value in semitones (69 = 440 Hz).
- **A:** Audio input for controlling the amplitude. The output signal moves between +A and -A.
- **W:** Audio input for controlling the pulse width (PWM). Range of values is -1 to 1. Symmetric waveform (square wave) 50:50 at **W** = 0, Lo:Hi-ratio 33:66 at -0.33, 66:33 at 0.33, 75:25 at 0.5, 90:10 at 0.8. $\text{Lo:Hi} = (1 + \mathbf{W}) / (1 - \mathbf{W})$.
- **Out:** Audio signal output for the pulse waveform.

Pulse FM

Oscillator

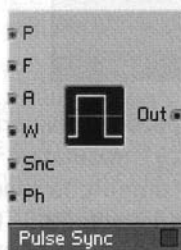


Oscillator for pulse wave with logarithmic pitch control, linear frequency modulation (FM), pulse width modulation (PWM) and linear amplitude modulation.

- **P:** Logarithmic event input for controlling the pitch (oscillator frequency). Value in semitones (69 = 440 Hz).
- **F:** Audio input for linear frequency modulation. Value in Hz. **P** and **F** together determinethe oscillator frequency.
- **A:** Audio input for controlling the amplitude. The output signal moves between +A and -A.
- **W:** Audio input for controlling the pulse width (PWM). Range of values is -1 to 1. Symmetric waveform (square wave) 50:50 at **W** = 0, Lo:Hi-ratio 33:66 at -0.33, 66:33 at 0.33, 75:25 at 0.5, 90:10 at 0.8. $\text{Lo:Hi} = (1 + W) / (1 - W)$.
- **Out:** Audio signal output for the pulse waveform.

Pulse Sync

Oscillator



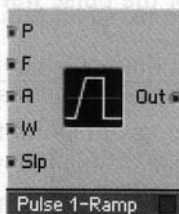
Oscillator for pulse wave with synchronization, logarithmic pitch control, linear frequency modulation (FM), pulse width modulation (PWM) and linear amplitude modulation.

Whenever the synchronization signal leaves zero to positive values (rising edge) the phase of the oscillator is reset to a position specified by the phase input.

- **P:** Logarithmic event input for controlling the pitch (oscillator frequency). Value in semitones ($69 = 440 \text{ Hz}$).
- **F:** Audio input for linear frequency modulation. Value in Hz. **P** and **F** together determine the oscillator frequency.
- **A:** Audio input for controlling the amplitude. The output signal moves between $+A$ and $-A$.
- **W:** Audio input for controlling the pulse width (PWM). Range of values is -1 to 1 . Symmetric waveform (square wave) 50:50 at $W = 0$, Lo:Hi-ratio 33:66 at -0.33 , 66:33 at 0.33 , 75:25 at 0.5 , 90:10 at 0.8 . $\text{Lo:Hi} = (1 + W) / (1 - W)$.
- **Snc:** Audio input for controlling synchronization of the waveform. A rising edge restarts the oscillator.
- **Ph:** Input for specifying the phase (position in the waveform) to which the oscillator is reset when synchronization occurs.
- **Out:** Audio signal output for the pulse waveform.

Pulse 1-ramp

Oscillator



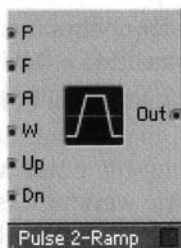
Oscillator for rectangular trapezoid waveform with logarithmic pitch control, linear frequency modulation (FM), pulse width modulation (PWM) and linear amplitude modulation. The waveform is a mixture of pulse and sawtooth: The rising edge of a pulse wave can be flattened and its slope adjusted. The falling edge is vertical.

- **P:** Logarithmic event input for controlling the pitch (oscillator frequency). Value in semitones ($69 = 440 \text{ Hz}$).

- **F**: Audio input for linear frequency modulation. Value in Hz. **P** and **F** together determinethe oscillator frequency.
- **A**: Audio input for controlling the amplitude. The output signal moves between +A and -A.
- **W**: Audio input for controlling the pulse width (PWM). Range of values is -1 to 1.
- **Slp**: Audio input for controlling the slope of the rising edge of the waveform. When **Slp** is zero (or when the input is not connected) the waveform does not rise and the output is always zero, i.e. there is no sound. Typical range of values: 1 ... 20
- **Out**: Audio signal output for the trapezoid waveform.

Pulse 2-ramp

Oscillator



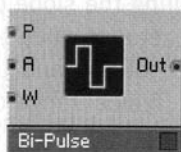
Oscillator for trapezoid waveform with logarithmic pitch control, linear frequency modulation (FM), pulse width modulation (PWM) and linear amplitude modulation. The waveform is a mixture of pulse, sawtooth and triangle: Both edges of a pulse wave can be flattened and their slope adjusted.

- **P**: Logarithmic event input for controlling the pitch (oscillator frequency). Value in semitones (69 = 440 Hz).
- **F**: Audio input for linear frequency modulation. Value in Hz. **P** and **F** together determinethe oscillator frequency.
- **A**: Audio input for controlling the amplitude. The output signal moves between +A and -A.
- **W**: Audio input for controlling the pulse width (PWM). Range of values is -1 to 1.
- **Up**: Audio input for controlling the slope of the rising edge of the waveform.

- **Dn:** Audio input for controlling the slope of the falling edge of the waveform.
- **Out:** Audio signal output for the trapezoid waveform.

Bi-Pulse

Oscillator



Oscillator for bipolar pulse wave with zero-phase. With logarithmic pitch control, pulse width modulation (PWM) and linear amplitude modulation.

- **P:** Logarithmic event input for controlling the pitch (oscillator frequency). Value in semitones ($69 = 440$ Hz).
- **A:** Audio input for controlling the amplitude. The output signal moves between $+A$ and $-A$.
- **W:** Audio input for controlling the pulse width (PWM). Range of values is 0 to 1. Symmetric waveform (square wave) 50:50 at $W = 0$, larger W results in a shorter pulse and longer zero-phase.
- **Out:** Audio signal output for the bipolar pulse waveform.

Impulse

Oscillator



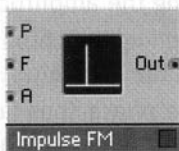
Oscillator for impulse train signal with logarithmic pitch control (P) and linear amplitude modulation (A).

- **P:** Logarithmic event input for controlling the pitch (oscillator frequency). Value in semitones ($69 = 440$ Hz).
- **A:** Audio input for controlling the amplitude.

- **Out:** Audio signal output for the impulse waveform.

Impulse FM

Oscillator

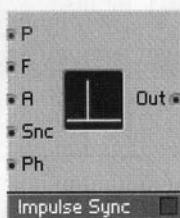


Oscillator for impulse train signal with logarithmic pitch control (P), linear frequency modulation (F) and linear amplitude modulation (A).

- **P:** Logarithmic event input for controlling the pitch (oscillator frequency). Value in semitones ($69 = 440$ Hz).
- **F:** Audio input for linear frequency modulation. Value in Hz. **P** and **F** together determine the oscillator frequency.
- **A:** Audio input for controlling the amplitude.
- **Out:** Audio signal output for the impulse waveform.

Impulse Sync

Oscillator



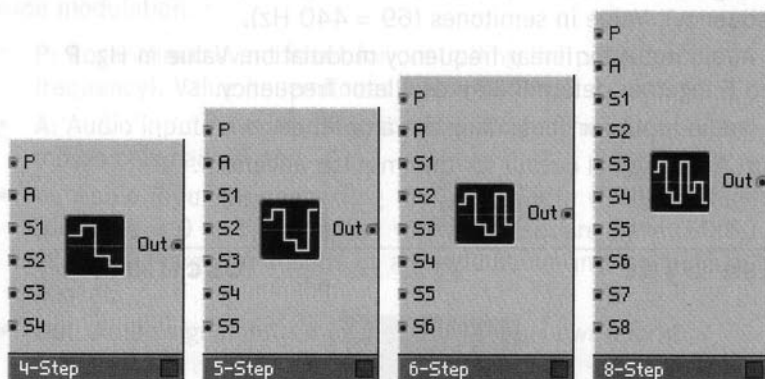
Oscillator for synchronizable impulse train signal with logarithmic pitch control (P), linear frequency modulation (F), linear amplitude modulation (A) and Sync input (Snc).

Whenever the synchronization signal leaves zero to positive values (rising edge) the phase of the oscillator is reset to a position specified by the phase input.

- **P**: Logarithmic event input for controlling the pitch (oscillator frequency). Value in semitones (69 = 440 Hz).
- **F**: Audio input for linear frequency modulation. Value in Hz. **P** and **F** together determine the oscillator frequency.
- **A**: Audio input for controlling the amplitude.
- **Snc**: Audio input for controlling synchronization of the waveform. A rising edge restarts the oscillator.
- **Ph**: Input for specifying the phase (position in the waveform) to which the oscillator is reset when synchronization occurs.
- **Out**: Audio signal output for the impulse waveform.

Multi-Step

Oscillator



4-Step

Oscillator

Oscillator for 4-step waveform with logarithmic pitch control and linear amplitude modulation. The level of each step can be set independent of the others.

- **P**: Logarithmic event input for controlling the pitch (oscillator frequency). Value in semitones (69 = 440 Hz).
- **A**: Audio input for controlling the amplitude. The output signal moves between +A and -A.
- **S1**: Audio input for controlling the level of the first step.
- **S2**: Audio input for controlling the level of the second step.

- **S3:** Audio input for controlling the level of the third step.
- **S4:** Audio input for controlling the level of the fourth step.
- **Out:** Audio signal output for the step waveform.

5-Step Oscillator

Like 4-Step but with 5 steps.

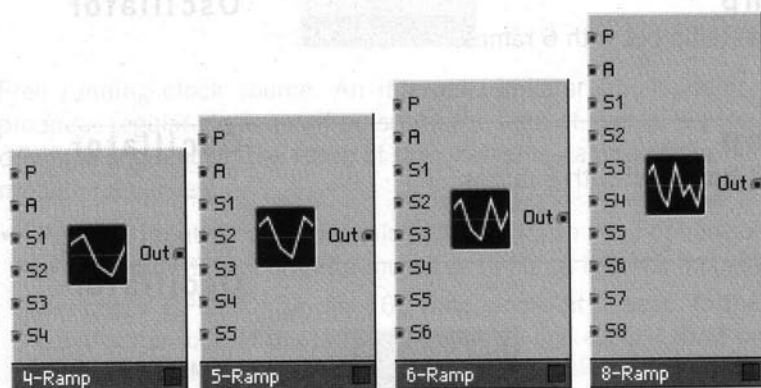
6-Step Oscillator

Like 4-Step but with 6 steps.

8-Step Oscillator

Like 4-Step but with 8 steps.

Multi-Ramp Oscillator



4-Ramp Oscillator

Oscillator for 4-ramp waveform with logarithmic pitch control and linear amplitude modulation. The level of each of the breakpoints which are connected by ramps can be set independent of the others.

- **P**: Logarithmic event input for controlling the pitch (oscillator frequency). Value in semitones (69 = 440 Hz).
- **A**: Audio input for controlling the amplitude. The output signal moves between +A and -A.
- **S1**: Event input for controlling the level of the first breakpoint.
- **S2**: Event input for controlling the level of the second breakpoint.
- **S3**: Event input for controlling the level of the third breakpoint.
- **S4**: Event input for controlling the level of the fourth breakpoint.
- **Out**: Audio signal output for the ramp waveform.

5-Ramp

Like 4-Ramp but with 5 ramps.

Oscillator

6-Ramp

Like 4-Ramp but with 6 ramps.

Oscillator

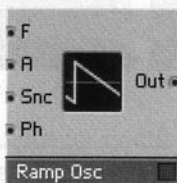
8-Ramp

Like 4-Ramp but with 8 ramps.

Oscillator

Ramp

Oscillator

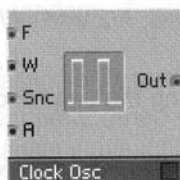


Oscillator to produce a ramp waveform, typically used as a control signal, for example to use an Audio Table module as a waveform oscillator. The signal ramps up from 0 to A and then resets instantly. There is no anti-aliasing or other complications.

- **F:** Audio input for control of the oscillator frequency in Hz. To control the pitch in semitones, use an Event Expon. (F) module.
- **A:** Input for controlling the amplitude of the ramp signal. Typ. Value: 1
- **Snc:** Audio input for controlling synchronization of the waveform. A rising edge resets the oscillator to **Ph**.
- **Ph:** Audio input for the synchronization phase. When the oscillator is synchronised, its output jumps to this value times A. Typ. Range: 0...1
- **Out:** Audio output for the ramp signal. Range 0...A.

Clock Oscillator

Oscillator



Free running clock source. An internal oscillator (monophonic) produces regular clock on/off pulses in the form of events, e.g. for driving a sequencer. The value of the on-events can be set in the module properties.

- **F:** Event input for control of clock frequency in Hz. For control of Tempo in BPM, compute the value in Hz as follows: $\text{clocks-per-beat} \times \text{BPM}/60$. So, for 16th note clocks at 4 beats to the bar (that is four 16ths per beat), you get $F = 4/60 \times \text{BPM}$ or $0.0667 \times \text{BPM}$.
- **W:** Event input for control of pulse width, i.e. the ratio of the duration of the on-period to the off-period. Range of values is -1 to 1. Symmetric waveform (equal duration on and off) at $W = 0$; Lo:Hi = $(1 + W) / (1 - W)$.
- **Snc:** Event input for synchronization of the waveform. A positive event synchronizes the clock source.

- **A:** Input for controlling the amplitude of the clock signal. You must connect something here, otherwise only zero-value events will be produced. Typ. Value:1.
- **Out:** Event output for the clock signal, alternating between on-value and zero.

Noise

Oscillator



Noise generator. Produces white noise, i.e. a random signal containing equal amounts of all possible frequency components. The signal consists of only two values, $A/2$ and $-A/2$, but which appear in a random sequence.

- **A:** Audio input for controlling the amplitude of the output signal.
- **Out:** Audio signal output for the noise waveform.

Random

Oscillator



Random value generator. Produces step waveform where the level of each step is random in the given interval with equal distribution. It works like a noise generator with uniform distribution followed by a sample & hold circuit clocked at a regular frequency.

- **P:** Logarithmic event input for controlling the step rate (sample & hold frequency). Value in semitones ($69 = 440 \text{ Hz}$).
- **A:** Audio input for controlling the amplitude. The output signal is a random value between $+A$ and $-A$.
- **Out:** Audio signal output for the random step waveform.

Geiger

Oscillator



Generates events at random intervals, much like a Geiger Counter radiation particle detector. The average rate of events can be controlled at the **P** input. **Rnd** controls the randomness of the event timing.

- **P:** Control input for logarithmic control of the average rate or density of the randomly occurring output events. Typ. range: [-50 ... 50]
- **Rnd:** Control input for the randomness of the event distribution: 0 = completely random, 1 = completely regular. Typ. range: [0 ... 1]
- **Out:** Audio output for randomly timed clicks.
- **Out:** Output for the randomly timed events. Use to trigger an envelope (**G**), for example.

Samplers

If it generates audio and it's not an oscillator, it's a sampler. Reaktor's samplers include basic sample players as well as sophisticated sample processors for granular synthesis, pitch and time shifting, and beat slicing. There's even one for picking out individual samples by number.

Depending on the selected Sampler module the following settings are available in the Properties.

Properties - Function page

- **Waveform-Button:** A click on the button with the waveform icon opens the Sample Map Editor.
- **Embed Samples In Ensemble** allows to store your samples with the saved Ensemble.

- **No Stereo** stops stereo playback (Only the left channel is used). Activation reduces the processor load.
- The **Quality** drop-down menu sets the playback quality of the sample in three options (**Poor**, **Good** and **Excellent**). Higher quality increases the processor load. The term “quality” refers to the absence of noise. Naturally, noise may actually improve the musical “quality” of a sample.
- **Oscil. Mode** places the module into oscillator mode.

Samplers in oscillator mode assume that the samples used contain waveforms or WaveSets. A waveform is a representation of a single vibration. Through repeated playback it is possible to recreate periodic vibration. The module then becomes a digital oscillator. Sampler modules in oscillator mode interpret the values at the **P** input in the same way as oscillators in REAKTOR – as MIDI note numbers – and produce periodical vibrations at the corresponding pitch.

In oscillator mode the **Sampler** and **Sampler FM** modules interpret the entire sample as one vibration. For example to produce a sound at 440 Hz, the entire sample is played 440 times per second, independent of the duration of the sample.

In waveform mode **Sampler Loop** interprets the loop length as vibrations. The loop length is read from the sound file, but can be changed at the **LL** input of the module (see the Module Reference). Therefore a sample can contain numerous waveforms, also known as WaveSets. Assuming a waveform contains 100 cycles, then 100 such waveforms fit into a sample the size of 10,000 cycles. The first waveform covers cycles 0-99, the second 100-199 etc. If the loop length sets the vibration of a waveform, the position of the loop logically sets the choice of waveforms from the WaveSet. **Sampler Loop** uses the **LS** input to control the loop start point. In waveform mode the values at this input are quantized, so that glitch free playback between waveforms is achieved. The position in a WaveSet can be easily controlled by velocity, etc. Obviously such WaveSet samples need to be either created or generated from a sample. The library contains many WaveSet ex-

amples. **Sampler Loop** integrates WaveSet Synthesis with REAKTOR's existing synthesis capabilities. In this way, WaveSet Synthesis is now available for the first time in combination with FM synthesis.

Properties - Appearance page

- **Picture:** Tick this checkbox to see a waveform display for the sampler module in the control panel.
- **Size X/Size Y:** Allows setting the size of the waveform display for the sampler module in pixel.
- **Scroll Bar:** Tick this checkbox to see a scroll bar under the waveform display for navigation purposes. The Scroll bar allows moving (drag the scroll bar in the middle) and zooming (drag the scroll bar at one of its ends to the left or right).

Sampler

Sampler



This is a player used for the polyphonic and transposed playback of a sample or sample map.

Sample management is carried out in the **Sample Map Editor**.

If **Loop** is activated, the whole sample is repeated continuously and is restarted by a positive event at the trigger input. If **Loop** is deactivated and a positive trigger event arrives at the trigger input, the sample will run from start to finish or, if the direction parameter is set to **Backward**, will run from the end to the beginning.

If **Oscillator Mode** is active, the playback rate will be adjusted to suit the length of the current sample in order to produce the correct pitch when operating as a waveform oscillator.

- **P**: Logarithmic control input for the playback rate (pitch) and for selecting a sample from the loaded sample map. If **P** is equal to the **Root Key** of the current sample, the sample will be played back at its original pitch.
- **Trig**: A positive event at this input triggers the sample to be played back from the beginning again.
- **A**: Control input for the output amplitude.
- **Out**: The sample player's output.

Sampler FM

Sampler



This is a player used for the polyphonic and transposed playback of a sample or a sample map. The **F** input and the starting point control input (**St**) allow you to manipulate sample playback.

Sample management is carried out in the **Sample Map Editor**.

If **Loop** is activated, the whole sample is repeated continuously; a positive event at the trigger input will make playback jump back to the starting point that has been set via the **St** input. If **Loop** is deactivated and a positive trigger event arrives at the trigger input, the sample will run from the starting point to the end or, if the direction parameter is set to **Backward**, will run from the starting point to the beginning.

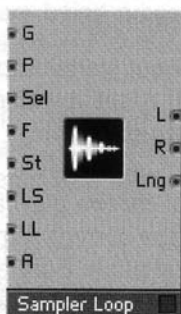
If **Oscillator Mode** is active, the playback rate will be adjusted to suit the length of the current sample in order to produce the correct pitch when operating as a waveform oscillator.

- **P**: Logarithmic control input for the playback rate (pitch) and for selecting a sample from the loaded sample map. If **P** is equal to the **Root Key** of the current sample, the sample will be played back at its original pitch.

- **F:** Linear control input for modulating the playback rate. The effect achieved is frequency modulation – the same as for the oscillator modules in REAKTOR. If a large negative value is present, the sample is played backwards.
- **St:** Control input for the starting point to be used when the next trigger occurs. The position is set in milliseconds from the start of the sample.
- **Trig:** A positive event at this input triggers the sample to be played from the beginning again.
- **A:** Control input for the output amplitude.
- **Out:** The sample player's output.
- **Lng:** Polyphonic event output for the length of the current sample in milliseconds.

Sampler Loop

Sampler



Ⓢ This is a universal player for the polyphonic and transposed playback of mono or stereo samples, sample maps and WaveSets.

Sample management is carried out in the **Sample Map Editor**.

After a positive **Gate** event, sample playback starts from the starting point (configurable via the **St** input). If **Loop** is deactivated, the sample will run once from the starting point to the end or, if the direction parameter is set to **Backward**, will run from the starting point to the beginning. If **Loop** is activated, the sample will be continuously repeated within the loop range as soon as playback enters this range. The loop range is preset using data from the sound file from which the sample was loaded. If the sound file

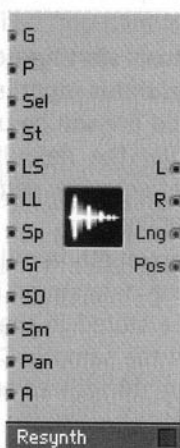
does not contain any loop data, the beginning of the sample is taken to be the beginning of the loop and the sample length is taken as the loop length. The loop data from the sound file are the default settings. These defaults are always used if the Loop Start (**LS**) or Loop Length (**LL**) inputs are not connected to other modules.

If the **Loop in Release** option is deactivated, loop playback will fade or fade out completely upon a **Gate** event occurring; depending on the direction set, the sample will run to its beginning or end and will then fade out. If the **Loop in Release** option is activated or if loop playback is switched off, the **Gate** events will only have a slight effect or none at all.

If the **No Stereo** option is activated (you can set this in the properties dialog box), stereo samples will be treated as mono samples, i.e. the same signal is sent to the **L** and **R** outputs. If this is the case, **Sampler Loop** only uses the left channel of stereo samples. This option has been made available because mono playback requires less processing capacity.

- **G**: A positive event at this input starts output from the position that is set at the **St** input. If negative values are present at the input, loop playback is interrupted unless the **Loop in Release** option has been activated.
- **P**: Logarithmic control input for the playback rate (pitch). If **P** is equal to the **Root Key** of the current sample, the sample will be played back at its original pitch. Furthermore, if the **Sel** input is not connected, **P** determines the selection of samples from the sample map. In **Oscillator Mode**, the **Sampler Loop** functions as a digital oscillator. In the same way as in the oscillator modules in REAKTOR, **P** determines the fundamental pitch of the generated oscillation.
- **Sel**: Control input for selecting a sample from the sample map. If this input is not connected, the values present at the **P** input are used instead.
- **F**: Linear control input for modulating the playback rate. The effect achieved is frequency modulation – the same as for the oscillator modules in REAKTOR.

- **St:** Control input for the starting point to be used when the next trigger occurs. The position is set in milliseconds from the start of the sample.
- **LS:** Control input for the loop starting point in milliseconds from the start of the sample. If this input is not connected, the loop data stored in the sound file will be used. If no loop data are stored in the sound file, the default is used: **LS** = 0. Changes at this input take effect when samples are retriggered and when a loop limit is reached.
- **LL:** Control input for the loop length in milliseconds. If this input is not connected, the loop data stored in the sound file will be used. If no loop data are stored in the sound file, the default is used: **LL** = length of the sample. Changes at this input take effect when samples are retriggered and when a loop limit is reached.
- **A:** Control input for the output amplitude.
- **L:** Audio output for the left channel of the sample player. When mono samples are being processed or if the **No Stereo** option has been activated, the same signal is present here as at the **R** output.
- **R:** Audio output for the right channel of the sample player. When mono samples are being processed or if the **No Stereo** option has been activated, the same signal is present here as at the **L** output.
- **Lng:** Polyphonic event output for the length of the current sample in milliseconds.



This is a real-time resynthesizer for the polyphonic, transposed playback of mono or stereo samples and sample maps. **Sample Resynth** allows you to control pitch and playback speed independently and in real-time, and also lets you extensively manipulate samples.

"Standard" samplers like the familiar hardware samplers or the **Sampler**, **Sampler FM** and **Sampler Loop** modules in REAKTOR all maintain a pointer for each voice. This pointer points to the current position in the sample. The amplitude at the output of the sampler is always the same as the amplitude of the sample at the pointer position. The pointer moves more quickly or less quickly through the sample and therefore generates an amplitude at the outputs which varies with time – i.e. an oscillation.

The speed of the pointer movement determines the speed of the sample playback (e.g. the speed of a beat loop). At the same time it also determines the pitch that is heard: the more slowly the signal (which has been recorded in the sample) is sampled, the longer the periods of the oscillations occurring at the output are – the pitch therefore decreases. If the pointer stops moving, the output amplitude stops changing. No audible signal is produced.

In the same way as a conventional sampler, **Sample Resynth** uses a pointer at the current sample position. However, the signal at the outputs is not simply the sample amplitude occurring at the pointer position. What actually happens is that the output signal is generated by a synthesizer inside the module. This synthesizer *resynthesizes* the signal at the pointer position. The pitch of the signal generated by this synthesizer is independent of the pointer's speed. Even if the pointer is not moving at all, the synthesizer continues to produce a sound. While the pitch of the output signal is determined, as is usual, over the **P** input, the **Sp** input determines the pointer speed.

Of course, the slowed down or "frozen" signal does not always correspond to what you might have imagined it to be. What does a hammer hitting a nail sound like if its is slowed down to infinity? The resynthesis algorithm used by **Sample Resynth** is designed in such a way that a broad range of sounds can be processed in a (musically-speaking) sensible, subtle or drastic manner. The algorithm can be adjusted by configuring the **G** (Granularity) and **Sm** (Smoothness) parameters. This means that you can manually adjust it to suit the sample and to produce drastic, strange sound effects. In the properties dialog box you can activate the **Signal-Informed Granulation** option separately for each sample in a sample map. If this option is switched on, the resynthesize algorithm in **Resynth** takes information on the sample into account. This information was collected during the analysis that was carried out when the sample was loaded for the first time. What this means is that the algorithm reacts to the characteristics of the audio material. If this option is deactivated, the results produced are generally quite "electronic".

Sample management is carried out in the **Sample Map Editor**.

After a positive **Gate** event, sample playback starts from the starting point (configurable via the **St** input). If **Loop** is deactivated, the sample will run once from the starting point to the end or, if the direction parameter is set to **Backward**, will run from the starting point to the beginning. If **Loop** is activated, the sample will be continuously repeated within the loop range as soon as playback enters this range. The loop range is preset using data from the sound file from which the sample was loaded. If the sound file does not contain any loop data, the beginning of the sample is

taken to be the beginning of the loop and the sample length is taken as the loop length. The loop data from the sound file are default settings. These defaults are always used if the Loop Start (**LS**) or Loop Length (**LL**) inputs are not connected to other modules.

If the **Loop in Release** option is deactivated, loop playback will fade or fade out completely upon a **Gate** event occurring. Depending on the direction set, the sample will run to its beginning or end and will then fade out. If the **Loop in Release** option is activated or if loop playback is switched off, the **Gate** events will only have a slight effect or none at all.

If the **No Stereo** option is activated (you can set this in the properties dialog box), stereo samples will be treated as mono samples, i.e. the same signal is sent to the **L** and **R** outputs. In this case, **Resynth** uses only the left channel of stereo samples. This option has been made available because mono playback requires less processing capacity.

All of the **Resynth's** inputs, except **Gate**, are designed as audio inputs. The values at these inputs are applied when samples are (re-)triggered (i.e. during positive **Gate** events). Changes to values during sample playback take effect in the **Granularity** interval (configured at the **Gr** input).

- **G**: A positive event at this input begins output from the position that is set at the **St** input. If negative values are present, loop playback is interrupted unless the **Loop in Release** option has been activated.
- **P**: Logarithmic audio control input for the playback rate (pitch). If **P** is equal to the **Root Key** of the current sample, the sample will be played back at its original pitch. Furthermore, if the **Sel** input is not connected, **P** also determines the selection of samples from the sample map.
- **Sel**: Audio control input for selecting a sample from the sample map. If this input is not connected, the values present at the **P** input are used instead.
- **St**: Audio control input for the starting point to be used when the next trigger occurs. The position is set in milliseconds from the start of the sample.

- **LS:** Audio control input for the loop starting point in milliseconds from the start of the sample. If this input is not connected, the loop data stored in the sound file will be used. If no loop data are stored in the sound file, the default is used: **LS** = 0.
- **LL:** Audio control input for the loop length in milliseconds. If this input is not connected, the loop data stored in the sound file will be used. If no loop data are stored in the sound file, the default is used: **LL** = length of the sample. If **LL** = 0, the movement within the sample stops when the loop starting point is reached; the sound is frozen at this point.
- **Sp:** Audio control input for pitch-independent output speed. Values that are present at the input are interpreted as a factor: When **Sp** = 1, playback occurs at the original speed; **Sp** = 2 corresponds to double the speed; and **Sp** = 0 means stop. If this input is not connected, the transposed original speed is taken as the default so that – as in conventional samplers – the speed reduces as pitch decreases.
- **Gr:** Audio control input for the granularity of the resynthesis process in milliseconds. This parameter determines the size of the sound particles used for resynthesis. If the **Signal-Informed Granulation** option is active, the values at the input will be used as the default; the values that are actually used are adjusted to suit the material.
- **SO:** Audio control input for modulation of the sample position (Sample Offset) in milliseconds. This input is used in order to modulate the position in the sample independent of pitch, e.g. via a noise generator.
- **Sm:** Audio control input for the *Smoothness* of the resynthesis process. The sound particles are adjusted using this. Small values generally lead to a rougher resulting sound.
- **Pan:** Audio control input for the position in the stereo field (-1 = Left, 0 = Center, 1 = Right).
- **A:** Audio control input for the output amplitude.
- **L:** Audio output for the left channel of the resynthesizer.
- **R:** Audio output for the right channel of the resynthesizer.
- **Lng:** Polyphonic event output for the length of the current sample in milliseconds.

- **Pos:** Polyphonic event output for the current position in the sample in milliseconds. Events are generated at time intervals corresponding to the set Granularity (**Gr**).

Grain Pitch Former

Sampler



This is a real-time resynthesizer for the polyphonic playback of mono or stereo samples and sample maps or WaveSets. **Sample Pitch Former** is a WaveSet synthesizer in which not only WaveSets can be loaded but also any samples you like. **Sample Pitch Former** removes the pitch development from a sample and gives the sample any new pitch you wish. Besides independent real-time control over the playback speed, **Sample Pitch Former** also allows you to carry out a spectral transposition, i.e. a pitch-independent transposition of the timbre.

"Standard" samplers like the familiar hardware samplers or the **Sampler**, **Sampler FM** and **Sampler Loop** modules in REAKTOR all maintain a pointer for each voice. This pointer points to the current position in the sample. The amplitude at the output of the sampler is always the same as the amplitude of the sample at the pointer position. The pointer moves more quickly or less quickly through the sample and therefore generates an amplitude at the outputs which varies with time – i.e. an oscillation.

The speed of the pointer movement determines the speed of the sample playback. At the same time it also determines the pitch that is heard: the more slowly the signal (that has been recorded in the sample) is sampled, the longer are the periods of the oscillations occurring at the output – the pitch therefore decreases. If the pointer stops moving, the output amplitude stops changing. No audible signal is produced.

In the same way as a conventional sampler, **Sample Pitch Former** uses a pointer at the current sample position. However, the signal at the outputs is not simply the sample amplitude occurring at the pointer position. What actually happens is that the output signal is generated by a synthesizer inside the module. This synthesizer resynthesizes the signal at the pointer position. The pitch of the signal generated by this synthesizer is independent of the pointer's speed. Even if the pointer is not moving at all, the synthesizer continues to produce a sound. While the pitch of the output signal is determined, as is usual, via the **P** input, the **Sp** input determines the pointer speed.

While conventional samplers and the **Sample Resynth** module in REAKTOR achieve a *relative* pitch change by *transposing* a sample, **Sample Pitch Former** forces any *absolute and definite* pitch that you wish onto a sample. **Sample Pitch Former** can therefore be used like a REAKTOR oscillator. Depending on the type of processed sound material and the settings used, the result can sound "electronically" altered to a greater or lesser degree. **Sample Pitch Former** will also generate signals with a certain pitch if the processed sample has no unique pitch of its own (e.g. recordings of cymbals or gongs). **Sample Pitch Former** produces fewer sound alterations the more restricted the fundamental pitch of the processed material is to be defined, and the less the original pitch deviates from the generated pitch.

In conventional samplers and in the **Sample Resynth** module in REAKTOR, the transposing of the fundamental pitch goes hand in hand with the transposing of *all* the spectral components. This is often considered a limitation since the transposition also affects certain spectral components which the listener would not expect to hear changed. The "Mickey Mouse effect" that occurs when speech recordings are detuned can be traced back to the transposition of the formants whose position for a "real" human speaker

would be largely independent of the fundamental pitch. Within certain limits, **Sample Pitch Former** decouples pitch and formant position from one another. The formant position can be shifted independent of pitch via the formant shift input (**FS**). Since the human ear uses all the spectral components to identify the fundamental pitch, you can manipulate this parameter to create interesting substitutions of fundamental pitch and timbre, especially at very low pitches. Similar sound effects are often created by synthesizer experts using *oscillator synchronization*.

Sample management is carried out in the **Sample Map Editor**.

After a positive **Gate** event, sample playback starts from the starting point (configurable via the **St** input). If **Loop** is deactivated, the sample will run once from the starting point to the end or, if the direction parameter is set to **Backward**, will run from the starting point to the beginning. If **Loop** is activated, the sample will be continuously repeated within the loop range as soon as playback enters this range. The loop range is preset using data from the sound file from which the sample was loaded. If the sound file does not contain any loop data, the beginning of the sample is taken to be the beginning of the loop and the sample length is taken as the loop length. The loop data from the sound file are default settings. These defaults are always used if the Loop Start (**LS**) or Loop Length (**LL**) inputs are not connected to other modules.

If the **Loop in Release** option is deactivated, loop playback will fade or fade out completely upon a **Gate** event occurring. Depending on the direction set, the sample will run to its beginning or end and will then fade out. If the **Loop in Release** option is activated or if loop playback is switched off, **Gate** events that are smaller than or equal to zero have no effect.

If the **No Stereo** option is activated (you can set this in the properties dialog box), stereo samples will be treated as mono samples, i.e. the same signal is sent to the **L** and **R** outputs. In this case, **Sample Pitch Former** uses only the left channel of stereo samples. This option has been made available because mono playback requires less processing capacity.

All of **Sample Pitch Former's** inputs, except **Gate**, are designed as audio inputs. The values at the inputs are applied when samples are (re-) triggered (i.e. during positive **Gate** events). During sample playback, changes to values take effect at intervals of the fundamental pitch period (you can set this via the **P** input).

- **G**: A positive event at this input starts output from the position that is set at the **St** input. If negative values are present, loop playback is interrupted unless the **Loop in Release** option has been activated.
- **P**: Logarithmic audio control input for the playback rate (pitch). Furthermore, if the **Sel** input is not connected, **P** also determines the selection of samples from the sample map.
- **Sel**: Audio control input for selecting a sample from the sample map. If this input is not connected, the values present at the **P** input are used instead.
- **St**: Audio control input for the starting point to be used when the next trigger occurs. The position is set in milliseconds from the start of the sample.
- **LS**: Audio control input for the loop starting point in milliseconds from the start of the sample. If this input is not connected, the loop data stored in the sound file will be used. If no loop data are stored in the sound file, the default is used: **LS** = 0.
- **LL**: Audio control input for the loop length in milliseconds. If this input is not connected, the loop data stored in the sound file will be used. If no loop data are stored in the sound file, the default is used: **LL** = length of the sample. If **LL** = 0, the movement within the sample stops when the loop starting point is reached; the sound is frozen at this point.
- **Sp**: Audio control input for pitch-independent output speed. Values that are present at the input are interpreted as a factor: when **Sp** = 1, playback occurs at the original speed; **Sp** = 2 corresponds to double the speed; and **Sp** = 0 means stop. If this input is not connected, the value 1 is taken as the default.
- **FS**: Audio control input for pitch-independent transposing of the formant position in semi-tones.

- **SO:** Audio control input for modulation of the sample position (Sample Offset) in milliseconds. This input is used to modulate the position in the sample independent of pitch, e.g. via a noise generator.
- **Sm:** Audio control input for the *Smoothness* of the resynthesis process. The sound particles are adjusted using this. Small values generally lead to a rougher resulting sound.
- **Pan:** Audio control input for the position in the stereo field (-1 = Left, 0 = Center, 1 = Right).
- **A:** Audio control input for the output amplitude.
- **L:** Audio output for the left channel of the resynthesizer.
- **R:** Audio output for the right channel of the resynthesizer.
- **Lng:** Polyphonic event output for the length of the current sample in milliseconds.
- **Pos:** Polyphonic event output for the current position in the sample in milliseconds. Events at this input are generated at time intervals corresponding to the current fundamental pitch period.

Grain Cloud

Sampler



Stereo-multi-sample granular-synthesizer with independent control over pitch **P**, pitch-slide **PS**, sample selection **Sel**, sample-position **Pos** and length **Len** of each grain. The envelope of each grain can be controlled with attack **Att** and decay **Dec**.

For each grain the delta time to the start of the next grain can be set with **dt**. The maximum number of simultaneous grains can be set in the Properties.

There are several jitter-inputs which set a range for the respective input.

Sample management is carried out in the **Sample Map Editor**.

- **Trig**: Event input for the Gate signal. (G) > 0 starts immediately next gain.
- **P**: Audio input for logarithmic pitch control (in semitones). The pitch is independent of the speed traversal. The sample plays at the original pitch when P=Root key. Typ. range: [0...127], Default:60.

- **D/F**: Audio input for direction control if P is connected. Otherwise it is a frequency controls. The sample plays at the original pitch when F=1, in reverse direction when F=-1. Typ. range: [-4...4], Default: 1.
- **PJ**: Audio input for pitch jitter control (in semitones). Typ. range: [0...3].
- **PS**: Audio input for logarithmic pitch shift control of current grain in semitones. Typ. range: [-3...3].
- **Sel**: Audio input for multi-sample selection (in semitones). When unconnected, the values at the (P) input are used. Typ. range: [0...127].
- **Pos**: Audio input for setting the sound file position in ms. Range [0...<length of sample in ms>].
- **PsJ**: Audio input for sound file position jitter control in ms. Typ. range: [0...<length of sample in ms>].
- **Len**: Audio input for setting the grain length in ms. Typ range: [10...100]. Default: 20 ms.
- **LnJ**: Audio input for grain length jittercontrol in ms. Typ range: [10...100]. Default: 0 ms.
- **Att**: Audio input for setting the attack time. Range: [0...1]. Default: 0.2.
- **Dec**: Audio input for setting the decay time. Range: [0...1]. Default: 0.2.
- **dt**: Audio input for setting the delta time before starting the next grain in ms. Typ range: [5...100]. Default: 20.
- **dtJ**: Audio input for delta time jitter control in ms. Typ range: [10...100]. Default: 20 ms.
- **Pan**: Audio input for setting the position in the stereo field. Range: [-1(Left)...1(Right)].
- **PnJ**: Audio input for pan jitter control. Range: [0...1].
- **A**: Audio input for amplitude control. Typ range: [0...1]. Default: 1.
- **L**: Polyphonic left channel audio output. Typ range: [-1...1].
- **R**: Polyphonic right channel audio output. Typ range: [-1...1].
- **Lng**: Event output for the length of samples in ms. Typ range: [0...<length of samples in ms>].

- **GTr:** Event output for grain trigger. Outputs 1 when a new grain starts, 0 when it stops.

Beat Loop

Sampler



This is a real-time resynthesizer for the synchronized playback of beat-loop samples. The transposing of beat-loops can be set via the **P** input independent of playback speed. **Beat Loop** synchronizes itself to a 96th note clock source (24 ppq) that is connected to the **C** input or, if the **C** input is not connected, it can sync with the global REAKTOR clock. Therefore, by default all **Beat Loops** in REAKTOR run in sync with one another independent of the sample's internal speed. Furthermore, **Beat Loop** also allows you to easily link internal REAKTOR sequencer modules and MIDI clocks to rhythmic sample material.

Sample management is carried out in the **Sample Map Editor**.

Beat Loop requires samples that have been cut exactly, and which contain 2, 4, 8, 16, or 32, etc. beats. The speed of the beat loops used should be between 87 and 174 BPM. If the samples being used fulfill these conditions, **Beat Loop** can output them in good quality even if the playback speed is changed. By activating the

sample-related **Pitched Sound** option (accessible in the properties dialog box), possible pitch falsification of basslines (and similar) can be avoided. However, in doing so you will have to accept a deterioration in rhythmic precision.

The loop range of the sample is configured using the Loop Start (**LS**) and Loop Length (**LL**) inputs. If **LS** and **LL** are not connected, the whole sample will be played back as a loop. Using positive **Rst** events, sample playback will be continued from the starting point (you can set this via the **St** input).

If the **No Stereo** option is activated (you can set this in the properties dialog box), stereo samples will be treated as mono samples, i.e. the same signal is sent to the **L** and **R** outputs. In this case, **Beat Loop** uses only the left channel of stereo samples. This option has been made available because mono playback requires less processing capacity.

All of **Beat Loop**'s inputs, except **C** and **Rst**, are designed as audio inputs. During sample playback, changes to values take effect at intervals of sixteenths of a note value.

- **C**: A positive event at this input switches the **Beat Loop** module by one 96th note value further. If this input is not connected, the module synchronizes itself with the global REAKTOR clock.
- **Rst**: A positive event at this input resets playback to the position set at the **St** input.
- **P**: Logarithmic audio control input for the playback rate (pitch). Furthermore, if the **Sel** input is not connected, **P** determines the selection of samples from the sample map.
- **Sel**: Audio control input for selecting a sample from the sample map. If this input is not connected, the values present at the **P** input are used instead.
- **St**: Audio control input for the starting point to be used when the next trigger occurs. The position is set in sixteenths of a note value from the start of the sample.
- **LS**: Audio control input for the loop starting point in sixteenths of a note value from the start of the sample. If this input is not connected, the default is used: **LS** = 0.

- **LL:** Audio control input for the loop length in sixteenths of a note value. If this input is not connected, the default is used: **LL** = length of the sample.
- **SO:** Audio control input for modulation of the sample position (Sample Offset) in sixteenths of a note value. This input is used to reach steps in the sample which, for instance, can be controlled by a sequencer module.
- **Sm:** Audio control input for the *Smoothness* of the resynthesis process. The sound particles are adjusted using this. Very small values generally lead to a "cracking" sound every sixteenth interval.
- **Pan:** Audio control input for the position in the stereo field (-1 = Left, 0 = Center, 1 = Right).
- **A:** Audio control input for the output amplitude.
- **L:** Audio output for the left channel of the resynthesizer.
- **R:** Audio output for the right channel of the resynthesizer.
- **Len:** Polyphonic event output for the length of the current sample in milliseconds.
- **L16:** Polyphonic event output for the length of the current sample in sixteenths of a note value.
- **P16:** Polyphonic event output for the current position in the sample in sixteenths of a note value.
- **Ct16:** Polyphonic event output for counting the sixteenths of a note value that have passed since the start / reset.

Sample Lookup

Sampler



This module makes samples available as function value look-up tables. A position within the sample in milliseconds is set via the **Pos** input. The value of the sample at this point is sent to the outputs.

You can load sound files using the **Load Sound...** entry in the context menu.

Sample management is carried out in the **Sample Map Editor**.

The properties dialog box allows you to select one of three levels of quality that is to be used for interpolation during transposed sample playback (**Poor, Good, Excellent**). If set to **Poor**, the sample values are not interpolated during output. Higher playback quality is achieved at the expense of processing capacity.

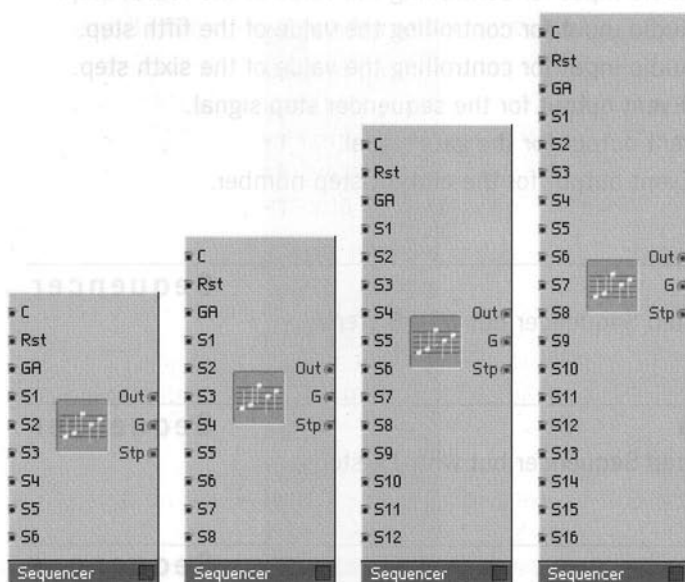
- **Pos:** Audio input for the position in the sample in milliseconds.
- **A:** Audio input for amplitude modulation.
- **L:** Audio output for the left channel of the sample. When processing mono samples, the same signal is output here as is sent to the **R** output.
- **R:** Audio output for the right channel of the sample. When processing mono samples, the same signal is output here as is sent to the **L** output.
- **Lng:** Polyphonic event output for the length of the current sample in milliseconds.

Sequencer

Reaktor's sequencers include gated step-sequencers in four sizes as well as a selectable-position sequencer. You'll also find clocks to drive them here.

Sequencer

Sequencer



6-Step

Sequencer

Sequencer with 6 steps. The output value for each step (e.g. to control oscillator pitch) can be set independently. In addition a gate signal is output with the amplitude given by the current value of the gate amplitude input at the time the step advances.

- **C:** Audio input for clock control. A positive zero crossing switches to the next step. Typically you would connect a Pulse Oscillator or MIDI Sync module here.
- **Rst:** Audio input for a reset signal. A positive zero crossing puts the sequencer back to the first step. Typically you would connect a button or MIDI Start module here.
- **GA:** Audio input for controlling the amplitude of the gate output. When the value is zero or the input is not connected, no signal appears on the gate output.
- **S1:** Audio input for controlling the value of the first step.
- **S2:** Audio input for controlling the value of the second step.

- **S3:** Audio input for controlling the value of the third step.
- **S4:** Audio input for controlling the value of the fourth step.
- **S5:** Audio input for controlling the value of the fifth step.
- **S6:** Audio input for controlling the value of the sixth step.
- **Out:** Event output for the sequencer step signal.
- **G:** Event output for the gate signal.
- **St:** Event output for the current step number.

8-Step

Sequencer

Like 6-Step Sequencer but with 8 steps.

12-Step

Sequencer

Like 6-Step Sequencer but with 12 steps.

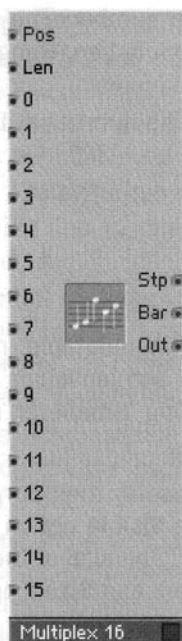
16-Step

Sequencer

Like 6-Step Sequencer but with 16 steps.

Multiplex 16

Sequencer



Value Selector, useful as a sequencer. An event at **Pos** addresses with its value one of the 16 inputs and the current value at this input is forwarded to **Out**.

If the **Pos** input is driven by a signal which is incremented stepwise, the output **Out** provides a sequence of the values at the inputs **0...15**. The values at the **Pos** input are folded into the number range $[0 \dots (\text{Len} - 1)]$ to allow a variable sequence length **Len**.

The **Pos** input can also be driven by a control element, a **Beat Loop** module, a random event source, or by the master clock.

- **Pos**: Input for controlling the selection. Each event at this input results in an event at the outputs. To run Select 16 as a sequencer, the position is set to the number of 16th notes that have passed since start or reset.
- **Len**: Input to set the sequence length. Range: $[1 \dots 16]$.

- **0-15:** Audio input for controlling the value of the appropriate step.
- **Stp:** Current sequence step number. The number at this output is calculated as **Pos** modulo **Len**. Range: [0 ... <sequence length >].
- **Bar:** Current bar number. The number at this output is calculated as $\text{Integer}(\text{Pos} / \text{Len})$.
- **Out:** Current sequence step output value.

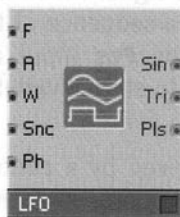
LFO, Envelope

This category includes a fully modulatable, multiwaveform LFO, a random control generator, and envelope generators of just about any description including time/level ramp generators in three sizes.

All Envelopes can optionally display their curve in a panel graphic. This can be achieved with the **Visible** option on the **Appearance** page in the module Properties. The size of the display can be set in the Properties using the **Size X** and **Size Y** options. The timeline of the curve in the display is not scaled.

LFO

LFO, Envelope



Low Frequency Oscillator with Pulse, Triangle and Parabol Wave outputs. It is typically used as a source for modulation signals (for vibrato, tremolo etc.). The output signal is a stream of events at the **Control Rate** selected in the **Settings** menu.

Since the LFO operates at the Control Rate, it is much more CPU efficient than an audio oscillator which could do the same job.

- **F**: Audio input for control of the oscillator frequency in Hz. For control with Tempo in BPM, you can compute the required value in Hz as follows: $F = \text{oscillations-per-beat} \times \text{BPM}/60$. So, for example, for 3 oscillations per bar at 4 beats to the bar (that is $\frac{3}{4}$ oscillations per beat), you get $F = (\frac{3}{4})/60 \times \text{BPM}$ or $0.0125 \times \text{BPM}$.
- **A**: Input for controlling the amplitude. The output signal moves between $+A$ and $-A$.
- **W**: Event input for control of pulse width, i.e. the ratio of the duration of the on-period to the off-period for the pulse output, and appropriate warping of the other waveforms. Range of values is -1 to 1. Symmetric waveforms at $W = 0$.
- **Snc**: Event input for synchronization of the LFO waveform. A positive event synchronizes the oscillator, resetting it to the phase given by the **Ph** input.
- **Ph**: Input for specifying the phase (position in the waveform) to which the oscillator is reset when synchronization occurs. **Ph** = 0: Phase = 0 deg (middle of rising slope), **Ph** = 0.5: Phase = 180 deg (middle of falling slope), **Ph** = 1: Phase = 360 deg (same as 0 deg).
- **Sin**: Event output for the sine waveshape signal.
- **Tri**: Event output for the triangle waveshape signal.
- **Pls**: Event output for the pulse waveshape signal.

Slow Random

LFO, Envelope



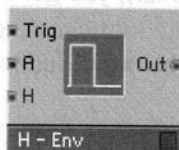
Low Frequency Oscillator which produces a random step waveform.

- **F**: Control input for the step frequency in Hz.

- **A:** Control input for the amplitude. The output value is somewhere in the range $-A$ to $+A$.
- **Out:** Event output for the random signal.

H - Env

LFO, Envelope

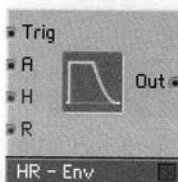


Envelope generator with hold characteristic. When the envelope is triggered the output value jumps to the current value of the amplitude input and stays there until the hold time has passed, after which the output jumps back to zero. The envelope can be triggered at any time, including retriggering during the hold time.

- **T:** Audio input for triggering the envelope on a rising edge (value leaves zero in positive direction).
- **A:** Audio input for the output value during the hold time. The input is sampled at the triggering instant after which the value is held at the output.
- **H:** Logarithmic event input for controlling the hold time. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (i.e. in $\text{dB}_{1\text{ms}}$).
- **Out:** Audio output for the envelope signal.

HR - Env

LFO, Envelope



Envelope generator with hold-release characteristic. When the envelope is triggered the output value jumps to the current value of the amplitude input and stays there until the hold time has passed, after which the output decays exponentially with the release time back to zero.

- **T:** Audio input for triggering the envelope on a rising edge (value leaves zero in positive direction).
- **A:** Audio input for the output value during the hold time. The input is sampled at the triggering instant after which the value is held at the output.
- **H:** Logarithmic event input for controlling the hold time. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (i.e. in dB_{1ms}).
- **R:** Logarithmic event input for controlling the release time. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (i.e. in dB_{1ms}).
- **Out:** Audio output for the envelope signal.

D - Env

LFO, Envelope

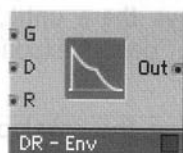


Envelope generator with decay characteristic. When the envelope is triggered the output value jumps to the current value of the amplitude input, after which the output decays exponentially with the decay time back to zero.

- **T:** Audio input for triggering the envelope on a rising edge (value leaves zero in positive direction).
- **A:** Audio input for the initial output value. The input is sampled at the triggering instant.
- **D:** Logarithmic event input for controlling the decay time. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (i.e. in dB_{1ms}).
- **Out:** Audio output for the envelope signal.

DR - Env

LFO, Envelope

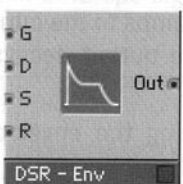


Envelope generator with decay-release characteristic. When the envelope is triggered with a gate event the output value jumps to the amplitude value of the gate signal, after which the output decays exponentially with the decay time back to zero. A gate event with amplitude zero (at note off) sets the decaying time to the release time value which is normally set to be shorter.

- **G:** Event input for the gate signal that triggers the envelope. The amplitude of the gate signal determines the initial output value.
- **D:** Logarithmic event input for controlling the decay time. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (i.e. in $\text{dB}_{1\text{ms}}$).
- **R:** Logarithmic event input for controlling the release time. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (i.e. in $\text{dB}_{1\text{ms}}$).
- **Out:** Audio output for the envelope signal.

DSR - Env

LFO, Envelope

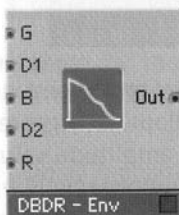


Envelope generator with decay-sustain-release characteristic. When the envelope is triggered with a gate event the output value jumps to the amplitude value of the gate signal, after which the output decays exponentially with the decay time to the sustain level (multiplied by the amplitude). After a gate event with amplitude zero (at note off) the output decays exponentially with the release time back to zero.

- **G:** Event input for the gate signal that triggers the envelope. The amplitude of the gate signal determines the initial output value.
- **D:** Logarithmic event input for controlling the decay time. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (i.e. in $\text{dB}_{1\text{ms}}$).
- **S:** Event input for controlling the sustain level. Typical range of values is: 0 (decay to zero) to 1 (hold at initial level), but sustain can be greater than 1 or even less than 0.
- **R:** Logarithmic event input for controlling the release time. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (i.e. in $\text{dB}_{1\text{ms}}$).
- **Out:** Audio output for the envelope signal.

DBDR - Env

LFO, Envelope



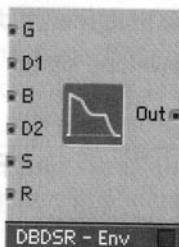
Envelope generator with decay-breakpoint-release characteristic. When the envelope is triggered with a gate event the output value jumps to the amplitude value of the gate signal, after which the output decays exponentially with the decay-1 time until it reaches the breakpoint level (multiplied by the amplitude). Next it continues decaying exponentially with the decay-2 time back to zero. A gate event with amplitude zero (at note off) sets the decaying time to the release time value which is normally set to be shorter.

- **G:** Event input for the gate signal that triggers the envelope. The amplitude of the gate signal determines the initial output value.
- **D1:** Logarithmic event input for controlling the first decay time. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (i.e. in $\text{dB}_{1\text{ms}}$).
- **B:** Event input for controlling the breakpoint level. Range of values: 0 (never use decay-2 time) to 1 (immediately use decay-2 time).

- **D2:** Logarithmic event input for controlling the second decay time. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (i.e. in $\text{dB}_{1\text{ms}}$).
- **R:** Logarithmic event input for controlling the release time. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (i.e. in $\text{dB}_{1\text{ms}}$).
- **Out:** Audio output for the envelope signal.

DBDSR-Env

LFO, Envelope



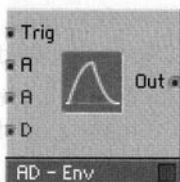
Envelope generator with decay-breakpoint-decay-sustain-release characteristic. When the envelope is triggered with a gate event the output value jumps to the amplitude value of the gate signal, after which the output decays exponentially with the decay-1 time until it reaches the breakpoint level (multiplied by the amplitude). Next it continues decaying with the decay-2 time to the sustain level (multiplied by the amplitude). The output is held at the sustain level until a gate event with amplitude zero (at note off) arrives, after which the output decays exponentially with the release time back to zero.

- **G:** Event input for the gate signal that triggers the envelope. The amplitude of the gate signal determines the initial output value.
- **D1:** Logarithmic event input for controlling the first decay time. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (i.e. in $\text{dB}_{1\text{ms}}$).
- **B:** Event input for controlling the breakpoint level. Range of values: 0 (never use decay-2 time) to 1 (immediately use decay-2 time).
- **D2:** Logarithmic event input for controlling the second decay time. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (i.e. in $\text{dB}_{1\text{ms}}$).

- **S:** Event input for controlling the sustain level. Typical range of values is: 0 (decay to zero) to 1 (hold at end of attack), but sustain can be greater than 1 or even less than 0.
- **R:** Logarithmic event input for controlling the release time. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (i.e. in $\text{dB}_{1\text{ms}}$).
- **Out:** Audio output for the envelope signal.

AD - Env

LFO, Envelope

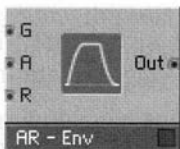


Attack-Decay envelope, triggered by the positive slope of the T signal. The decay starts immediately, when the attack time is over.

- **T:** Input for the trigger signal. A positive slope starts the envelope.
- **A:** Control input for the output amplitude.
- **A:** Control input for the attack time of the envelope. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (i.e. in $\text{dB}_{1\text{ms}}$).
- **D:** Control input for the decay time of the envelope. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (i.e. in $\text{dB}_{1\text{ms}}$).
- **Out:** Output for the envelope signal.

AR - Env

LFO, Envelope

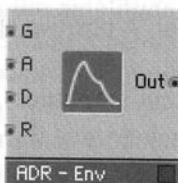


Envelope generator with attack-release characteristic. When the envelope is triggered with a gate event the output value rises during the attack time with a linear slope to the amplitude value of the gate signal. The output is then held at the maximum level until a gate event with amplitude zero (at note off) arrives, after which the output decays exponentially with the release time back to zero.

- **G:** Event input for the gate signal that triggers the envelope. The amplitude of the gate signal determines the maximum output value.
- **A:** Logarithmic event input for controlling the attack time. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (i.e. in $\text{dB}_{1\text{ms}}$).
- **R:** Logarithmic event input for controlling the release time. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (i.e. in $\text{dB}_{1\text{ms}}$).
- **Out:** Audio output for the envelope signal.

ADR-Env

LFO, Envelope



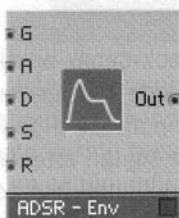
Envelope generator with attack-decay-release characteristic. When the envelope is triggered with a gate event the output value rises during the attack time with a linear slope to the amplitude value of the gate signal, after which it decays with the decay time to zero. When a gate event with amplitude zero (at note off) arrives, the output decays exponentially with the release time back to zero.

- **G:** Event input for the gate signal that triggers the envelope. The amplitude of the gate signal determines the maximum output value.
- **A:** Logarithmic event input for controlling the attack time. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (i.e. in $\text{dB}_{1\text{ms}}$).

- **D:** Logarithmic event input for controlling the decay time. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (i.e. in $\text{dB}_{1\text{ms}}$).
- **R:** Logarithmic event input for controlling the release time. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (i.e. in $\text{dB}_{1\text{ms}}$).
- **Out:** Audio output for the envelope signal.

ADSR - Env

LFO, Envelope

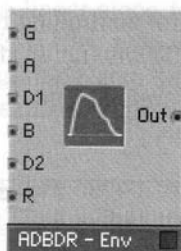


Envelope generator with the classic attack-decay-sustain-release characteristic. When the envelope is triggered with a gate event the output value rises during the attack time with a linear slope to the amplitude value of the gate signal, after which it decays exponentially with the decay time to the sustain level (multiplied by the amplitude). The output is held at the sustain level until a gate event with amplitude zero (at note off) arrives, after which the output decays exponentially with the release time back to zero.

- **G:** Event input for the gate signal that triggers the envelope. The amplitude of the gate signal determines the maximum output value.
- **A:** Logarithmic event input for controlling the attack time. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (i.e. in $\text{dB}_{1\text{ms}}$).
- **D:** Logarithmic event input for controlling the decay time. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (i.e. in $\text{dB}_{1\text{ms}}$).
- **S:** Event input for controlling the sustain level. Typical range of values is: 0 (decay to zero) to 1 (hold at end of attack), but sustain can be greater than 1 or even less than 0.
- **R:** Logarithmic event input for controlling the release time. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (i.e. in $\text{dB}_{1\text{ms}}$).
- **Out:** Audio output for the envelope signal.

ADBDR - Env

LFO, Envelope

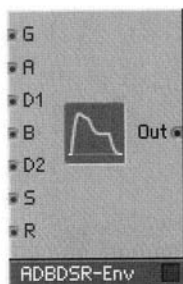


Envelope generator with attack-decay-breakpoint-decay-release characteristic. When the envelope is triggered with a gate event the output value rises during the attack time with a linear slope to the amplitude value of the gate signal, after which it decays exponentially with the decay-1 time until it reaches the breakpoint level (multiplied by the amplitude). Next it continues decaying exponentially with the decay-2 time back to zero. A gate event with amplitude zero (at note off) sets the decaying time to the release time value which is normally set to be shorter.

- **G:** Event input for the gate signal that triggers the envelope. The amplitude of the gate signal determines the maximum output value.
- **A:** Logarithmic event input for controlling the attack time. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (i.e. in $\text{dB}_{1\text{ms}}$).
- **D1:** Logarithmic event input for controlling the first decay time. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (i.e. in $\text{dB}_{1\text{ms}}$).
- **B:** Event input for controlling the breakpoint level. Range of values: 0 (never use decay-2 time) to 1 (immediately use decay-2 time).
- **D2:** Logarithmic event input for controlling the second decay time. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (i.e. in $\text{dB}_{1\text{ms}}$).
- **R:** Logarithmic event input for controlling the release time. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (i.e. in $\text{dB}_{1\text{ms}}$).
- **Out:** Audio output for the envelope signal.

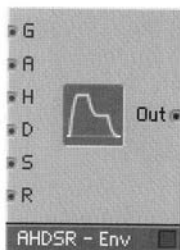
ADBD SR-Env

LFO, Envelope



Envelope generator with attack-decay-breakpoint-decay-sustain-release characteristic. When the envelope is triggered with a gate event the output value rises during the attack time with a linear slope to the amplitude value of the gate signal, after which it decays according to the first decay time with a linear slope to the breakpoint level. From there it continues with the second decay time to the sustain level (the levels are multiplied by the amplitude). The output is held at the sustain level until a gate event with amplitude zero (at note off) arrives, after which the output decays exponentially with the release time back to zero.

- **G:** Event input for the gate signal that triggers the envelope. The amplitude of the gate signal determines the maximum output value.
- **A:** Logarithmic event input for controlling the attack time. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (i.e. in dB_{1ms}).
- **D1:** Logarithmic event input for controlling the first decay time. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (i.e. in dB_{1ms}).
- **B:** Event input for controlling the break point level. Typical range of values is: 0 to 1.
- **D2:** Logarithmic event input for controlling the second decay time. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (i.e. in dB_{1ms}).
- **S:** Event input for controlling the sustain level. Typical range of values is: 0 to 1.
- **R:** Logarithmic event input for controlling the release time. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (i.e. in dB_{1ms}).
- **Out:** Audio output for the envelope signal.

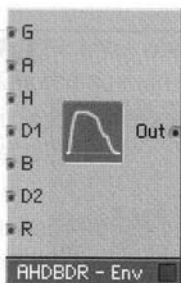


Envelope generator with attack-hold-decay-sustain-release characteristic. When the envelope is triggered with a gate event the output value rises during the attack time with a linear slope to the amplitude value of the gate signal and stays there until the hold time has passed, after which it decays according to the decay time with a linear slope to the sustain level. The output is held at the sustain level until a gate event with amplitude zero (at note off) arrives, after which the output decays exponentially with the release time back to zero.

- **G:** Event input for the gate signal that triggers the envelope. The amplitude of the gate signal determines the maximum output value.
- **A:** Logarithmic event input for controlling the attack time. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (i.e. in $\text{dB}_{1\text{ms}}$).
- **H:** Logarithmic event input for controlling the hold time. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (i.e. in $\text{dB}_{1\text{ms}}$).
- **D:** Logarithmic event input for controlling the decay time. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (i.e. in $\text{dB}_{1\text{ms}}$).
- **S:** Event input for controlling the sustain level. Typical range of values is: 0 (decay to zero) to 1 (hold at end of attack), but sustain can be greater than 1 or even less than 0.
- **R:** Logarithmic event input for controlling the release time. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (i.e. in $\text{dB}_{1\text{ms}}$).
- **Out:** Audio output for the envelope signal.

AHDBDR - Env

LFO, Envelope



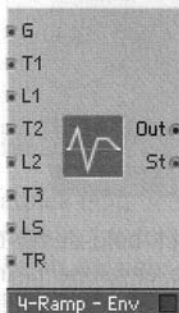
Envelope generator with attack-hold-decay-breakpoint-decay-release characteristic. When the envelope is triggered with a gate event the output value rises during the attack time with a linear slope to the amplitude value of the gate signal and stays there until the hold time has passed, after which it decays exponentially with the decay-1 time until it reaches the breakpoint level (multiplied by the amplitude). Next it continues decaying exponentially with the decay-2 time back to zero. A gate event with amplitude zero (at note off) sets the decaying time to the release time value which is normally set to be shorter.

- **G:** Event input for the gate signal that triggers the envelope. The amplitude of the gate signal determines the maximum output value.
- **A:** Logarithmic event input for controlling the attack time. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (i.e. in $\text{dB}_{1\text{ms}}$).
- **H:** Logarithmic event input for controlling the hold time. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (i.e. in $\text{dB}_{1\text{ms}}$).
- **D1:** Logarithmic event input for controlling the first decay time. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (i.e. in $\text{dB}_{1\text{ms}}$).
- **B:** Event input for controlling the breakpoint level. Range of values: 0 (never use decay-2 time) to 1 (immediately use decay-2 time).
- **D2:** Logarithmic event input for controlling the second decay time. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (i.e. in $\text{dB}_{1\text{ms}}$).
- **R:** Logarithmic event input for controlling the release time. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (i.e. in $\text{dB}_{1\text{ms}}$).

- **Out:** Audio output for the envelope signal.

4-Ramp

LFO, Envelope



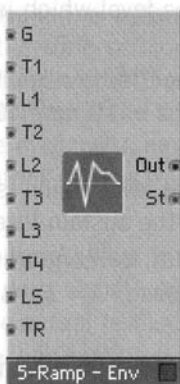
4-stage envelope generator with linear slopes. When the envelope is triggered with a gate event the output value rises to the first level, reaching it within the time set for the first stage. Then it continues to the second level within the time set for the second stage, and so on. The third level is the sustain level, at which the output is held until a gate event with amplitude zero (at note off) arrives, after which the output returns to zero within the last time-period.

- **G:** Event input for the gate signal that triggers the envelope. The amplitude of the gate signal combines with all the level values to determine the actual levels reached.
- **T1:** Logarithmic event input for controlling the time for the first stage. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (i.e. in dB_{1ms}).
- **L1:** Input for controlling the level which is reached at the end of the first stage.
- **T2:** Logarithmic event input for controlling the time for the second stage. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (i.e. in dB_{1ms}).
- **L2:** Input for controlling the level which is reached at the end of the second stage.

- **T3:** Logarithmic event input for controlling the time for the third stage. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (i.e. in $\text{dB}_{1\text{ms}}$).
- **LS:** Input for controlling the level which is reached at the end of the third stage. This is the sustain level.
- **TR:** Logarithmic event input for controlling the time for the last stage which is the release. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (i.e. in $\text{dB}_{1\text{ms}}$).
- **Stg:** Event output for the envelope's current stage (1, 2 ...). After the end of the release stage and before a new trigger, the value is 0. With appropriate processing this value can be used to chain other envelopes, or for the envelope to retrigger itself.
- **Out:** Audio output for the envelope signal.

5-Ramp

LFO, Envelope

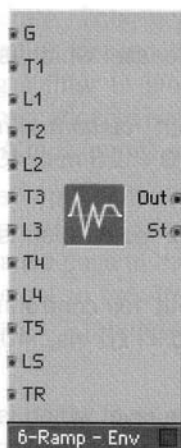


5-stage envelope generator with linear slopes. When the envelope is triggered with a gate event the output value rises to the first level, reaching it within the time set for the first stage. Then it continues to the second level within the time set for the second stage, and so on. The fourth level is the sustain level, at which the output is held until a gate event with amplitude zero (at note off) arrives, after which the output returns to zero within the last time-period.

- **G:** Event input for the gate signal that triggers the envelope. The amplitude of the gate signal combines with all the level values to determine the actual levels reached.
- **T1:** Logarithmic event input for controlling the time for the first stage. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (i.e. in $\text{dB}_{1\text{ms}}$).
- **L1:** Input for controlling the level which is reached at the end of the first stage.
- **T2:** Logarithmic event input for controlling the time for the second stage. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (i.e. in $\text{dB}_{1\text{ms}}$).
- **L2:** Input for controlling the level which is reached at the end of the second stage.
- **T3:** Logarithmic event input for controlling the time for the third stage. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (i.e. in $\text{dB}_{1\text{ms}}$).
- **L3:** Input for controlling the level which is reached at the end of the third stage.
- **T4:** Logarithmic event input for controlling the time for the fourth stage. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (i.e. in $\text{dB}_{1\text{ms}}$).
- **LS:** Input for controlling the level which is reached at the end of the fourth stage. This is the sustain level.
- **TR:** Logarithmic event input for controlling the time for the last stage, which is the release. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (i.e. in $\text{dB}_{1\text{ms}}$).
- **Stg:** Event output for the envelope's current stage (1, 2 ...). After the end of the release stage and before a new trigger, the value is 0. With appropriate processing this value can be used to chain other envelopes, or for the envelope to retrigger itself.
- **Out:** Audio output for the envelope signal.

6-Ramp

LFO, Envelope



6-stage envelope generator with linear slopes. When the envelope is triggered with a gate event the output value rises to the first level, reaching it within the time set for the first stage. Then it continues to the second level within the time set for the second stage, and so on. The fifth level is the sustain level, at which the output is held until a gate event with amplitude zero (at note off) arrives, after which the output returns to zero within the last time-period.

- **G:** Event input for the gate signal that triggers the envelope. The amplitude of the gate signal combines with all the level values to determine the actual levels reached.
- **T1:** Logarithmic event input for controlling the time for the first stage. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (i.e. in dB_{1ms}).
- **L1:** Input for controlling the level which is reached at the end of the first stage.
- **T2:** Logarithmic event input for controlling the time for the second stage. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (i.e. in dB_{1ms}).
- **L2:** Input for controlling the level which is reached at the end of the second stage.

- **T3:** Logarithmic event input for controlling the time for the third stage. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (i.e. in $\text{dB}_{1\text{ms}}$).
- **L3:** Input for controlling the level which is reached at the end of the third stage.
- **T4:** Logarithmic event input for controlling the time for the fourth stage. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (i.e. in $\text{dB}_{1\text{ms}}$).
- **L4:** Input for controlling the level which is reached at the end of the fourth stage.
- **T5:** Logarithmic event input for controlling the time for the fifth stage. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (i.e. in $\text{dB}_{1\text{ms}}$).
- **LS:** Input for controlling the level which is reached at the end of the fifth stage. This is the sustain level.
- **TR:** Logarithmic event input for controlling the time for the last stage, which is the release. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (i.e. in $\text{dB}_{1\text{ms}}$).
- **Stg:** Event output for the envelope's current stage (1, 2 ...). After the end of the release stage and before a new trigger, the value is 0. With appropriate processing this value in can be used to chain other envelopes, or for the envelope to retrigger itself.
- **Out:** Audio output for the envelope signal.

Filter

Filters make up Reaktor's largest collection of signal processors. You'll find 22 of them here covering everything from standard low-, high-, and bandpass, to emulations of classic synth filters from Sequential and Moog. There's also an allpass filter for reverb and dispersion circuits as well as integrating and differentiating filters.

All of REAKTOR's filters can operate at any frequency, from 0 Hz (constant signal) through the entire audio range right up to the limit set by the sample rate. This means they are all equally suited for audio processing or for smoothing control signals (e.g. portamento). When using a filter to process the input to a port which is of the type that only accepts events (e.g. P as opposed to F) you need to insert an **A to E (perm)** module for conversion.

All the filters and EQs can optionally display their frequency response in a graph on the panel. This can be achieved with the **Visible** option on the **Appearance** page in the module Properties. The size of the display can be set in the Properties using the **Size X** and **Size Y** options. The graph's frequency axis is logarithmic and ranges from 10 Hz to 20 kHz.

HP/LP 1-Pole

Filter

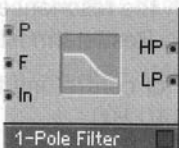


1-pole filter with high pass and low pass outputs (6 dB/octave falloff) and logarithmic control of cutoff frequency.

- **P:** Logarithmic event input for controlling the cutoff frequency. Value in semitones (69 = 440 Hz).
- **In:** Audio signal input for the signal to be filtered
- **HP:** Audio output for the high pass filtered signal
- **LP:** Audio output for the low pass filtered signal

HP/LP 1-Pole FM

Filter

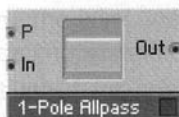


1-pole filter with high pass and low pass outputs (6 dB/octave falloff), logarithmic and linear control of cutoff frequency.

- **P**: Logarithmic event input for controlling the cutoff frequency. Value in semitones ($69 = 440 \text{ Hz}$).
- **F**: Audio input for linear control of the cutoff frequency. Value in Hz. **P** and **F** together determine the oscillator frequency.
- **In**: Audio signal input for the signal to be filtered
- **HP**: Audio output for the high pass filtered signal
- **LP**: Audio output for the low pass filtered signal

Allpass 1-Pole

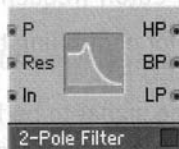
Filter



????????????????????

Multi 2-Pole

Filter



2-pole filter with high pass and low pass outputs (12 dB/octave falloff), band pass (above and below each 6 dB/octave falloff), variable resonance, and logarithmic control of cutoff frequency.

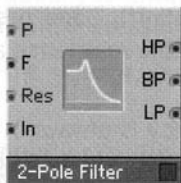
The pass band gain is always 1 (0 dB), whereas the gain at the cutoff frequency increases with the resonance. Caution: Very large amplitudes are generated when **Res** is almost 1.

- **P**: Logarithmic event input for controlling the cutoff frequency. Value in semitones ($69 = 440 \text{ Hz}$).

- **Res:** Event input for controlling the filter's resonance/damping. Range of values 0 (maximal damping, no resonance) to 1 (no damping, maximal resonance, self oscillation!). At high resonance the filter's Q-factor = frequency [Hz] / bandwidth [Hz] $1 / (2 - 2 \text{ Res})$.
- **In:** Audio signal input for the signal to be filtered
- **HP:** Audio output for the high pass filtered signal
- **BP:** Audio output for the band pass filtered signal
- **LP:** Audio output for the low pass filtered signal

Multi 2-Pole FM

Filter



2-pole filter with high pass and low pass outputs (12 dB/octave falloff), band pass (above and below each 6 dB/octave falloff), variable resonance, logarithmic and linear control of cutoff frequency.

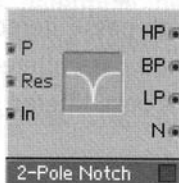
The pass band gain is always 1 (0 dB), whereas the gain at the cutoff frequency increases with the resonance. Caution: Very large amplitudes are generated when **Res** is almost 1.

- **P:** Logarithmic event input for controlling the cutoff frequency. Value in semitones (69 = 440 Hz).
- **F:** Audio input for linear control of the cutoff frequency. Value in Hz. **P** and **F** together determine the oscillator frequency.
- **Res:** Event input for controlling the filter's resonance/damping. Range of values 0 (maximal damping, no resonance) to 1 (no damping, maximal resonance, self oscillation!). At high resonance the filter's Q-factor = frequency [Hz] / bandwidth [Hz] $1 / (2 - 2 \text{ Res})$.
- **In:** Audio signal input for the signal to be filtered
- **HP:** Audio output for the high pass filtered signal

- **BP:** Audio output for the band pass filtered signal
- **LP:** Audio output for the low pass filtered signal

Multi/Notch 2-Pole

Filter



2-pole filter with band reject (notch) output, high pass and low pass outputs (12 dB/octave falloff), band pass (above and below each 6 dB/octave falloff), variable resonance and logarithmic control of cutoff frequency.

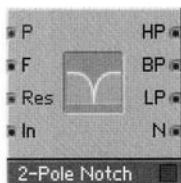
The gain at the cutoff frequency is always 1 (0 dB), whereas the pass band gain decreases with increasing resonance.

At the notch filter output the selected frequency is completely removed.

- **P:** Logarithmic event input for controlling the cutoff frequency. Value in semitones ($69 = 440 \text{ Hz}$).
- **Res:** Event input for controlling the filter's resonance/damping. Range of values 0 (maximal damping, no resonance) to 1 (no damping, maximal resonance, self oscillation!). At high resonance the filter's Q-factor = frequency [Hz] / bandwidth [Hz] $1 / (2 - 2 \text{ Res})$.
- **In:** Audio signal input for the signal to be filtered
- **HP:** Audio output for the high pass filtered signal
- **BP:** Audio output for the band pass filtered signal
- **LP:** Audio output for the low pass filtered signal
- **N:** Audio output for the band reject (notch) filtered signal

Multi/Notch 2-Pole FM

Filter



2-pole filter with band reject (notch) output, high pass and low pass outputs (12 dB/octave falloff), band pass (above and below each 6 dB/octave falloff), variable resonance, logarithmic and linear control of cutoff frequency.

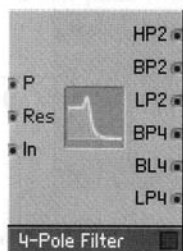
The gain at the cutoff frequency is always 1 (0 dB), whereas the pass band gain decreases with increasing resonance.

At the notch filter output the selected frequency is completely removed.

- **P**: Logarithmic event input for controlling the cutoff frequency. Value in semitones (69 = 440 Hz).
- **F**: Audio input for linear control of the cutoff frequency. Value in Hz. **P** and **F** together determine the oscillator frequency.
- **Res**: Event input for controlling the filter's resonance/damping. Range of values 0 (maximal damping, no resonance) to 1 (no damping, maximal resonance, self oscillation!). At high resonance the filter's Q-factor = frequency [Hz] / bandwidth [Hz] $1 / (2 - 2 \text{ Res})$.
- **In**: Audio signal input for the signal to be filtered
- **HP**: Audio output for the high pass filtered signal
- **BP**: Audio output for the band pass filtered signal
- **LP**: Audio output for the low pass filtered signal
- **N**: Audio output for the band reject (notch) filtered signal

Multi/LP 4-Pole

Filter



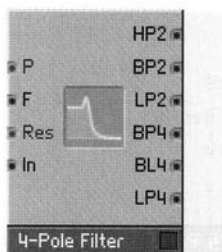
4-pole filter with low pass (24 dB/oct falloff), band pass (12/12 dB/oct) and band/low pass (6/18 dB/oct) outputs, 2-pole high and low pass (12 dB/oct) and band pass (6/6 dB/oct) outputs, variable resonance and logarithmic control of cutoff frequency.

The pass band gain is always 1 (0 dB), whereas the gain at the cutoff frequency increases with the resonance. Caution: Very large amplitudes are generated when **Res** is almost 1.

- **P**: Logarithmic event input for controlling the cutoff frequency. Value in semitones (69 = 440 Hz).
- **Res**: Event input for controlling the filter's resonance/damping. Range of values 0 (maximal damping, no resonance) to 1 (no damping, maximal resonance, self oscillation!).
- **In**: Audio signal input for the signal to be filtered
- **HP2**: Audio output for the 2-pole high pass filtered signal
- **BP2**: Audio output for the 2-pole band pass filtered signal
- **LP2**: Audio output for the 2-pole low pass filtered signal
- **BP4**: Audio output for the 4-pole band pass filtered signal
- **BL4**: Audio output for the 4-pole band/low pass filtered signal
- **LP4**: Audio output for the 4-pole low pass filtered signal

Multi/LP 4-Pole FM

Filter



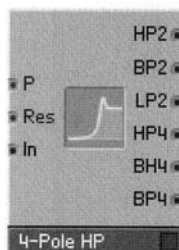
4-pole filter with low pass (24 dB/oct falloff), band pass (12/12 dB/oct) and band/low pass (6/18 dB/oct) outputs, 2-pole high and low pass (12 dB/oct) and band pass (6/6 dB/oct) outputs, variable resonance, logarithmic and linear control of cutoff frequency.

The pass band gain is always 1 (0 dB), whereas the gain at the cutoff frequency increases with the resonance. Caution: Very large amplitudes are generated when **Res** is almost 1.

- **P**: Logarithmic event input for controlling the cutoff frequency. Value in semitones (69 = 440 Hz).
- **F**: Audio input for linear control of the cutoff frequency. Value in Hz. **P** and **F** together determine the oscillator frequency.
- **Res**: Event input for controlling the filter's resonance/damping. Range of values 0 (maximal damping, no resonance) to 1 (no damping, maximal resonance, self oscillation!).
- **In**: Audio signal input for the signal to be filtered
- **HP2**: Audio output for the 2-pole high pass filtered signal
- **BP2**: Audio output for the 2-pole band pass filtered signal
- **LP2**: Audio output for the 2-pole low pass filtered signal
- **BP4**: Audio output for the 4-pole band pass filtered signal
- **BL4**: Audio output for the 4-pole band/low pass filtered signal
- **LP4**: Audio output for the 4-pole low pass filtered signal

Multi/HP 4-Pole

Filter



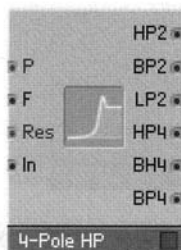
4-pole filter with high pass (24 dB/oct falloff), band pass (12/12 dB/oct) and band/high pass (18/6 dB/oct) outputs, 2-pole high and low pass (12 dB/oct) and band pass (6/6 dB/oct) outputs, variable resonance and logarithmic control of cutoff frequency.

The pass band gain is always 1 (0 dB), whereas the gain at the cutoff frequency increases with the resonance. Caution: Very large amplitudes are generated when **Res** is almost 1.

- **P**: Logarithmic event input for controlling the cutoff frequency. Value in semitones (69 = 440 Hz).
- **Res**: Event input for controlling the filter's resonance/damping. Range of values 0 (maximal damping, no resonance) to 1 (no damping, maximal resonance, self oscillation!).
- **In**: Audio signal input for the signal to be filtered
- **HP2**: Audio output for the 2-pole high pass filtered signal
- **BP2**: Audio output for the 2-pole band pass filtered signal
- **LP2**: Audio output for the 2-pole low pass filtered signal
- **HP4**: Audio output for the 4-pole high pass filtered signal
- **BH4**: Audio output for the 4-pole band/high pass filtered signal
- **BP4**: Audio output for the 4-pole band pass filtered signal

Multi/HP 4-Pole FM

Filter



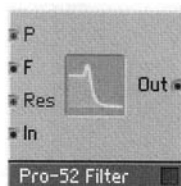
4-pole filter with high pass (24 dB/oct falloff), band pass (12/12 dB/oct) and band/high pass (18/6 dB/oct) outputs, 2-pole high and low pass (12 dB/oct) and band pass (6/6 dB/oct) outputs, variable resonance, logarithmic and linear control of cutoff frequency.

The pass band gain is always 1 (0 dB), whereas the gain at the cutoff frequency increases with the resonance. Caution: Very large amplitudes are generated when **Res** is almost 1.

- **P**: Logarithmic event input for controlling the cutoff frequency. Value in semitones (69 = 440 Hz).
- **F**: Audio input for linear control of the cutoff frequency. Value in Hz. **P** and **F** together determine the oscillator frequency.
- **Res**: Event input for controlling the filter's resonance/damping. Range of values 0 (maximal damping, no resonance) to 1 (no damping, maximal resonance, self oscillation!).
- **In**: Audio signal input for the signal to be filtered
- **HP2**: Audio output for the 2-pole high pass filtered signal
- **BP2**: Audio output for the 2-pole band pass filtered signal
- **LP2**: Audio output for the 2-pole low pass filtered signal
- **HP4**: Audio output for the 4-pole high pass filtered signal
- **BH4**: Audio output for the 4-pole band/high pass filtered signal
- **BP4**: Audio output for the 4-pole band pass filtered signal

Pro-52 Filter

Filter



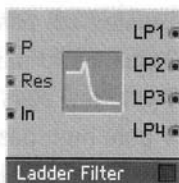
Filter taken from the Pro-52 virtual analog synthesizer. It is a 4-pole low pass filter (24 dB/oct falloff) with variable resonance and logarithmic and linear control of cutoff frequency.

The filter goes into self-oscillation when **Res** approaches the value 1. The amplitude of the self-oscillation is approximately 1.

- **P**: Logarithmic event input for controlling the cutoff frequency. Value in semitones ($69 = 440$ Hz).
- **F**: Audio input for linear control of the cutoff frequency. Value in Hz. **P** and **F** together determine the oscillator frequency.
- **Res**: Event input for controlling the filter's resonance/damping. Range of values 0 (maximal damping, no resonance) to 1 (no damping, maximal resonance, self-oscillation).
- **In**: Audio signal input for the signal to be filtered
- **Out**: Audio output for the 4-pole low pass filtered signal

Ladder Filter

Filter

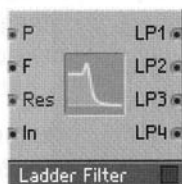


Same as **Ladder Filter FM** but without the frequency modulation input **F**.

????????????????

Ladder Filter FM

Filter



Filter modelled on the classic ladder circuit patented by Bob Moog. It is a 4-pole filter with different low pass outputs: 24 dB/oct, 18 dB/oct, 12 dB/oct and 6 dB/oct falloff. It also has variable resonance and logarithmic and linear control of cutoff frequency.

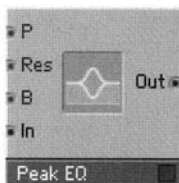
The saturation characteristic of the analog circuit can optionally be simulated by turning on **Distortion** in the properties. When Distortion is enabled, the filter goes into self-oscillation when the value of **Res** is 1 or more. The amplitude of the self-oscillation is approximately 1 when **Res** is just above 1, but can be much larger when driving Res even higher.

The filter also has options for selecting the quality of the simulation: **Standard**, **High** and **Excellent**. The effect is particularly noticeable when distortion is turned on. Of course, the higher quality settings come at the price of increased CPU usage.

- **P**: Logarithmic event input for controlling the cutoff frequency. Value in semitones ($69 = 440$ Hz).
- **F**: Audio input for linear control of the cutoff frequency. Value in Hz. **P** and **F** together determine the oscillator frequency.
- **Res**: Event input for controlling the filter's resonance/damping. Range of values 0 (no resonance) to 1 (maximal resonance, self-oscillation). Values above 1 are possible in Distortion mode.
- **In**: Audio signal input for the signal to be filtered
- **LP1**: Audio output for the 6 dB/oct low pass filtered signal
- **LP2**: Audio output for the 12 dB/oct low pass filtered signal
- **LP3**: Audio output for the 18 dB/oct low pass filtered signal
- **LP4**: Audio output for the 24 dB/oct low pass filtered signal

Peak EQ

Filter

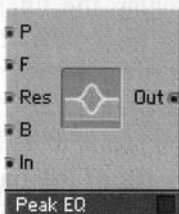


Parametric equalizer with adjustable boost/cut, band width and logarithmic frequency control. With the Peak EQ a specific frequency in the signal and a narrow or wide band of frequencies around it can be amplified or attenuated. More distant frequencies are not affected.

- **P:** Logarithmic event input for controlling the cutoff frequency. Value in semitones ($69 = 440 \text{ Hz}$).
- **Res:** Event input for controlling the filter's resonance (Q-factor). Range of values 0 (minimal resonance, maximal band width) to 1 (maximal resonance, minimal band width). At high resonance the filter's Q-factor = frequency [Hz] / bandwidth [Hz] $1 / (2 - 2 \text{ Res})$.
- **B:** Event input for controlling boost/cut in dB. At value **B** = 0 the signal remains unchanged.
- **In:** Audio signal input for the signal to be equalized
- **Out:** Audio output for the equalized signal

Peak EQ FM

Filter

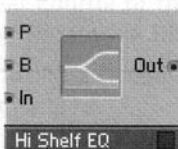


Parametric equalizer with adjustable boost/cut, band width and logarithmic and linear frequency control. With the Peak EQ a specific frequency in the signal and a narrow or wide band of frequencies around it can be amplified or attenuated. More distant frequencies are not affected.

- **P:** Logarithmic event input for controlling the cutoff frequency. Value in semitones ($69 = 440 \text{ Hz}$).
- **F:** Audio input for linear control of the cutoff frequency. Value in Hz. **P** and **F** together determine the oscillator frequency.
- **Res:** Event input for controlling the filter's resonance (Q-factor). Range of values 0 (minimal resonance, maximal band width) to 1 (maximal resonance, minimal band width). At high resonance the filter's Q-factor = frequency [Hz] / bandwidth [Hz] $1 / (2 - 2 \text{ Res})$.
- **B:** Event input for controlling boost/cut in dB. At value **B** = 0 the signal remains unchanged.
- **In:** Audio signal input for the signal to be equalized
- **Out:** Audio output for the equalized signal

High Shelf EQ

Filter



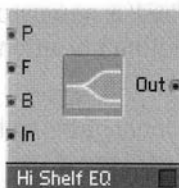
Parametric equalizer with high shelving characteristic, adjustable boost/cut, band width and logarithmic frequency control.

The level of frequencies above the corner frequency is raised or reduced by the set amount. Low frequencies are not affected.

- **P:** Logarithmic event input for controlling the cutoff frequency. Value in semitones ($69 = 440 \text{ Hz}$).
- **B:** Event input for controlling boost/cut in dB. At value **B** = 0 the signal remains unchanged.
- **In:** Audio signal input for the signal to be equalized
- **Out:** Audio output for the equalized signal

High Shelf EQ FM

Filter



Parametric equalizer with high shelving characteristic, adjustable boost/cut, band width and logarithmic and linear frequency control.

The level of frequencies above the corner frequency is raised or reduced by the set amount. Low frequencies are not affected.

- **P:** Logarithmic event input for controlling the cutoff frequency. Value in semitones (69 = 440 Hz).
- **F:** Audio input for linear control of the cutoff frequency. Value in Hz. **P** and **F** together determine the oscillator frequency.
- **B:** Event input for controlling boost/cut in dB. At value **B** = 0 the signal remains unchanged.
- **In:** Audio signal input for the signal to be equalized
- **Out:** Audio output for the equalized signal

Low Shelf EQ

Filter



Parametric equalizer with low shelving characteristic, adjustable boost/cut, band width and logarithmic frequency control.

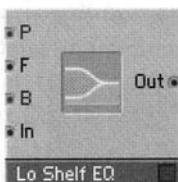
The level of frequencies below the corner frequency is raised or reduced by the set amount. High frequencies are not affected.

- **P:** Logarithmic event input for controlling the cutoff frequency. Value in semitones (69 = 440 Hz).

- **B:** Event input for controlling boost/cut in dB. At value **B** = 0 the signal remains unchanged.
- **In:** Audio signal input for the signal to be equalized
- **Out:** Audio output for the equalized signal

Low Shelf EQ FM

Filter



Parametric equalizer with low shelving characteristic, adjustable boost/cut, band width and logarithmic and linear frequency control.

The level of frequencies below the corner frequency is raised or reduced by the set amount. High frequencies are not affected.

- **P:** Logarithmic event input for controlling the cutoff frequency. Value in semitones (69 = 440 Hz).
- **F:** Audio input for linear control of the cutoff frequency. Value in Hz. **P** and **F** together determine the oscillator frequency.
- **B:** Event input for controlling boost/cut in dB. At value **B** = 0 the signal remains unchanged.
- **In:** Audio signal input for the signal to be equalized
- **Out:** Audio output for the equalized signal

Differentiator

Filter

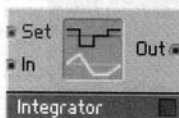


The differentiator gives the slope of the input signal in change units per millisecond. The effect is like a high pass filter where the amplification is proportional to frequency. Unity gain at 159 Hz.

- **In:** Audio signal input for the signal to be differentiated
- **Out:** Audio output for the differentiated signal

Integrator

Filter



Integrator with Reset input. The output value changes with a slope given by the input in change units per millisecond. The effect is like a low pass filter where amplification is proportional to $1/\text{frequency}$. Unity gain at 159 Hz.

- **In:** Audio signal input for the signal to be integrated
- **Set:** Event input for the reset. When an event is received, the output signal is set to the value of the event.
- **Out:** Audio output for the integrated signal

Delay

Reaktor provides six types of delay to accomplish most delay functions. Those include two tap delays, two granular delays, a diffuser (handy for reverb circuits), and a unit delay useful in loop-back processing and physical modeling.

Single Delay

Delay



Polyphonic delay line for audio and event signals. The input signal appears at the output with a delay corresponding to the set time. The delay time is controlled at the **Dly** input.

The upper limit for the delay time can be adjusted in the module's properties dialog window in the **Max Delay Buffer** field (default: 1 second) and affects the memory usage. How big this value can be set depends on the amount of available RAM storage in the computer. At a sample rate of 44.1 kHz you need 172 kB of RAM for each second of buffer length and for each voice. For one minute you need 10 MB. If the Delay is used as an Event Delay (the **In**-port is red indicating that the module is in Event processing mode) you can set the **Max Count Of Buffered Events** at the same place.

This module replaces several modules of former REAKTOR version:

- It behaves like a REAKTOR 3 **Static Delay** if you connect an audio signal to the **In**-input and an event signal to the **Dly**-input. If the delay time does not correspond to an integer number of samples, the output signal is generated by interpolation. The interpolation method can be chosen in the properties. Linear interpolation may reduce high frequency components in the sound somewhat.
- It behaves like a REAKTOR 3 **Modulation Delay** if you connect an audio signal to the **In**-input and an audio signal to the **Dly**-input. The delay time can be continuously modulated by an audio signal. If the delay time does not correspond to an integer number of samples, the output signal is generated by interpolation. The interpolation method can be chosen in the properties. Linear interpolation may reduce high frequency components in the sound somewhat.
- It behaves like a REAKTOR 3 **Event Delay** if you connect an event signal to the **In**-input.

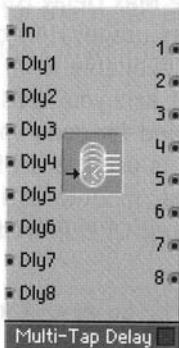
Ports

- **Dly**: Hybrid input for controlling the delay time. Value in milliseconds.
- **In**: Hybrid input for the signal to be delayed.

- **Out:** Hybrid output for the delayed signal.

Multi-Tap Delay

Delay



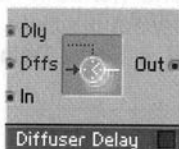
Multi tap delay line for audio signals. If the set delay time corresponds to a non-integer number of sample, interpolation occurs. The interpolation method can be set in the properties.

The outputs are usually connected to a mixer or scanner.

- **In:** Audio input for the signal to be delayed.
- **Dly1...8:** Audio inputs for control of the delay time in milliseconds. Typ range: [0...1000].
- **1...8:** Audio outputs for the delayed input signal. **1** is delayed by time **Dly1**, **2** by **Dly2** etc.

Diffuser Delay

Delay



The diffuser is an all-pass filter containing a delay line with feedback. Its function is to smear (decorrelate) the input signal without emphasizing any frequency components. Its typical use is as a building block for reverb type effects, where you would connect several of these modules in series and give them different delay times of a few milliseconds.

The delay time is controlled at the **Dly** input. The delay time can be continuously modulated by an audio signal. If the delay time does not correspond to an integer number of samples, the output signal is generated by interpolation. The interpolation method can be chosen in the properties. Linear interpolation may reduce high frequency components in the sound somewhat.

The amount of internal feedback is controlled at the **Dffs** input.

When the delay time is set to zero, the Diffuser functions as a 1-pole all pass filter. Like this you can construct a phaser, for example, by connecting several diffusers in series and modulating the **Dffs** parameter.

The upper limit for the delay time can be adjusted in the module's properties dialog window (default: 200 ms) and affects the memory usage.

- **Dly**: Hybrid input for controlling the delay time. Value in milliseconds.
- **Dffs**: Event input for controlling the diffusion coefficient. Range of values: -1 ... 1. **Dffs** = 0 : the module is a pure delay, **Dffs** = 1 : output = input, **Dffs** = -1 : output = -input. The most useful values for **Dffs** are near 0.5.
- **In**: Audio input for the signal to be diffused.
- **Out**: Audio output for the diffused signal.



A delay line and pitch-shifter for audio signals. The input signal appears at the outputs with a delay corresponding to the set time at the **Dly** input, transposed by the amount set at the **P** input in semitones.

The input signal is “chopped-up” into sound particles, whose size is controlled by the *Granularity* input (**Gr**). The “roughness” of the resulting sound can be controlled via the *Smoothness* input (**Sm**). The position of the sound particles in the stereo field is set at the **Pan** input.

The delay time of the **Grain Delay** can be varied without affecting the pitch. Interesting effects are possible in combination with random/noise generators.

In the properties dialog window the playback quality and also the upper limit of the delay time can be set. The available maximum delay time can deviate from the set amount by up to 50 %.

- **P**: Logarithmic audio control input for transposing in semitones (Pitch).
- **Dly**: Audio control input for the delay time in milliseconds.
- **Gr**: Audio control input for the granularity of the re-synthesis process, in milliseconds. This parameter sets the size of the sound particles used for the re-synthesis.
- **Sm**: Audio control input for the *Smoothness* of the re-synthesis process. This affects the formation of the sound particles. Low settings result generally in a rougher sound.
- **Pan**: Audio control input for stereo field positioning (-1 = Left, 0 = Middle, 1 = Right).

- **A:** Audio control input for the output amplitude.
- **In:** Input for the audio signal to be delayed.
- **L:** Audio output for the left channel of the delay.
- **R:** Audio output for the right channel of the delay.
- **Dly:** Polyphonic event output, which is set to the current delay time. This output always produces an event whenever a new sound particle is made.

Grain Cloud Delay

Delay



The Grain Cloud Delay module is similar to the Grain Cloud module (Samplers section) except that instead of operating on audio files loaded into its memory, it processes a constantly changing audio buffer that is fed by the module's bottom input port (labeled "In").

- **Trig:** Event input for triggering the next grain. Values >0 trigger the next grain. Values $=-1$ suppress the next grain (see the Dist input).
- **Frz:** Event input to freeze the audio buffer. Values >0 freeze the buffer.
- **P:** Audio input for logarithmic control of pitch-shift in semitones. The pitch is independent of the speed of traversal. Typical range -20 to 20.
- **D/F:** Audio input setting the direction (if the P input is wired) or linear frequency (if the P input is not wired) of grain playback. The audio buffer plays at the original pitch when $F=1$, in reverse direction when $F=-1$. Typical range -4 to 4. Default 1.
- **PJ:** Audio input for pitch jitter (in semitones). Typical range 0 to 3.
- **PS:** Audio input for logarithmic pitch shift of current grain in semitones. Typical range -3 to 3.
- **Dly:** Audio input for setting the delay time in ms. Allowed range 0 to buffer length.
- **DIJ:** Audio input for delay-time jitter in ms. Allowed range 0 to buffer length.
- **Len:** Audio input for the grain length in ms. Typical range 10 to 100. Default 30.
- **LnJ:** Audio input for grain-length jitter in ms. Typical range 10 to 100. Default 0.
- **Att:** Audio input for setting individual grain playback attack time. Range 0 to 1. Default 0.2.
- **Dec:** Audio input for setting individual grain playback decay time. Range 0 to 1. Default 0.2.
- **Dist:** Audio input for setting the delta-time between grains in ms. Grains are automatically triggered at this rate. Typical range 5 to 100. Default 20.
- **DisJ:** Audio input for delta-time jitter in ms. Typical range 5 to 100. Default 20.
- **Pan:** Audio input for setting the pan position in the stereo field. Range -1 to 1.
- **PnJ:** Audio input for setting the pan jitter. Range 0 to 1.

- **A:** Audio input for amplitude control. Typical range 0 to 1. Default 1.
- **In:** Audio input for the audio signal to be delayed.
- **L:** Polyphonic left channel audio output.
- **R:** Polyphonic right channel audio output.
- **Dly:** Polyphonic delay time output at every grain start.
- **Gtr:** Polyphonic Grain Trigger: 1 when a new grain starts, 0 when it stops.

Unit Delay



Delay

Delays audio signal by one sample period ($1/\text{samplerate}$). Every structure that uses some kind of feedback must have a unit delay in the loop. If no explicit unit delay is there, the program will insert an invisible unit delay somewhere in the loop.

- **In:** Input for the signal to be delayed by one sample period.
- **Out:** Output for the delayed signal.

Audio Modifier

Reaktor's audio processing modules provide various forms of distortion (shaping, clipping, and mirroring, for example). There are also slew limiter, peak detector, sample and hold, and frequency divider modules.

Saturator

Audio Modifier

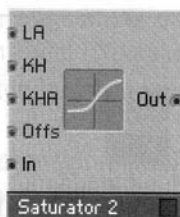


Distortion with a smoothly rounded input-output curve for soft transition to saturation. The output value is limited to ± 2 (reached for input values bigger than ± 4). Very small input values are not changed.

- **In:** Audio input for signal to be saturated
- **Out:** Audio output for saturated signal

Saturator 2

Audio Modifier

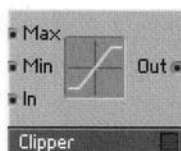


Saturator 2 is a fourth-order asymmetric parabolic saturator and offers control over the saturation curve.

- **LA:** Level asymmetry. At $LA = 0$ the saturation levels for positive and negative signals are equal. For $LA > 0$ the positive level is reduced. At $LA = 1$ it becomes zero. For $LA < 0$ the negative level is reduced. At $LA = -1$ it becomes zero.
- **KH:** Knee hardness. At $KH = 0$ the saturation rises as soft as possible. The full range between zero and the saturation level is used for the rounded curve. With growing values of KH the curve range is reduced to $(1 - KH)$ of the saturation level. At $KH = 1$ the signal is clipped hard at the saturation level.
- **KHA:** Knee hardness asymmetry. At $KHA = 0$ the knee hardness for positive and for negative signals is the same. For $KHA > 0$ the knee hardness for positive signals is reduced. At $KHA = 1$ it becomes zero. For $KHA < 0$ the knee hardness for negative signals is reduced. At $KHA = -1$ it becomes zero.
- **Offs:** This input adds an offset to the input signal and shifts it relative to the saturator curve. For a zero signal the offset is fully compensated at the output.
- **In:** Input for the signal to be distorted.
- **Out:** Output for the distorted signal.

Clipper

Audio Modifier



Distortion with a hard clipping and adjustable upper and lower limit. When the input signal exceeds the upper limit it is clipped to the limit value, similarly for the lower limit. Signal values between the limits are passed unchanged.

- **Max:** Audio input for controlling the upper limit of the signal
- **Min:** Audio input for controlling the lower limit of the signal
- **In:** Audio input for signal to be clipped
- **Out:** Audio output for clipped signal

Mod. Clipper

Audio Modifier



Distortion with a hard clipping and variable limit. When the absolute value of the input signal exceeds limit it is clipped to that value. Smaller signal values are passed unchanged.

- **M:** Audio input for controlling the limit on the absolute value of the signal
- **In:** Audio input for signal to be clipped
- **Out:** Audio output for clipped signal

Mirror 1 Level

Audio Modifier



Distortion by signal value mirroring with adjustable mirror level. Signal values above the mirror level are “reflected” at the level to end up smaller. Signal values below the mirror level are passed unchanged.

- **Max:** Audio input for controlling the mirror level
- **In:** Audio input for signal to be modified
- **Out:** Audio output for the modified signal

Mirror 2 Levels

Audio Modifier



Distortion by double signal value mirroring with adjustable mirror levels. Signal values above the upper mirror level are “reflected” at the level to end up smaller, those below the lower mirror level are “reflected” at that level to end up larger. Signal values in the middle are passed unchanged.

- **Max:** Audio input for controlling the upper mirror level
- **Min:** Audio input for controlling the lower mirror level
- **In:** Audio input for signal to be modified
- **Out:** Audio output for the modified signal

Chopper

Audio Modifier



Chopper Modulator that switches amplification of the input signal between a variable factor and one.

When the modulation signal is positive, the input signal is multiplied by the value **X**. When the modulation signal is negative, the input is unchanged.

- **M**: Audio input for the modulation signal (only the sign is relevant).
- **X**: Audio input for controlling the amplification factor used when **M** is positive.
- **In**: Audio input for the signal to be chopped.
- **Out**: Audio output for the chopped signal

Shaper 1 BP

Audio Modifier



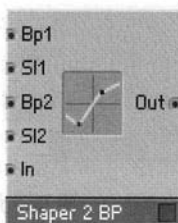
Signal shaper with piecewise linear input/output curve and one breakpoint.

The slope of the curve above the breakpoint can be adjusted. Signal values below the breakpoint are passed unchanged.

- **Bp**: Audio input for controlling the level of the breakpoint
- **Sl**: Audio input for controlling the slope of the upper part of the input/output curve (1 = no change in signal).
- **In**: Audio input for the signal to be shaped.
- **Out**: Audio output for the shaped signal.

Shaper 2 BP

Audio Modifier



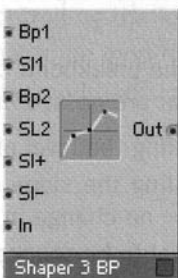
Signal shaper with piecewise linear input/output curve and two breakpoint.

The slope of the curve above the upper and below the lower breakpoint can be adjusted. Signal values between the breakpoints are passed unchanged.

- **Bp1:** Audio input for controlling the level of the upper breakpoint
- **SI1:** Audio input for controlling the slope of the upper part of the input/output curve (1 = no change in signal).
- **Bp2:** Audio input for controlling the level of the lower breakpoint.
- **SI2:** Audio input for controlling the slope of the lower part of the input/output curve (1 = no change in signal).
- **In:** Audio input for the signal to be shaped.
- **Out:** Audio output for the shaped signal.

Shaper 3 BP

Audio Modifier



Signal shaper with piecewise linear input/output curve and three breakpoint.

The slope of the curve above the upper breakpoint, below the lower breakpoint and between the breakpoints and zero can be adjusted. Signal values between the breakpoints are passed unchanged.

- **Bp1:** Audio input for controlling the level of the upper breakpoint
- **Sl1:** Audio input for controlling the slope of the upper part of the input/output curve
- **Bp2:** Audio input for controlling the level of the lower breakpoint
- **Sl2:** Audio input for controlling the slope of the lower part of the input/output curve (1 = no change in signal).
- **Sl+:** Audio input for controlling the slope of the input/output curve between zero and the upper breakpoint (1 = no change in signal).
- **Sl-:** Audio input for controlling the slope of the input/output curve between the lower breakpoint and zero (1 = no change in signal).
- **In:** Audio input for the signal to be shaped
- **Out:** Audio output for the shaped signal

Shaper Parabolic

Audio Modifier



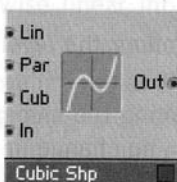
Signal shaper with parabolic (2nd order polynomial) input/output curve.

The linear and square parts of the signal can be adjusted (**Out** = **Lin In** + **Par In**²).

- **Lin:** Audio input for controlling the level of the linear, undistorted part
- **Par:** Audio input for controlling the level of the square, distorted part
- **In:** Audio input for the signal to be shaped
- **Out:** Audio output for the shaped signal

Shaper Cubic

Audio Modifier



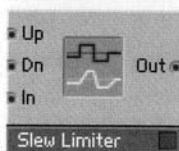
Signal shaper with cubic parabolic (3rd order polynomial) input/output curve.

The linear, square and cubic parts of the signal can be adjusted ($\text{Out} = \text{Lin In} + \text{Par In}^2 + \text{Cub In}^3$).

- **Lin:** Audio input for controlling the level of the linear, undistorted part
- **Par:** Audio input for controlling the level of the square, distorted part
- **Cub:** Audio input for controlling the level of the cubic, distorted part
- **In:** Audio input for the signal to be shaped
- **Out:** Audio output for the shaped signal

Slew Limiter

Audio Modifier



Slew rate limiter with separately adjustable maximum rate for rising and falling signals. The output signal tracks the input signal, but for fast movement and jumps at the input, the output follows with a limited rate of change (ramp) until its value reaches that of the input signal.

- **Up:** Audio input for controlling the maximum slew rate for rising signals. Value in units of 1/sec
- **Dn:** Audio input for controlling the maximum slew rate for falling signals. Value in units of 1/sec
- **In:** Audio input for signal to be slew rate limited
- **Out:** Audio output for slew rate limited signal

Peak Detector

Audio Modifier



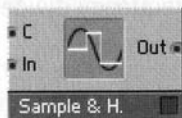
Detector for the peak amplitude. The input signal is rectified and smoothed with an adjustable release time. The attack time is zero.

The result of the Peak Detector's action is that the output value follows the amplitude envelope of the input – use this module as an envelope follower.

- **Rel:** Logarithmic event input for controlling the release time. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (i.e. in dB_{1ms}).
- **In:** Audio input for signal to be detected
- **Out:** Audio output for the signal

Sample & Hold

Audio Modifier



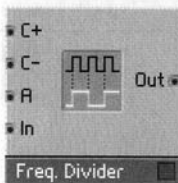
Sample & Hold with clock input.

When the clock signal rises above zero, the current input value is passed to the output and held there until the next clock pulse. The result is a step waveform at the output.

- **C:** Audio input for the clock signal. The input is sampled on a rising edge here.
- **In:** Audio input for the signal to be sampled
- **Out:** Audio output for the sampled signal

Frequency Divider

Audio Modifier



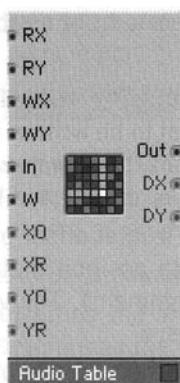
Frequency divider (pulse sub-oscillator) with independently controllable high and low level duration.

A pulse wave is generated by counting the zero crossings of the input signal. The frequency of the output waveform will be a fraction of the frequency of the input waveform. The frequency ratio can be adjusted ($f_{\text{out}} = 2 f_{\text{in}} / (C+ + C-)$). An asymmetric waveform results when **C+** and **C-** have different values.

- **C+:** Audio input for controlling the number of zero crossings of the input signal during the high phase of the output.
- **C-:** Audio input for controlling the number of zero crossings of the input signal during the low phase of the output.
- **A:** Audio input for controlling the amplitude. The output signal moves between +A and -A.
- **In:** Audio input for the signal to be frequency-divided
- **Out:** Audio output for the frequency divided signal.

Audio Table

Oscillator



Holds a table of data values. The table can be read out as an audio signal, audio can be stored in the table and the table's content can be displayed and edited graphically. The table can be 1-dimensional (a row of values) addressed by X, or 2-dimensional (a matrix of rows and columns, or a set of independent rows) addressed by X and Y.

The value at the output is taken from the table by reading at the position given by the inputs **RX** and **RY**. A signal at the module's **In**-port are stored in individual cells of the table according to the write position given by the inputs **WX** and **WY**.

X is the horizontal position from left to right and Y is the vertical position from top to bottom. The count always starts at 0 for the first element.

The module's panel display can show all the data or a limited region of it. Many options in the properties allow customizing the behaviour.

For full details on properties, menus and keyboard shortcuts, please see the section *Table Modules* on page 160.

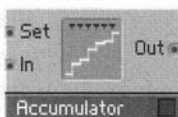
- **RX**: Audio input for the X-position of a table cell from which the data is read.
- **RY**: Audio input for the Y-position of a table cell from which the data is read. This is used in 2D-mode or for addressing the row number if more than one row exists.

- **WX**: Audio input for the X-position of a table cell in which the data is written.
- **WY**: Audio input for the Y-position of a table cell in which the data is written.
- **W**: Audio input for activating table write operation.
- **In**: Audio input for the signal to be written into the table. When the value at **W** is bigger than 0, the value at **In** is written into the table at the position given by **WX** and **WY**.
- **XO**: Event input for the horizontal offset of the displayed data region. **XO** controls the data position that appears in the display (according to View Alignment). The value of **XO** is in the units specified in properties.
- **XR**: Event input for the horizontal range of the displayed data region. **XR** controls how many units of data fit in the display, i.e. it lets you zoom into the data.
- **YO**: Event input for the vertical offset of the displayed data region. **YO** controls the data position that appears in the display (according to View Alignment). The value of **YO** is in the units specified in properties.
- **YR**: Event input for the vertical range of the displayed data region in 2D-mode. **YR** controls how many units of data fit in the display, i.e. it lets you zoom into the data.
- **Out**: Audio output for signal read from the table at the position controlled by the **RX** and **RY** inputs.
- **DX**: Event output for the size of the horizontal table rows in units.
- **DY**: Event output for the size of the vertical table columns in units.

Event Processing

Event modules are used for counting, for logical operations, for splitting and merging control signals, and for timing. You'll also find modules here for shaping and randomizing control signals. Finally there's a full-featured lookup table module-the control equivalent of the audio table in the oscillator section.

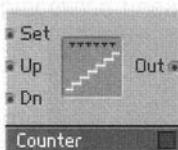
Accumulator Event Processing



Accumulator (sum) for event values. The value of each event at the input is added to the total value stored in the module. The value of the updated sum is sent as an event at the output.

- **In:** Input for the events to be accumulated.
- **Set:** Event input for (re)setting the internal sum. The value of an event at this input determines the new value of the internal sum.
- **Out:** Event output for the accumulated total value.

Counter Event Processing



Counter controlled by events. A positive event at the appropriate input increases or decreases the output value by one.

- **Up:** Input for counting up. Only events with a positive, non-zero value have an effect here, increasing the output by one.
- **Dwn:** Input for counting down. Only events with a positive, non-zero value have an effect here, decreasing the output by one.
- **Set:** Event input for (re)setting the counter value. The value of an event at this input determines the new value of the counter.
- **Out:** Event output for the counter value.

Randomizer

Event Processing



Event randomizer with adjustable spread.

The arriving events are modified with a random positive or negative offset in the given range (**Out** = **In** + X, where $-Rng \leq X \leq Rng$).

- **Rng**: Audio input for controlling the range of the random signal modification
- **In**: Event input for signal to be randomized
- **Out**: Event output for randomized signal

Frequency Divider

Event Processing



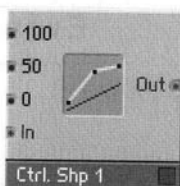
The frequency of the output events is the frequency of the input events divided by a number controlled by the input **N**. The output signal returns to zero after a certain number of input events, depending on the pulse length which is set with the input **W**.

- **N**: Control input for the number of input events per output period. ($N < 2$: no division, $2 \dots 2.99$: division by 2, $3 \dots 3.99$: division by 3, etc.).
- **W**: Control input for the pulse width of the output signal. Range of values: $0 \dots 1$ (0% ... 100%). **W** = 0 : the gate signal at **Out** returns to zero already at the next event at **In**, **W** = 0.5 : the gate signal at **Out** returns to zero after half the period, **W** = 1 : the gate signal at **Out** stays on all the time.

- **In:** Input for the event (clock) signal to be frequency-divided (e.g. MIDI Clock pulses). Only events with a positive, non-zero value have an effect here.
- **Rst:** Event input for resetting the internal counter in order to force a synchronous start (connect to MIDI Start signal if using the **Event Freq. Divider** for MIDI synchronization). Requires an event with positive non-zero value.
- **Out:** Event output for the frequency divided signal

Ctrl. Shaper 1 BP

Event Processing

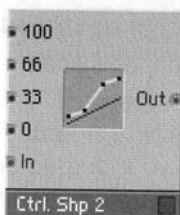


Event signal shaper with piecewise linear input/output curve and one fixed breakpoint. The output produced by linear interpolation between the given points.

- **100:** Output value at $In = 1$ (100%).
- **50:** Output value at the breakpoint at $In = 0.5$ (50%).
- **0:** Output value at $In = 0$ (0%).
- **In:** Event input for the signal to be shaped
- **Out:** Event output for the shaped signal

Ctrl. Shaper 2 BP

Event Processing

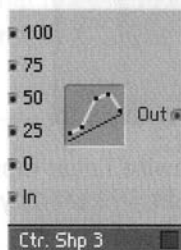


Event signal shaper with piecewise linear input/output curve and two fixed breakpoints. The output produced by linear interpolation between the given points.

- **100:** Output value at **In** = 1 (100%).
- **66:** Output value at the upper breakpoint at **In** = 0.66 (66%).
- **33:** Output value at the lower breakpoint at **In** = 0.33 (33%).
- **0:** Output value at **In** = 0 (0%).
- **In:** Event input for the signal to be shaped
- **Out:** Event output for the shaped signal

Ctrl. Shaper 3 BP

Event Processing



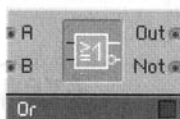
Event signal shaper with piecewise linear input/output curve and three fixed breakpoints. The output produced by linear interpolation between the given points.

- **100:** Output value at **In** = 1 (100%).
- **75:** Output value at the upper breakpoint at **In** = 0.75 (75%).
- **50:** Output value at the middle breakpoint at **In** = 0.5 (50%).
- **25:** Output value at the lower breakpoint at **In** = 0.25 (25%).
- **0:** Output value at **In** = 0 (0%).
- **In:** Event input for the signal to be shaped
- **Out:** Event output for the shaped signal

Logic AND**Event Processing**

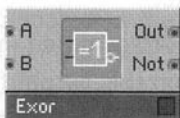
Logic gate for event signals. The output is the logic AND of the two input values, i.e. the output is 1 when both inputs are positive, otherwise the output is 0.

The inputs treat values of zero and below as the logic False state, values above zero are logic True.

Logic OR**Event Processing**

Logic gate for event signals. The output is the logic OR of the two input values, i.e. the output is 1 when one or both inputs are positive, otherwise the output is 0.

The inputs treat values of zero and below as the logic False state, values above zero are logic True.

Logic EXOR**Event Processing**

Logic gate for event signals. The output is the logic EXOR of the two input values, i.e. the output is 1 when one but not both inputs are positive, otherwise the output is 0. So in effect one input inverts the logic value of the other input.

The inputs treat values of zero and below as the logic False state, values above zero are logic True.

Logic NOT

Event Processing

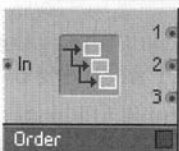


Logic gate for event signals. The output is the logic complement of the input value, i.e. the output is 0 when the input is positive, otherwise the output is 1.

The input treats values of zero and below as the logic False state, values above zero are logic True.

Order

Event Processing

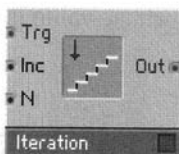


Event-Order. An event arriving at the input is transmitted with the same value on all the outputs, but in a well defined order: first it goes to **1**, then to **2** and finally to **3**. The event travels through ALL the modules in the chain connected to **1**, before going to the first module connected to **2** etc.

- **In:** Input for events to be re-transmitted in a defined order.
- **1:** An event is transmitted on this output first.
- **2:** An event is transmitted on this output second.
- **3:** An event is transmitted on this output third.

Iteration

Event Processing



An event arriving at the **Trg** input is passed to the output and triggers a series of **N** additional output events. Each subsequent event has the value of its predecessor incremented by the value at the **Inc** input. All events are sent before the next audio sample is processed. This module can be used where an iterative event calculation is needed. It helps to avoid the construction of event loops which can lead to instable operation of Reaktor.

- **Trg**: Trigger input. An event at this input triggers $N+1$ events at the output.
- **Inc**: Increment for the value of each subsequent event.
- **N**: Number of additional output events. The value has to be greater than or equal to the integer number (decimal places will be cut).

Separator

Event Processing



All received events are compared to the threshold value and are sent to either one or the other output.

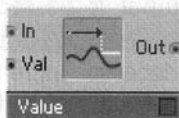
One use for the separator would be to convert a gate signal to a trigger signal by filtering out the zero events (**Thld** = 0, **Hi** = trigger output).

- **Thld**: Audio control input for the threshold value
- **In**: Input for the events to be separated and sent to the two outputs.

- **Hi:** Output for those events whose value is greater than the threshold level.
- **Lo:** Output for those events whose value is less than or equal to the threshold level.

Value

Event Processing



Value-changer for events. Events arriving at the input **In** have their value replaced with the current value at the **Val** input, and are then sent with the new value from the output.

This module can also be used as an event-controlled sample&hold module.

- **In:** Input for events to be changed in value.
- **Val:** Input for specifying the new value for the events.
- **Out:** Event output for the value-changed events.

Merge

Event Processing



Merger for event signals. When more than one wire is connected at the input, the output value is that of the last received input event, irrespective of which wire it came from.

The module has a dynamic in-port management. The number of in-ports can be defined with **Min Num Port Groups** on the **Function** page of the Properties.

Unlike in previous REAKTOR versions subsequent events with the same value will not be filtered out anymore. Therefore use the **Step Filter** module.

Step Filter Event Processing



An event is only passed if the input value is larger/smaller than the previous input value +/- tolerance.

- **In:** Input for events to be filtered.
- **Tol:** Input for the tolerance level.
- **Out:** Event output.

Router M->1 Event Processing



Router multiple to one. The events at the selected input will be let through other events will be filtered. The inputs are selected by the **Pos** input. When **Wrap** mode is selected in the Properties, **Pos** wraps around so that Max+1 is the same as 0, Max+2 is 1 etc..

The module has a dynamic in-port management. The number of in-ports can be defined with **Min Num Port Groups** on the **Function** page of the Properties.

- **Pos:** Input for selecting the thru input.
- **In:** Signal input.
- **Out:** Output for the selected input signal.

Router 1,2

Event Processing



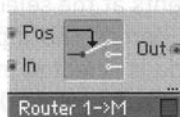
On/off switch for one or toggle switch for two event signals, with control input. The last event passed through is held at the output even while off.

The module has a dynamic in-port management. The number of in-ports can be defined with **Min Num Port Groups** on the **Function** page of the Properties between 1 and 2.

- **Ctrl**: Hybrid input for control of switching. While the value is greater than zero all input events are passed to the output.
- **In**: Event input for the signal to be switched
- **Out**: Event output for the switched signal. While **Ctrl** is smaller than zero, the last value is held at the output.

Router 1->M

Event Processing



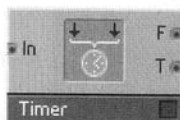
Router one to multiple. The events at the input will be let through to the selected output. The output is selected by the Pos input. When Wrap mode is selected in the Properties, Pos wraps around so that Max+1 is the same as 0, Max+2 is 1 etc..

The module has a dynamic out-port management. The number of out-ports can be defined with **Min Num Port Groups** on the **Function** page of the Properties.

- **Pos**: Input for selecting the thru output.
- **In**: Signal input.
- **Out**: Output for the input signal.

Timer

Event Processing

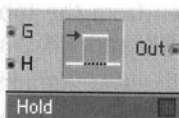


The elapsed time between the two last received events is measured and output as an event. Also, the frequency whose period of oscillation is equal to the measured duration is calculated and output.

- **In:** Polyphonic input for the events whose distance in time is to be measured.
- **F:** Polyphonic event output for the frequency of input events in Hz.
- **T:** Polyphonic event output for the time between events in milliseconds.

Hold

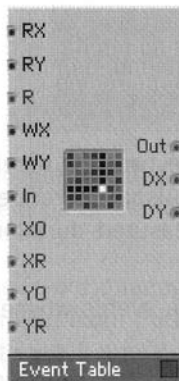
Event Processing



Hold envelope.

When the module is triggered by a positive event at the **G** input, the event's value is held as the output value until the hold time has passed, after which the output jumps back to zero. The module can be triggered at any time, including retriggering during the hold time.

- **G:** Event input for triggering the envelope. Only events with a positive values (not zero) have an effect here.
- **H:** Input for controlling the hold time in milliseconds.
- **Out:** Event output for the envelope signal.



Holds a table of data values. Event values can be read from the table, event values can be stored in the table and the table's content can be displayed and edited graphically. The table can be 1-dimensional (a row of values) addressed by X, or 2-dimensional (a matrix of rows and columns, or a set of independent rows) addressed by X and Y.

The value at the output is taken from the table by reading at the position given by the inputs **RX** and **RY**. Values arriving at the module's **W** input are stored in individual cells of the table according to the write position given by the inputs **WX** and **WY**.

X is the horizontal position from left to right and Y is the vertical position from top to bottom. The count always starts at 0 for the first element.

The module's panel display can show all the data or a limited region of it. Many options in the properties allow customizing the behaviour.

For full details on properties, menus and keyboard shortcuts, please see the section *Table Modules* on page 160.

- **RX**: Audio input for the X-position of a table cell from which the data is read.
- **RY**: Audio input for the Y-position of a table cell from which the data is read. This is used in 2D-mode or for addressing the row number if more than one row exists.

- **R**: Event input for triggering table read operation at the position given by **RX** and **RY**. Each event here produces an event at the **Out** output.
- **WX**: Audio input for the X-position of a table cell in which the data is written.
- **WY**: Audio input for the Y-position of a table cell in which the data is written.
- **W**: Event input for triggering table write operation at the position given by **WX** and **WY**. The value of the data written into the table cell is the value of the event arriving at **W**.
- **XO**: Event input for the horizontal offset of the displayed data region. **XO** controls the data position that appears in the display (according to View Alignment). The value of **XO** is in the units specified in properties.
- **XR**: Event input for the horizontal range of the displayed data region. **XR** controls how many units of data fit in the display, i.e. it lets you zoom into the data.
- **YO**: Event input for the vertical offset of the displayed data region. **YO** controls the data position that appears in the display (according to View Alignment). The value of **YO** is in the units specified in properties.
- **YR**: Event input for the vertical range of the displayed data region in 2D-mode. **YR** controls how many units of data fit in the display, i.e. it lets you zoom into the data.
- **Out**: Event output for data from the cell controlled by the **RX**, **RY** and **R** inputs.
- **DX**: Event output for the size of the horizontal table rows in units.
- **DY**: Event output for the size of the vertical table columns in units.

Auxiliary

If you can't find it anywhere else, it's probably here. First you'll find tape decks for recording and playing back audio files. Then there are modules for managing polyphonic voices, for converting audio signals to control signals, and for reporting the status of various Reaktor processes such as tuning and tempo.

Tapedeck 1-Ch

Auxiliary



1-channel Tapedeck module for recording and playback of audio signals. Audio files can be read from and write to either memory or harddisk depending on the properties setting.

In harddisk mode the files are written into the folder you specify in the Reaktor preferences. Reaktor does sample rate converting on the fly and writes the file in the sample rate which is currently used by your audio system.

In memory mode you can display the waveform of a loaded audio file in the panel by ticking the checkbox **Picture** in the **Appearance** tab in the properties. The whole waveform for the audio file will fit automatically to the **Size** of the picture which you enter in the **Appearance** tab.

Memory mode

By ticking the checkbox **Keep audio only in memory** in the properties the tapedeck module starts working in memory mode and the **Memory** section with the **Save**, **Save as...** and **Reload** buttons becomes active.

The maximum recording time is set with **Max Recording Size (sec)** in seconds. How big this value can be depends on the amount of available RAM storage in the computer. At a sample rate of 44.1 kHz you need 86 kB of RAM for each second of buffer length, for one minute you need 5 MB. If the buffer memory for the TapeDeck has been chosen too large, virtual memory swaps by the operating system can cause significant hard disk activity during recording. This can prevent REAKTOR from processing audio smoothly.

Audio files can be imported for playback with the button **Select File...** in the properties of the TapeDeck module. A file that is already imported can be loaded again by clicking on the **Reload** button, e.g. if it has been changed using a sample editor.

When importing an audio file, the length of the tapedeck's memory buffer is adjusted to match the loaded data. If you later want to make a recording which is longer than this, you will first need to set a bigger value under **Max Recording Size (sec)** in the module's properties dialog window. The loaded file will automatically set to the current sample rate of Reaktor.

Note: Be aware that Reaktor converts all audio files stored in the tape decks in memory mode to the new sample rate whenever Reaktor switches to a new rate. So you should avoid changing the sample rate if your ensemble contains memory based audio samples. If the audio file is stored on harddisk you can use the **Reload** button after changing the sample rate to import the file again.

A recording can be exported using **Save** in the properties. If the file has already been exported, you will be asked if you want to overwrite the file, e.g. if you have made a new recording in the TapeDeck. With **Save as...** you can store the file under a new name.

Harddisk mode

By ticking the checkbox **Stream audio from/to harddisk** in the properties the tapedeck module starts working in harddisk mode. Audio files are written to and read from harddisk directly. You can define an audio file for playback using the **Select File...** button. With the **New File** button you create a new file named "untitled" (if a file with the name is already existing in your Audio folder you have specified in the Reaktor Preferences the new name will contain an additional number). You can edit this name directly in the name field inside the Properties.

If **Value** is activated in the properties, a display for the file name will appear in the panel. By opening the context menu on this box files can also be imported and exported.

- **Rec:** Event input for switching recording mode on and off, e.g. with a gate signal. Start of recording on an event with a positive value. End of recording on an event with negative or zero value or when the maximum recording time is reached.
- **Play:** Event input for switching playback mode on and off, e.g. with a gate signal. Start of playback on an event with a positive value, playback stops on an event with negative or zero value.
- **Lp:** Event input for switching loop playback mode on and off. When this input receives a positive value, playback starts from the beginning when the end of playback is reached. The result is an infinite loop while playback is switched on.
- **In:** Monophonic audio input for the signal to be recorded. Maximum value ± 1 . To record a polyphonic signal a Voice Combiner has to be inserted.
- **Pse:** Pause control input. A value greater than zero pauses, else continues. Typ. range: [0 ... 1].
- **Pos:** Input for setting playback position in ms. Typ. range: [0 ... 20000].
- **Spd:** Variable playback speed of the Tape. Values must be greater than 0. A value of 1 means normal speed. Typ. range: [0.8 ... 1.2].
- **Out:** Audio output for the playback signal or the signal being recorded.
- **Rec:** 1 = Tapedeck is recording, else 0. Typ. range: [0 ... 1].

- **Play:** 1 = Tapedeck is playing back, else 0. Typ. range: [0 ... 1].
- **Wrp:** An Event with value 1 is sent each time the tape is wrapping around the loop.
- **Pos:** 1 = Tapedeck is paused, else 0. Typ. range: [0 ... 1].
- **Time:** current playback/recording position in ms [0 ... <length of sample in ms>].
- **Lng:** Length of recording in ms. Typ. range: [0 ... <length of sample in ms>].

Tapedeck 2-Ch

Auxiliary



This works just like **Tapedeck 1-Ch** but has two channels for recording and playback of audio signals. It imports and exports stereo files.

- **In L:** Monophonic audio input for the signal to be recorded on the left channel. Maximum value ± 1 . To record a polyphonic signal a Voice Combiner has to be inserted.
- **In R:** Monophonic audio input for the signal to be recorded on the right channel. Maximum value ± 1 . To record a polyphonic signal a Voice Combiner has to be inserted.
- **L:** Audio output for the playback signal or the signal being recorded on the left channel.
- **R:** Audio output for the playback signal or the signal being recorded on the right channel.

Audio Voice Combiner

Auxiliary



Audio Voice Combiner. Converts a polyphonic audio signal to monophonic by summing all the voices.

- **In:** Polyphonic audio input for the signal to be combined to mono
- **Out:** Monophonic audio output for the combined signal

Event V.C. All

Auxiliary



Event Voice Combiner. Converts a polyphonic event signal to monophonic by sending the events of all the voices to the same monophonic voice.

- **In:** Polyphonic event input for the signal to be combined to mono
- **Out:** Monophonic event output for the combined signal

Event V.C. Max

Auxiliary



Event Voice Combiner with maximum value selection. Converts a polyphonic event signal to monophonic by finding the voice with the highest current value and sending it to the mono output. A polyphonic gate signal input is necessary to recognize active voices.

- **In:** Polyphonic event input for the signal whose maximum value is to be output
- **G:** Polyphonic event input for the gate signal of the polyphonic instrument
- **Out:** Monophonic event output for the maximum value signal

Event V.C. Min

Auxiliary



Event Voice Combiner with minimum value selection. Converts a polyphonic event signal to monophonic by finding the voice with the lowest current value and sending it to the mono output. A polyphonic gate signal input is necessary to recognize active voices.

- **In:** Polyphonic event input for the signal whose minimum value is to be output
- **G:** Polyphonic event input for the gate signal of the polyphonic instrument
- **Out:** Monophonic event output for the minimum value signal

A to E

Auxiliary



Audio to event converter. The audio signal is sampled using the Control Rate specified in the **Settings** menu and the values are sent out as a stream of events.

A to E (Trig)

Auxiliary

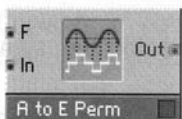


Audio to event converter with trigger input. When the trigger input signal leaves zero to positive values (rising edge) the audio signal is sampled and output as an event.

- **T:** Audio input for triggering the conversion. Triggers on a rising edge
- **In:** Audio input for signal to be sampled and output as events
- **Out:** Event output for the sampled signal

A to E (Perm)

Auxiliary

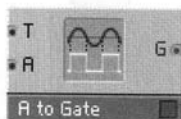


Audio to event converter with permanent conversion at regular intervals and adjustable sampling frequency. The audio signal is sampled with the given frequency and output as a stream of events. When running this module at a high frequency (above 1000 Hz) the CPU-Load caused by event processing can rise significantly. Typically $F = 200$ Hz is sufficient.

- **F:** Audio input for controlling the sample frequency. Value in Hz
- **In:** Audio input for signal to be sampled and output as events
- **Out:** Event output for the sampled signal

A to Gate

Auxiliary



Audio to gate event converter with gate amplitude input. When the trigger signal leaves zero to positive values (rising edge), the gate signal is switched on with an amplitude of the current value of the amplitude input. When the trigger input returns to zero or negative values (falling edge) the gate signal is switched off.

- **T:** Audio input for triggering the gate signal
- **A:** Audio input for controlling the amplitude of the gate events
- **Out:** Event output for the gate event signal.

To Voice

Auxiliary



Monophonic input signals are transmitted to one voice of the polyphonic output signal. The voice number is given by the current value of the **V** input. Signals are discarded when the value at **V** is not a valid voice number.

This is useful for making polyphonic sequencers, for example.

- **V:** Monophonic input for defining the number of the voice to which an event is to be sent. Typ. Range: [1 ... 10].
- **In:** Monophonic hybrid input for signals to be sent to one of the polyphonic voices.
- **Out:** Polyphonic hybrid output. The input signals are transmitted (with their original value) on one voice of this polyphonic signal.

From Voice

Auxiliary



One voice is selected from the polyphonic input signal by the current value of the **V** input. Only events on the selected voice are transmitted to the monophonic output. All events on other voices are discarded.

- **V:** Monophonic input for defining the number of the voice from which events are allowed through. Typ. Range: [1 ... 10].
- **In:** Hybrid input. One voice of this polyphonic signal is sent to the output.
- **Out:** Monophonic hybrid output. The input signal belonging to one voice are transmitted (with their original value) on this monophonic output.

Audio Smoother

Auxiliary



Smoother for monophonic event signals with audio output. Typically, a smoother is connected after a fader or button to get smooth transitions.

The jumps in the value of the input event signal are smoothed to ramps. The **Transition Time** (in milliseconds) can be adjusted in the properties. Exactly after this time has elapsed, the output reaches the same value as the input, assuming that no further jumps occurred at the input. The larger the **Transition Time**, the stronger the smoothing effect.

- **In:** Mono event input for the signal to be smoothed.
- **Out:** Mono audio output for the smoothed signal.

Event Smoother

Auxiliary



Smoother for monophonic event signals. Typically, a smoother is connected after a fader or button to get smooth transitions.

The jumps in the value of the input event signal are smoothed to ramps. The **Transition Time** (in milliseconds) can be adjusted in the properties. Exactly after this time has elapsed, the output reaches the same value as the input, assuming that no further jumps occurred at the input. The larger the **Transition Time**, the stronger the smoothing effect.

During the transition, events are output with the rate selected as the **Control Rate** in the **Settings** menu. The higher the rate, the higher the resolution of the smoothing transition.

- **In:** Mono event input for the signal to be smoothed.
- **Out:** Mono event output for the smoothed signal.

Master Tune/Level

Auxiliary



Controls the overall level and tuning.

- **Tun:** Control input for the master tuning. Scale: 1 semitone per unit. At 0.0 the note a3 is tuned to 440 Hz Typ. Range: [-1 ... 1].
- **Lvl:** Control input for the master level at the output converters. Scale: 1 dB per unit 0.0 = unity gain = 0.0 dB Typ. Range: [-60 ... 0].
- **Tun:** Output for for the master tuning.
- **Lvl:** Output for for the master level.

Tempo Info

Auxiliary

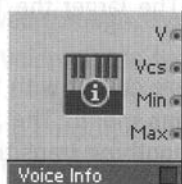


Source for the current Tempo measured in Beats per Second. To get the BPM value, multiply by 60.

- **Out:** Event output for the current tempo in Beats per Second (Hz).

Voice Info

Auxiliary



Voice Info module.

- **V:** Polyphonic output for the ID Number of each Voice [1, 2, 3 ... Voices].
- **Vcs:** Output for the current Number of Voices in the instrument.
- **Min:** Output for the Min Unison Voices setting of the Instrument. This is the minimum number of voices assigned to one key for playing in unison.
- **Max:** Output for the Max Unison Voices setting of the Instrument. This is the maximum number of voices assigned to one key for playing in unison.

Tuning Info

Auxiliary

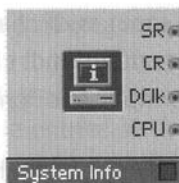


Control for and information about the settings of the instrument's tuning parameters.

- **Tun:** Control for tuning the instrument in semitones.
- **Spr:** Control for the amount of unison spread in semitones.
- **Shft:** Control for the shifting incoming MIDI notes in semitones.
- **Tun:** Output for the tuning of the instrument in semitones.
- **Spr:** Output for the tuning spread amount in semitones.
- **Shft:** Output for the amount of the note shift of incoming notes in semitones.

System Info

Auxiliary



Source for information about the system: Sampling rate (in samples/sec), control rate (in Hz) and CPU load (in %).

- **SR:** Output for the current sampling rate in samples per second.
- **CR:** Output for the current control rate in Hz.
- **DClk:** Output for the current display rate in frames per second. Event is sent just before each display update.
- **CPU:** Output for the current CPU load in percentages.

Note Range Info

Auxiliary



Note Range Info module.

- **Upr:** Input for selecting the upper limit for received midi notes.
- **Lwr:** Input for selecting the lower limit for received midi notes.
- **Upr:** Output for the upper note limit for received midi notes.
- **Lwr:** Output for the lower note limit for received midi notes.

MIDI Channel Info

Auxiliary

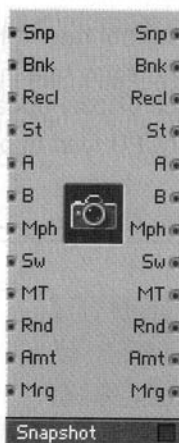


Midi Channel Info module.

- **Ich:** Input for selecting the input midi channel.
- **Och:** Input for selecting the output midi channel.
- **Ich:** Output for the current input midi channel of the instrument.
- **Och:** Output for the current output midi channel of the instrument.

Snapshot

Auxiliary



The Snapshot module allows you to change snapshots and morph between snapshots from within Reaktor. The module has a built-in list processor that works exactly like the List module (see the Panel modules section).

Appearance

The panel representation of this module changes depending on the chosen style on the properties' **Appearance** page. The following styles are available:

- **Button:** Each module in-port creates a button. All buttons will be arrayed vertically in the instrument panel. The currently activated button will be displayed in the Indicator color of the Instrument.
- **Menu:** Each module in-port creates a new entry in a drop down list.
- **Text Panel:** Each module in-port creates a new entry in a list which displays multiple entries at the same time. If you have created more entries than fit into the text panel display specified by the **Size X** and **Size Y** fields on the Appearance tab, you will get scrollbars in the panel.
- **Spin:** Each module in-port creates a new entry in a list. You can switch through the list using a + and a - button right hand of the list entry panel display.

The **Size X** and **Size Y** fields are controlling the display size of the control element in the panel.

Ports

- **Snp:** Audio input for selecting the snapshot to be recalled or stored. Range: 1 ... 128.
- **Bnk:** Audio input for selecting the snapshot bank. Range: 1 ... 16.
- **Recl:** A positive event recalls the snapshot selected by the Snp and Bnk inputs.
- **St:** A positive event stores the snapshot to a position selected by the Snp and Bnk inputs.

- **A:** A positive event takes the current values from the Snp and Bnk inputs to select a snapshot for the Morph position A.
- **B:** A positive event takes the current values from the Snp and Bnk inputs to select a snapshot for the Morph position B.
- **Mrph:** This input controls the morph position between the snapshots loaded for A and B. Value ≤ 0.0 : A, Value $0.0 - 1.0$: morphing between A and B. Value ≥ 1.0 : B.
- **Sw:** Events with negative and zero values set the switches/buttons to their position in snapshot A. Positive events set the switches/buttons to their position in snapshot B.
- **MT:** Event input for the morph time in ms.
- **Rnd:** A positive event triggers the snapshot randomize function.
- **Amt:** Input for the randomization amount. Range: $0.0 \dots 1.0$ ($1.0 = 100\%$)
- **Mrg:** A positive event triggers the random merge function.
- **Snp:** Output for the index of the current snapshot. An event is sent every time a snapshot is recalled or stored.
- **Bnk:** Output for the index of the current snapshot bank. An event is sent every time a snapshot is recalled or stored.
- **Recl:** An event with value = 1.0 is sent when a snapshot was recalled.
- **St:** An event with value = 1.0 is sent when a snapshot was stored.
- **A:** Output for the index of the snapshot of morph position A. An event is sent when a snapshot was recalled or selected for position A.
- **B:** Output for the index of the snapshot of morph position B. An event is sent when a snapshot was recalled or selected for position B.
- **Mrph:** Output for the morph position. Value = 0.0 : A, Value $0.0 - 1.0$: morphing between A and B, Value = 1.0 : B.
- **Sw:** Output for Switch Position A/B. Value = 0.0 if the switches/buttons are set to their position in snapshot A. Value = 1.0 if the switches/buttons are set to their position in snapshot B.
- **MT:** Output for the morph time in ms.

- **Rnd**: An event (with value = 1.0) is sent when the randomize function was triggered.
- **Amt**: Output for the randomization amount. Range: 0.0 ... 1.0 (1.0 = 100%)
- **Mrg**: An event (with value = 1.0) is sent when the random merge function was triggered.

Set Random

Auxiliary



Sets the random seed for the pseudo-random number generator used in all **Event Randomize**, **Slow Random** and **Geiger** modules. Only modules in the same instrument are affected. Each unique value starts a different pseudo-random sequence.

Unison Spread

Auxiliary



Polyphonic event source for a constant value used to spread out parameters for unison voices.

In unison mode, the different voices that play the same note need to have their parameters spread out a little to make them slightly different so that their sum results in a sound that is fatter, and not just louder. For note pitch this happens automatically and is controlled by the **Unison Spread** parameter in the instrument properties. Other parameters can be spread out by adding an offset generated by the Unison Spread module.

The value at the module's input determines the amount of spread. At the output you get a value which is different for each of the voices playing the same note. The value only changes once a new note is played.

Snap Value

Auxiliary



This module stores the input value with a snapshot and outputs the value, when the snapshot is recalled.

- **In:** Input for the value to be stored in a snapshot. The input signal is sampled in the moment when the snapshot is stored. If connected to an event source, input events are passed to the output.
- **Out:** Output for the value stored in the snapshot that has been recalled most recently.

Terminal

Terminals are used to route audio and control signals in and out of Reaktor's Instrument and Macro sub-structures. You can create terminals automatically when you drag a cable to an Instrument or Macro within the structure window while holding down the Ctrl key.

In Port

Terminal



Terminal for audio and event signals to create in-ports for Instruments and Macros.

Out Port

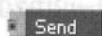
Terminal



Terminal for audio and event signals to create out-ports for Instruments and Macros.

Send

Terminal

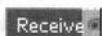


Send terminal for audio and event signals to create a cableless connection within an Instrument.

Each inserted Send module creates an entry in a list of available signal sources in the Properties of a Receive module.

Receive

Terminal



Receive terminal for audio and event signals to create a cableless connection within an Instrument.

Properties - Function page

For each Send module inserted in the same Instrument a list entry is created. The list entry name will be created according to the label of the Send module. The **Up** and **Down** buttons above the list serves for moving a selected entry up or down.

The list contains the following columns which can be sorted by a click on the appropriate header entry:

- **#:** Indicates the list position of the Send module. The default value refers to this number.
- **Label:** The name of the Send module. If you rename a Send module the label in the list will be updated.
- **State:** Indicates if a connection to this send module is possible. Only establish a connection if this field displays **OK**.
- **Use:** Click on this field to set a connection between to the appropriate Send module. An existing connection is indicated by a cross.

The **Mouse Resolution** only applies to the panel control if you choose the **Spin** style on the **Appearance** page, where you can click on the control entry and drag up or down to change the entry.

The **Default** value is used whenever an initialization of the control happens. In this case the entry # in the list will be selected according to the **Default** value entered here.

Appearance

The panel representation of this module changes depending on the chosen style on the properties' **Appearance** page. The following styles are available:

- **Button:** Each module in-port creates a button. All buttons will be arrayed vertically in the instrument panel. The currently activated button will be displayed in the Indicator color of the Instrument.
- **Menu:** Each module in-port creates a new entry in a drop down list.
- **Text Panel:** Each module in-port creates a new entry in a list which displays multiple entries at the same time. If you have created more entries than fit into the text panel display specified by the **Size X** and **Size Y** fields on the Appearance tab, you will get scrollbars in the panel.
- **Spin:** Each module in-port creates a new entry in a list. You can switch through the list using a + and a - button right hand of the list entry panel display.

The **Size X** and **Size Y** fields are controlling the display size of the control element in the panel.

OSC Send

Terminal



OSC messages can be communicated using the **OSC Send** and **OSC Receive** modules. Both modules are dynamic - new in- or out-ports are added by wiring to blank regions in the port areas while holding down the **Ctrl** key (left edge if the Send and right edge of the Receive Module).

Multiple connections to the **OSC Send** module result in OSC messages with multiple arguments. For example, if you wire the **MX** and **MY** outputs of an **XY** Module to an **OSC Send** module, each OSC message will have both the **X** and **Y** value. You must wire multiple outputs from the **OSC Receive** Module to receive multiple arguments-one for each argument up to a maximum of 10.

If you have created more than one in-port for the **OSC Send** module, always the first in-port of the module will trigger the OSC message.

OSC messages can also be sent and received directly from some Reaktor Modules, for example panel controls. The send and receive settings are the same as in the OSC terminal modules.

The module has a dynamic in-port management. The number of in-ports can be defined with **Min Num Port Groups** on the **Function** page of the Properties.

OSC Receive

Terminal



See OSC Send module.

Click on the **PAGE NUMBERS** to **HYPER-jump** to that **page**. Finally, a **PDF manual** that **ROCKS!**

Numerics

4-Ramp 265, 308

4-Step 264

A

about 77

Accumulator 347

AD - Env 301

ADBDR - Env 304

ADBDSR-Env 305

adder 231

ADSR - Env 302, 303, 306, 307

aftertouch 223, 224, 228, 229

Akai import 158

all-pass filter 331

Amp 246

amplifier 246

AR - Env 301

ASIO 18

audio signals 117, 126

Audio Table 269

audio to event converter 365,
366, 367

Audio Units 19

Audio-In module 38, 88, 89

Audio-Out module 38, 89

B

backup 68

basics 35

Beat Loop 287

Bi-Pulse 262

Bi-Saw 249

Button 206

button 120, 137, 206

button, definition 132

C

Ch. Aftertouch 223, 228

Chopper 339

chopper 339

Clipper 337

clipper 337

clipping distortion 337

clock 266, 267

Compare 235

Constant 231

constant 122, 231

context menu 56, 88, 89, 98,
112, 117, 128

control source 130

control source, range of values
121

control sources 120

Controller 223, 228

controller 223, 228

copy 62, 98, 113, 119

Core Audio 18

Counter 347

CPU load 63, 66, 67, 78, 103,
113, 117, 118, 119, 123,
210

create control 117, 120

Crossfade 246

crossfade 243

crossfade with controls 121

Ctrl. Shaper 1 BP 349

Ctrl. Shaper 2 BP 349

Ctrl. Shaper 3 BP 350

Cubase 21

cut 62, 98, 113, 119

D

D - Env 297

DBDR - Env 299

deactivation of modules, auto-
matic 123

Delay 328

delay line 329, 332

delete 56, 62, 98, 113, 119,
125

Differentiator 327

differentiator 328

diffuser 331

Diffuser Delay 330

DirectSound 18

divider 233, 234

double-click 56

Click on the PAGE NUMBERS to HYPER-jump to that page. Finally, a PDF manual that ROCKS!

DR - Env 298
drum-maps 152
DSR - Env 298
DXi 25
DXi 2 18

E

ensemble, definition 87
Envelope 294
envelope 296, 297, 298, 299,
300, 301, 302, 303, 304,
305, 308, 309, 311
equalizer 324, 325, 326, 327
Event 1/x 234
Event Processing 346
event separator 353
event signal 117, 127
event signals 126
Event Table 358
Event x/y 234
exit 61

F

Fader 203
fader 120, 137, 203
fader, definition 131
File menu 59
Filter 312
filter 313, 314, 315, 316, 317,
318, 319, 320, 321, 322,
323

Frequency Divider 344, 348
frequency divider 344
From Voice 368

G

Gate 221
gate 137, 207, 221, 222, 227
Geiger 269
Grain Cloud 285
Grain Delay 332
granularity 332

H

H - Env 296
Hi Shelf EQ 325, 326

Hidden Files 13
hide hints 79
hierarchy 57, 97, 111, 114,
115, 129
High Shelf EQ 325
High Shelf EQ FM 326
hints 79, 124, 126
hold control 109
HP/LP 1-Pole 313
HP/LP 1-Pole FM 313
HR - Env 296

I

Impulse 262
Impulse FM 263
Impulse Sync 263
incremental mode 139
instrument, definition 97
instrument, name 91, 101
instrument, number of voices
102
instrument, properties 90, 100
instruments 128
Integrator 328
integrator 328
inverter 232

K

key control 61, 62, 119, 125,
138
Knob 206
knob 120, 137, 206, 211
knob, definition 131

L

Ladder Filter 322
Ladder Filter FM 323
Lamp 214
lamp 214
Level Lamp 214
Level Meter 217
level meter 217
LFO 37, 246, 294, 295
load measuring 66
lock 136

Click on the PAGE NUMBERS to HYPER-jump to that page. Finally, a PDF manual that ROCKS!

Logic 24
 Logic AND 351
 Logic EXOR 351
 Logic NOT 352
 Logic OR 351
 loop length 270
 low frequency oscillator 294,
 295
 Low Shelf EQ 326, 327
 Low Shelf EQ FM 327

M

macro 43
 macro, definition 111
 master tuning 39
 master volume 39
 measure CPU usage 66, 67
 memory 148
 Meter 216
 meter 216
 MIDI Channel Aftertouch events
 223, 228
 MIDI Controller events 139, 223,
 228
 MIDI controller incremental mode
 139
 MIDI input 35
 MIDI Learn 138, 139
 MIDI Note events 139, 220,
 221, 222, 227
 MIDI note numbers 109
 MIDI Out 227
 MIDI output 140
 MIDI Pitchbend events 220, 227
 MIDI Poly Aftertouch events 139,
 224, 229
 MIDI Program Change events 92,
 104, 142, 225, 229
 MIDI remote control 138
 MIDI source modules 120, 139,
 221
 MIDI source, range of values 121
 mirror 338

Mirror 1 Level 338
 Mirror 2 Levels 338
 missing sound files 149
 Mixer 245
 mixer 245
 MME 18
 Mod. Clipper 337
 module, definition 115
 module, new 115
 mono 113, 118, 364, 365
 Multi 2-Pole 314
 Multi 2-Pole FM 315
 Multi/HP 4-Pole 320
 Multi/HP 4-Pole FM 321
 Multi/LP 4-Pole 318
 Multi/LP 4-Pole FM 319
 Multi/Notch 2-Pole 316
 Multi/Notch 2-Pole FM 317
 multiplier 232
 Multi-Ramp 265
 Multi-Sine 257
 Multi-Step 264
 Multi-Tap Delay 330
 mute 88, 89, 98, 112, 118

N

new ensemble 59
 new module 115
 Noise 268
 noise 268
 note limit 109
 Note Pitch 220
 note pitch 220, 227
 Note Pitch/Gate 227
 note shift 109
 Nuendo 22
 number of voices 102

O

Off Velocity 222
 On Velocity 222
 on/off switch 356
 open 59
 Open Sound Control (OSC) 30

Click on the PAGE NUMBERS to HYPER-jump to that page. Finally, a PDF manual that ROCKS!

Order 352
Oscillator 246
oscillator mode 270
oscillator selection 123
oscillator sync 248, 251, 253,
256, 259, 263
oscillator, impulse 262
oscillator, multi-ramp 265
oscillator, multi-step 264
oscillator, parabol 252, 253,
254
oscillator, pulse 258, 259, 260,
261, 262, 263
oscillator, sawtooth 247, 248,
249
oscillator, sine 255, 256
oscillator, triangle 250, 251,
252

P

panel 41, 57, 98, 129
panel, definition 130
panner 244
Par FM 253
Par PWM 254
Par Sync 253
Parabol 252
parabolic oscillator 254
parametric equalizer 324, 325,
326, 327
parent 129
paste 62, 128
Peak Detector 343
peak detector 343
Peak EQ 324
Peak EQ FM 324
pitch 220
Pitchbend 220, 227
pitchbend 220, 227
pitch-shifter 332
PlugIn 17
Poly Aftertouch 224
polyphonic processing 118

ports, Audio-In 89
ports, instrument 98, 123
ports, macro 112, 123
ports, module 116
preferences 68
Pro-52 Filter 322
processor 66
Program Change 224, 229
program change 142, 225, 229
properties 89, 98, 113, 118,
119, 129
Pulse 258
Pulse 1-ramp 260
Pulse 2-ramp 261
Pulse FM 259
Pulse Sync 259
pulse wave oscillator 262

Q

quality 270
Quantizer 344, 348
quantizer 237

R

Ramp 269
Random 268
random 268
Randomizer 351
randomizer 348
Rectifier 338
rectifier 235
redo 62
Relay 346
relay 356
reverb 331
right mouse button 41, 42, 43,
44, 45, 46, 47, 50, 51, 56
root key 151
Router 357
RTAS 18
run audio 66, 78
S
Sample & Hold 343
sample & hold 344

Click on the PAGE NUMBERS to HYPER-jump to that page. Finally, a PDF manual that ROCKS!


sample analysis 150
 sample editors 150
 Sample Lookup 289
 sample maps 151, 152
 sample rate 64, 78, 92, 148
 Sample-Player 271, 272, 273
 Sampler 271
 sampler 42
 Sampler FM 272
 Sampler Loop 273
 samples 149
 Saturator 335
 saturator 336
 save 59
 save as 59, 98, 113, 128
 Saw FM 247
 Saw Pulse 248
 Saw Sync 248
 Sawtooth 247
 sawtooth oscillator 249
 Scanner 242
 Scope 213
 Sel. Note Gate 222
 Sel. Poly AT 224, 229
 selection 55
 Separator 353
 Sequencer 290, 291
 sequencer 291
 serial number 77
 shaper 339, 340, 341, 342,
 349, 350
 Shaper 1 BP 339
 Shaper 2 BP 340
 Shaper 3 BP 340
 Shaper Cubic 342
 Shaper Parabolic 341
 show hints 79
 Sin/Cos 240
 Sine 255
 Sine FM 255
 Sine Sync 256
 Single Trig. Gate 221

Slew Limiter 342
 slew limiter 343
 Slow Random 295
 smoother 368, 369
 smoothness 332
 snap isolate 145
 snapshot, recall 92, 104, 142
 snapshot, store 223, 224
 snapshots, definition 141
 soft takeover 140
 software version 77
 Sonar 25
 Song Pos 226, 230
 sound files 148, 150
 SoundManager 18
 source module 131, 132, 135,
 203, 206, 211, 221
 source, definition 120
 Square Root 239
 stand-alone program 16
 Start/Stop 225, 230
 Stereo Amp 245
 Stereo Pan 246
 stereo pan 244
 stop audio 66, 78
 structure 50, 57, 98, 113
 structure, definition 114
 sustain control 109
 switch 138, 209
 switch, definition 122, 132
 Sync Clock 226
 synchronization 349
 System menu 66
 System Requirements 14
 system requirements 11
T
 tapedeck 360, 363
 terminal, definition 123
 terminals 98, 112, 123
 threshold 353
 Timer 357
 timer 357

Click on the PAGE NUMBERS to HYPER-jump to that page. Finally, a PDF manual that ROCKS!

To Voice 360
toggle 137, 207
toolbar 136, 139
tooltipssee hints 79
transposing 109
Tri FM 250
Tri Sync 251
Tri/Par Symm 251
Triangle 250
trigger 137, 207, 221
tuning, master 39
tutorial 35
U
undo 61
unison 102, 103, 375
V
Value 354
velocity 221, 222, 227

voice combiner 364, 365
voices, number of 102
volume, master 39
VST 2.0 Plug-in 21
W
WAV export 361
WAV import 361
waveform 148
windows 57
wire, creating 44, 118, 124,
125
wire, definition 124
wire, deleting 125
wiring 47
wiring, rules 125
X
XY 211



www.native-instruments.com | info@native-instruments.com

SUPPORT N.I PLEASE! THEY MAKE THE BEST SOFTWARE ON THE PLANET!



NATIVE INSTRUMENTS
SOFTWARE SYNTHESIS

N.I - THIS IS HOW YOU SHOULD DISTRIBUTE YOUR MANUALS - ENVIRONMENTALLY RESPONSIBLE PDF!