



NATIVE INSTRUMENTS
SOFTWARE SYNTHESIS

REAKTOR 5

Operation Manual

The information in this document is subject to change without notice and does not represent a commitment on the part of Native Instruments Software Synthesis GmbH. The software described by this document is subject to a License Agreement and may not be copied to other media. No part of this publication may be copied, reproduced or otherwise transmitted or recorded, for any purpose, without prior written permission by Native Instruments Software Synthesis GmbH. All product and company names are trademarks of their respective owners.

And also, if you're reading this, it means you bought the software rather than stole it. It's because of people like you that we can continue to create great tools and update them. So, thank you very much.

Users Guide written by: Rick Scott, Marius Wilhelmi, Len Sasso, Stephan Schmitt, Erik Wiegand, James Walker-Hall, Julian Ringel
Special thanks to Henri Hagenow, Dan Santucci and Hanna Felski.

© Native Instruments Software Synthesis GmbH, 2005. All rights reserved.
First Edition, May 2005

REAKTOR is a trademark of Native Instruments Software Synthesis.



Germany

Native Instruments GmbH
Schlesische Str. 28
D-10997 Berlin
Germany
info@native-instruments.de
www.native-instruments.de

USA

Native Instruments USA, Inc.
5631 A Hollywood Boulevard
Los Angeles, CA 90028
USA
info@native-instruments.com
www.native-instruments.com

Table of Contents

1. Introduction	15
1.1. What is REAKTOR?	15
1.2. New/Changed Features in REAKTOR 5	15
1.3. Event Initialization	15
1.4. REAKTOR Core Technology	16
1.6. Changed Primary Modules	18
1.7. New Functions	19
1.8. Changed Functions	20
1.9. Discarded and Reassigned Functions	21
1.10. Opening REAKTOR 3 Ensembles	21
2. Product Authorization	22
2.1. What is the Product Authorization?	22
2.2. Conducting the Product Authorization	23
2.3. Method A: REAKTOR 5 computer has direct access to the internet ...	23
2.4. Method B: Internet Connection on another computer	26
2.5. Method C: No Internet Connection available	28
2.6. Registration support	30
3. Installation under Windows XP	31
3.1. System Requirements and Recommendations	31
3.2. Software Installation	31
3.3. VST plug-in Installation	32
3.4. DXi 2 plug-in Setup	32
3.5. RTAS plug-in installation	33
4. Installation under MacOS X	33
4.1. System Requirements and Recommendations	33
4.2. Installing REAKTOR 5 OS X	34
4.3. MacOS Audio Unit plug-in Installation	34
4.4. RTAS plug-in installation	34
5. Audio Interfaces	35
5.1. Stand-alone Application	35
6. REAKTOR 5 as Standalone	39
6.1. Soundcard (Audio Interface)	39
6.2. Routing	41
6.3. MIDI	42

7. REAKTOR 5 as Plug-in	43
7.1. Automation ID editing	44
7.2. Total Recall	44
7.3. VST 2.0 Plug-In	48
7.3.1. Using the REAKTOR 5 plug-in in Cubase SX 3	48
7.3.2. Using the REAKTOR 5 plug-in in Nuendo 2.0	49
7.4. Audio Units Plug-ins	50
7.4.1. Use in Logic 7.x	50
7.4.2. Use in Digital Performer 4.5	52
7.4.3. Use in Garage Band	53
7.5. DXi 2 plug-in	54
7.5.1. Use in Sonar 4	54
7.7. Using REAKTOR RTAS with Pro Tools 6.x (Mac/Windows)	55
8. Open Sound Control (OSC)	57
8.1. Application areas	57
8.2. OSC System Setup	58
9. First Steps in REAKTOR	61
9.1. Opening and Playing Examples	61
9.2. Your First DIY Synthesizer	72
9.3. Your First DIY Structure	84
10. Basic Operation	90
10.1. Mouse	90
10.2. Context Menus	91
10.3. Key Commands	91
10.4. Ensemble Panel and Structure Windows	91
11. Menus	93
11.1. File Menu	93
11.2. Edit Menu	95
11.3. Settings Menu	97
11.4. System Menu	99
11.8. View Menu	106
11.5. ? Menu	114
12. REAKTOR Toolbars	115
12.1. Main Toolbar	115
12.2. Ensemble Panel Toolbar	117
12.3. Structure Toolbar	119
13. The Browser	120
13.1. Accessing Files	121
13.2. Auditioning Files	124

14. Ensemble	125
14.1. Ensemble Structure Window	127
14.2. Ensemble Panel Window.....	129
14.3. Ensemble Properties Dialog	130
15. Instruments.....	138
15.1. Adding Instruments to an Ensemble.....	138
15.2. Ports.....	139
15.3. Context Menu.....	139
15.4. Instrument Header.....	140
15.5. Instrument Properties	142
16. Primary Macros	155
16.1. What is a Primary Macro?	155
16.2. Adding Macros to a Structure	156
16.3. Ports.....	157
16.4. Context Menu.....	158
16.5. Macro Properties	159
17. Primary Structures	164
17.1. What is a Primary Structure?.....	164
17.2. Modules	165
17.3. Source Modules.....	170
17.4. Switches.....	172
17.5. Terminals.....	173
17.6. Wires	173
17.7. Signal Processing in REAKTOR	176
17.8. Context Menu.....	181
18. Panel Editing.....	182
18.1. What Is a Panel?.....	182
18.2. What are Panel Controls?	183
18.3. Panel Controls.....	184
18.4. Panel Control Skins.....	191
18.5. Connection Properties of Panel Controls	196
18.6. Editing the Panels.....	199
19. Panel Operation	200
19.1. Mouse Control	200
19.2. Using Keys to Change Control Settings	204
19.3. MIDI Control	204
19.4. MIDI Out	206
19.5. Customized Panels.....	206
20. Snapshots.....	212

21. Sampling and Resynthesis	222
21.1. Sample Management.....	222
21.2. Sample Maps	225
21.3. Sample Map Editor	228
21.4. Akai Import.....	238
22. Table Modules.....	240
22.1. Properties	240
22.2. Context Menu.....	248
22.3. Advanced Operation	252
23. “Classic Modular” Macro Collection	253
23.1. Display	254
23.2. MIDI	255
23.3. Mixer/Amp	256
23.4. Oscillator	259
23.5. Sampler.....	260
23.6. Sequencer	261
23.7. LFO, Envelope	267
23.8. Filter	269
23.9. Delay.....	271
23.10. Audio Modifier.....	272
23.11. Event Processing	273

Module Reference	275
Panel	277
Fader	277
Knob.....	279
Button	280
List.....	281
Switch.....	282
Lamp	283
Level Lamp.....	284
RGB Lamp	285
Meter	285
LevelMeter.....	286
Picture	286
Multi Picture.....	287
Text	288
Multi Text	288
XY	289
Scope	290
Multi Display and Poly Display	291
Mouse Area	293
Stacked Macro	295
IC Send.....	296
IC Receive	296
MIDI In	297
Note Pitch	297
Pitchbend.....	297
Gate	298
Single Trig. Gate.....	298
Sel. Note Gate	298
On Velocity	299
Off Velocity.....	299
Controller	299
Ch. Aftertouch	300
Poly Aftertouch	300
Sel. Poly AT	300
Program Change.....	301
Start/Stop	301
1/96 Clock.....	301
Sync Clock	302
Song Pos.....	302
Channel Message	302

MIDI Out.....	304
Note Pitch/Gate	304
Pitchbend.....	304
Controller	304
Ch. Aftertouch	305
Poly Aftertouch	305
Sel. Poly AT	305
Program Change.....	306
Start/Stop	306
1/96 Clock.....	306
Song Pos.....	307
Channel Message	307
Math.....	309
Constant	309
Add	309
Subtract.....	310
Invert, -X.....	310
Multiply.....	310
$a * b + c$	311
Reciprocal $1/x$	311
Divide x/y	311
Modulo $x \% y$	312
Rectifier	312
Rect./Sign	312
Compare	313
Compare/Equal	313
Quantize.....	314
Expon. (A)	314
Expon. (F)	314
Log (A).....	315
Log (F).....	315
Power x^y	315
Square Root.....	316
$1 / \text{Square Root}$	316
Sine.....	316
Sine/Cos	317
Arcsin	317
Arccos	317
Arctan.....	318

Signal Path	319
Selector/Scanner	319
Relay 1,2.....	319
Crossfade	320
Distributor/Panner	320
Stereo Pan	321
Amp/Mixer.....	321
Stereo Amp/Mixer.....	322
Oscillator	323
Sawtooth.....	323
Saw FM.....	323
Saw Sync	324
Saw Pulse	325
Bi-Saw	325
Triangle	326
Tri FM.....	326
Tri Sync.....	327
Tri/Par Symm.....	327
Parabol	328
Par FM.....	328
Par Sync	329
Par PWM.....	330
Sine.....	330
Sine FM	331
Sine Sync	331
Multi-Sine	332
Pulse	333
Pulse FM.....	334
Pulse Sync	334
Pulse 1-ramp	335
Pulse 2-ramp	336
Bi-Pulse	337
Impulse.....	337
Impulse FM	338
Impulse Sync	338
Multi-Step	339
4-Step	339
5-Step	340
6-Step	340
8-Step	340

Multi-Ramp	340
4-Ramp.....	340
5-Ramp	341
6-Ramp	341
8-Ramp	341
Ramp.....	341
Clock	342
Noise	342
Random	343
Geiger	343
Samplers	344
Sampler	345
Sampler FM.....	346
Sampler Loop	347
Grain Resynth	349
Grain Pitch Former	353
Grain Cloud	357
Beat Loop	359
Sample Lookup	361
Sequencer	362
Sequencer	362
6-Step	362
8-Step	363
12-Step.....	363
16-Step.....	363
Multiplex 16	363
LFO, Envelope.....	365
LFO	365
Slow Random.....	366
H - Env	366
HR - Env	367
D - Env	368
DR - Env	368
DSR - Env	369
DBDR - Env	369
DBDSR-Env	370
AD - Env	371
AR - Env	371
ADR-Env	372

ADSR - Env	373
ADBDR - Env	373
ADBDSR-Env	374
AHDSR - Env	375
AHDBDR - Env	376
4-Ramp.....	377
5-Ramp	378
6-Ramp	380
Filter.....	382
HP/LP 1-Pole.....	382
HP/LP 1-Pole FM	383
Allpass 1-Pole.....	383
Multi 2-Pole.....	384
Multi 2-Pole FM	384
Multi/Notch 2-Pole	385
Multi/Notch 2-Pole FM.....	386
Multi/LP 4-Pole	387
Multi/LP 4-Pole FM	388
Multi/HP 4-Pole	389
Multi/HP 4-Pole FM	390
Pro-52 Filter	391
Ladder Filter	391
Ladder Filter FM	392
Peak EQ.....	393
Peak EQ FM	393
High Shelf EQ.....	394
High Shelf EQ FM	394
Low Shelf EQ.....	395
Low Shelf EQ FM	396
Differentiator	396
Integrator	397
Delay	398
Single Delay.....	398
Multi-Tap Delay	399
Diffuser Delay	400
Grain Delay.....	401
Grain Cloud Delay.....	402
Unit Delay	404

Audio Modifier.....	405
Saturator	405
Saturator 2	405
Clipper	406
Mod. Clipper	407
Mirror 1 Level	407
Mirror 2 Levels	407
Chopper	408
Shaper 1 BP	409
Shaper 2 BP	409
Shaper 3 BP	410
Shaper Parabolic	411
Shaper Cubic	411
Slew Limiter	412
Peak Detector	412
Sample & Hold	413
Frequency Divider	413
Audio Table	414
Event Processing.....	416
Accumulator	416
Counter	416
Randomizer	417
Frequency Divider	417
Ctrl. Shaper 1 BP	418
Ctrl. Shaper 2 BP	418
Ctrl. Shaper 3 BP	419
Logic AND	419
Logic OR	420
Logic EXOR	420
Logic NOT	420
Order	421
Iteration	421
Separator	422
Value	422
Merge	423
Step Filter	423
Router M->1	423
Router 1,2	424
Router 1->M	424
Timer	425

Hold	425
Event Table	426
Auxiliary	428
Tapedeck 1-Ch	428
Tapedeck 2-Ch	431
Audio Voice Combiner	431
Event V.C. All	432
Event V.C. Max	432
Event V.C. Min	433
A to E	433
A to E (Trig)	433
A to E (Perm)	434
A to Gate	434
To Voice	435
From Voice	435
Voice Shift	436
Audio Smoother	437
Event Smoother	437
Master Tune/Level	438
Tempo Info	438
Voice Info	438
Tuning Info	439
System Info	439
Note Range Info	440
MIDI Channel Info	440
Snapshot	441
Set Random	443
Unison Spread	443
Snap Value	444
Snap Value Array	444
In Port	446
Out Port	446
Send	446
Receive	446
IC Send	448
IC Receive	448
OSC Send	449
OSC Receive	449
Appendix	450
Index	451

1. Introduction

1.1. What is REAKTOR?

REAKTOR is a powerful and flexible program that turns your computer into a professional-strength synthesis, sampling, and audio-processing system. With REAKTOR's completely modular structure, you can build virtually any digital audio device that you can imagine. From relatively simple analog synths to large modular systems, from basic sample players to exotic granular (re)samplers, 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 kinds. 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 it enables you to get right down to the business of making music.

If something in the library doesn't do exactly what you need, its modular structure and its control elements are accessible for you to modify. Nothing is hidden. And there's an active user community and online library with new instruments and effects being added all the time. In short, you decide how to use REAKTOR. Fire up a pre-built ensemble (combination of instruments) today, add some snapshots (presets) and make some modifications tomorrow, build your own instrument from the ground up the next day. Just get started!

1.2. New/Changed Features in REAKTOR 5

REAKTOR 5 represents a major advancement in flexibility, power, and sonic potential over REAKTOR 4. The following sections present a short overview of new and changed features in REAKTOR 5.

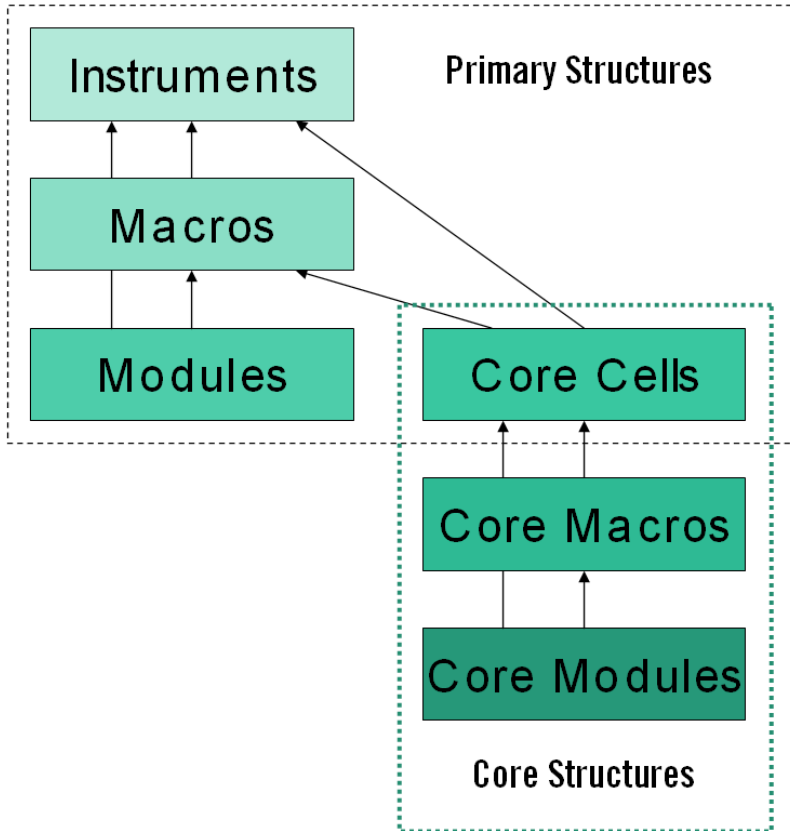
1.3. Event Initialization

REAKTOR 5 has a new initialization scheme for event inputs that is used if the REAKTOR 4 Legacy Mode option is disabled (in the Ensemble Properties dialog). We strongly recommend that you disable REAKTOR 4 Legacy Mode in your ensembles for the sake of future compatibility!

1.4. REAKTOR Core Technology

The biggest change to REAKTOR 5 is that it provides two levels of functionality: the primary level and the core level.

The primary level comprises the instruments, macros, and modules as they are known from REAKTOR 4.



The core level, also known as the REAKTOR Core, comprises three new objects: core cells, core macros, and core modules. A core cell (*.rcc file) is a macro/module hybrid that serves as a bridge between the primary and core levels of REAKTOR. Just as primary-level structures consist of primary macros (*.mdl files) and primary modules, core-cell structures consist of core macros (*.rcm files) and core modules.

Note that primary and core macros are stored in external files (*.mdl and *.rcm, respectively), but that primary and core modules are built into the REAKTOR program. For this reason, modules are referred to as built-in modules.

Core cells, and the core macros and modules they contain, are built upon new concepts of signal propagation and runtime compilation. Using REAKTOR Core technology enables builders to design sophisticated signal-processing structures, that would not have been possible in REAKTOR 4. For a comprehensive introduction to REAKTOR Core technology, please refer to the separate REAKTOR Core manual.

1.5. New Primary Modules

There are no new primary modules for audio generation and processing in REAKTOR 5, because this type of low-level functionality is, from now on, realized at the core level (i.e. within core cells), not at the primary level. There is a comprehensive, ever-growing library of core cells, core macros, and core modules for low-level DSP processing.

The new primary modules focus on the user interface, data storage, voice routing. MIDI input/output, and internal connections.

The new modules are:

- **Mouse Area (Panel)** - enables other modules (such as Multi Display and Poly Display) to process mouse actions (button clicks, mouse drags, changes in position, etc.).
- **Multi Display** and **Poly Display (Panel)** - enable REAKTOR users to generate and manipulate multiple graphical objects (rectangles, pictures, animations, etc.).
- **Stacked Macro** and **Panel Index (Panel)** - enable multiple macros to share the same display area in an instrument panel, where one macro is displayed at a time.
- **Channel Message (MIDI In)** and **Channel Message (MIDI Out)** - receive/send all types of MIDI channel messages from/to external MIDI devices (keyboard, sequencer, file, etc.) or internal instruments.
- **Voice Shift (Auxiliary)** - shifts specified input voices (e.g. 1, 2) to specified output voices (e.g. 3, 4).
- **Snap Value Array (Auxiliary)** - stores/recalls arrays of values to/from the edit buffer and snapshots.

- **IC Send** (Terminal) and **IC Receive** (Terminal) - send/receive monophonic event signals anywhere in the ensemble. IC stands for internal connection.
- **Numeric Readout** – is a panel element to display numeric values.

For detailed information on each of these modules, see the **Primary Modules Reference**.

1.6. Changed Primary Modules

The appearance and functionality of several REAKTOR 4 modules has been changed in REAKTOR 5:

- **Invert, Rectify** (Math), and **Merge, Order, Value, Logic AND, Logic OR, Logic XOR, Logic NOT** (Event Processing) - the structure icons for all of these modules are different from those in REAKTOR 4.
- **Meter, Lamp, Multi Picture, Multi Text** (Panel), **MIDI In Controller, MIDI Out Controller** - the Internal Connections list in the Properties dialog has been removed from all these modules; Internal connections are now established by the IC Send and IC Receive modules.
- **Snap Value** (Auxiliary) - can now be run in monophonic or polyphonic mode. (In REAKTOR 4, **Snap Value** is a monophonic-only module.)
- **Panel Controls** (Panel) - the functionality of several REAKTOR 4 panel control modules has been changed in REAKTOR 5. Control and port labels can be edited in panel view (in unlocked mode). Control values can be set in panel view (in locked mode). Most panel controls can have panel skins. There are new options for instrument and primary macro background pictures. For detailed information on each of these modules, see the **Primary Modules Reference**.

1.7. New Functions

There are several new functions in REAKTOR 5:

- **Panelsets** - an enhanced replacement for REAKTOR 4 screensets.
- **Bookmarking a structure** - you can bookmark a structure so that you can jump straight to it from any other structure in the ensemble.
- **Locking an instrument's voice allocation settings** - an instrument's voice allocation settings (Voices, Max Unison V, and Min Unison V) can now be locked by turning on the Lock Voices option (in the Properties dialog).
- **Voice & MIDI Slave option** - an instrument's voice allocation and MIDI In settings can now be controlled from another instrument in the ensemble.
- **Panel skins** - REAKTOR 5 enables you to customize the appearance of several panel controls by applying skins to them: faders, knobs, buttons, lists, switches, Receive modules, lamps, and meters.
- **Instrument and macro borders** - you can now add borders (blank margins) to instrument panels and framed primary macros.
- **Auditioning audio files in the Browser and Sample Map Editor** - the REAKTOR 5 Browser and Sample Map Editor both support audio-file auditioning (pre-listening).
- **Initialization** - REAKTOR 5 has a new initialization scheme for event inputs that is used if the **REAKTOR 4 Legacy Mode** option is disabled (in the Ensemble Properties dialog).
- **User Content folders** - during installation, REAKTOR 5 creates separate folders for its system files (ensembles, instruments, primary macros, core cells, core macros), and for user files that are created/maintained by the user (ensembles, instruments, primary macros, core cells, core macros, audio, imported files, pictures, snapshots, tables).
- **Deleting wires** - wires can now be deleted by dragging the mouse from the input port to which the wire is connected to a blank part of the structure.
- **Debug option - Show Event Initialization Order** numbers modules in a structure to show their initialization sort order.
- **CPU peak meter** – The CPU meter has been extended. It now also features a bar to show the average CPU drain (white), peak above average (yellow), CPU overload (red).

1.8. Changed Functions

Several REAKTOR 4 functions have been changed in REAKTOR 5:

- **Ensemble Panel window** - there is now only one panel window, the former Ensemble Panel window. All instrument panels reside within the Ensemble Panel window.
- **Structure windows** - in order to minimize Structure window clutter, REAKTOR 5 displays all structures (ensemble, instrument, primary macro, core cell, and core macro) in the same Structure window. You can bypass this feature and open a structure in a separate window by Alt+double-clicking the structure icon, or WindowsXP: Right-clicking / OS X: Ctrl+clicking the icon and selecting **Structure Window** from the context menu.
- **Main toolbar** - several aspects of the REAKTOR 4 Main toolbar have been changed in REAKTOR 5. The number of Main toolbar elements has been reduced, because the Ensemble Panel window and Structure windows now have their own toolbars. In the OS X implementation of REAKTOR 5, the toolbar is now displayed as a toolbox that can be placed anywhere on the screen, in order to keep the window headers visible.

There are now two MIDI activity lamps: External MIDI In and External MIDI Out. During the compilation of a core structure, the CPU load indicator changes to a compilation progress bar.

- **Ensemble Panel and Structure toolbars** - Ensemble Panel window and Structure windows now have their own toolbars, each with a set of the most commonly used functions in that window.
- **Instrument header** - Several aspects of the REAKTOR 4 Instrument header have been changed in REAKTOR 5. The A, B, and Minimize buttons have been moved to the left. The panel Lock/Unlock function now has its own button (wrench icon). The Mute and Solo buttons have been removed. There are now four MIDI activity lamps: External and Internal MIDI In, and External and Internal MIDI Out. The In and Out drop-down menus provide access to all of the input and output connections (MIDI and wiring) of the instrument.
- **Browser item access** - The REAKTOR 5 Browser provides dedicated buttons that enable you to fast access system and custom folders.

1.9. Discarded and Reassigned Functions

Several REAKTOR 4 functions have been discarded or reassigned in REAKTOR 5:

- There are no longer separate panels for instruments. All instrument panels are displayed in the Ensemble Panel window. The new Panelset bar provides easy (one-click) access to all of an ensemble's instrument panels.
- REAKTOR 4 screensets (storage slots for ensemble layouts) have been replaced by REAKTOR 5 panelsets (see **New Functions, panelsets**).
- The Browser no longer supports wiring (this has been reassigned to the In and Out menus in an instrument's panel header), structure browsing, and module loading.
- The internal MIDI connections of an instrument are no longer set in the instrument's Properties dialog; they are set in the instrument panel header's In and Out menus.

1.10. Opening REAKTOR 3 Ensembles

Ensembles saved with REAKTOR 3 will not open in REAKTOR 5 unless the REAKTOR 3 USB copy protection key is plugged in. If you have the key, install it and plug it into your USB port, then open the REAKTOR 3 ensembles in REAKTOR 5 and save them as REAKTOR 5 files. Once you've done this, you'll be able to open the ensemble files without using the key. There is also a Batch Processing function to perform the conversion of many files at once.

2. Product Authorization

Part of the REAKTOR 5 installation is a **Product Authorization** which has to be fulfilled in order to make permanent use of the software. We recommend that you take notice of this chapter first, then proceed with the software installation as described in the following chapters and finally return to this chapter.

2.1. What is the Product Authorization?

The **Product Authorization** includes a full registration. After having completed the authorization, you can make use of all online services related to the registered product. On the Native Instruments website you can read online tutorials, get technical support, participate in the NI forums and download updates.

REAKTOR 5 requires the **Product Authorization** in order to use the software permanently. You can run REAKTOR 5 for 30 days without any limitations. As long as the software runs unauthorized a message at every program start reminds you that the authorization has not been completed yet and indicates how many more days the software is running without an authorization.

The **Product Authorization** is performed by a small application called **Registration Tool**. The **Registration Tool** generates a so called **System ID** which serves as request code for receiving an **Authorization Key**. The **System ID** is based on some hardware components of your computer system, the operating system version and the serial number you have entered at the REAKTOR 5 installation.

Note: Exchanging your audio card, MIDI interface or external equipment will not start the request for a new **Authorization Key**. Only exchanging a basic hardware component in your computer or installing a new operating system might produce a new **Product Authorization** request. In this case the **Registration Tool** will reflect the change by displaying a new **System ID** and you have to repeat the **Product Authorization**.

The **System ID** has to be sent to Native Instruments in order to receive the **Authorization Key** which allows the permanent use of REAKTOR 5. Since the **Product Authorization** is part of the license agreement REAKTOR 5 will stop launching after 30 days if it was not authorized until then. Of course, it is also possible to complete the **Product Authorization** after 30 days. In this case the software will launch again as soon as the **Product Authorization** has been completed.

Note: The data transfer at the online Product Authorization is done via a secure connection using 128 bit encryption. Native Instruments keeps your personal data like email and postal address in confidence. No data will be passed to a third party.

THIS IS YOUR SYSTEM ID

The System ID is based on certain hardware components of your computer system. During the registration process it will be sent to NI in order to generate an Authorization Key, which will be sent to you per email. (Exchanging your audio card, MIDI interface or external equipment will not require a new Authorization Key.)

COPY

FAQ

1 REGISTRATION

The first step is to send your System ID to Native Instruments and register your product. Choose one of the following options:

A. DIRECT ONLINE REGISTRATION
If you have direct access to the Internet from this computer, this is the fastest way to register your product.

B. SEND REGISTRATION FROM ANOTHER COMPUTER
Use this option if you have access to the Internet via another computer.

C. OFFLINE REGISTRATION
This allows you to fill out an HTML form that can be printed and sent to NI.

REGISTER NOW

OR

SAVE REGISTRATION FILE

OR

FILL OUT FORM

2 AUTHORIZATION KEY

Once you receive the email with your authorization key please copy the entire key and paste it into the field below, or open the file that was attached to the mail.

PASTE FROM CLIPBOARD

OR

OPEN FILE

Registration Tool

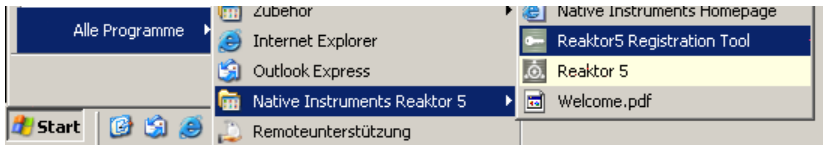
2.2. Conducting the Product Authorization

Native Instruments has set a high value on making the authorization procedure as easy and comfortable as possible. In the following sections we describe three methods of conducting the **Product Authorization**. We recommend **Method A** and **Method B** since these result in the easiest and fastest way of receiving the **Authorization Key**.

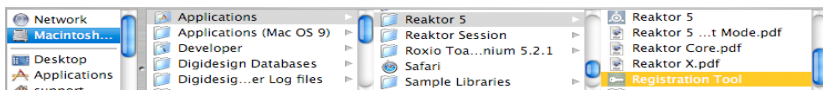
2.3. Method A: REAKTOR 5 computer has direct access to the internet

Important: This method requires a valid email address to complete the **Product Authorization**, since the registration code will be sent to you by email.

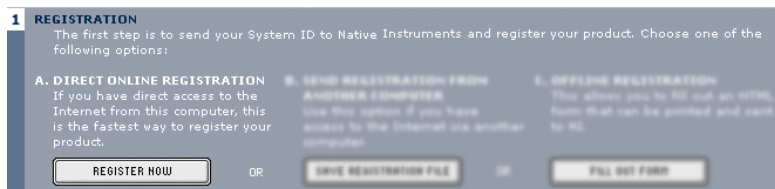
- **Windows:** Start the Registration Tool from the start menu (Native Instruments REAKTOR 5 ⇔ REAKTOR 5 Registration Tool) or from the REAKTOR 5 installation folder (default path: C:\Program Files\Native Instruments\REAKTOR 5\).



- **MacOS:** Start the Registration Tool from the REAKTOR 5 installation folder (default path: Applications\REAKTOR 5\).



- A click on the Register Now button opens the Native Instruments registration webpage. Therefore your standard Internet Browser will be opened and an internet connection will be established according to your system settings. Your System ID will be automatically transmitted to the registration form.



- On the first online page you are asked if it is your First Registration at Native Instruments or if you want to do an Additional Registration.
- Depending on the option you have chosen on the first online page you now get a login page asking for your username and password or a form where you can fill out your address data. Please fill out all required fields and follow the instructions on the screen to complete the registration.

Register Product

Additional Registration: Log in with your existing username and password to add this registration to your account.

First Registration: Your first product registration. Select a new username.

- On the last online registration page your Authorization Key is directly shown in the browser. Please copy the full number (12 x 5 digits) and paste it to the registration tool. Within a few minutes you will also receive an email from the Native Instruments registration system containing the Authorization key. The Authorization key is available in the email body and additionally as text attachment. This email also contains the password which is required for using the online services.

The product is now authorized. The Authorization Key for the current hardware setup is:

87145 70100 17888 76455 78008 09008
80708 51174 04001 00745 00708 00000

Please copy the entire Authorization Key (both lines) to the clipboard and paste it into the appropriate field of the Registration Tool.

The Authorization Key has also been sent to the following email address:

native@nativeinstruments.de

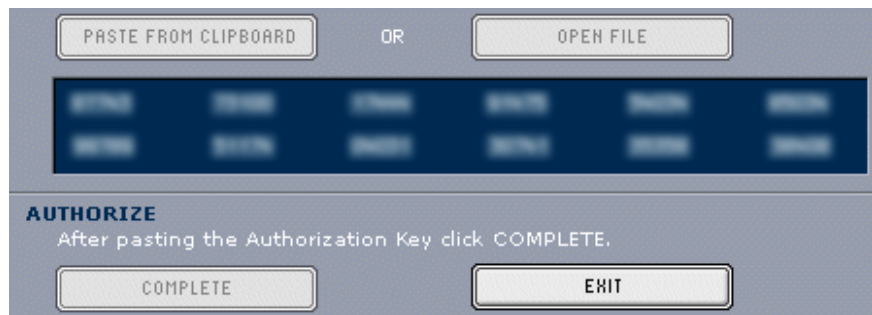
- Start the Registration Tool again and either copy the Authorization Key from the email and press the Paste from Clipboard. button in the Registration Tool or use the Open File button in the Registration Tool to open the email attachment which you previously have saved to hard disk.

2 AUTHORIZATION KEY

Once you receive the email with your authorization key please copy the entire key and paste it into the field below, or open the file that was attached to the mail.

OR

Click on the **Complete** button.



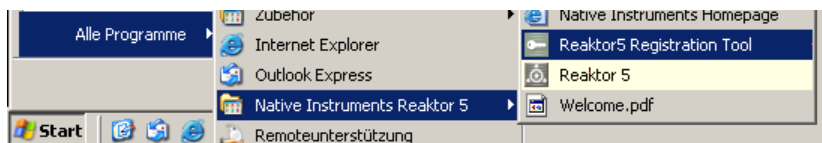
After completing the Product Authorization the Registration Tool looks like this

Now the REAKTOR 5 **Product Authorization** has been completed. The authorization message at every REAKTOR 5 start has disappeared and you can use the software permanently:

2.4. Method B: Internet Connection on another computer

Important: This method requires a valid email address to complete the **Product Authorization**, since the **Authorization Key** will be send to you by email.

- **Windows:** Start the Registration Tool from the start menu (Native Instruments REAKTOR 5 ⇒ REAKTOR 5 Registration Tool) or from the REAKTOR 5 installation folder (default path: C:\Program Files\Native Instruments\REAKTOR 5\).



- **MacOS:** Start the Registration Tool from the REAKTOR 5 installation folder (default path: Applications\REAKTOR 5\).



- A click on the Save Registration File button opens a Save dialog for saving a HTML file. Save the HTML file to any storage medium.

1 REGISTRATION
The first step is to send your System ID to Native Instruments and register your product. Choose one of the following options:

A. DIRECT ONLINE REGISTRATION
If you have direct access to the Internet from this computer, this is the fastest way to register your product.

B. SEND REGISTRATION FROM ANOTHER COMPUTER
Use this option if you have access to the Internet via another computer.

C. OFFLINE REGISTRATION
This allows you to fill out an HTML form that can be printed and sent to NI.

REGISTER NOW OR **SAVE REGISTRATION FILE** OR FILL OUT FORM

- Transfer the HTML file to another computer where you have internet access (via floppy disk, CDR etc.).
- Open the HTML file in your internet browser.
- The HTML page contains a link for the registration page on the Native Instruments website. When you click on this link an internet connection will be established according to your system settings.
- On the first online page you are asked if it is your First Registration at Native Instruments or if you want to do an Additional Registration.

Register Product

Additional Registration: Log in with your existing username and password to add this registration to your account.

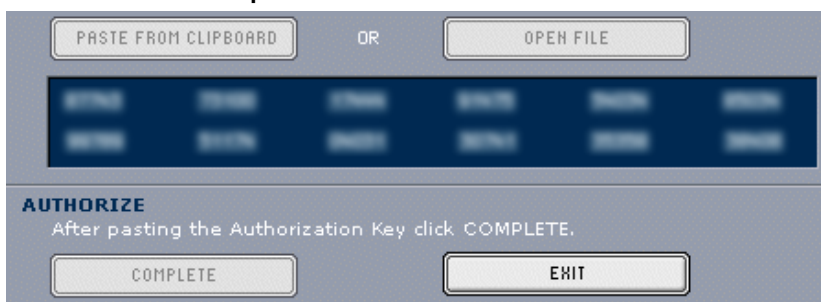
First Registration: Your first product registration. Select a new username.

- Depending on the option you have chosen on the first online page you now get a login page asking for your username and password or a form where you can fill out your address data. Please fill out all required fields and follow the instructions on the screen to complete the registration.
- Within a few minutes you will receive an email from the Native Instruments registration system containing the Authorization key. The Authorization key is available in the email body and additionally as text attachment. This email also contains the password which is required for using the online services.
- Transfer the text attachment to the computer where you have installed REAKTOR 5.

- Start the Registration Tool again and use the Open File button in the Registration Tool to open the email attachment which you previously have saved to hard disk.



- Click on the **Complete** button.



After completing the Product Authorization the Registration Tool looks like this

Now the REAKTOR 5 **Product Authorization** has been completed. The authorization message at every REAKTOR 5 start has disappeared and you can use the software permanently:

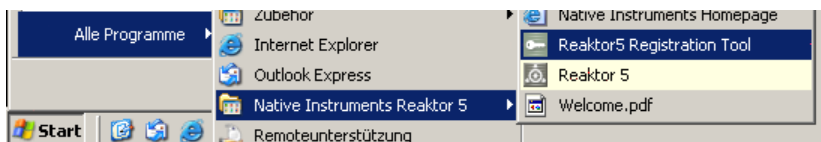
2.5. Method C: No Internet Connection available

Important: Following this method you will have to fill in a form which you send to Native Instruments. You will receive the **Authorization Key** either by email (recommended), by postal mail or by fax. If you do not provide Native Instruments with a valid email address in the form, be prepared to type in the Authorization Key manually (about 60 digits).

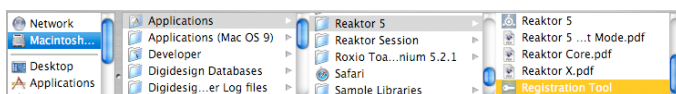
If you do not have access to the internet or if you do not have a working email address, the **Product Authorization** can also be done via postal mail or fax. Although Native Instruments goes after a fast handling of your authorization

requests, it is generally recommended that you prefer **Method A** or **Method B** for shortest return times and most comfortable operation. Please note the following instructions to fulfill the **Product Authorization**:

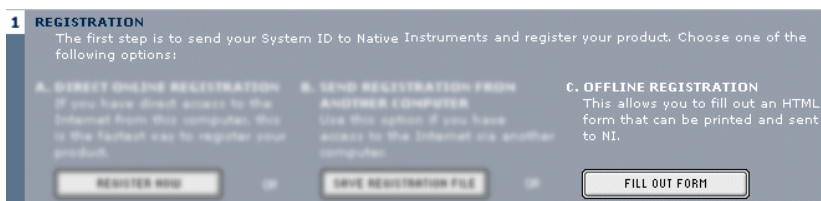
- **Windows:** Start the **Registration Tool** from the Windows start menu (**Native Instruments REAKTOR 5** ⇒ **REAKTOR 5 Registration Tool**) or from the REAKTOR 5 installation folder (default path: **C:\Program Files\Native Instruments\REAKTOR 5**).



- **MacOS:** Start the **Registration Tool** from the REAKTOR 5 installation folder (default path: **Applications\REAKTOR 5**)



- A click on the **Fill Out Form** button opens a local HTML file in the operating system's standard browser or another program which you have defined as standard application for opening HTML files.



- The HTML file contains all information Native Instruments requires for completing the **Product Authorization** and registration. Please fill in the required data and print it out, or write a letter containing the data.

If you write a letter please attend to a legible handwriting to avoid mistakes at the Native Instruments registration team. Illegible email or postal addresses can cause problems with the **Authorization Key** delivery.

Send the form to Native Instruments using one of the following contact addresses:

Native Instruments GmbH

Registration
Schlesische Straße 28
10997 Berlin
Germany
Fax: +49 30 6110352400

Native Instruments USA

5631 A Hollywood Boulevard
Los Angeles CA 90028
USA
Fax: +1-323-372-3676

- You will receive the **Authorization Key** via email (recommended), fax or mail.
- Start the Registration Tool again and either copy the **Authorization Key** from the email and press the **Paste from Clipboard** button in the Registration Tool or use the **Open File** button in the Registration Tool to open the email attachment which you previously have saved to hard disk. If you have received the Authorization Key by mail or fax, type it in manually.
- Click on the **Complete** button.
- Now the REAKTOR 5 **Product Authorization** has been completed. The authorization message at every REAKTOR 5 start has disappeared and you can use the software permanently:

2.6. Registration support

If you run into problems during the **Product Authorization** the Native Instruments registration support team will be happy to help you. In this case send a support request on the NI website using the following URL:

<http://www.native-instruments.com/register-support.info>

Please describe the occurred problem as accurate as possible and provide the registration support team with the necessary details to solve the problem.

3. Installation under Windows XP

3.1. System Requirements and Recommendations

To use the REAKTOR 5 software, you need a computer with the following minimum specifications:

Hardware

- Pentium III 1 GHz/ Athlon XP 1.33 GHz.
- 512 MB RAM
- Up to 1GB free HD space.
- A sound card compatible with Windows XP
- CD drive

Software

- Windows XP

Important: REAKTOR 5 only runs on processors supporting SSE

3.2. Software Installation

- Insert the REAKTOR 5 CD into the CD drive.
- Use the Windows Explorer to view the contents of the CD.
- Start the installation by double-clicking **REAKTOR 5 Setup.exe**.
- The setup program will suggest **C:\Program Files\Native Instruments\REAKTOR 5** as the path for the destination folder. You may also choose another folder if you wish.

Installed Folders, Files, and Links

The setup program creates a new folder called **REAKTOR 5** in the installation directory (**Program Files\Native Instruments**). This folder contains the files required to operate the software. If you do not choose a different program group during the installation, links to REAKTOR 5 and a ReadMe file are added to the **Start** menu under **Programs\Native Instruments**.

3.3. VST plug-in Installation

- Insert the Installation CD into the CD drive.
- Use the Windows Explorer to view the contents of the CD. To start, double-click the **REAKTOR 5 Setup.exe** file.
- When the choice is given by the installer, select **VST plug-in** from the list of components to install.
- 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.

Note: If more than one host program for VST 2.0 plug-ins is installed on your computer, the installer lets you install to multiple VST-folders by shift-clicking them. If you decide to install them at a later date, simply copy the “**REAKTOR 5 VST.dll**” file into the VST plug-ins folders of these programs. Windows: If the VST plug-in files are not visible in the Windows Explorer, select the **Show All Files** option. This option is located in the Explorer menu **View** ⇒ **Folder Options...** on the **View** tab below **Hidden Files**. Optionally, you can set up your programs so that they all use the same VST plug-ins folder.

3.4. DXi 2 plug-in Setup

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.

Installation

- Insert the Installation CD into the CD drive of your computer.
- Use the Windows Explorer to view the contents of the CD and double-click the **REAKTOR 5 Setup.exe** file to start the installation.
- Continue the REAKTOR 5 installation until you come to the **Choose plug-ins** page. Tick the checkbox **DXi plug-in**.

The installation program copies the REAKTOR 5 plug-in to your hard disk and registers it as a DXi 2 plug-in for use in DXi 2-compatible host programs. After the installation, REAKTOR 5 appears as a plug-in in the host program.

3.5. RTAS plug-in installation

- Launch the REAKTOR Installer from the CD.
- Select the Custom installation type.
- Select only RTAS from the list of components to install.

4. Installation under MacOS X

4.1. System Requirements and Recommendations

To use the REAKTOR 5 software, you need a computer with the following minimum specifications:

Hardware

- Apple PowerMac G4 1 GHz or faster
- 512 MB RAM
- Audio interface compatible with Core Audio
- CoreMIDI compatible MIDI interface for connecting a MIDI keyboard or an external sequencer (only for the stand-alone version)
- Up to 1GB free HD space
- CD drive

Software

- MacOS 10.2.6

Important: REAKTOR 5 only runs on processors supporting AltiVec

4.2. Installing REAKTOR 5 OS X

- Insert the Installation CD into the CD drive of your computer.
- Double-click the installation program **Install REAKTOR 5** 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 5 folder; if you do not select a different destination, the REAKTOR 5 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.

4.3. MacOS Audio Unit plug-in Installation

- Launch the REAKTOR 5 Installer from the CD
- Select the **Custom** installation type.
- Select only **Audio Unit** from the list of components to install.

4.4. RTAS plug-in installation

- Launch the REAKTOR Installer from the CD
- Select the Custom installation type.
- Select only RTAS from the list of components to install.

5. Audio Interfaces

Audio interfaces, which include software routines called drivers, allow REAKTOR 5 (and other programs you have installed, if present) to communicate with your computer’s audio hardware. This section describes how to use various audio interfaces with REAKTOR 5.

There are two main ways to implement REAKTOR 5:

As a “stand-alone” device that requires no host software. REAKTOR 5’s audio and MIDI connections interact directly with your computer’s audio/MIDI hardware interface.

As a plug-in that works in conjunction with a “host” program, such as sequencing or hard disk recording software. In this case, the host program interacts directly with the computer’s hardware interface. REAKTOR 5 connects to the host program via “virtual patch cords.” REAKTOR 5’s audio outputs appear as signals in the host’s mixer, and the host passes MIDI data to REAKTOR 5.

We’ll describe each mode in detail, but first let’s look at the various interface drivers and plug-in formats used by different operating systems and programs.

5.1. Stand-alone Application

REAKTOR 5 works in stand-alone mode with ASIO, MME, DirectSound, and Core Audio. The REAKTOR 5/computer combination acts as an instrument, similar to a hardware digital synthesizer. The table shows you which drivers are available under which Operating System:

Driver	Windows	MacOS X
ASIO 2.0	•	• (only Jaguar)
DirectSound	•	
MME	•	
Core Audio		•

Plug-In

Used as a plug-in, REAKTOR 5 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 REAKTOR 5 and audio mix-down of the MIDI tracks within a single program
- Comfortable automation of REAKTOR 5 parameters in the sequencer
- Further processing of REAKTOR 5 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 song file of the sequencer) is loaded
- Integration with other instruments into a “virtual studio”

The shortcuts / key commands do not work in all sequencers. This is due to the fact that the host capture keys for themselves and do not pass them on to the plug-in.

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

Plug-in Interface	Host-Program	Windows	Mac
VST 2.0 Plug-in	Cubase, Nuendo	•	•
DXi	Sonar	•	
Audio Units	Logic		•

Note: Some hosts include “wrappers” that allow running REAKTOR 5 with a choice of plug-in protocols. Try each one, as one may offer better performance than another. *Example:* With Sonar, running REAKTOR 5 as a DXi instrument allows using multiple outputs, while running it as a VST instrument provides more automation options

Interface Details

The interfaces described below represent different ways REAKTOR 5 can communicate with your sound card. Available interfaces depend on your computer, the audio interface (sound card) you’re using, and your computer platform (REAKTOR 5 supports Windows XP or MacOS X). Choose the fastest interface protocol supported by your interface, which will likely be ASIO with Windows, or Core Audio for Mac. For Windows, you can also use DirectSound and Multimedia (also called MME), but expect a significant delay (called *latency*) between the time you play a note and the time you hear it.

ASIO (Audio Streaming Input Output): This cross-platform plug-in protocol was developed by Steinberg. It is highly recommended for its low latency, multi-channel audio card support, and high performance.

DirectSound: Developed by Microsoft, this is a component of DirectX 5.0 or higher for Windows. How well DirectX works well depends on your sound card. If you adjust the interface for an acceptable amount of latency, you may hear glitches and clicks in the audio output that can only be fixed if you increase latency.

MME (Multi Media Extension): This is the standard Windows audio driver. Most sound cards support this interface and work with it quite well. However, MME is even less suitable than DirectSound for real-time applications due to its comparatively high latency.

Core Audio: This audio interface for MacOS X is integrated tightly into the operating system, and works with external audio hardware as well as the Mac's integrated audio output.

Plug-In Details

VST (Virtual Studio Technology): Like ASIO, this cross-platform plug-in technology was developed by Steinberg. It is the most common plug-in format, and many programs are optimized to work with VST plug-ins.

DXi2 (DirectX Instrument 2): Based on Microsoft DirectX technology, this plug-in interface for software synthesizers and instruments is designed for low latency and high performance on the Windows platform. Cakewalk Sonar and Image Line FL Studio are the most well-known hosts that support DXi.

RTAS (Real Time Audio Suite): This interface protocol from Digidesign allows using plug-ins with ProTools (or other Digidesign-compatible software). Unlike traditional TDM effects that depend on using Digidesign hardware, RTAS plug-ins are “native;” the host processor performs all computations needed for the plug-in.

AU (Audio Units): This plug-in format is exclusively for the Macintosh OS X platform, and is tied in closely with the operating system.

More About Latency

As with any digital device (including hardware signal processors) that convert audio to data and back again, a computer adds a certain amount of delay (“latency”) when processing audio signals. Fortunately, with today’s computers and low-latency sound card drivers, this delay can be so small that you can’t hear it (*e.g.*, under 3 milliseconds, which is about the same delay caused by moving your head one meter further away from a speaker). However, typical computers are generally not set up for low latency; attempting to play in real time through REAKTOR will probably be unsatisfying because of the delay.

If your computer is already configured for low-latency operation, keep reading.

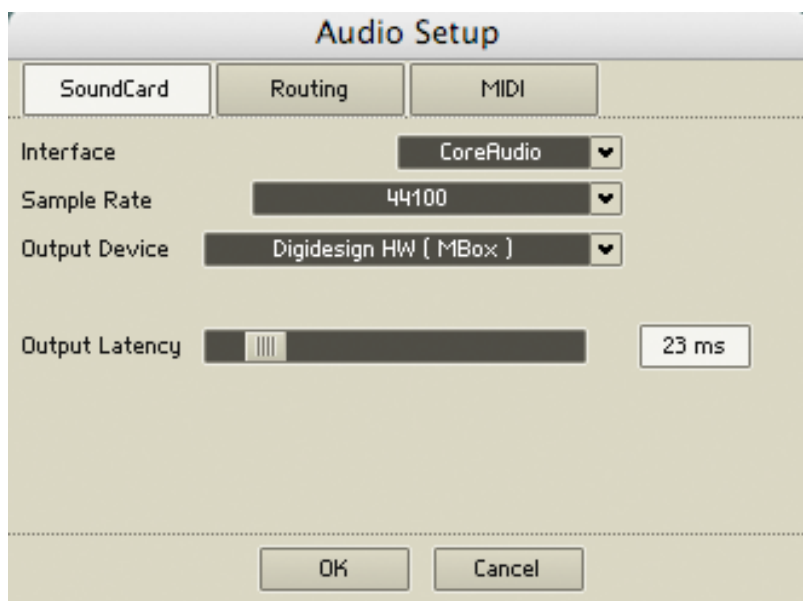
6. REAKTOR 5 as Standalone

When used as a plug-in, the host program has already set up its audio and MIDI connections, and REAKTOR 5 simply “plugs in” to these. However, with stand-alone operation REAKTOR 5 communicates directly with your audio interface. It’s therefore necessary to specify audio and MIDI settings, as well as the preferred driver protocol (of course, plug-in format is not an issue).

Setup for Mac and Windows machines is essentially identical, except where indicated. Note that if you change your audio interface, you will almost certainly need to re-adjust these settings.

Call up the **Audio + MIDI Settings** setup dialog from the **Setup** menu. You’ll see three tabs for **Soundcard**, **Routing** (audio output patching), and **MIDI**.

6.1. Soundcard (Audio Interface)



Audio + MIDI Settings dialog

Interface: Choose the fastest interface protocol supported by your interface, which will be ASIO or Core Audio. For Windows, you can also use DirectSound and Multimedia (also called MME), but expect a significant delay between the time you play a note and the time you hear it.

Windows only: Avoid using any drivers listed as “emulated,” as they provide poorer performance than other drivers. For example, although DirectSound drivers generally outperform MME drivers, MME drivers will outperform emulated DirectSound drivers.

Sample rate: The drop-down menu will display compatible sample rates for your audio interface. 44.1kHz is the same sample rate used for CDs, and is the most “universal” choice. However, some audio interfaces offer 48kHz and 96kHz (REAKTOR 5 accepts up to 96kHz sample rates). These higher rates stress your computer more, but offer somewhat better high frequency response. If you are using REAKTOR 5 standalone, choose whichever rate you prefer. When used as a plug-in with a host program (e.g., Cubase, Digital Performer, Logic, Sonar, etc.), the host will determine the sample rate.

Output Device: Use ASIO written specifically for your audio interface (not “ASIO DirectX” or “ASIO Multimedia,” unless no other choices are available), or for the Mac, Core Audio.

Output Latency: This field displays the output latency. For some drivers you can adjust the latency individually using a fader.

Adjust latency for the fastest possible setting that gives consistent audio performance. The CPU may not be able to keep up with fast settings, resulting in possible crackles or pops in the audio. Slower settings will give more consistent audio performance, but the amount of delay may be musically unsatisfying.

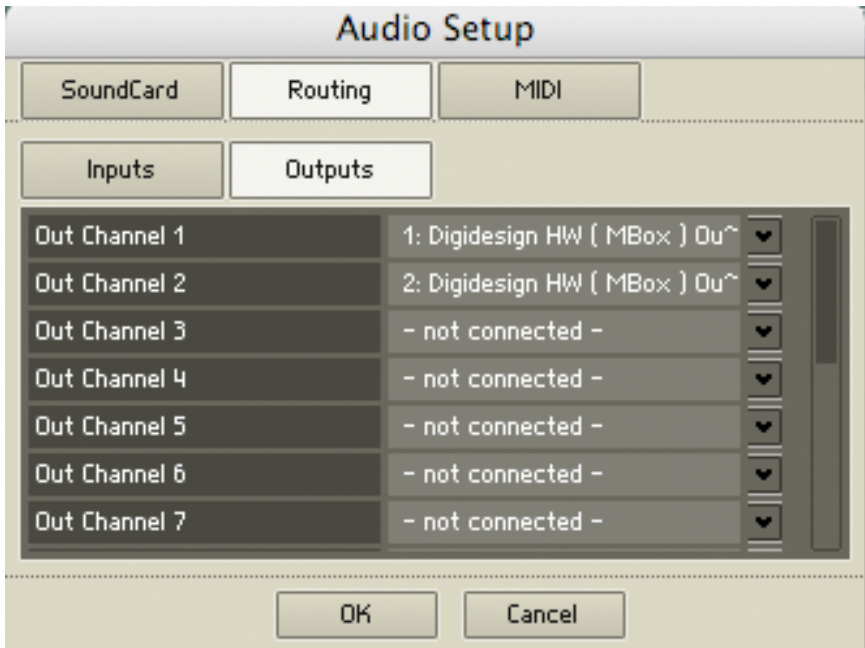
Experiment with the latency setting until you find the best compromise between consistent audio performance and fast response. A quick way to adjust latency is as follows:

Select any instrument and play it while moving the Latency slider.

Move the Latency slider to the left until you start to hear clicks in the audio output.

Now move the slider to the right until the clicks disappear. This is the optimum setting.

6.2. Routing



Using the drop-down menu, REAKTOR 5's Output can be assigned to an output from a multi-output sound card.

If your sound card offers multiple inputs and outputs, you can choose which ones connect to REAKTOR 5. Click on Inputs to choose the desired inputs from the drop-down menus, and click on Outputs to select the outputs, also from drop-down menus. Note that the right and left channels are independent and can be assigned to any inputs/outputs – not just stereo pairs – as well as disconnected from audio ins and outs.

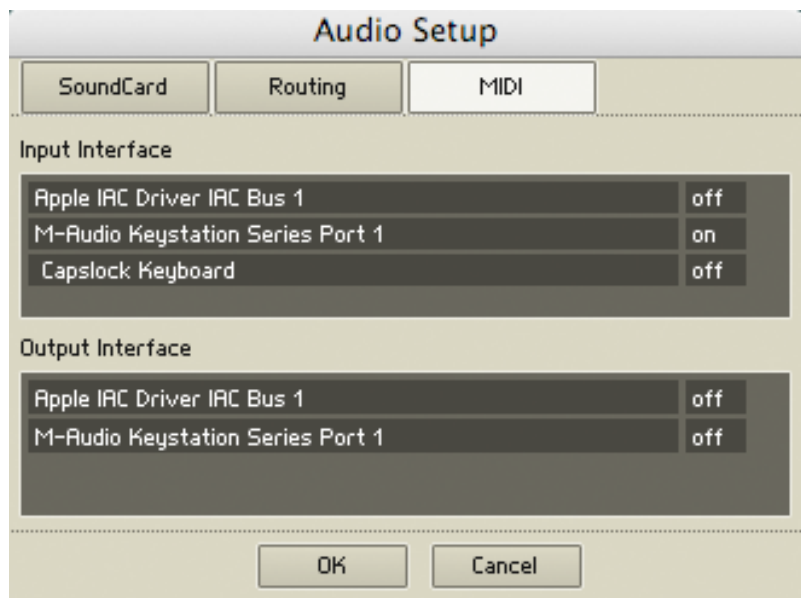
REAKTOR 5 provides up to 8 monophonic outs. Using all of these in stand-alone mode requires an audio interface with eight output channels.

However, it is not necessary to have this many channels. For example, if a notebook computer has built-in audio and offers only a single stereo output (two channels), you can assign all the REAKTOR 5 sounds to the stereo output.

Each drop-down list shows what outputs are available from the driver/audio interface selected under the Interface tab. Assign each REAKTOR 5 1/2 Right, Channel 3/4 Left, Channel 3/4 Right, Channel 5, Channel 6, Channel 7, and Channel 8) to the desired hardware output.

Windows only: The audio interface's overall level may be determined by a mixer applet included with your interface hardware, or the built-in Windows mixer. If you encounter excessively low or high levels, please check the Windows volume control by going Start ⇒ Programs ⇒ Accessories ⇒ Entertainment ⇒ Volume Control. Then check the Wave volume slider, and adjust its level as needed.

6.3. MIDI



When you click on the MIDI tab you'll see a list of MIDI I/O. Initially, each one will be Off. This field is a toggle – click on Off to turn an input or output On, click on On to turn an input or output Off.

If you enable more than one input, they will be merged.

7. REAKTOR 5 as Plug-in

The plug-in version of REAKTOR looks a bit different from the standalone version, but you have still access to all the software's main features (unless they are not applicable to plug-in operation).

A menu is available in the plug-in version as context menu. To call up the menu, WindowsXP: Right-click / OS X: 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 WindowsXP: Right-click / OS X: Ctrl + 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 on 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.

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.

7.1. Automation ID editing

Each control element in a REAKTOR ensemble is assigned a unique automation ID. This enables automation from a host application. The ID determines the order that the controls appear in the host parameter list. For this reason, the instrument properties has several functions for re-arranging the automation ID's. These functions are particularly important in certain host applications that only recognize a limited number of parameters.

- **Compress** ensures that there are no 'gaps' between automation ID's.
- **Sort and compress** additionally ensures that ID's of controls within the same macro are grouped together (so that they will appear together in the host parameter list).
- Each instrument in an ensemble also has a unique **base ID**. This determines the overall order of Instrument controls within the parameter list. **Instrument up** and **Instrument down** increase or decrease the priority of the instrument in the parameter list.

7.2. Total Recall

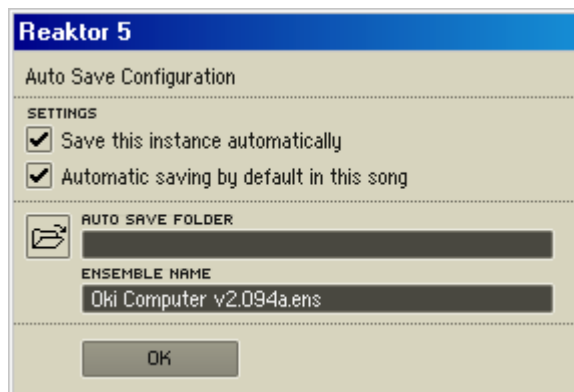
Loading a REAKTOR Plug-in

After inserting a REAKTOR plug-in in your host application, you will be presented with two options: **New Ensemble** and **Load Ensemble....**

Selecting **New Ensemble** will load **New.ens** (an empty structure in which you can construct your own ensemble). Selecting **Load Ensemble...** will present a file dialog to select an existing file.

It is also possible to load an ensemble by drag and drop from the browser or from an outside window.

The Auto Save Configuration Dialog



After loading an ensemble or creating a new one, the **Auto Save Configuration** dialog with the following options will appear.

- **Save this instance automatically:** Enabling this option is always recommended. It ensures that the ensemble and all changes made to it are saved to a separate .ens file when you save your project. This is safest as amendments to the original ensemble will not affect your project. When enabled, you are required to choose a destination folder and a filename for the Auto Save location. When disabled, a copy of the ensemble is not made, thus saving disk space. The current setting of the ensemble panel controls will be stored within your project file. But please note, sharing ensembles in this way means that amendments to the structure of the original ensemble will affect all projects using the ensemble. Further, changes to event / audio table data and sampler file references will not be stored when saving the project.
- **Automatic saving by default in this project:** When this option is on, the Auto Save Configuration dialog will open up with each REAKTOR plug-in that you insert in this project. Keeping this option active guarantees that the latest state of all ensembles used in this project will be saved in their own files.
- **Auto Save Folder:** Here you specify the folder where the ensemble shall be automatically saved. You can either enter a directory from the keyboard, or select it from a file browser by clicking on the folder icon. It is recommended to store ensemble files in the same folder as the project file itself. This ensures that they will be kept together when moving the project folder.

- **Ensemble Name:** Here you can specify a filename for the ensemble. We recommend choosing a name that is unique and cannot be confused with the names used for other plug-in instances in the current project or other projects.
- **OK:** Selecting OK confirms the Auto Save configuration. If Auto Save is on, but a folder or ensemble name have not yet been defined, the option to select “OK” is disabled.
- **Cancel:** Discards changes and closes the dialog

Auto Save Functions in the Plug-in Header



- The first field shows the file name of the currently loaded ensemble. This is updated with the file name that has been given in the Auto Save Dialog.
- The **Menu** button opens the main menu.
- The lamp between Menu and Auto Save” buttons shows whether the plug-in is in Auto Save mode or not.
- The **Auto Save** button opens the Auto Save Configuration dialog. This allows you to reconfigure Auto Save settings. If you change the Auto Save folder, the ensemble will be automatically moved to the new destination. If there are any other instances using the old folder, you will be asked whether to change the Auto Save folder for those instances too.

Replacing an Ensemble

If Auto Save is enabled, and you replace the ensemble, the Auto Save configuration dialog will appear. Thus, when auditioning different ensembles it might be advantageous to disable Auto Save until you decide which ensemble to use.

Loading a Project and the Ensemble is not Found

If an ensemble is not found when opening a project, a message will appear in the plug-in window. You can locate the ensemble via a file browser by clicking the **Locate Ensemble** button, by dragging the correct ensemble into the window, or by re-configuring the Auto Save dialog. If other plug-in instances have missing ensembles, an option to try the new folder location for those other instances will now appear.

Working with multiple REAKTOR Plug-ins

If you load another REAKTOR plug-in into your project, the Auto Save mode and folder are taken from the last REAKTOR plug-in instance.

What is Saved with a Project

When the project is saved (or the host queries the plug-in data for whatever reason), the following happens:

- If Auto Save is turned on, the ensemble is saved as a file to the AutoSave folder.
- If Auto Save is turned off, the path of the currently used ensemble is saved in the project data.

In either case, the following are also saved:

- The current setting of all panel controls
- The last Auto Save mode and folder
- The current window size and mode (minimized, auto-layout or fixed)

Saving a copy of an Ensemble

In the Plug-in the ‘Save As’ command is replaced by ‘Save A Copy As’. This allows saving a copy of the ensemble elsewhere without changing the Auto Save filename and path.

Global Auto Save default

In the Preferences, the **Auto Save off by default** option determines the default Auto Save configuration for new instances.

Plug-in Size Functions



Several functions are available for controlling the window size of a plug-in.

These are the last four buttons on the right hand side of the plug-in toolbar. The first two buttons minimize and maximize the screen respectively. The third button (‘Manual resize’) resizes the plug-in window to fit the ensemble when pressed. The fourth button (‘Automatic Resize’) is a toggle button. When

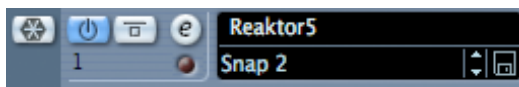
activated, the plug-in window will always be automatically resized when required (for example, switching windows, toggling between A and B panels).

The maximum plug-in window size can now be configured in the preferences.

7.3. VST 2.0 Plug-In

7.3.1. Using the REAKTOR 5 plug-in in Cubase SX 3

- Launch Cubase, go to the **Devices** menu option and select the **VST Instruments** menu option or press F11 on your keyboard.
- A window showing the instrument rack appears. Click on an empty slot and choose REAKTOR 5 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 5's output just like any other existing audio track in your Cubase song.
- Click on the **Edit** button to call up the REAKTOR 5 interface. Here you can control and edit all the features and functions that REAKTOR 5 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 out ports to assign to this MIDI track. Choose **REAKTOR 5** from the list.

Note: If REAKTOR 5 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 library you should be able to trigger it via MIDI using a keyboard controller. REAKTOR 5'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 instrument.
- Make sure that you have properly configured your sound card for use with Cubase.

(please refer to your Cubase manual for more information)

7.3.2. Using the REAKTOR 5 plug-in in Nuendo 2.0

- 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 on an empty slot and choose **REAKTOR 5 VST** 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 5's output just like any other existing audio track in your Nuendo project.
- Click on the **Edit** button to call up the REAKTOR 5 interface. Here you can control and edit all the features and functions that REAKTOR 5 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 5 VST** 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 5 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 library you should be able to trigger it via MIDI using a keyboard controller. REAKTOR 5'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 instrument.
- Make sure that you have properly configured your sound card for use with Nuendo

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

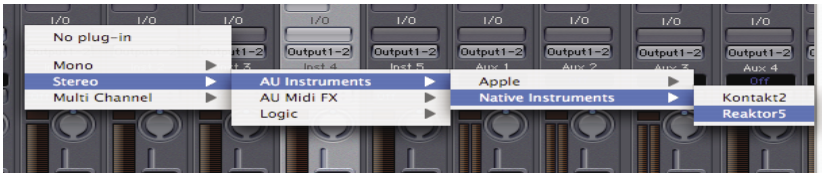
7.4. Audio Units Plug-ins

7.4.1. Use in Logic 7.x

- 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** ⇨ **Inst 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 5 Audio Unit plug-in in the appropriate insert slot of the instrument mixer bus, either in the arrange or mixer window. Then click onto the insert slot, hold down the mouse button and choose **Stereo** ⇒ **Audio Units** ⇒ **Native Instruments** ⇒ **REAKTOR 5**. (REAKTOR 5 is also available as a multi-channel insert)



- 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 5's output just like any other existing audio track in Logic.
- If the REAKTOR 5 interface is not already open, double click on the mixer's REAKTOR 5 slot to call up the REAKTOR 5 interface. Here you can control and edit all the features and functions that REAKTOR 5 has to offer.

Note: If REAKTOR 5 does not appear in the list of available AUinstruments inside your AU host application, then it is not installed correctly. Please refer to the previous section on installing the plug-in for the Mac platform for more assistance on setting this up.

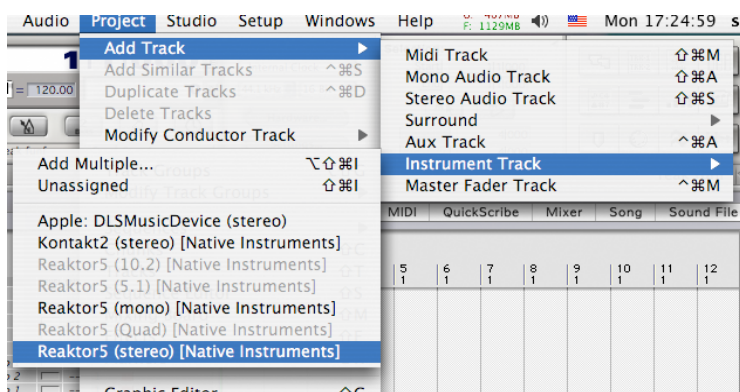
After having loaded an Instrument from the library you should be able to trigger it via MIDI using a keyboard controller. REAKTOR 5's sound will generate through the mixer and directly to your sound card. If the plug-in does not receive MIDI or generate audio, then make sure to check the following two areas:

- Make sure the Inst track is selected in the Arrange window.
- The MIDI channel of your MIDI track must correspond to the receive channel of the loaded instrument.
- Make sure that you have properly configured your sound card for use with Logic.

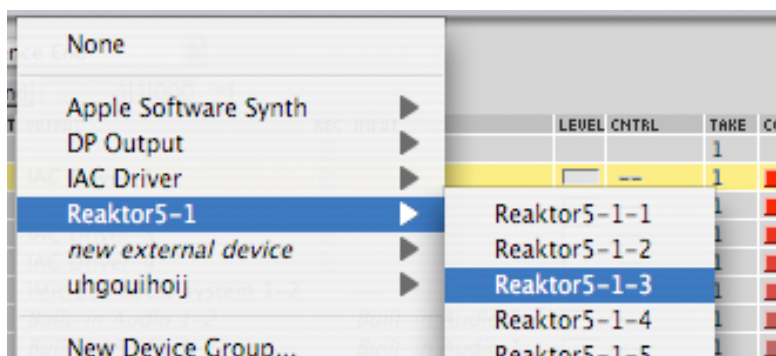
(please refer to your Logic manual for more information).

7.4.2. Use in Digital Performer 4.5

- Launch Digital Performer and create an instrument track by selecting **Project ⇒ Add Track ⇒ Instrument Track ⇒ REAKTOR 5**.



- Create a MIDI track by selecting **Project ⇒ Add Track ⇒ Midi Track**. In Digital Performer's track overview window (or in the sequence editor window) assign the output of this MIDI track to "REAKTOR 5-1" and a MIDI channel. If you instantiate further REAKTOR 5 Plug-Ins they will be named "REAKTOR 5-2", "REAKTOR 5-3" etc.



- The plug-in is now ready to use. The mixer of Digital Performer will allow you to mix, pan, and process REAKTOR 5's output just like any other existing audio track.
- To play REAKTOR 5 with your keyboard, record enable the MIDI track which you have routed to REAKTOR 5 and make sure **Midi Patch Through** is enabled in the Studio menu of Digital Performer.

- Double click on the REAKTOR 5 slot in Digital Performers mixing board to call up the REAKTOR 5 interface. Here you can control and edit all the features and functions that REAKTOR 5 has to offer.

Note: If REAKTOR 5 does not appear in the list of available Audio Unit plug-ins inside your Audio Units host application, then it is not installed correctly. Please refer to the previous section on installing the plug-in for Mac platforms for more assistance on setting this up.

After having loaded an Instrument from the library you should be able to trigger it via MIDI using a keyboard controller. REAKTOR 5's sound will generate through Digital Performers 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:

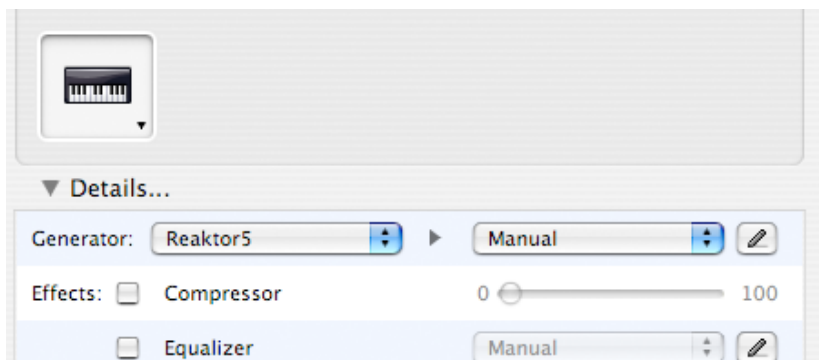
- Make sure **Midi Patch Through** is enabled in the Studio menu of Digital Performer.
- The MIDI channel of your MIDI track must correspond to the receive channel of the loaded instrument.
- Make sure that the instruments track output is correctly set.
- Make sure that you have properly configured your sound card for use with Digital Performer.

(please refer to your Digital Performer manual for more information).

7.4.3. Use in Garage Band

- Launch Garage Band
- Press the “+” button to create a new “Software Instrument” Track. From here you can choose the icon you wish to use.
- Double-click the instrument track icon or press the “I” icon to get the Track Info.
- From the Info window expand the Details triangle underneath the Instrument icon to expose the track settings.
- From the Generator drop-down menu, choose REAKTOR 5 from among Audio Unit plug-ins.
- Clicking on the pencil icon next to the “Manual” drop-down menu will open the REAKTOR 5 interface for editing.

- REAKTOR 5 can now be played using an external MIDI keyboard.



7.5. DXi 2 plug-in

DXi is a Microsoft DirectX technology based plug-in format

7.5.1. Use in Sonar 4

- Launch Sonar
- In the synth rack choose **REAKTOR 5 DXi 2**.



Loading the REAKTOR 5 DXi 2 plug-in in the synth rack

- Route a MIDI track to the DXi 2-Plug-in by selecting **REAKTOR 5** in the Out drop down list.



Assign a MIDI track to the REAKTOR 5-DXi-Plug-in

After having loaded an Instrument from the library you should be able to trig-

ger it via MIDI using a keyboard controller. REAKTOR 5's sound will generate through Sonar's 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:

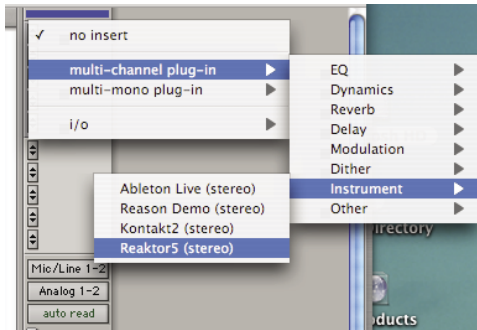
- Make sure **Midi Patch Through** is enabled in the Studio menu of Sonar.
- The MIDI channel of your MIDI track must correspond to the receive channel of the loaded instrument.
- Make sure that the instruments track output is correctly set.
- Make sure that you have properly configured your sound card for use with Sonar.

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

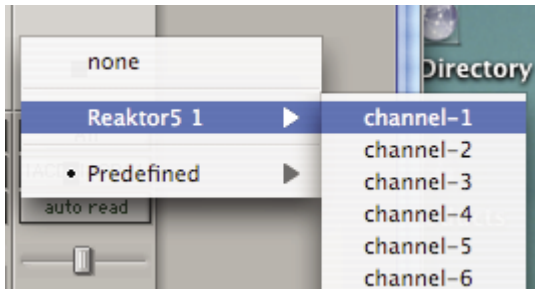
7.7. Using REAKTOR RTAS with Pro Tools 6.x (Mac/Windows)

The RTAS format is an interface protocol for Mac OS and Windows 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.

- Launch Pro Tools
- Create a new **AUX** track **File a New Track**
- **Create a new MIDI** track the same way
- Locate the channel mixer **Windows ⇒ Show mix**
- The dark grey box at the topmost section of the AUX channel is the RTAS insert section. Click on the first empty slot to show all available RTAS plug-ins.
- Choose **REAKTOR 5** from the menu



- Now locate the **MIDI** channel you just created
- In the output slot, choose REAKTOR and the appropriate channel



After record enabling the MIDI track, you will be able to play REAKTOR with your MIDI keyboard.

Important: If you will to use REAKTOR's multiple outputs (provided the REAKTOR ensemble supports this) in Pro Tools 6.7 or higher, you must create additional stereo/mono Auxiliary tracks in your Pro Tools session. After creating new tracks, set the aux input source to "plug-in" and choose the corresponding output from the drop-down list.

(Please refer to your Pro Tools manual for more information on how to record REAKTOR's output).

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.

8.1. Application areas

The OSC implementation of REAKTOR allows for easy setup of

- Internet-based collaborative international music making
- Sound installations with dozens of computers 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 version of REAKTOR only supports transmission of event data between two or more REAKTOR computers, but not audio data. In addition to REAKTOR's general requirements you will need an ethernet card to use OSC. Also, TCP/IP and UDP protocol stacks must be installed on your computer.

8.2. OSC System Setup

OSC Setup

OSC
☒ Activate

LOCAL IP ADDRESS
127.0.0.1

LOCAL PORT
10000

LOCAL IDENTIFIER
Reaktor5-1

Apply

CLOCK SYNC
☐ Master ☒ Off

NETWORK DLY (MS)
0.000

TIME SYNC
☐ Master ☒ ok

TIME OFFSET (MS)
0

Identifier	IP Address	Port
------------	------------	------

Scan
Edit
Delete

IDENTIFIER
[]

REMOTE IP ADDR.
[]

REMOTE PORT
0

Apply

OSC MESSAGE
[]

MONITOR OPTIONS
select

OSC MONITOR
[]

OK Cancel

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 the 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 10,015.
- **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. The client constantly polls the master for the time stamp, 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 received one second later by the client. This only applies if the participating clients are in Time Sync mode.

OSC Member List

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

You can edit and delete entries in this list. To do so, select an entry and press the **Edit** button. To apply any changes you have made, 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 met:

- 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 10,000 and 10,015 must have been entered.

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

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.

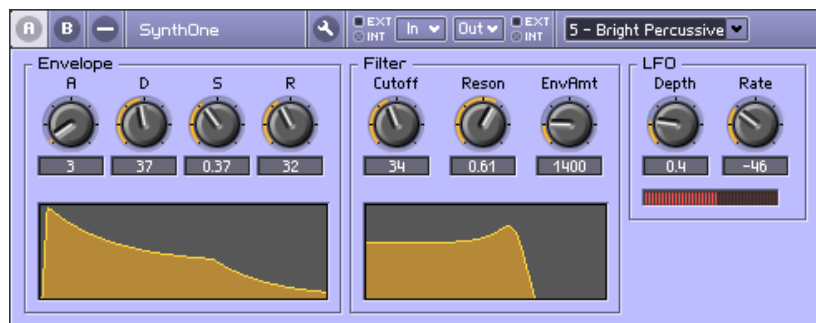
But don't worry. Although you can work with REAKTOR at a complex level, you don't have to. As you will see in our first tour, it is possible to make music with the software using a number of different instruments, even without any knowledge of synthesis methods or processing structures. You simply help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

9.1. Opening and Playing Examples

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 Interface** on the **MIDI** tab of the **Audio + MIDI Settings...** dialog. (For more information about activating MIDI ports, see **REAKTOR Standalone**). 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 play notes (see **Appendix** for key mapping).

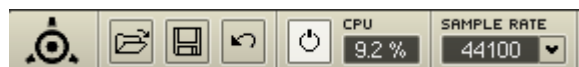
SynthOne



Panel Window for SynthOne

Now open the folder **Tutorial Ensembles** in the REAKTOR Library folder, select **SynthOne.ens** and click on **Open**, or drag the ensemble from the REAKTOR Browser into the main REAKTOR window.

You should first take a quick look at the Main toolbar.



Is the **Run/Stop Audio** button on, and does the field next to it show the current CPU usage? If so, you can now play **SynthOne**. If not then click the **Run/Stop Audio** button to turn on your new synth, and then let 'er rip! By the way, if the CPU usage field shows **Over** or a warning message pops up (“Processor Overload!”) to inform you that audio processing has been turned off, then you will need either to reduce the number of **Voices** in the Instrument header of the Synth 1 panel from **6** to a smaller number, or to 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 sound card is being overloaded – in which case you should reduce the ensemble volume (by using the Main fader).

With every note you play, the MIDI In lamp in the Instrument Header should light up red.



This lamp indicates that the SynthOne is receiving MIDI data, while the MIDI In Ext lamp in the Instrument header indicates that the MIDI data is being received by the instrument.

SynthOne is a replica of a simple 6 voice analog synthesizer. It contains one

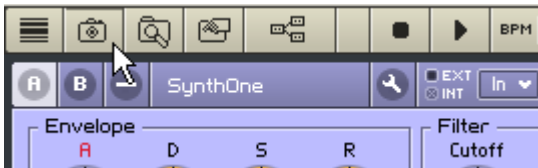
sawtooth oscillator, a 24-dB lowpass filter, an LFO that affects the pitch, and an ADSR envelope that modulates the sawtooth amplitude and the filter cutoff frequency.

In the **SynthOne** 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, **EnvAmt** 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 had your hands on a synthesizer before, then dealing with this straightforward device should not be much of a challenge. On the other hand, if **SynthOne** 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 may want to save it. To do so:

- Open the **Snapshots** window by clicking on the camera icon in the toolbar, which opens the snapshot window.



The **Snapshots** window is used for managing (saving, renaming, deleting, etc.) snapshots, which are like the “patches”, “presets” or “programs” found in other programmable synthesizers.

- Click the **Append** button twice (the first click lights the button, the second un-lights it) to save your current **SynthOne** sound settings to the first empty slot in the snapshots list.
- REAKTOR gives your new snapshot a default name. To rename it, double-click on the current name, type your new name, and press the **Enter** key.

Padecho



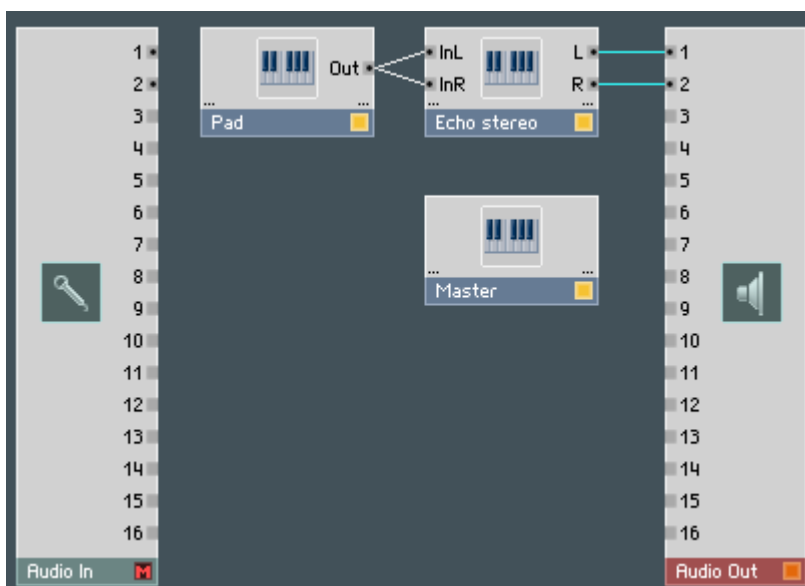
The next ensemble we are going to have a look at is called **Padecho.ens**. (Pad-echo = Pad with echo.) You will find it in the same folder from where we previously pulled out **SynthOne**.

At this point you will be asked whether you want to save the changes you have made in SynthOne. You probably want to answer **No** here, unless you have just created a snapshot(s) worth keeping.

You will now see three instrument panels, which are labeled Pad, Echo Stereo and **Master**. Click on the Structure button (its icon is three little boxes connected by wires) in the Panel toolbar in order to open the Ensemble Structure window.



The ensemble is the highest level in REAKTOR and the Ensemble Structure window provides a bird's eye view of the complete working environment that is available to you. In this case it contains the synthesizer **Pad**, the stereo delay effect **Echo Stereo**, and **Master** which holds the master ensemble controls **Main** and **Tune**.



The output of **Pad** is connected to the two inputs of **Echo Stereo**. The outputs of **Echo Stereo** are then connected to the top 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 sound card, but it can also be the Plug-In connection to another piece of software. Its counterpart, which is also present in every ensemble, is the **Audio In** module, which represents the audio inputs of your sound card (or the Plug-In connection). In this case **Audio In** is muted (as indicated by the red **M** in its title bar), because **Padecho** doesn't use 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 is that its generative power is not restricted to synthesizers, because **Echo Stereo** is an effects unit.

You can switch between the Structure and Panel windows by double-clicking the black background of the Panel window, or double-clicking the dark grey background of the Structure window. Give it a try.

You can play the ensemble with the Structure window open (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 of a synth quite a bit.

The panel **of the Master Instrument** gives access to two knobs: **Main** for controlling the master volume of the ensemble, and **Tune** for setting the master tuning.

Together with the panels for **Pad** and **Stereo Echo** you see all control elements of the ensemble.

Before we give you some time alone with the **Padecho** ensemble, we'll 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, coarsely with the knob labeled **Interval**, and finely 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 LFO 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 **SynthOne**.

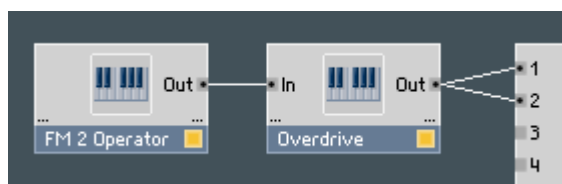
The **Echo Stereo** consists of two delay lines, one of which processes the left and the other the right stereo channel. Their delay times can be controlled independently of each other using **Del L** and **Del R**, where a setting of zero means that there is no delay. The desired number of echo repeats 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, controlling the strength of the effect.

FM Overdrive



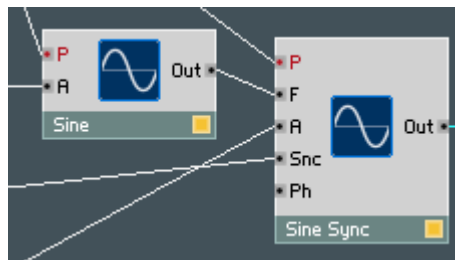
The next ensemble we want to present to you is called **FM-Overdrive.ens**. (**FM-Overdrive** = FM with 2 Operators + Overdrive.) You can also find this in the by now familiar **Tutorial Ensembles** folder.

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



The **FM 2 Operator** synthesizer demonstrates 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 aren't 6 operators (as in the DX7), or 4 like in the smaller DX models, but only 2 operators, so the whole structure should be quite clear.



Both operators consist of an oscillator that generates a sine wave. One, called the carrier, is 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 sound's timbre.

Play a few notes on your MIDI instrument. Not very exciting, is it? Now slowly turn the **FM** knob upwards and listen to 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

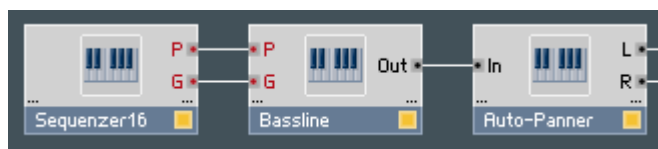
your FM sound creation with some amount of acoustic grit. The best thing is probably if you first try out the various snapshots before dedicating yourself to the following explanation of this device.

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** (Squnc16 = 16-Step Sequencer) with which we are now going to experiment, can also be found in the **Tutorial Ensembles** folder.



A look at the Ensemble structure tells us something about the construction of this ensemble: **Sequencer16**, a 16-Step sequencer (another device that REAKTOR can provide) controls **Bassline**, a kind of 303 clone, whose signal reaches the audio output via the **Auto Panner**.

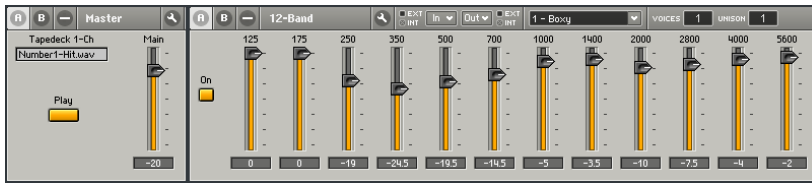
The panel of **Sequencer16** is already visible and, if the **Run** button is already pushed, you should hear the pattern playing. Press **Run** once to stop the pattern and another time to restart it. You can change the pitch for every step with the **Pitch** faders in the top row, and the volume for 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.

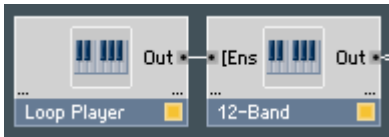
The **Bassline** instrument corresponds to what you may know from a 303; but even if you have never seen such a beast, with such a small number of controls it's unlikely that any confusion will arise. Just turn any knobs you want and listen to the result.

As you have probably already noticed, the sound in this ensemble is always moving back and forth between the left and right speakers. The **Auto Panner** is responsible for this. 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 will find it with the previous REAKTOR examples in the **Tutorial Ensembles** folder.



This ensemble is made up of the units **Loop Player** and **12-Band**. Before you can hear anything, you first need to load a sample into **Loop Player**. Windows XP: Right-click / OS X: Ctrl+click on the sample slot displaying “untitled*.wav” in the panel and choose **Load Audio in Tapedeck...** from the context menu. In the Open Audio File dialog select any one of the WAV or AIF files on your hard drive and load it by clicking on **Open**. Presto, a click on the **Play** button and you should hear the newly loaded sample as a loop. If **Play** is already pressed and you don't hear anything, or if the sample fails to loop, press **Play** twice to reinitialize it and start playback.

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. Each fader has a number that indicates what frequency band (measured in Hz) it controls. 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 a good old-fashioned analog synthesizer? 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 **macros**.

Note: REAKTOR supports two kinds of macros: primary macros (macros that reside within the primary level of REAKTOR) and core macros (macros that reside within the core level of REAKTOR). In this section, we are speaking exclusively about primary macros. For DIY information on core macros, see the REAKTOR Core manual.

In REAKTOR terminology, macros are functional blocks that make the construction of complex structures 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 **Run/Stop Audio** button in the Main toolbar, so that you don't get startled when the half-finished construction suddenly starts making noises.

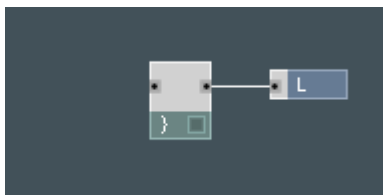


To begin with we will prepare the workspace in which to construct the synth. Please open the **File** menu at the top of REAKTOR and choose the entry **New Ensemble**. After this you will see (in the Ensemble structure) our old friends **Audio Out** and **Audio In**, as well as two instruments, **Instrument** and **Master**.



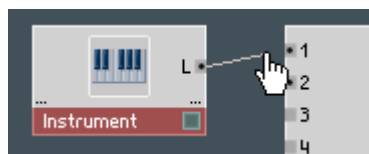
To proceed, delete the default **Instrument** since we want to see how to build an ensemble from ground up. The other instrument called **Master** should remain, since it contains the important global controls **Level** and **Tune** which appear in the **Panel** window.

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. XP: Right-click / OS X: Ctrl+click on a blank part of the Ensemble Structure window and in the context menu choose **Insert Instrument** ⇒ **New - 2In2Out**. An empty instrument named **Instrument** appears in the structure. Double-click **Instrument** to open its structure and delete all terminals inside it except for the **L** output terminal and the **Audio Voice Combiner** (}) in front of it.

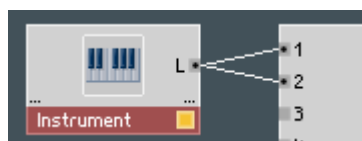


Double-click on a blank part of the **Instrument** structure to redisplay the Ensemble structure window. (Double-clicking on a structure is a great shortcut for moving one structure up in the ensemble hierarchy.) Click on the **L** output

port of **Instrument**, drag the mouse pointer to **1** input port of the **Audio Out** module, and release the mouse button.



Do you now see a wire connecting the two components? If not, try again. If so, we congratulate you on creating your first virtual wire! In the same way connect the **L** output of **Instrument** to the **2** input of **Audio Out** so that you can later hear the sound on both channels.



Choice of Components

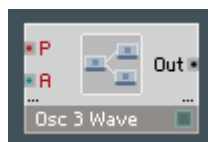
For our synth we need one or more oscillators whose output signal should go through a filter and whose volume should be controlled by an envelope, and that's it. We will now assemble these components.

Take a look at the **Master** and **Instrument** panels (not structures!) in the Ensemble Panel window.



The **Instrument** panel is empty (because we haven't added any visible objects to its structure yet), and **Master** contains only the **Level** and **Tune** controls.

Open the Structure window for **Instrument**. It is here that we will insert the components mentioned above. XP: Right-click / OS X: Ctrl+click in **Instrument's** Structure window and in the context menu choose **Macro** ⇒ **Building Blocks** ⇒ **Oscillators** ⇒ **Osc (pls, saw, tri)** . In the Structure window you can now see the insert macro, which is named **Osc 3 Wave**.



A quick look at the **Instrument** panel, which was empty until just a few moments ago, shows the controls contained in **Osc 3 Wave**.

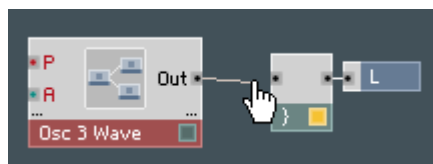


We're making progress.

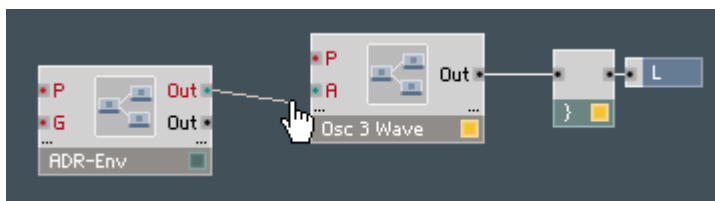
Before we go on we should make sure that the oscillator is working. As a precautionary measure, set the **Level** fader of **Master** to, let's say, **-10** to avoid any nasty surprises during the following audio test.



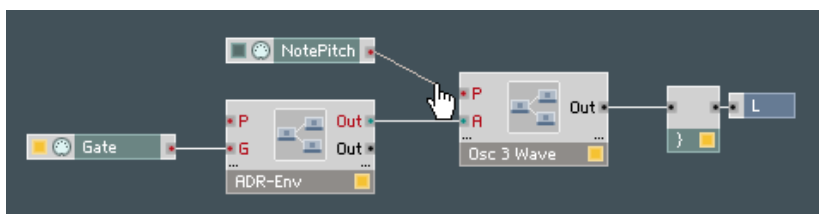
In the **Instrument** structure, connect a wire from the **Out** port of the **Osc 3 Wave** macro to the input port of the **Audio Voice Combiner** module (**}**). To do this, click and drag your mouse from one port to the other (the direction doesn't matter). The **Audio Voice Combiner** serves to convert a polyphonic signal into a monophonic one. This conversion is especially important in front of an instrument output port, since instrument output ports are generally monophonic.



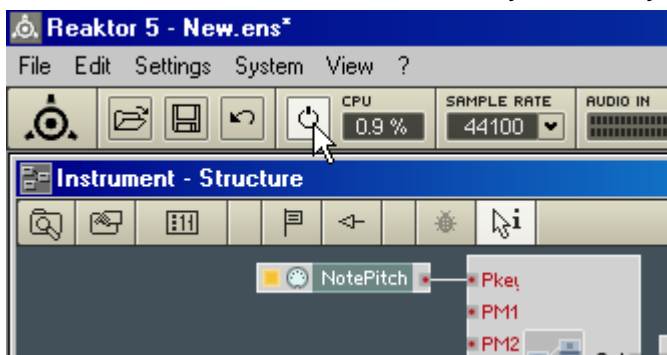
Let's add an **ADSR** volume envelope to the **Osc 3 Wave** macro by using the context menu again and choosing **Macro** ⇒ **Building Blocks** ⇒ **Envelopes** ⇒ **ADSR - Env**. Connect the lower **Out** output port (the black one, not the red one) of the **ADSR-Env** macro to the **A** input of the **Osc 3 Wave** macro.



Now we only need two more important MIDI modules to get a connection to an external MIDI input device. Insert the **NotePitch** module (**Built-In Module** ⇒ **MIDI In** ⇒ **Note Pitch**) and connect it to the **P** input of the **Osc 3 Wave** macro. Finally we need a **Gate** module (**Built-In Module** ⇒ **MIDI In** ⇒ **Gate**), which has to be connected with the **G** input of the **ADSR-Env** macro.



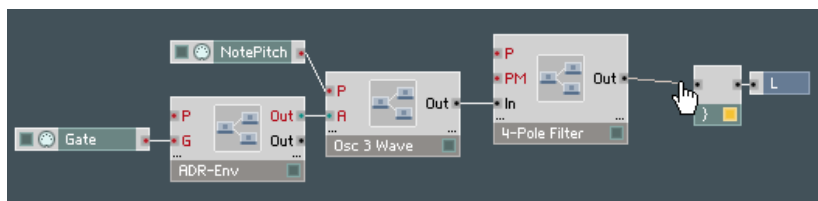
Turn on the **Run/Stop Audio** button in the Main toolbar, and press some keys on your MIDI keyboard (or computer keyboard). You should hear sounds from the synth. (If not, check your wiring, and/or save/reload the ensemble.) This is it, our oscillator in its raw form – not really beautiful (yet) but audible.



Before loading the next component, the filter, first remove the wire you just drew between **Osc 3 Wave** and the **Audio Voice Combiner**. Simply drag from one port to another again, as if connecting a second wire, and the connection is gone. Alternatively, just click on the wire to select it (it changes color) and press the **Del** key.

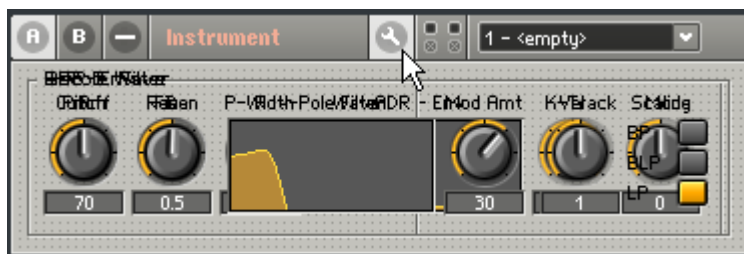


Now load a Filter macro in the same manner as you did the other macros. You will find it in the context menu under **Macro** ⇒ **Building Blocks** ⇒ **Filter** ⇒ **4 Pole Filter (BP, BLP, LP)**. Connect the **Out** port of the **Osc 3 Wave** macro to the **In** port of the **4 Pole Filter** macro, and then connect the **Out** port of the **4 Pole Filter** to the input of the **Audio Voice Combiner** so that you get a signal at the output again.

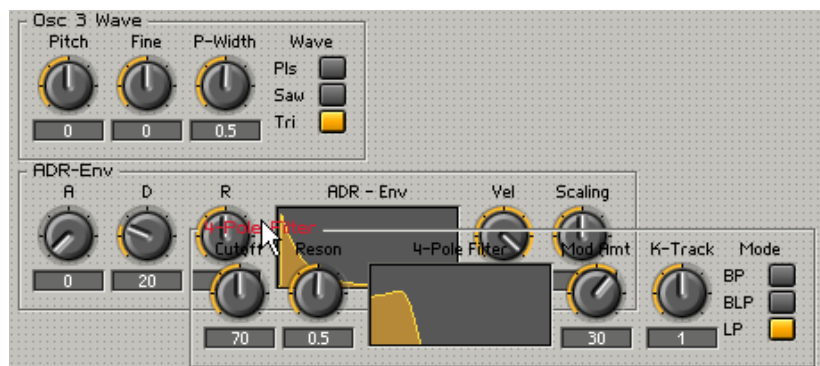


Note that several new filter controls have appeared in the panel. If you cannot see these controls, because your **Instrument** panel is a jumble of overlapping controls, do this to clean it up:

- 1) Click the **Lock/Unlock Panel** button (wrench icon) in the **Instrument** panel header to unlock the panel;



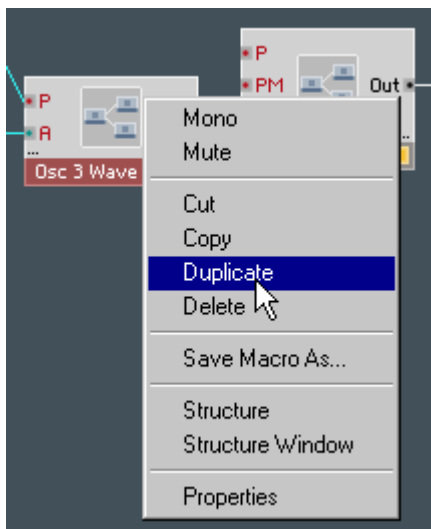
- 2) Drag each macro (by its title at the top of its frame) to its own area in the panel;



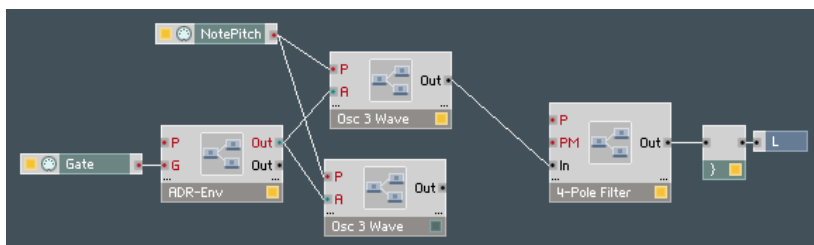
- 3) click the **Lock/Unlock Panel** button again to re-lock the panel. You should now be able to see all three macros clearly: **Osc 3 Wave**, **ADR-Env**, and **4 Pole Filter**.



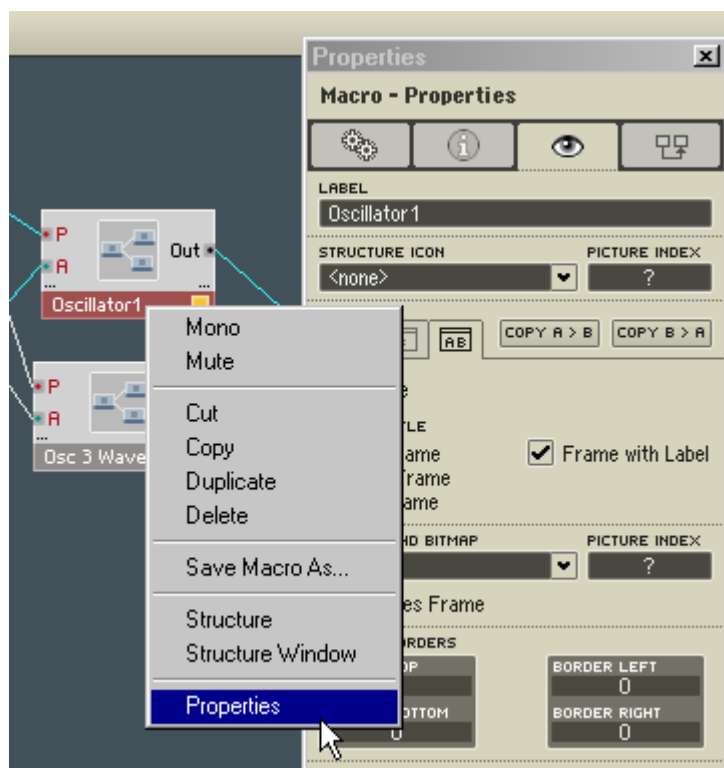
On considering these components you may wonder if a one-oscillator synth is the be all and end all. After all, it's well known that two oscillators simply give a fatter sound. OK, so let's add another one. In the Instrument structure, XP: Right-click / OS X: Ctrl+click on the macro **Osc 3 Wave** (careful, don't hit one of the ports), select **Duplicate** from the context menu. Done.



It is important to maintain a clean design – especially when dealing with a complex synthesizer. It's easy to create a chaotic layout and wind up spending hours in search of a problem's cause. So - if you didn't do so before like in the screenshots above - let's clean up the Structure window a bit. Move the **4 Pole Filter** macro to the left of the **Out** port, with the **Audio Voice Combiner** (**}**) module in-between. And place the two **Osc 3 Wave** macros neatly above one another to the left of the **4 Pole Filter** macro.



For the next step, let's remove some of the confusion between the two oscillators, which at present are identical, even sharing the same name. Let's give them different labels. To do so, XP: Right-click / OS X: Ctrl+click on the upper **Osc 3 Wave** macro and choose **Properties** from the context menu. A window will appear with a field called **Label** in the top left; here you can enter a new name, say, **Oscillator1**. Do the same for the lower **Osc 3 Wave** macro, but label it **Oscillator2**. You can also rename the macro **4 Pole Filter** to simply **Filter**.

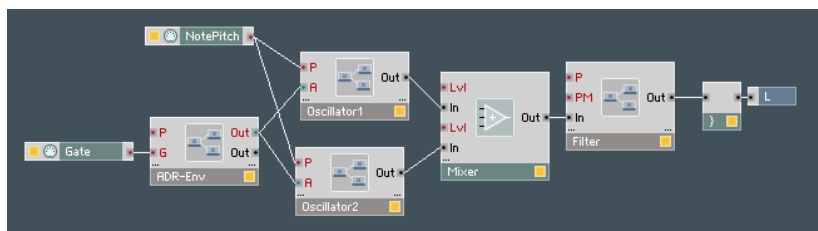


Now we want the signals of both oscillators to enjoy treatment by the filter, so we need a second wire from **Out** port of **Oscillator2** to the **In** port of **Filter**. We clearly have a problem here, because REAKTOR does not allow you to connect two wires to the same port. Of course that's not 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 oscillator macros and pass the sum on to the **In** port of **Filter**. And this component, a mixer for audio signals, is the **Amp/Mixer** module.

To insert the **Amp/Mixer** module, XP: Right-click / OS X: Ctrl+click on a blank part of the Structure window and in the context menu choose **Built-In Module** ⇒ **Signal Path** ⇒ **Amp/Mixer**.



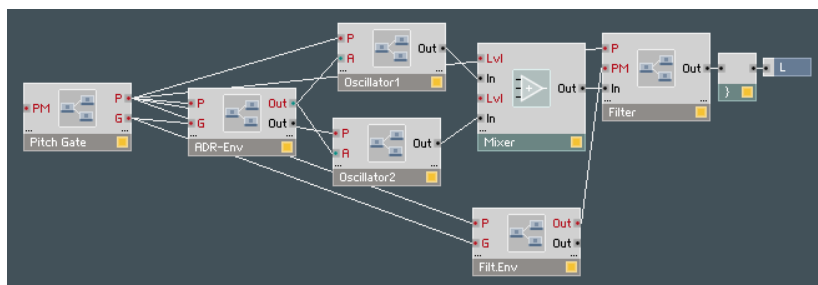
Now place the **Amp/Mixer** module between **Oscillator1 / Oscillator2** and **Filter** (we want things to be neat), and connect the output of **Oscillator1** to the **In** port of **Amp/Mixer**. We need to connect the output of **Oscillator2** to **Amp/Mixer** also. But **Amp/Mixer** only has one In port and, as we know, you cannot connect two wires to the same port. Fortunately, this is not a problem, because the **Amp/Mixer** module supports dynamic Input port handling. Simply XP: Ctrl+drag / OS X: ⌘+drag (i.e. hold down Ctrl/APPLE while dragging) from the **Out** port of **Oscillator2** to a spot just below the occupied **In** port of **Amp/Mixer**. This causes **Amp/Mixer** to create another In port, to which you should now complete your connection. The rest is child's play: a wire from the output of **Amp/Mixer** to **In** of the **Filter**, and a wire from **Out** of the **Filter** to the input of the **Audio Voice Combiner**. Done.



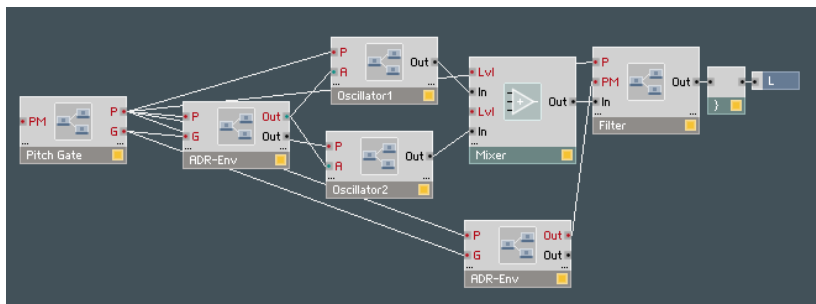
As soon as you complete the last connection, the status LEDs of all modules should light up to indicate that we now have a functional structure.

If you did not use the Duplicate option to get a copy of Oscillator 1 (see above) but did the good old Copy/Paste action you have to connect the out ports of the modules **Gate** and **Note Pitch** to the **P** and **A** in ports of **Oscillator2** as you did for **Oscillator1** to transmit MIDI notes to **Oscillator2** also.

An extra ADR envelope for the Filter makes our synthesizer sound richer. Duplicate the **ADR-Env** macro and assign it to the filter by connecting its upper **Out** (red) port to the **PM** input of the **Filter** macro. Rename it to **Filt.Env**



And, finally, let's exchange the two modules **NotePitch** and **Gate** for a single macro which does the same but also has an integrated module for Pitchbending. Delete **NotePitch** and **Gate** from the structure. Select **Macro** ⇒ **Building Blocks** ⇒ **Pitch+Gate** ⇒ **Pitch + Gate** to insert the **Pitch Gate** macro into the structure. Connect the **P** output of **Pitch Gate** to the **P** inputs of the two **ADR-Env** macros, **Oscillator 1** and **Oscillator 2**, and **Filter**. The **Pitch Gate G** output has to be connected to the **G** inputs of both **ADR-Env** macros.



Now the synthesizer can be played properly. The pitch and volume of the incoming MIDI notes are recognized by the synth, and even the pitchbend wheel on the MIDI keyboard works, because the **Pitch Gate** macro is set up to handle all of these tasks.

Arranging the Panel

Have a look now at the **Instrument** panel. You see a bunch of knobs that are wrapped up within frames to form groups. Each frame corresponds to one of the macros that we have inserted, so we know exactly which controls belong to the **Filter**, which to **Oscillator2**, etc.

Now you can start polishing the panel design. For example, at the moment the controls for **Oscillator2** are still overlaying those for **Oscillator1**. To change this, unlock the panel by clicking the **Lock/Unlock Panel** button (wrench icon) in the instrument panel header. You can tell that the panel is unlocked, because the **Lock/Unlock Panel** button is lit, and the panel is overlaid with a grid. (Note that you can play an instrument when its panel is unlocked, but you cannot change any of its control settings.) Drag all five macros to suitable locations in the panel.



Possible result after polishing the panel layout

Once you are happy with the layout you can freeze it in its current state to make sure that knobs or frames aren't moved inadvertently. Simply click again on the **Lock/Unlock Panel** button to lock the panel.

Saving

You probably want to save your two-oscillator synthesizer instrument so that you can reuse it in another ensemble. First, let's name it something other than its default name, **Instrument**. Double-click on the name **Instrument** in the panel header to open its Properties dialog. In the Label field, type **My DIY Synth** (or similar) and press the **Enter** key. Now let's save the instrument. XP: Right-click / OS X: Ctrl+click on your new name in the panel header and choose **Save Instrument as...** from the context menu.

In the Save Instrument dialog that appears, choose a folder in which to store your instrument file, specify a file name (or use the one that REAKTOR suggests: **My DIY Synth**) 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 **My DIY Synth** instrument) has already been saved.

Note: To save the entire ensemble, rather than just one instrument in the ensemble, use **File⇒Save Ensemble...** from the main menu

Luxury

If after some time you feel like adding more features, 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. First, try to change the Envelope type from ADR to ADSR to change the percussive sound of your first DIY synth and get something like a lead synthesizer.

9.3. Your First DIY Structure

Our DIY synthesizer project was executed mainly using prebuilt macros. We would now like to introduce you to the art of constructing a synthesizer completely from scratch. Contrary to the recommendation we gave above, which was to always separate larger functional units into macros, this new synthesizer will be constructed entirely with modules in a single Structure window, no macros. 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 **File**⇒**New Ensemble** from the main menu to open a new ensemble. In the Ensemble Structure window, delete the default **Instrument**. All that should be left is the default **Master** instrument (containing Level and Tune controls), and the **Audio In** and **Audio Out** modules.

XP: Right-click / OS X: Ctrl+click on a blank part of the Structure window and choose **Insert instrument** ⇒ **New - 2In2Out** from the context menu. An empty instrument named **Instrument** appears in the structure. Rename **Instrument** to **My DIY Struct** (or similar). Double-click **My DIY Struct** to open its structure and delete all modules inside it except for the **L** output terminal and its connected **Audio Voice Combiner** (}). Double-click on a blank part of the **My DIY Struct** structure to move one level up to the Ensemble structure. Connect the **L** output port of **My DIY Struct** to input ports **1** and **2** of **Audio Out**.

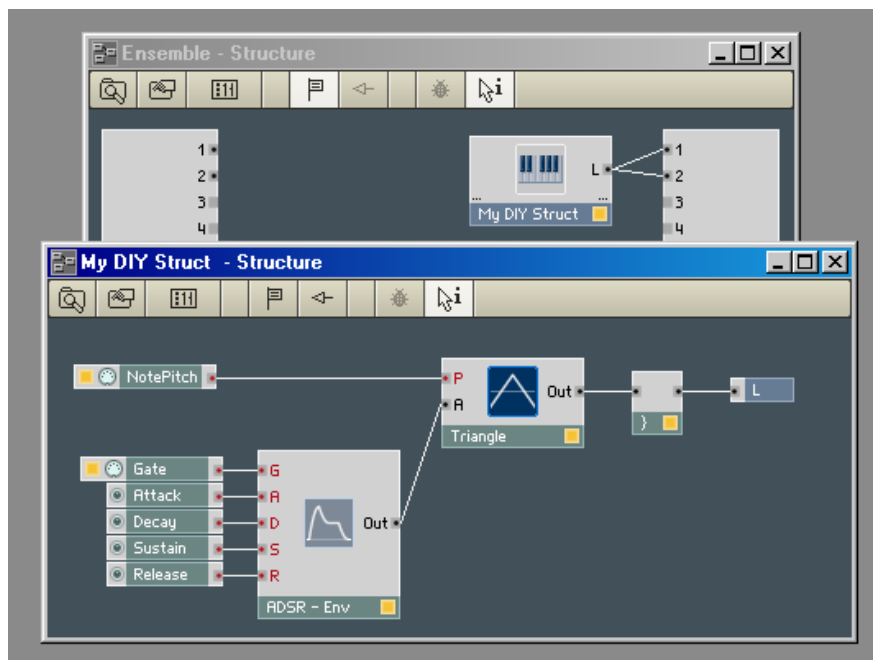
Open the **My DIY Struct** structure by double-clicking its icon in the Ensemble structure. It is here, in the **My DIY Struct** Structure window that we will implement our synthesizer circuitry.

First we insert an oscillator. Our choice this time is an oscillator module that generates a triangle wave. XP: Right-click / OS X: Ctrl+click on a blank part of the Structure window and in the context menu choose **Built-In Module** ⇒ **Oscillator** ⇒ **Triangle**.

The next step is to add modules that will tell the synthesizer about the volume (gate) and pitch of the incoming MIDI notes. To that end, use the context menu to select **Built-In Module** ⇒ **MIDI In** ⇒ **Gate** and then **Built-In Module** ⇒ **MIDI In** ⇒ **Note Pitch**. For the envelope we choose an **ADSR-Env** module (**Built-In Module** ⇒ **LFO, Envelope** ⇒ **ADSR**).

Now position and interconnect the modules according to the following il-

illustration. Use this shortcut to create the **Attack, Decay, Sustain, Release** controls that connect to the **A, D, S, R** input ports of the **ADSR-Env** module: XP: Right-click / OS X: Ctrl+click on each of the input ports and select **Create Control** from the context menu. Experienced builders use this “trick” all the time to create input-port controls (that they then modify, as necessary, to meet their needs).



How does it all work?

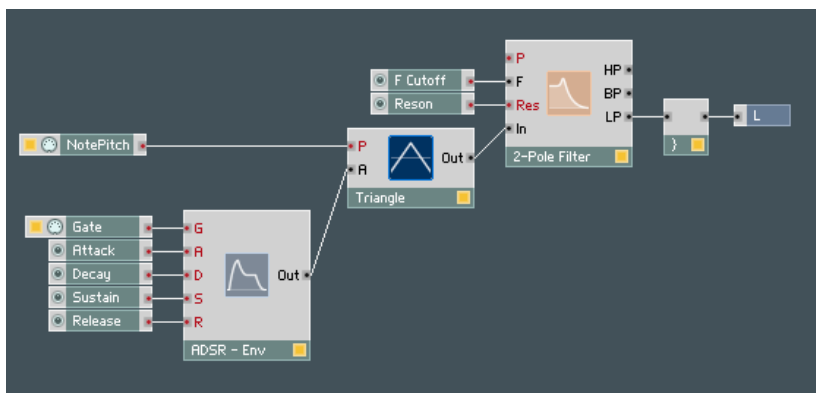
On studying this structure, the following functionality becomes apparent: The **ADSR-Env** module generates an envelope whose shape is specified with the **Attack, Decay, Sustain** and **Release** knobs in the **My DIY Struct** panel window (keep it tidy!). The envelope is triggered by a rising signal at the gate input **G**, in our case generated by the press of a MIDI key. The pitch of the incoming MIDI note is received by the **NotePitch** module and sent to the **Triangle** module through its **P**(itch) input.

You can already play this synthesizer, but you will very likely soon get tired of the sound - other than the volume envelope there's simply nothing that can be adjusted. The whole thing, however, becomes much more interesting when a filter is brought into play.

Adding a Resonant Filter

Use the context menu to insert a 2-pole filter (with FM) into **My DIY Struct** structure (**Built-In Module** ⇒ **Filter** ⇒ **Multi 2-PoleFM**). Then connect **Out** of the **Triangle** module to **In** of the **2-Pole Filter** module, and **LP** of **2-Pole Filter** to the **Audio Voice Combiner** (}).

Next, using **Create Control** (as discussed above), create controls for the **2-Pole Filter** inputs **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 **F Cutoff** and **Reson** in the **My DIY Struct** panel window. (To see these knobs, you'll have to unlock the panel, arrange the controls, and lock it again. If the Lock/Unlock Panel button is not visible in an instrument header, because the header is too narrow, use the header's context menu to select **Lock/Unlock Panel**.)



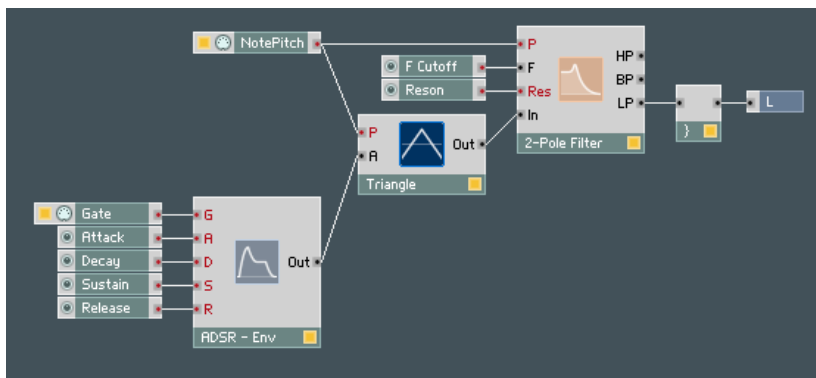
F Cutoff sets the filter's cutoff frequency. When utilizing the output **LP** (low pass) of the **2-Pole Filter** module, as done here, all frequencies above the cutoff frequency are removed. By using the other outputs of **2-Pole Filter** it can also be employed as a band pass (**BP**) or high pass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

Reson sets the resonance of the filter. The higher the resonance value, the more the frequencies near the cutoff frequency are boosted. If you set **Reson** very high, say ≥ 9.5 , the filter will begin to self-oscillate. Beware: This can cause extremely loud feedback-type sounds which can, potentially, damage your speakers (and ears)!

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.

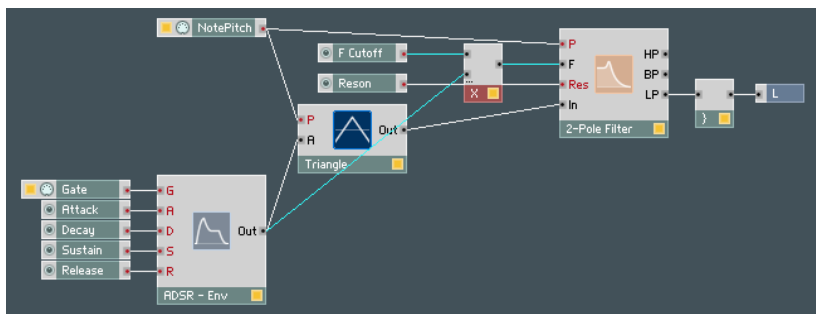
The **2-Pole Filter** circuit is designed to add the control signal at the **P** input to the frequency control signal at the **F** input. The sum of the two then determines the filter's cutoff frequency. If you play some high notes they will sound as you would expect.



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-Env** module take over this task, too. If you wanted to have a more sophisticated synth, you could add a separate envelope just for the filter.

For the envelope to affect the filter cutoff we first need another component, a multiplier (**Built-In Module** ⇒ **Math** ⇒ **Multiply**). Connect **Out** of **ADSR-Env** to one of the two input ports of **Multiply (X)**, the output of **F Cutoff** to the other **Multiply (X)** input, and the **Multiply (X)** output to the **F** input of **2-Pole Filter**, as illustrated below.



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

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

Variations

Here are some 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.

Be warned, however, that unexpected - and potentially very loud - sounds will be generated from time to time as you experiment with REAKTOR. To protect both your speakers and your ears, set your initial amplification levels low.

All right! Now get cracking!

10. Basic Operation

The REAKTOR user interface follows the conventions of your computer's operating system, so it is easy for someone who has already worked with OS X or Windows to get used to the software. Nevertheless, we want to explain some particular characteristics of REAKTOR 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:

- **Selecting** an object is done by clicking on it with the left mouse button. Selected objects (instruments, macros, modules, etc.) are recognized by their title bar which is colored red. If you want to select several objects, hold down the XP: **Ctrl** key or the OS X: **Shift** key on your 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 done by clicking the left mouse button on it and keeping the button pressed while dragging 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, and all the wiring to other objects will remain intact, merely stretching 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 and holding the left mouse button on the output port of the object that is to transmit the signal, and then dragging the mouse pointer, and with it the wire, to the desired input port of the object that is to receive the signal. Simply release the left mouse button and the connection is established. Wiring operations can also be carried out in the reverse direction (from input port to output port) the results will still be the same.
- **Double-clicking the left mouse button** on an object (or on the background of a window) will perform various actions, depending on the object. Actions that can be executed by a double-click appear in bold type in an object's corresponding context menu.

- **Clicking the right mouse button (XP) or holding down the Ctrl key and clicking the mouse button (OS X)** 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 REAKTOR's 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 if you need information about it, XP: right-click or OS X: Ctrl+click on it. A context menu will appear whose entries apply to the selected object. Click (with the left mouse button) on a menu item to select it. The menu will disappear and the operation will be carried out. For example, you can delete a module by selecting the entry **Delete** from its context menu.

10.3. Key Commands

Many functions in REAKTOR can be performed with keys or key combinations in addition to the mouse. Available key commands are listed in the menus next to the command.

10.4 Ensemble Panel and Structure Windows

The REAKTOR workspace comprises two windows: the Ensemble Panel window and the Structure window. The Ensemble Panel window contains the ensemble panel and all of the ensemble's instrument panels. The Structure window contains the structure (internal wiring) of the currently selected object (ensemble, instrument, primary macro, core cell/macro).

There is one Ensemble Panel window. By default, there is one Structure window also. You can, however, choose to open multiple Structure windows by **Alt +** double-clicking the desired objects, or selecting **Structure Window** from the **context menu**. Though we recommend that you work with a single Structure window to keep your screen (and mind!) uncluttered, REAKTOR allows you to open as many separate Structure windows as you like.

Here are some guidelines for managing your REAKTOR Ensemble Panel and Structure windows:

- To open the Ensemble Panel window, choose **View->Show Panel** from the main menu. Or, if you are working in a Structure window, use this trick: Double-click on a blank part of the Structure window to display its parent Structure window. Keep doing this until you reach the Ensemble Structure window. Double-click there and REAKTOR will display the Ensemble Panel window.
- To open a Structure window, double-click on the desired object (to open it in the shared Structure window) or **Alt +** double-click on the object (to open it in a separate Structure window).
- All open Ensemble Panel and Structure windows are listed at the bottom of the **View** menu. To jump to an open window, select it in the list (or, if the window is visible, simply click anywhere within it).
- To jump one structure up in the hierarchy – i.e. to the structure that *contains* the current structure – double-click on a blank part of the Structure window.
- You move, resize, minimize, and close REAKTOR windows just as you would for any other windows on your platform. If a window is too small to display its entire contents, scrollbars at its right and bottom edges enable you to scroll through the window's contents.

The following applies to the use of REAKTOR in Windows:

- As is the norm with Windows programs, all REAKTOR Ensemble Panel and Structure windows are contained inside the main REAKTOR application window. When this main window is resized, minimized, or covered by another application, all the contained windows are affected.
- When an Ensemble Panel or Structure window is maximized, it expands to fill the entire main REAKTOR window, and all other windows are also maximized until any one of them is reset to a smaller size.
- When a window is minimized, it appears as a small rectangular box at the bottom of the main window.
- To step through all open windows, use **Ctrl + Tab**.

11. Menus

In addition to the various **context menus**, the commands for using REAKTOR are accessible from the menu bar of the main window. The program's global functions, controlled from the menu bar, are described below.

11.1. File Menu

New Ensemble

Selecting **File->New Ensemble** (or pressing XP: **Ctrl + N** / OS X: **⌘ + N**) creates a new ensemble that contains a **Master** instrument and **Audio In** and **Audio Out** modules.

Open...

Selecting **File->Open...** (or pressing Windows: **Ctrl + O** / OS X: **⌘ + O**) loads an ensemble file (*.ens) stored on your disk.

Save Ensemble

Selecting **File->Save Ensemble** (or pressing XP: **Ctrl + S** / OS X: **⌘ + S**) stores the current ensemble together with all its instruments, structures, panels, and snapshots in an *.ens file.

Save Ensemble As...

Selecting **File->Save Ensemble As...** (or pressing XP: **Ctrl + Shift + S** / OS X: **⌘ + Shift + S**) is identical to Save Ensemble (see above), but it enables you to specify a new filename and/or folder for the ensemble.

Save Window As...

Selecting **File->Save Window As...** (or pressing XP: **Ctrl + E** / OS X: **⌘ + E**) enables you to (re)name and store the contents of the currently selected window.

If the Ensemble Panel window is selected, the ensemble will be saved (in an *.ens file), just as if you had used the **Save Ensemble** menu command.

If a instrument structure window is selected, the instrument containing the structure will be saved (in an *.ism file) together with all its structures, panels, and snapshots.

If a Macro structure window is selected, the macro containing the structure will be saved (in an *.mdl file).

Import MIDI File...

There is an integrated MIDI File Player in REAKTOR that enables 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.

Because it has an integrated MIDI File Player, REAKTOR can play arrangements without a separate sequencer. This option can be especially appealing to live performers: A sequencer running in the background on the same computer could cause glitches and make your performance more difficult, since you would have to load new files into the sequencer as well as into REAKTOR. On top of that, you would then have to alternate between the two programs in order to access important parameters.

There is another advantage to using the integrated MIDI File Player instead of an external sequencer: sample accurate timing. All notes in a MIDI file that begin at the same time will be played by REAKTOR simultaneously, so the timing is perfectly tight. Of course, the MIDI file's timing depends on the accuracy and resolution of the sequencer it was created on.

The REAKTOR MIDI File Player can be loaded either manually or automatically: For manual operation, use **File->Import MIDI File...** from the main menu to load a MIDI file from your disk. For automatic operation, REAKTOR will load a MIDI file upon opening an ensemble if that file is in the same folder and has the same name as the ensemble (but with the extension .mid); for example, mySynth.ens and mySynth.mid.

In the **Settings** menu there are three entries for navigating the MIDI File Player. When **Play MIDI File** is enabled, the MIDI file is played back when you start the REAKTOR clock (by clicking the **Start/Restart Clock** button in the **Ensemble Panel Toolbar**). The MIDI file will be played in an endless loop when **Loop MIDI File** is enabled. You can use this for instance to keep repeating a pattern or sequence of patterns. Finally, **Ignore Tempo Change**, when enabled, causes REAKTOR to ignore all tempo in the MIDI file and play the file back with the tempo set by the REAKTOR clock (BPM).

The transport functions of the MIDI File Player are controlled from the REAKTOR clock:

- Click the **Start/Restart Clock** button to start MIDI File playback from the beginning or to restart playback at the place where the file was paused.
- Click the **Pause/Stop Clock** button once to pause playback of the MIDI File. Click the **Pause/Stop Clock** button a second time to stop playback and rewind the MIDI file back to the beginning.

Batch Processing

Batch Processing enables batch conversion of REAKTOR 3 files to the REAKTOR 5 format, and the analysis of audio files for the granular sampler modules. Simply plug in your REAKTOR 3 USB key, select a source and destination folder, and click on **OK**.

Recent Ensembles

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

Exit

Exit closes the REAKTOR program and all its windows, including those in the taskbar. If any changes have been made to the current ensemble since it was last saved, REAKTOR asks if you want to save the file before exiting.

11.2. Edit Menu

Undo

Selecting **Edit->Undo** (or pressing XP: **Ctrl + Z** / OS X: **⌘ + Z**) reverses the effect of the last editing operation carried out in any of the structures. The Undo function does not apply to panel control setting changes; i.e. changing the value of a knob or fader. For this, you want the Compare function in the Snapshots window.

You can set the maximum number of consecutive **Undo** commands in the **Preferences dialog -->Options** page. If your computer runs low on memory, try reducing this number.

Redo

Selecting **Edit->Redo** (or pressing XP: **Ctrl + Y** / OS X: **⌘ + Y**) reverses the effect of the most recent **Undo** operation (i.e. it “undoes” the last **Undo**). You can execute **Redo** as many times as you previously executed **Undo** until you wind up back where you started.

Cut

Selecting **Edit->Cut** (or pressing XP: **Ctrl + X** / OS X: **⌘ + X**) cuts (removes) the current selection and copies it to the clipboard. From there it can be inserted in another place using the **Paste** command (**see below**).

Copy

Selecting **Edit->Copy** (or pressing XP: **Ctrl + C** / OS X: **⌘ + C**) copies the current selection to the clipboard. From there it can be inserted in another place using the **Paste** command (**see below**).

Paste

Selecting **Edit->Paste** (or pressing XP: **Ctrl + V** / OS X: **⌘ + V**) copies the current contents of the clipboard into the selected structure.

When using the keyboard shortcut for pasting, you can specify where to paste to by clicking in the desired Structure window.

Duplicate

Selecting **Edit->Duplicate** (or pressing XP: **Ctrl + D** / OS X: **⌘ + D**) creates a copy of the current selection. It is equivalent to selecting Copy, then Paste.

Delete

Selecting **Edit->Delete** (or pressing the **Del** key) deletes the current selection. You can also use **Delete** in the context menu of the selected object (module, wire, etc.).

Select All

Selecting **Edit->Select All** (or pressing XP: **Ctrl + A** / OS X: **⌘ + A**) selects all the objects in the current window. You can then unselect individual objects by XP: Ctrl+clicking / OS X: **⌘**+clicking on them.

11.3. Settings Menu

Sample Rate

Sample Rate sets the sample rate at which REAKTOR generates and processes audio signals. With higher sample rates you can achieve better sound quality, but the CPU load rises proportionally. You can change the internal sample rate to any of the values in the menu. The range of available values depends on your sound card or host plug-in. If the internal sample rate is different from the sound card's or host plug-in's sample rate, the **Audio In** and **Audio Out** modules will do the necessary sample-rate conversion.

Control Rate

Control Rate sets the control rate for REAKTOR event signals; i.e. the number of times per second that event-signal values are updated. The control rate is applied globally to all primary modules that generate or process events; e.g. **LFO**, **Slow Random**, **Event Hold**, **A-to-E**, **Event Smoother**, and more. Since the control rate is very low compared to the sample rate, these modules need very little CPU power. For this reason, good builders choose to work with event signals rather than audio signals whenever possible (i.e. whenever it doesn't degrade the sound).

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

MIDI Learn

Selecting **Settings->MIDI Learn** (or pressing XP: **Ctrl + T** / OS X: **⌘ + T**) activates 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 **MIDI Learn** button (midi connector icon) in the **Ensemble Panel Toolbar**.

Set Protected/Set Unprotected

Enables/disables Protection mode. In protection mode only a very limited edit of the ensemble panel and structure is possible. Insertion, deletion, movement, of panel controls and the alteration of voices is disabled.

Automatic Layout

Enables Automatic Layout mode for all instrument panels. (This is equivalent to turning on **Automatic Panel Layout** in an ensemble's Properties dialog, Appearance page.) By default this option is switched on.

External Sync

Toggles between the internal REAKTOR clock and an external clock (received via MIDI) for all **Sync Clock** and **1/96 Clock** modules. Also enables control of **Start/Stop** modules by external MIDI-Start/Stop messages. When **External Sync** is turned on, the tempo cannot be adjusted by the master clock BPM field in the Main toolbar; instead, the internal clock is adjusted according to the external clock.

MIDI Clock Out

If you enable this option, REAKTOR sends out MIDI clock ticks on all active MIDI out ports (as set in the Audio Setup dialog, MIDI page).

Clock Start

Starts the REAKTOR master clock which controls all of the clock-driven modules in the ensemble. This works with both internal and external clocks. It sets the output of all **Start/Stop** modules to "start". In the Ensemble Panel toolbar there is a **Start/Restart Clock** button for the same function.

Clock Stop

Stops the REAKTOR master clock which controls all the **Sync Clock** and **1/96 Clock** modules in the ensemble. This works with both internal and external clocks. It sets the output of all **Start/Stop** modules to "on". In the Ensemble Panel toolbar there is a **Pause/Stop Clock** button for the same function.

Play MIDI File, Loop MIDI File, Ignore Tempo Change

These commands act on REAKTOR's integrated MIDI File Player. For more information about this, please see the Imp.

11.4. System Menu

Run/Stop Audio

With this menu command all audio computations can be started (**Run Audio**) or stopped (**Stop Audio**). In effect, this is the main on/off switch for the REAKTOR software. The same function can be accessed by the **Run/Stop Audio** button in the Main toolbar.

Debug

The **Debug** menu provides four options: **Measure CPU Usage**, **Show Module Sorting**, **Show Event Initialization Order**, and **Optimization**.

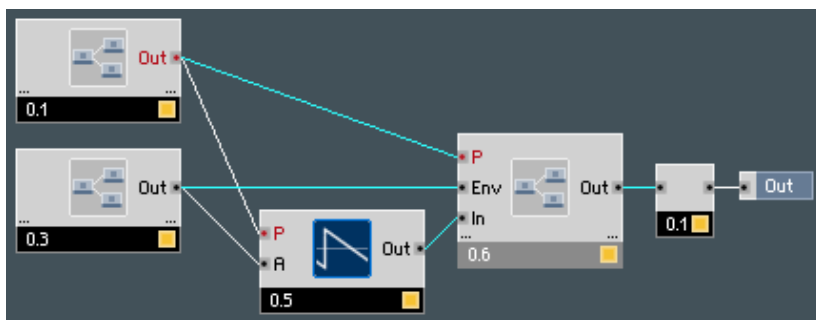
Measure CPU Usage

The **Measure CPU Usage** option switches all audio-processing components (instruments, macros, modules) to CPU-load measuring mode. The current CPU load is displayed in black labels on the components (in structure view). This feature is useful for determining how much of the total load is being caused by each component. This information can help you to streamline the structure, thus allowing for the generation of more voices.

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

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

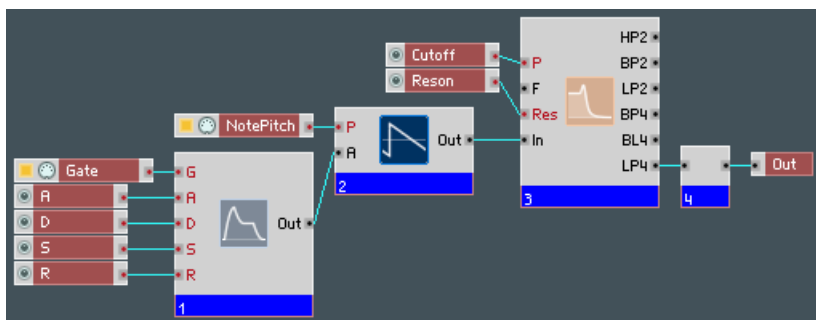
This mode is only available when **Run Audio** is active. During load measuring the audio output is switched off. You can enable the **Measure CPU Usage** option with the key combination XP: **Ctrl + U** / OS X: **⌘ + U**.



CPU Usage Display

Show Module Sorting

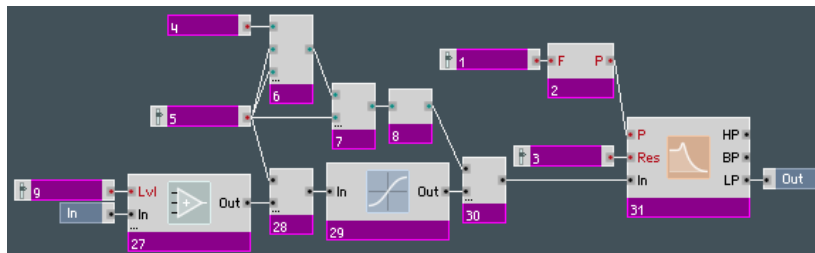
The **Show Module Sorting** option switches all audio-processing modules to sorting mode. In this mode, the current position of each module within the overall stream of audio processing is shown. This position will be displayed as a number in blue label on each module.



Show module Sorting mode

Show Event Initialization Order

This **Show Event Initialization Order** option displays the current position of each module within the overall stream of event initialization.



Show Event Initialization Order screenshot

Audio + MIDI Settings...

This menu item opens a dialog for selecting your audio and MIDI interfaces. Detailed instructions are found in the chapter **REAKTOR Standalone Version**

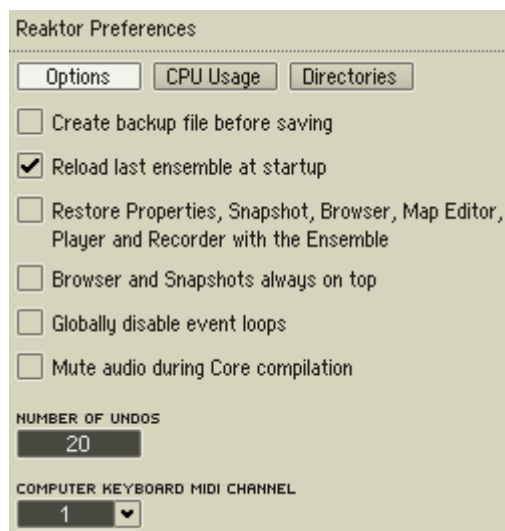
OSC Settings...

This menu item opens the OSC Setup dialog. Detailed instructions on how to use OSC are found in **Open Sound Control** chapter.

Preferences

This menu item opens the Preferences dialog in which you will find a number of REAKTOR preference options that you can set as desired. The Preferences dialog consists of three pages: Options, CPU Usage, and Directories.

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 previously saved version of a file if something should go wrong (by renaming the extension `.bak` to the original extension).
- **Reload last ensemble at startup** tells REAKTOR to load the ensemble that was active when it was last shut down.
- When enabled, **Restore Properties, Snapshots, Browser, Map Editor, Player and Recorder with the Ensemble** restores an ensemble's Properties dialog and Snapshots window, Browser, Sample Map Editor, Playerbox and Recorderbox just as they were when the ensemble was last saved.
- **Browser and Snapshots always on top** causes the Browser and Snapshots window to display on top of all other REAKTOR dialogs and windows.
- When turned on, **Globally disable event loops** prevents event-signal loops from occurring in ensembles. (If an event loop is about to occur, REAKTOR displays a message revealing the source of the loop and asking you how to proceed.) Event loops can lead to stack overflow crashes, which in turn can make ensembles un-playable and, in some cases, un-openable. If this happens, restart REAKTOR, turn on **Globally disable event loops**, open the problematic ensemble, and trace the source of the event loop

with the help of event-loop identification messages. (It can be useful to disable audio to prevent further loops occurring during this process.) We recommend that you globally disable event loops to maximize the stability of REAKTOR. To ensure backward compatibility, files saved in older versions of REAKTOR have event loops enabled by default.

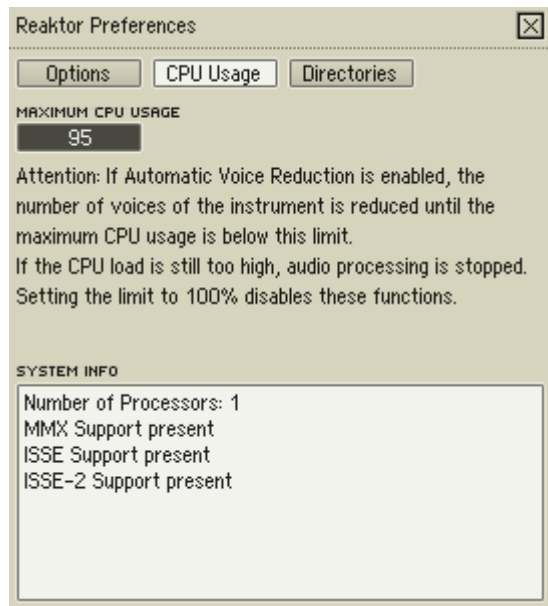
Note: In most cases, the Iteration module can avoid the need for creating event loops. The Iteration module has a limited speed option in its properties, which can avoid audio glitches caused by processing a large number of iterations too fast.

- **Mute audio during Core compilation**, when enabled, turns off all audio processing during the period of time that it takes to compile a core object. This makes compilation go faster.
- The **Number of Undos** box lets you set the maximum number of undos that can be performed in a row. Setting **Number of Undos** to 20, for example, enables you to undo the last 20 edits you've made by issuing 20 Undo commands in a row. Enter 0 to disable the undo function.

If you work on ensembles containing large audio files, you will need a lot of computer memory to continue to use the undo function. If you do run out of memory, try reducing the Number of Undos.

- **Computer Keyboard MIDI Channel** specifies the MIDI channel for MIDI notes played at the computer keyboard. This is set to 1 by default, because REAKTOR instruments are set to receive MIDI messages on channel 1 by default. If, however, you want to use the keyboard to play an instrument that receives MIDI on a different channel (e.g. 2), you would set Computer Keyboard MIDI Channel to this channel (2).

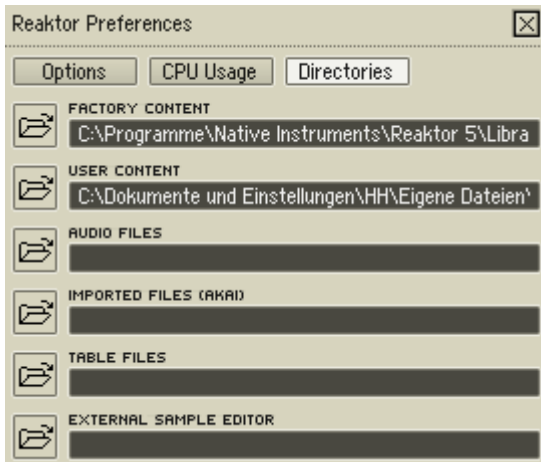
CPU Usage



REAKTOR can automatically reduce the number of voices in polyphonic instruments when the total CPU load of the ensemble reaches a certain limit. In this way, the polyphony can be adjusted according to the available processing power.

The upper CPU load limit is set by **Maximum CPU Usage** (where **80** = 80%, **95** = 95%, etc.). Note that REAKTOR changes the number of voices only for instruments in which **Automatic Voice Reduction** has been selected (Properties dialog, Function page). To disable automatic voice reduction for all instruments in an ensemble, set **Maximum CPU Usage** to 100.

Directories



Factory Content

This path specifies the default folder in which the REAKTOR system ensembles, instruments, primary macros, core cells, and core macros are stored; i.e. the objects that were automatically copied to your hard disk when REAKTOR was installed on your system.

The System Content path is used by all relevant context menus (e.g. Ensemble Structure and Panel windows, instrument and macro Structure windows, etc.) and by the top row of buttons in the Browser (Ens., Instr., Macro, Core. C., Core M.) to enable you to access system objects quickly and easily.

User Content

This path specifies the default folder in which custom user ensembles, instruments, primary macros, core cells, core macros, audio files, imported files, pictures files, snapshot files, and table files are stored. These are files that you create, acquire, modify, etc.

Just as with the Factory Content path, the User Content path is used by all relevant context menus (e.g. Ensemble Structure and panel windows, instrument and macro Structure windows, etc.) and by the User row of buttons in the Browser (Ens., Instr., Macro, Core. C., Core M.) to enable you to access your custom objects quickly and easily.

Warning: Never store user files in REAKTOR 5 system folders, because REAKTOR updates might delete them!

Audio Files

This path specifies the initial default folder for loading and saving audio files (*.wav, *.aif, *.aiff) from anywhere in REAKTOR: Player, Recorder, Sample Map Editor, Tapedeck and Sampler modules, etc.

Imported Files (Akai)

This path specifies the default folder for storing Map files (.map) that have been converted with the Akai-Import function.

Table Files

This path specifies the default folder for loading and saving table files (in the Properties dialog of a Table module). Table files have the file extension *.ntf and can be used in the Audio and Event Table modules.

External Sample Editor

Enter the pathname of your favorite sample editor here, and you can launch it by selecting **Edit** from the drop-down Edit Sample List menu in the Sample Map Editor window.

11.8. View Menu

Show/Hide Hints

Shows and hides the popup hints that appear when the mouse points to an object (toolbar button, module, port, etc.). The keyboard shortcut is XP: **Ctrl + I** / OS X: **⌘ + I**.

Show/Hide Toolbox

Shows and hides the Main toolbar. For details, see Toolbar. The keyboard shortcut is XP: **Ctrl + F1** / OS X: **⌘ + F1**.

Show/Hide Playerbox






Shows and hides the Playerbox. The keyboard shortcut is XP: **Ctrl + F2** / OS X: **⌘ + F2**.


The Playerbox enables you to route the audio output from an audio file (*.wav, *.aif, *.aiff) into your REAKTOR ensembles for processing (filtering, delay, reverb, etc.). The Playerbox output audio signal is made available to an ensemble from the top two ports of the Audio In module (in the Ensemble Structure window).





Important note: When you play an audio file in the Playerbox, all non-Playerbox audio signals at the top two ports of the Audio In module will be muted.

The Playerbox contains the following controls:

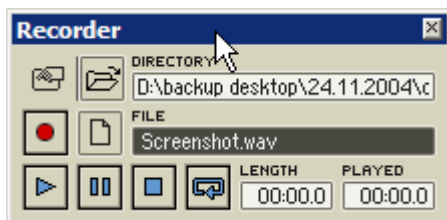
-  **Load:** Loads an audio file from the hard disk.
-  **File** drop-down menu: After having loaded an audio file, the folder from which it came will then be scanned for additional audio files. These files will appear in the **File** drop-down menu for quick access.
-  **Play:** Starts playback of a loaded audio file.

Note: The audio file will play at the current REAKTOR sample rate, as set in Sample Rate in the Main toolbar.

-  **Pause:** Pauses playback of an audio file. Press the Pause button again to proceed with playback.

-  **Stop:** Stops playback and goes back to the beginning of the audio file.
-  **Loop:** Plays an audio file in an endless loop.
-  **Length:** Shows the length of the loaded audio file.
-  **Played:** Shows the current playback position.

Show/Hide Recorderbox




Shows and hides the Recorderbox. The keyboard shortcut is XP: **Ctrl + F3** / OS X: **⌘ + F3**.

The Recorderbox enables you to record the audio output of an ensemble by writing it to an audio file on your hard disk (XP-systems record .wav files, OS X-systems record .aif files). The Recorderbox records the audio signal that is output to the top two ports of the Audio Out module (in the Ensemble Structure window).

The Recorder can also play an audio file by sending its output to the top two ports of the Audio Out module.

Note: When you play an audio file in the Recorderbox, all non-Recorderbox audio signals at the top two ports of the Audio Out module will be muted.

The Recorderbox has its own Recorder Settings dialog where you can define conditions for starting/stopping the Recorder. To open the Recorder Settings


dialog, click the **Recorder Settings** button  in the upper-left corner of the Recorderbox.




The Recorder Settings dialog provides the following options:


- **Record Start By (Manual, Note On, Clock Start):** When **Manual** is enabled, you start the Recorder by arming it using the Record button hitting the Play or Pause button. When **Note On** is enabled, the Recorder is started by a MIDI Note On event. When **Clock Start** is enabled, the Recorder starts when the REAKTOR clock is started with the Start/Restart Clock button in the Main toolbar.
- **Record Stop By (Manual Only, Note Off, Clock Stop, Loop Length):** When **Manual Only** is enabled, you stop the Recorder using the Stop button. It is also possible to stop the Recorder for a while and resume with the Pause button. When **Note Off** is enabled, the Recorder is stopped by a MIDI Note Off event. When **Clock Stop** is enabled, the Recorder stops when the REAKTOR clock is stopped with the Stop/Pause Clock button in the Main toolbar. When **Loop Length** is enabled, the Recorder is stopped when the loop length duration (as specified by the Loop Length option, see below) has been reached.
- **Start Offset (Bars)** specifies a time delay (in bars) before 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.
- **Loop Length (Bars)** specifies a duration (in bars) for the recording, when **Loop Length** is selected in the **Record Stop By** section above.
- **Slave Player Controls to Recorder:** When on, this causes the Playerbox transport controls to be slaved to the Recorderbox transport controls.


The Recorderbox contains the following controls:


-  **Load:** Creates a new audio file (or selects an existing one) to record to. Be careful: If you select an existing audio file, the recording will overwrite (delete) the file's previous contents. You can also use **Load** to load an audio file for playback (as described above).


-  **Directory:** Shows the current folder being used for recording.


-  **Record:** Arms the Recorder. Once armed, the actual recording will start when you click the **Pause** button (to un-pause the Recorder).


-  **New:** Creates a new, empty audio file on your hard disk in which to store the recording.
- **File:** Shows the name of the audio file currently being recorded (or about to be recorded). You can change the name of a new or existing audio file here.


-  **Play:** Starts playback of the currently loaded audio file (as displayed in the File field).

-  **Pause:** Pauses/un-pauses recording or playback.

-  **Stop:** Stops recording or playback and resets the recording/playback pointer to the beginning of the file.

-  **Loop:** Plays an audio file in an endless loop.

-  **Length:** Shows the length of the loaded audio file.

-  **Played:** Shows the current record/playback position.

Show/Hide Properties

Shows and hides the Properties dialog for the selected object (ensemble, instrument, macro, etc.). You can also open an object's Properties dialog by double-clicking on its title bar, or XP: right-clicking / OS X: **Ctrl**+clicking in the title bar and choosing **Object Properties** from the context menu, where **Object** is the name of the clicked-on object: **EchoBox Properties**, **"Fader" Properties**, etc. The Properties dialog always refers to the currently selected object and can stay open while you work. The keyboard shortcut is **F4**.

Show/Hide Sample Map Editor

Shows and hides the Sample Map Editor window. The keyboard shortcut is **F7**.

Show/Hide Browser

Shows and hides the Browser. The keyboard shortcut is **F5**.

Show/Hide Snapshots

Shows and hides the Snapshots window. The keyboard shortcut is **F6**.

Reset All Tool Window Positions

Resets all the tool windows (Playerbox, Recorderbox, Properties dialog, Sample Map Editor, and Browser) to their default sizes and locations. If you ever have trouble finding one of these windows, selecting **Reset All Tool Window Positions** should fix things right up.

Show/Hide Panel

Shows and hides the Ensemble Panel window.


Store Panelset

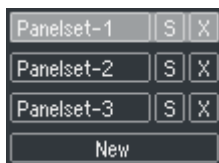
Enables you to store up to 8 different panelsets with an ensemble.

To store a panelset:

1. Create the panelset by arranging your Ensemble Panel window as desired (instrument positions, views, visibilities).

Select **View->Store Panelset->N**, where **N** is a number between 1 and 8. The keyboard shortcut is XP: **Ctrl + Alt + N** / OS X: **⌘ + Alt + N**.

You can also use the  Panelset bar, which is located in the panel Toolbar, to store panelsets. Using the Panelset bar, rather than **View->Store Panelset**, enables you to store as many panelsets with an ensemble as your computer memory will allow.




The panelset bar with three different panelset views stored

Recall Panelset

Enables you to recall the first 8 panelsets in an ensemble.

To recall a panelset:

1. Select **View->Recall Panelset->N**, where **N** is a number between 1 and 8. The keyboard shortcut is XP: **Ctrl + N** / OS X: **⌘ + N**.

You can also use the  Panelset bar to recall panelsets. Using the Panelset bar, rather than **View->Store Panelset**, enables you to recall as many panelsets as the ensemble contains (not just the first 8).

Close All Structures

Closes all open Structure windows. If you work with multiple Structure windows, this is a great way to un-clutter your view.

Cascade

Cascades all open REAKTOR windows. This function is available under Windows only. On OSX the functions are minimize and close.

Tile Horizontally

Tiles all open REAKTOR windows horizontally. This function is available under Windows only.

Tile Vertically

Tiles all open REAKTOR windows vertically. This function is available under Windows only.

Minimize

This function is available under Mac OS X only.

Close

This function is available under Mac OS X only.

Arrange Icons

The function arranges the minimized windows. This function is available under Windows only.

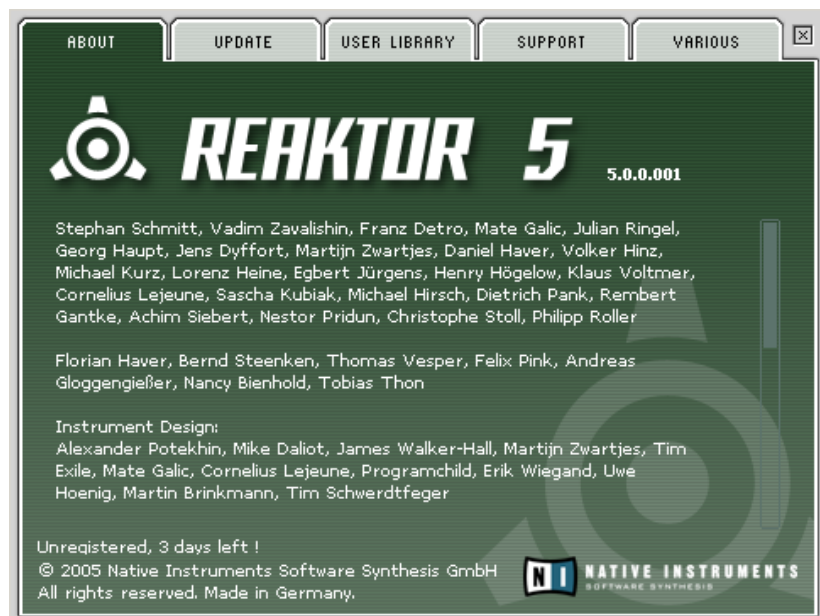
List of Open Windows

All currently open REAKTOR windows are listed at the bottom of the View menu. Clicking a window in the list selects it, brings it to the front of the REAKTOR workspace, and restores it (if it was minimized).

11.5. ? Menu

About

Choosing **About** from the XP: ? / OS X: **REAKTOR 5** menu opens the About REAKTOR 5 window. On the About page of this window is the REAKTOR version number and your personal REAKTOR serial number. On the other pages are web links for the REAKTOR user library and user forum, program updates, FAQs, and technical support.



12. REAKTOR Toolbars

There are three REAKTOR toolbars:

- **Main toolbar** - appears at the top of the REAKTOR application window. It contains tools for managing the REAKTOR program.



- **Ensemble Panel toolbar** - appears at the top of the Ensemble Panel window. It contains tools for working in ensembles.



- **Structure toolbar** - appears at the top of a Structure window. It contains tools for working in structures.







Note: The command bar at the top of an instrument panel is called a *header*, not a toolbar.







12.1. Main Toolbar




The Main toolbar

The Main toolbar has the following elements (left to right):

- Clicking on the  **NI icon** or the  **REAKTOR icon** opens the About window.
-  **Open** opens an existing ensemble. You can also press XP: **Ctrl + O** / OS X: **⌘ + O** to do this.
-  **Save** stores the current ensemble with all changes made to it. You can also press XP: **Ctrl + S** / OS X: **⌘ + S** to do this.

-  **Undo** undoes the last editing operation. You can also press XP: **Ctrl + Z** / OS X: **⌘ + Z** to do this.
-  **Run/Stop Audio** turns all REAKTOR audio processing on and off. When you are building or editing an ensemble and don't need sound, you can turn off audio processing to minimize your CPU load.
-  **CPU Load Display** displays the percentage of your CPU's processing capacity that is being used to run REAKTOR. In instances of CPU overload, the display will show **Over**. (The maximum allowed CPU usage is set in the CPU Usage page of the **Preferences** dialog.) Remember, a computer running REAKTOR has to do more than process audio! It also has to transfer audio to the sound card, process MIDI data, execute events, and draw REAKTOR's graphical displays, as well as run the computer operating system and any other programs that might be open. This is why REAKTOR can reach its limits at **CPU Load** levels well below 100%. The threshold for smooth ensemble operation is normally between 60% and 80%. To test your computer's limits, raise the number of voices in an ensemble until the CPU overload message appears.
-  The **Sample Rate** list box specifies the sample rate at which REAKTOR is running. You can select a different sample rate from the list; only those rates supported by your sound card are shown.
-  The **Audio In** level meter displays the level of the audio that is being input into REAKTOR; i.e. the audio signal (from the Playerbox or your sound card) that is made available to ensembles at the top two ports of the Audio In module.
-  The **Audio Out** level meter displays the level of the audio that is being output from REAKTOR; i.e. the audio signal generated by an ensemble that is made available to the outside world (sound card, speakers, headphones, host program for REAKTOR running as a plug-in, etc.) at the top two ports of the Audio Out module.








- The  **MIDI In** lamp lights whenever REAKTOR receives a MIDI event from an active MIDI Input port (as set in the MIDI page of the Audio Setup dialog).
- The **MIDI Out** lamp lights whenever REAKTOR sends a MIDI event to an active MIDI Output port (as set in the MIDI page of the Audio Setup dialog).

12.2. Ensemble Panel Toolbar






The

Ensemble Panel toolbar

- The  **Show/Hide Panelset Bar** button shows and hides the Panelset bar.
- The  **Snapshots** button opens the Snapshots window.
- The  **Browser** button opens the Browser.
- The  **Properties** button opens the Properties dialog.
- The  **Ensemble Structure** button opens the Ensemble Structure window.
- The  **Pause/Stop Clock** button stops the REAKTOR master clock. If a MIDI file has been imported (**File->Import MIDI File**); pushing **Pause/Stop Clock** once pauses the MIDI file playback; pushing **Pause/Stop Clock** a second time stops the playback and rewinds the MIDI file back to its beginning.
- The  **Start/Restart Clock** button starts the REAKTOR master clock. If a MIDI file has been imported, **Start** starts MIDI File playback from the








beginning or restarts playback at the place where the file was paused.

- Use the  **Tempo** selector to adjust the rate of the REAKTOR master clock in beats per minute (BPM). Double-click on the tempo display to enable direct entry of numerical values from your computer keyboard. Use the up/down arrows to adjust the tempo in 1 BPM increments.
- Use the  **MIDI Learn** button to quickly and easily assign a panel control (knob, fader, etc.) to an external MIDI controller (keyboard mod wheel, knob, fader, etc.). To do so: Select a panel control by clicking on it; click on the **MIDI Learn** button; operate the external MIDI controller (e.g. move the keyboard mod wheel). To cancel the controller assignment, open the control's Properties dialog and deselect its **Activate MIDI In** option.
- The  **Show/Hide Info** button shows and hides popup information. When **Show Info** is on (button lit), pointing your mouse to an object (instrument, primary or core macro/module, etc.) shows information on the object in a popup, and pointing to a wire shows the current values being sent along the wire. When **Hide Info** is on (button unlit), pointing to objects and wires shows no popups.

12.3. Structure Toolbar

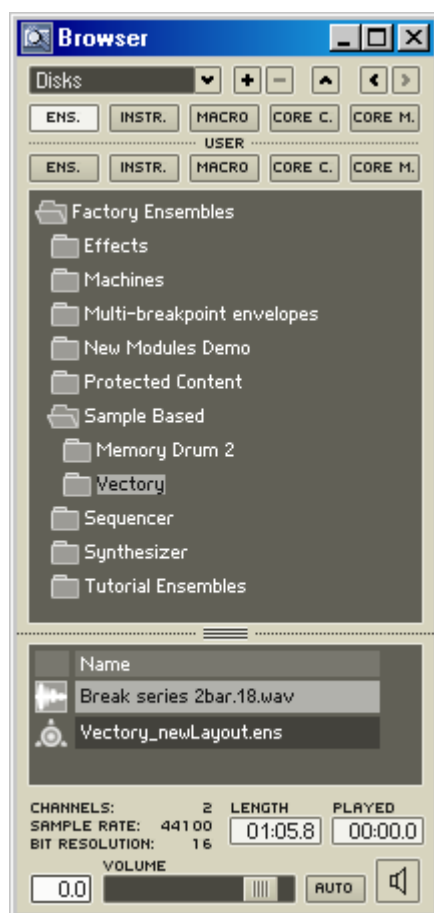


The Structure toolbar

- The  **Browser** button opens the Browser.
- The  **Properties** button opens the Properties dialog.
- The  **Ensemble Panel** button opens the Ensemble Panel window.
- The  **Bookmark** button bookmarks the current Structure window so that you can jump straight to it from any other Structure window in the ensemble.
- The  **Jump to Bookmark** button jumps from any Structure window in the ensemble to the bookmarked Structure window (if the **Jump** arrow is pointing toward the **Bookmark** button), or from the bookmarked Structure window to the last-displayed Structure window (if the **Jump** arrow is pointing away from the **Bookmark** button).
- The  **Debug** button is active within REAKTOR Core Cells only.
- The  **Show/Hide Info** button shows and hides popup information. When **Show Info** is on (button lit), pointing your mouse to an object (instrument, primary or core macro/module, etc.) shows information on the object in a popup, and pointing to a wire shows the current values being sent along the wire. When **Hide Info** is on (button unlit), pointing to objects and wires shows no popups.

13. The Browser

The Browser enables you to quickly and easily access the following types of files for use in REAKTOR:



The Browser

- Audio files (*.wav, *.aif, *.aiff), ensemble files (*.ens), instrument files (*.ism), primary macro files (*.mdl), core cell files (*.rcc), and core macro files (*.rcm) by using the **Disk Navigation** controls.



The **+** button adds a folder to the favorites list. The favorites list is appended to the **Disks** list.

- Factory (REAKTOR) ensembles, instruments, macros, core cells, and core macros by using the top row of **Ens.**, **Instr.**, **Macro**, **Core C.**, and **Core M.** buttons.



- Custom (user) ensembles, instruments, macros, core cells, and core macros by using the bottom (User) row of **Ens.**, **Instr.**, **Macro**, **Core C.**, and **Core M.** buttons.



13.1. Accessing Files

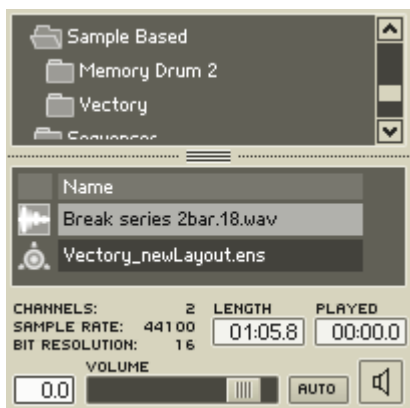
To open the Browser: select **View->Show Browser** from the main menu; or click on the **Show/Hide Browser** button in the Ensemble or Structure toolbar; or press **F5**.

The Browser is organized into two panes:



The upper pane of the browser containing the Disk Navigation






- In the upper pane, you locate files by using the self-explanatory **Disk Navigation** controls, or the **Ens.**, **Instr.**, **Macro**, **Core C.**, and **Core M.** buttons underneath. The top row of **Ens.**, **Instr.**, **Macro**, etc. buttons navigates to the system files that were installed along with the REAKTOR program. The bottom row of buttons navigates to your custom, user files. (The paths REAKTOR needs to find these files are in the **Preferences** dialog, Directories page. If necessary, you can change them there.)








The lower pane of the browser

- The lower pane enables you to access (audition, open, load, drag into structures) files.

You can use the Browser to access the following files:

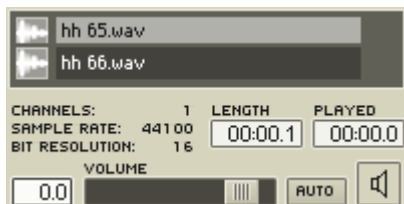
-  **Audio file:** You can load an audio file (*.wav, *.aif, *.aiff) by dragging it from the Browser (lower pane) to the Sample Map Editor, or to a sampler or tape deck display in an instrument panel. To locate an audio file, use the **Disk Navigation** controls at the top of the Browser.
-  **Ensemble file:** You can open a new ensemble file (*.ens) by dragging it from the Browser (lower pane) to the REAKTOR workspace. To locate an ensemble file, use either of the **Ens.** buttons.
-  **Instrument file:** You can insert an instrument file (*.ism) by dragging it from the Browser to the ensemble panel or ensemble Structure window. To locate an instrument file, use either of the **Instr.** buttons.
-  **Primary macro file:** You can load a primary macro file (*.mdl) by dragging it from the Browser to a primary Structure window. To locate a primary macro file, use either of the **Macro** buttons.
-  **Core cell file:** You can load a core cell file (*.rcc) by dragging it from the Browser to a primary Structure window. To locate a core cell file, use either of the **Core C.** buttons.

-  **Core macro file:** You can load a core macro file (*.rcm) by dragging it from the Browser to a core Structure window. To locate a core macro file, use either of the **Core M.** buttons.
-  **Sample map file:** You can load a sample map file (*.map) by dragging it from the Browser to the Sample Map Editor or to a sampler module panel display. To locate a sample map file, use the **Disk Navigation** controls.
-  **MIDI file:** You can load a standard MIDI file (*.mid) by dragging it from the Browser to the REAKTOR workspace. (This is equivalent to using **File->Import MIDI File** to load the file.) This loads the MIDI file into REAKTOR's MIDI File Player (for playback using the Pause/Stop and Start/Restart buttons in the Ensemble Panel toolbar). To locate a MIDI file, use the **Disk Navigation** controls.
-  **Table file:** You can load a table file (*.ntf) by dragging it from the Browser to a table display in an instrument panel. To locate a sample map file, use the **Disk Navigation** controls.
-  **Snapshot file:** You can load a snapshot file (*.ssf) by dragging it from the Browser to the Snapshots window. To locate a snapshot file, use the **Disk Navigation** controls.


Note: When you use the Browser (or a context menu) to insert an object (instrument, primary macro, core cell, or core macro) into a structure, the object's input and output ports are not automatically connected to anything (an instrument, macro, Audio In port, Audio Out port, etc.). You must wire all desired object connections manually.

13.2. Auditioning Files


The Browser supports audio-file auditioning (pre-listening):



The audition section of the browser

1. In the Browser (lower pane, see above section **Browser**), select an audio file (*.wav, *.aif, *.aiff) to audition. Playback controls appear at the bottom of the Browser, along with properties of the selected audio file (channels, sample rate, length, etc.).
2. Click on the  **Play** button (speaker icon) to start/stop auditioning the audio file. Use the **Volume** fader to set the volume.



3. If you turn  **Auto** on, auditioning starts automatically when you select an audio file.

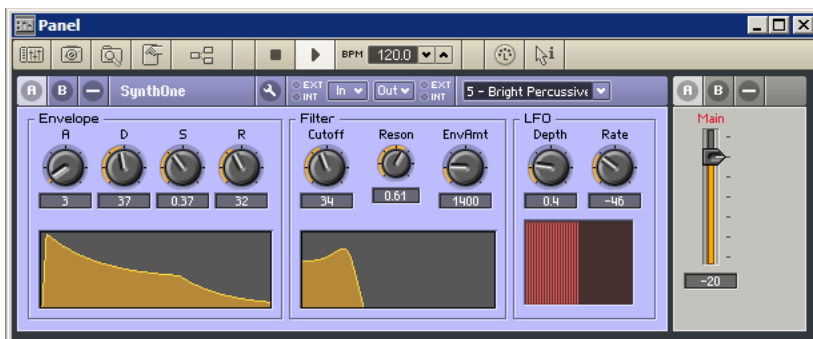
14. Ensemble

The ensemble is the highest object in the REAKTOR structural hierarchy. The entire contents of the current REAKTOR workspace (instruments, control settings, audio input/output connections, snapshots, etc.) are stored with the ensemble (in an *.ens file), and are restored when the ensemble is reloaded.

REAKTOR's structural hierarchy is as follows:

- An ensemble can contain instruments.
- An instrument can contain other instruments, as well as primary macros, primary modules, and core cells.
- A primary macro can contain other primary macros, as well as primary modules, and core cells.
- A core cell can contain core macros and core modules.
- A core macro can contain other core macros, as well as core modules.

In terms of panel display:



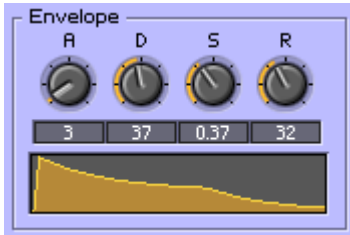
Ensemble Panel Window

- An ensemble has one panel window,



Instrument Panel

- Each instrument has a panel that you can choose to show or hide in the Ensemble Panel window.

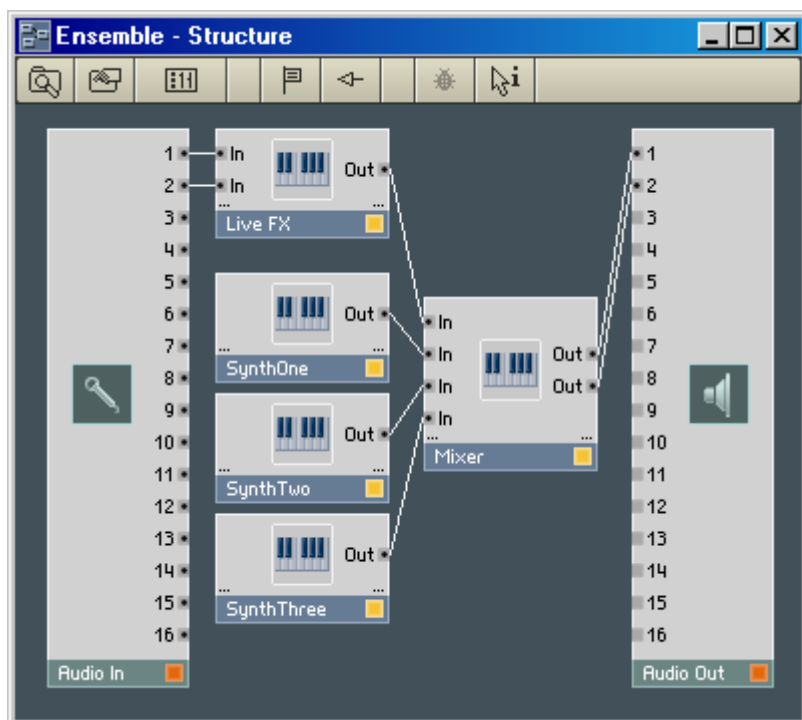


Frame of a macro in the Instrument panel

- Each primary macro can have a frame (that you choose to show or hide) in its instrument panel.

While the above is a simplified explanation, it should give you a clear idea of REAKTOR's basic architecture.

14.1. Ensemble Structure Window



Ensemble with five instruments: one live effect, three sound sources and a mixer. At left is the Audio In module.

The Ensemble Structure window gives a bird's eye view of the structure of the entire ensemble. It contains icons of all the instruments in the ensemble, and the Audio In and Audio Out modules, which provide access to your sound card or plug-in host.

Audio In Module

The **Audio In** module represents your REAKTOR audio inputs, as defined on the **SoundCard** and **Routing** pages of the **Audio Setup** dialog (**System->Audio + MIDI Settings...**). The **Audio In** module is a fixed part of the Ensemble Structure window and cannot be removed.

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

- **Mute** mutes (disables) the **Audio In** module. If you are not routing any audio input into an ensemble, it is good programming (though not required) to mute **Audio In**.
- **Properties** opens the **Audio Setup** dialog (just like **System->Audio + MIDI Settings...**).

The **Audio In** module provides a total of 16 ports for incoming audio (from the Playerbox, an external microphone, etc.). The actual number of available ports (i.e. those to which you can connect wires within an ensemble) is determined by the number of audio inputs your sound card supports. Available ports are marked with a black dot; unavailable ports are gray.

Audio Out Module

The **Audio Out** module represents your REAKTOR audio outputs, as defined on the **SoundCard** and **Routing** pages of the **Audio Setup** dialog (**System->Audio + MIDI Settings...**). The **Audio Out** module is a fixed part of the Ensemble Structure window and cannot be removed.

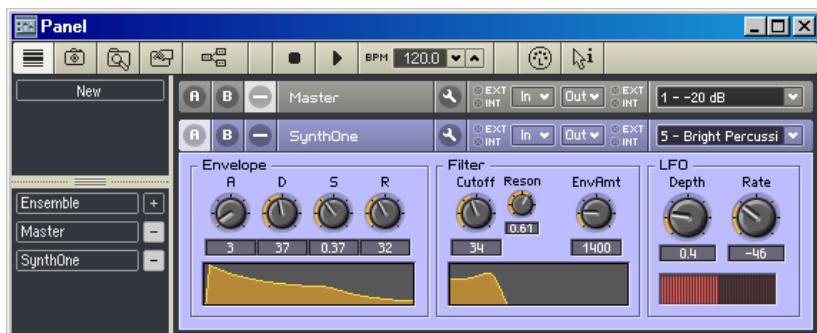
The context menu of the **Audio Out** module contains two entries:

- **Mute** mutes (disables) the **Audio Out** module. If your ensemble does not generate sound (e.g. if it only generates a display), it is good programming (though not required) to mute **Audio Out**.
- **Properties** opens the **Audio Setup** dialog (just like **System->Audio + MIDI Settings...**).

The **Audio Out** module provides a total of 16 ports for outgoing audio to your sound card of plug-in host. The actual number of available ports (i.e. those to which you can connect wires within an ensemble) is determined by the number of audio outputs your sound card supports. Available ports are marked with a black dot; unavailable ports are gray.

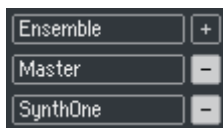
14.2. Ensemble Panel Window


The Ensemble Panel window can display all (or one, or none, etc.) of the instrument panels in the ensemble.



Ensemble Panel Window with its Panelsetbar on the left side

To show/hide instrument panels, you use the **Panelset bar**.



To show/hide the **Panelset bar**, click the  **Show/Hide Panelset bar** button in the Ensemble Panel toolbar.



Each instrument panel has three views: A, B, and minimized. You select these views by clicking the **A**, **B**, and **Minimize** (—) buttons in the instrument header.



The A and B views are set in the **Properties** dialog **Appearance** pages of the instrument controls (knobs, faders, XYs, macro frames, etc.). The minimized view displays the instrument header alone.

14.3. Ensemble Properties Dialog

There are many ways to open an ensemble's Properties dialog. Use whichever one suits you best:

- Windows XP: Right-click / OS X: Ctrl+click on a blank part of the Ensemble Panel window and select **Ensemble Properties** from the context menu.
- Click on the  **Show/Hide Properties** button in the Ensemble Structure window toolbar.
- Click on a blank part of the Ensemble Panel window and click on the  **Show/Hide Properties** button in its toolbar.
- Click on a blank part of the Ensemble Panel window and select **View->Show Properties** (or press **F4**).

Like all Properties dialogs, the Ensemble Properties dialog has four pages, each accessible from a picture button on the top of the dialog:



Function,



Info,



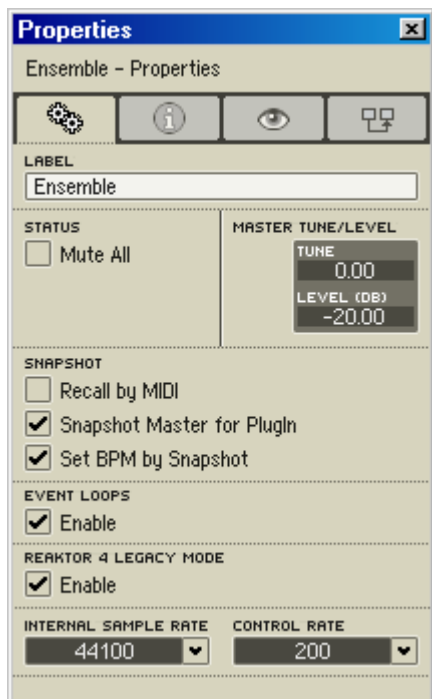
Appearance, and



Connection.

Tip: The contents of the Properties dialog automatically changes to display the values for the currently selected object (ensemble, instrument, primary macro, etc.). So, to compare object values, leave the Properties dialog open and toggle your selection between the two objects.

Function Page



Ensemble Properties dialog, Function page

Label

The **Label** field contains the label “Ensemble” and cannot be changed. Note that the **Label** field of all other REAKTOR objects (except for core modules) can be changed, enabling you to customize names of instruments, primary macros, core cells, etc.

Status

Mute All mutes (disables) all instruments in the ensemble. This reduces REAKTOR’s CPU usage to a minimal level (just enough to keep REAKTOR running), marks all instrument icons (in the Ensemble Structure window) with a red M (for Mute), and draws a red X over all primary output/input ports.

Master Tune/Level

Tune adjusts the ensemble's global pitch. The value is specified in fractional semitone units (12 semitones = an octave). Positive values raise the pitch; negative values lower it.

Level changes the ensemble's global volume level. The value is specified in dB (6 dB doubles/halves the volume). Positive values raise the level, negative values lower it.

Snapshot

When **Recall by MIDI** is enabled, an incoming MIDI Program Change message with the value N (where N is an integer from 0-127) will recall the snapshot with the value N+1 (if that snapshot exists). Thus a Program Change message of 0 will recall snapshot 1, a message of 1 will recall snapshot 2, and so on. This way you can quickly and easily recall snapshots from your MIDI controller (keyboard) by issuing MIDI Program Change messages of the desired snapshot number.

Snapshot Master for Plug-In

When Snapshot Master for Plug-In is enabled the ensemble snapshots are available in host programs. There can only be one snapshot master. This setting can alternatively be activated for single instruments.

Set BPM by Snapshot enables REAKTOR master clock BPM settings to be saved/recalled with snapshots.

Event Loops

When the **Event Loops** option is enabled, REAKTOR allows event-signal loops to occur within the ensemble. These loops can lead to stack overflow crashes, which in turn can make the ensemble un-playable and, in some cases, un-openable.

When **Event Loops** is disabled, event-signal loops are prevented from occurring. If a loop is about to occur, REAKTOR displays a message revealing the source of the loop and asking you how to proceed.

We recommend that you disable **Event Loops** to maximize the stability of REAKTOR. To ensure backward compatibility, ensemble files saved in older versions of REAKTOR have **Event Loops** enabled by default.

REAKTOR 4 Legacy Mode

REAKTOR 5 has a new initialization scheme for event inputs that is used if the **REAKTOR 4 Legacy Mode** option is disabled. We strongly recommend that you disable **REAKTOR 4 Legacy Mode** in your ensembles for the sake of future compatibility!

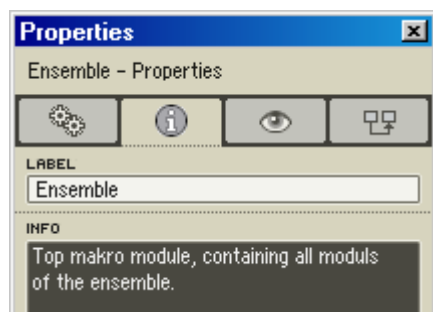
Internal Sample Rate

Internal Sample Rate sets the sample rate at which REAKTOR generates and processes audio signals. With higher sample rates you can achieve better sound quality, but the CPU load rises proportionally. You can change the internal sample rate to any of the values in the menu. The range of available values depends on your sound card or host plug-in. If the internal sample rate is different from the sound card's or host plug-in's sample rate, the **Audio In** and **Audio Out** modules will do the necessary sample-rate conversion.

Control Rate


Control Rate sets the control rate for REAKTOR event signals; i.e. the number of times per second that event-signal values are updated. The control rate is applied globally to all primary modules that generate or process events; e.g. **LFO**, **Slow Random**, **Event Hold**, **A-to-E**, **Event Smoother**, and more. Since the control rate is very low compared to the sample rate, these modules need very little CPU power. For this reason, good builders choose to work with event signals rather than audio signals whenever possible (i.e. whenever it doesn't degrade the sound).

Info Page

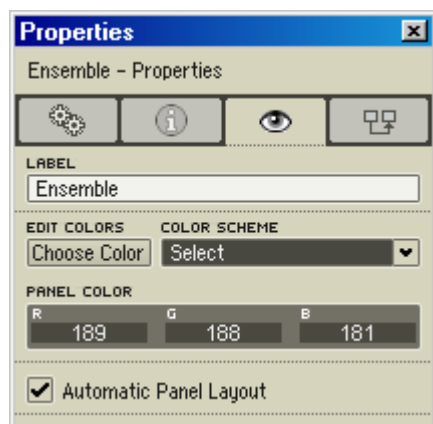


Ensemble Properties dialog, Info page

Enter desired information about your ensemble into the **Info** field of this page.

If  **Show Info** in the Ensemble Panel or Structure toolbar is enabled, your **Info** field text will be displayed in a popup whenever the mouse points to the ensemble panel header.

Appearance Page



Ensemble Properties dialog, Appearance page

Edit Colors

- **Choose Color:** Click this button to use the Color dialog palette to choose a color for the ensemble panel header. Note that an ensemble panel consists only of a header; there is no ensemble panel “body.”
- **Color Scheme: Set to Custom** sets the ensemble’s color scheme to the current custom color scheme. **Save as Custom** saves the ensemble’s color scheme as the new custom scheme. Note that REAKTOR only supports one custom color scheme; if you save a new scheme, you’ll overwrite the previous scheme. **Set to Default** sets the ensemble’s color scheme to the default scheme (grey panel with orange indicators).
- **Panel Color:** You can use the Color dialog palette to choose an ensemble panel header color (see **Choose Color**, above). Or you can mix your own color by using the **R**, **G**, and **B** fields to enter values for the color’s red, green, and blue components. Each field accepts values from 0 (none) to 256 (full). Entering 0 for all three fields creates black; entering 256 creates white. And so on.

Automatic Panel Layout

The **Automatic Panel Layout** option, when enabled (its default setting), causes all panels displayed in the Ensemble Panel window to be arranged tidily within the window. When **Automatic Panel Layout** is disabled, the panels can be placed anywhere within the window.

Connection Page



Ensemble Properties dialog, Connection page

MIDI

- The **MIDI In Device** drop-down menu specifies which of the available MIDI In devices the ensemble can receive messages from. (You make MIDI In devices available to ensembles in the Audio Setup dialog, MIDI page.) Typically, **MIDI In Device** is set to **All**, enabling the ensemble to receive messages from all available MIDI In devices. In some cases, however, you might want to prevent an ensemble from receiving messages from a certain MIDI In device.
- **Channel** specifies the MIDI Channel number used by the ensemble for MIDI input. The ensemble receives only those MIDI messages that are sent on the specified MIDI Channel number.
- **Morph** activates the ensemble snapshot morph.
- **Controller** specifies the controller number used for the snapshot morph.

OSC

- The **OSC Source** drop-down menu lets you choose the OSC source(s) from which the ensemble receives OSC data.
- The **OSC Target** drop-down menu lets you choose the OSC target(s) to which the ensemble sends OSC data.
- **OSC Connections** lists the active OSC connections.

External Sync

When **External Sync** is enabled: an external clock signal (received via MIDI) controls all **Sync Clock** and **1/96 Clock** modules in the ensemble; MIDI Start/Stop messages control all **Start/Stop** modules; and the clock tempo cannot be adjusted in the BPM field on the Main toolbar.

Note: You can also enable/disable **External Sync** from the REAKTOR Settings menu.

When **External Sync** is disabled: the internal REAKTOR master clock controls the **Sync Clock** and **1/96 Clock** modules; the REAKTOR **Pause/Stop Clock** and **Start/Restart Clock** buttons control the **Start/Stop** modules; and the clock tempo can be adjusted in the BPM field.

15. Instruments

A REAKTOR instrument is an object that has its own internal structure, MIDI processing, control panel, and snapshots. In the Ensemble Structure window, instrument objects can be recognized by their blue label and keyboard icon.



Instrument object

An instrument can contain other instruments, as well as primary macros, primary modules, and core cells. It may also be called a different name.

15.1. Adding Instruments to an Ensemble

You add instruments to an ensemble by loading them from the REAKTOR system library or from your user content storage area (set in Preferences dialog, Directories page).

To add an instrument to an ensemble you can use any of these methods:

- XP: Right-click / OS X: Ctrl+click on a blank part of the structure to which you want to add the instrument. Normally, this is the ensemble structure; but you can also add instruments to instrument structures.
- Use **Insert Instrument** from the context menu to find and insert the desired instrument.
- Open the Browser (**View->Show Browser** or **F5**). Use the upper **Instr.** button to find a system instrument, or the lower **Instr.** button to find a user instrument (from your user content storage area). In the lower pane, drag the desired instrument to the structure.
- Use the Browser's **Disk Navigation** controls (top row) to navigate to the desired instrument folder, then drag the instrument to the structure.



The system library provides a generous selection of premade sound-generation and effects instruments. If you want to start developing a new instrument, you first need to load an empty one (i.e. one that begins with **_New**) from the system library.


When inserting an instrument in a structure, you are in effect creating a copy of the instrument file that is stored on your disk. The instrument copy and the

instrument file are completely independent. Changes you make to the instrument copy will not affect the instrument file, and vice-versa. If you want to change the instrument file, you must change the copy (in an ensemble), and then use **Save Instrument As...** (from the instrument's context menu) to save the changed copy over the existing instrument file.

Note: Instruments can also be inserted within another instrument.

15.2. Ports

There is no fixed arrangement of input and output ports for instruments. The type and number of ports in an instrument is determined by the user by the insertion of **Terminals** ( , ) in the instrument structure.

The connections into and out of an instrument through the terminals must always be **monophonic** (rather than polyphonic). For this reason, if the instrument is polyphonic, an **Audio Voice Combiner**  module needs to be inserted before the output port(s) to convert the polyphonic signal to a monophonic signal before it is output.

15.3. Context Menu

The context menu of an instrument object in the Ensemble Structure window contains the following entries:

- **Mute**, when on, disables the selected instrument.
- **Solo**, when enabled, connects the output of the selected instrument directly to the Audio Out module (i.e. to the sound card or plug-in host). All instruments that lie upstream from the selected instrument (i.e. that feed into the selected instrument) remain active. All instruments that lie downstream from the selected instrument (i.e. into which the selected instrument feeds) are muted. For example, say a Synth instrument signal is fed to a Chorus instrument, the Synth + Chorus signal is then fed to a Compressor instrument, and the Synth + Chorus + Compressor signal is finally connected to the Audio Out module. If the Chorus instrument is in **Solo** mode, the Synth + Chorus signal is connected to the Audio Out module. The Compressor is not included in the circuit, because it lies downstream from the **Solo** Chorus instrument (i.e. Chorus feeds into Compressor).


- **Cut** removes the selected instrument from the structure and stores it temporarily in the clipboard. From there, the instrument can be pasted (using the **Paste** command) to a different structure (or another location in the same structure) .
- **Copy** does the same thing as **Cut**, but it does not remove the instrument from the structure.
- **Duplicate** creates a copy of the selected instrument in the same structure. Choosing **Duplicate** is equivalent to choosing **Copy**, then **Paste**.
- **Delete** deletes the selected instrument from the structure.
- **Save Instrument As...** enables the selected instrument to be saved to an *.ism file on your disk. Builders typically use **Save Instrument As...** to save new or modified instruments to their user content Instruments folder.
- **Structure** opens the selected instrument's structure in the main structure window. Choosing **Structure** is equivalent to double-clicking on the instrument's structure icon.
- **Structure Window** opens the selected instrument's structure in a separate (i.e. not the main) structure window. Doing so enables multiple structure windows to be open at the same time. Choosing **Structure Window** is equivalent to Alt+double-clicking on the instrument's structure icon.
- **Properties** opens the selected instrument's Properties dialog. For details, see Instrument Properties below.









15.4. Instrument Header

The Instrument header contains the following elements:



The Instrument header

- Instruments have two panels views: A and B. The  **A** and **B** buttons in the instrument header enable you to choose which of these panel views to display. You can specify if an object (knob, fader, meter, etc.) is to appear in panel A, B, both, or neither by using the **A**, **B**, **AB**, and **Visible** options in its **Properties** dialog (Appearance page).

- Clicking on the  **Minimize** (—) button hides the instrument panel and leaves only the header visible. Clicking on **Minimize** (—) again redisplay the panel.
- The instrument name displays the text in the  **Label** field of the instrument's Properties dialog.
- The  **Lock/Unlock Panel** button locks and unlocks the instrument panel. Locking a panel freezes all its elements (controls, displays, etc.) in their current locations. Unlocking a panel enables you to move elements to new locations. Note that, when locked, panel controls can be adjusted (e.g. knobs can be turned), but not moved (to different panel locations); when unlocked, panel controls can be moved, but not adjusted.
- The four MIDI activity lamps –  **External** and **Internal** MIDI In, and **External** and **Internal** MIDI Out – light when external/internal MIDI events arrive at the active MIDI In ports or are sent to the active MIDI Out ports. (You configure your REAKTOR MIDI ports in the **Audio Setup** dialog, MIDI page.)
- The  **In** and **Out** drop-down menus enable you to set all the instrument's input and output connections (MIDI and wiring).
- The  **Snapshot** drop-down menu enables you to recall snapshots for the instrument.
-  **Voices** displays (and allows you to change) the number of polyphonic voices allocated to the instrument. This value can also be changed in the **Voices** field of the instrument's **Properties** dialog (Function page).
-  **Unison** displays (and allows you to change) the maximum number of unison voices per note that the instrument will play. The richness and chorus-like quality of the “unison” effect (made famous by hardware synthesizers) is enabled by setting **Unison** to 2 or higher. The unison voices are detuned (with respect to one another) by the value specified by **Unison Sprd** in the instrument's **Properties** dialog (Function page). The minimum number of unison voices per note can be set there as well (**Min Unison V.**).

Note: REAKTOR enables you to save the current **Unison** value with a snapshot, i.e. to specify a different number of maximum unison voices for each snapshot. But the current **Voices** value applies to the entire instrument (all snapshots).

15.5. Instrument Properties

There are many ways to open an instrument's Properties dialog. Use whichever one suits you best:

- Double-click on the instrument's title bar (not its keyboard icon!) in any structure window. Or double-click on the instrument's name in its panel header.
- XP: Right-click / OS X: Ctrl+click on the instrument in any structure window, and select **Properties** from the context menu. Or XP: Right-click / OS X: Ctrl+click on the instrument's name in its panel header, and select **Properties** from the context menu
- Select the instrument in the Ensemble Panel or any structure window, and select **View->Show Properties** from the main menu (or press **F4**).

Like all Properties dialogs, an instrument's Properties dialog has four pages, each accessible from a picture button on the top of the dialog:



Function,



Info,



Appearance, and



Connection.

Tip: The contents of the Properties dialog automatically changes to display the values for the currently selected object (ensemble, instrument, primary macro, etc.). So, to compare object values, leave the Properties dialog open and toggle your selection between the two objects.

Function Page

Properties [X]

Instrument - Properties

Icon: [Settings] [Info] [Eye] [Grid]

LABEL
Instrument

STATUS
☐ Solo
☐ Mute

TUNING
TUNE: 0.00
UNISON SPRD: 0.05

VOICE ALLOCATION
VOICE & MIDI SLAVE TO: None
☐ Lock Voices

VOICES	MAX UNISON V	MIN UNISON V
1	1	1

☐ Automatic Voice Reduction

VOICE ASSIGN
☐ Oldest ☐ Reassign
☐ Newest
☐ Nearest

SNAPSHOT
☐ Store by Parent ☐ Only if changed
☐ Recall by Parent ☒ Recall by MIDI
☐ Snapshot Master for PlugIn

EVENT LOOPS
☐ Enable

Properties dialog of an instrument, Function page

Label

The **Label** field specifies the name of the instrument; i.e. the name that appears in the instrument panel header. You can change this field to (re)name your instrument.

Status

- **Solo**, when enabled, connects the output of the instrument directly to the Audio Out module (i.e. to the sound card or plug-in host). All objects that lie upstream from the instrument (i.e. that feed into the instrument) remain active. All objects that lie downstream from the instrument (i.e. into which the instrument feeds) are muted. For an example, see above, **Context Menu**.
- **Mute** mutes (disables) the instrument and all objects that lie upstream from it (i.e. that feed into it). In structure view, an instrument whose Mute option is turned on has a red M over its status LED and red crosses over its input/output ports.

Tuning

- **Tune** adjusts the instrument's pitch with respect to the master ensemble tuning (as set in the **Ensemble Properties** dialog, Function page). The value is specified in fractional semitone units (12 semitones = an octave). Positive values raise the pitch; negative values lower it. A typical application is to detune two instruments to get a fatter sound; a good value for this is **Tune** = 0.05 (equivalent to 5 cents or 1/20th of a semitone).
- **Unison Spread** determines the degree of detuning between each of the instrument's unison voices. (Note that the instrument must have 2+ unison voices for **Unison Spread** to have an effect.) As with **Tune**, the **Unison Spread** value is specified in fractional semitone units. A typical value is 0.05, which detunes each of the unison voices by 5 cents (1/20th of a semitone) with respect to one another, making for a fatter sound.

Voice Allocation

Each instrument has its own polyphonic voice allocation settings.

- **Voice & MIDI Slave To**, when turned on, enables the instrument's voice allocation and MIDI In settings to be controlled from another instrument in the ensemble.
- **Lock Voices**, when enabled, locks the instrument's voice allocation settings (**Voices**, **Max Unison V**, and **Min Unison V**). If you need to change any of these settings, simply disable **Lock Voices**.
- **Voices** specifies the total number of polyphonic voices that the instrument can play. This number applies to all the polyphonic modules in

the instrument (i.e. all modules whose Properties dialog Mono option is disabled).

- **Max Unison V.** specifies the maximum number of unison voices that an instrument can play per note. If, for example, you set **Max Unison V.** to 3, the instrument will play a maximum of 3 unison voices per note. (The minimum number is set by **Min Unison V.**, see below.) The amount of detuning between voices is set with **Unison Spread** (see above).
- **Min Unison V.** specifies the minimum number of unison voices that an instrument can play per note. If, for example, you set **Min Unison V.** to 2, the instrument will play a minimum of 2 unison voices per note.

The **Voices**, **Max Unison V.**, and **Min Unison V.** values are interdependent. **Voices** specifies the total number of polyphonic voices that the instrument can play. If the instrument does not make use of unison playing, **Max Unison V.** and **Min Unison V.** are both set to 1. If the instrument does make use of unison, **Max Unison V.** and **Min Unison V.** specify the maximum and minimum number of unison voices that the instrument can play per note.

For example, say the Voices, Max Unison V., and Min Unison V. settings were 24/1/1; the instrument could play up to 24 notes at the same time, with no unison effect. **Say the settings were 24/3/3;** the instrument could play up to 8 notes at the same time with 3 unison voices per note ($8 * 3 = 24$). **Now say the settings were 24/3/2;** the instrument could play up to 8 notes at the same time with 3 unison voices per note ($8 * 3 = 24$), or up to 12 notes at the same time with 2 unison voices per note ($12 * 2 = 24$). And so on.

If **Max Unison V.** and **Min Unison V.** are different, (e.g. 3 and 2, as in our example), REAKTOR automatically switches between the **Max** and **Min** values depending on how many notes are being played at the same time. With a 24/3/2 configuration, if the number of notes being played at the same time is ≤ 8 , each note will have 3 unison voices; if the number of notes being played at the same time is > 8 , each note will have 2 unison voices.

- When **Automatic Voice Reduction** is enabled, REAKTOR will automatically reduce the number of instrument voices (specified by **Voices**) when the CPU load exceeds the limit set in the Preferences dialog (CPU Usage page). This way, polyphony can be adjusted according to the available processing power.

Voice Assign

When the number of voices in an instrument is not sufficient to process all the notes being played at the same time, REAKTOR needs to choose intelligently which voice (or voices, if the instrument is in **Unison** mode) to “steal” from an existing note and reassign to a new note. There are three options for such voice assignment: Oldest, Newest, and Nearest.

- When **Oldest** is enabled, the voice which has been held longest is stopped and assigned to the new note. This is the most common voice assignment strategy.
- When **Newest** is enabled, the most recently played voice is stopped and assigned to the new note. This can be useful for playing a melody over held notes, because none of the held voices will get stopped and re-assigned.
- When **Nearest** is enabled, the voice closest in pitch to the new note is stopped and assigned to the new note. This is good when using polyphonic portamento (glide).
- **Reassign** determines what happens when the same note is played again. Either the voice already playing the note is reused or another voice is used. **Reassign** mode is good for making efficient use of a limited number of voices, and it is also what you are used to from playing the piano.

Snapshot

You find more about the principle Snapshots in REAKTOR in the **Snapshot** section.

- When **Recall by Parent** is enabled and **Store by Parent** is disabled, you can recall an instrument's snapshots by recalling a snapshot in its parent object (usually the ensemble, but sometimes another instrument). For example, let's say an instrument's **Recall by Parent** is enabled and **Store by Parent** is disabled. If you store the snapshot **iSnap1** in the instrument, then store **eSnap1** in the ensemble, recalling **eSnap1** in the ensemble automatically recalls **iSnap1** in the instrument.
- When **Recall by Parent** and **Store by Parent** are both enabled, you can store and recall an instrument's snapshots by storing and recalling a snapshot in its parent object. For example, let's say an instrument's **Recall by Parent** and **Store by Parent** are enabled. If you create the settings for a new snapshot in the instrument, then store the snapshot **Snap1** in the ensemble, a snapshot with the same name (**Snap1**) is stored in

the instrument. Recalling **Snap1** in the ensemble automatically recalls **Snap1** in the instrument.

- When **Recall by Parent** is disabled and **Store by Parent** is enabled, you can store an instrument's snapshots by storing a snapshot in its parent object. For example, let's say an instrument's **Recall by Parent** is disabled and **Store by Parent** is enabled. If you create the settings for a new snapshot in the instrument, then store the snapshot **Snap1** in the ensemble, a snapshot with the same name (**Snap1**) is stored in the instrument. However, since **Recall by Parent** is disabled, recalling **Snap1** in the ensemble does not recall **Snap1** in the instrument.
- When **Only if changed** and **Store by Parent** are both enabled, if you store a new snapshot in the parent (ensemble), a snapshot with the same name will only be stored in the child (instrument) if the settings for the new snapshot in the child are different from the settings for the current snapshot. This saves space in the instrument's snapshot list.
- When **Recall by MIDI** is enabled, an incoming MIDI Program Change message with the value N (where N is an integer from 0-127) will recall the snapshot with the value N+1 (if that snapshot exists). Thus a Program Change message of 0 will recall snapshot 1, a message of 1 will recall snapshot 2, and so on. This way you can quickly and easily recall snapshots from your MIDI controller (keyboard) by issuing MIDI Program Change messages of the desired snapshot number.

When Snapshot Master for PlugIn is enabled, When Snapshot Master for Plug-In is enabled the instrument snapshots are available in the host program. There can only be one snapshot master. This setting can alternatively be activated for the entire ensemble (see ensemble properties.)

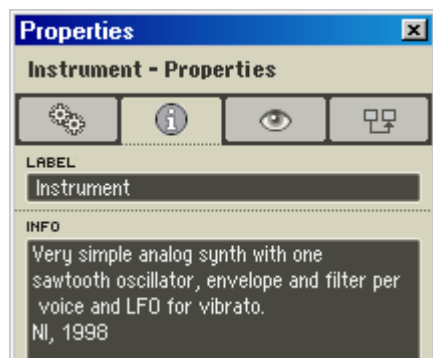
Event Loops

When the **Event Loops** option is enabled, REAKTOR allows event-signal loops to occur within the instrument. These loops can lead to stack overflow crashes, which in turn can make the ensemble un-playable and, in some cases, un-openable.

When **Event Loops** is disabled, event-signal loops are prevented from occurring. If a loop is about to occur, REAKTOR displays a message revealing the source of the loop and asking you how to proceed.


We recommend that you disable **Event Loops** to maximize the stability of the instrument.

Info Page



Properties dialog of an instrument, Info page

Enter desired information about your instrument into the **Info** field of this page.

If  **Show Info** in the Ensemble Panel or Structure toolbar is enabled, your **Info** field text will be displayed in a popup whenever the mouse points to the instrument panel header.

Appearance Page

Properties

Instrument - Properties

LABEL

Instrument

EDIT COLORS **COLOR SCHEME**

Choose Color Select

ITEM	R	G	B
Panel	189	188	181
Indicator	255	160	0
Graph Line	255	213	53
Graph Fill	175	127	46
Graph BG	76	76	76

STRUCTURE ICON **PICTURE INDEX**

<none> ?

☒ Available in Panelsets

ALL CONTROLS

Visible Invisible

BACKGROUND PICTURE **PICTURE INDEX**

<none> ?

PICTURE BORDERS

BORDER TOP	0
BORDER BOTTOM	0
BORDER LEFT	0
BORDER RIGHT	0

Properties dialog of an instrument, Appearance page

Edit Color

- **Choose Color:** Click this button to use the Color dialog palette to choose a color for the selected entry in the **Item** list below.
- **Color Scheme:** **Set to Custom** sets the instrument's color scheme to the current custom color scheme. **Save as Custom** saves the instrument's color scheme as the custom scheme. Note that REAKTOR only supports

one custom color scheme; if you save a new scheme, you'll overwrite the previous scheme. **Set to Default** sets the instrument's color scheme to the default scheme (grey panel with orange indicators).

- **Item list:** This area lists all the instrument panel items that can have customized colors. You can use the Color dialog palette to choose colors for these items (see **Choose Color**, above). Or you can mix your own colors by using the **R**, **G**, and **B** fields to enter values for the color's red, green, and blue components. Each field accepts values from 0 (none) to 256 (full). Entering 0 for all three fields creates black; entering 256 creates white. And so on. Here are the color-customizable items:
 - **Panel:** Color of the panel background, if the panel has no background picture.
 - **Indicator:** Color of control (knob, fader, button, etc.) indicators.
 - **Graph Line:** Color of graph lines in tables, XY cursors, and fill outlines in filter and envelope displays.
 - **Graph Fill:** Color of graph fills in tables, XY objects, and fills in filter and envelope displays.
 - **Graph BG:** Color of the background of tables, XYs, and envelope and filter displays.
 - **Grid:** Color of the grid in table displays.
 - **2D Table Min:** Color of the minimum value in the 2D table displays.
 - **2D Table Max:** Color of the maximum value in the 2D table displays.
 - **2D Table Default:** Color of the default value in the 2D table displays.

Structure Icon

- **Structure Icon:** Lets you replace the default instrument structure icon (keyboard) with your own picture.
- **Picture Index:** If you choose a structure icon picture that contains multiple smaller pictures, you can select the index for the desired picture (after having set **Num Animations** in the Picture Properties dialog).

Available in Panelsets

When **Available in Panelsets** is enabled, the instrument is included in the list at the bottom of the **Panelset** bar, and its panel can be shown (or hidden) in the Ensemble Panel window. When **Available in Panelsets** is disabled, the instrument is not listed in the Panelset bar, and its panel cannot be shown in the Ensemble Panel window.

Panel Controls

- **A, B, AB:** These determine if changes you make to the appearance of the instrument are applied to panel A (**A**), panel B (**B**), or both panels A and B (**AB**). This affects two things: the **All Controls Visible** and **Invisible** commands (see below) and the **Picture Borders** settings (see below). If **A** is enabled, the **All Controls Visible** and **Invisible** commands and **Picture Borders** settings will only be applied to panel A of the instrument. If **B** is enabled, the commands and settings will only be applied to panel B. If **AB** is enabled, the commands and settings will be applied to both panels, A and B.
- **Copy A > B, Copy B > A:** Click on one of these buttons to copy the full content and appearance of one panel to the other.

All Controls

- **Visible** displays all of the instrument's controls in the panel(s) specified by **A**, **B**, and **AB** (see above).
- **Invisible** hides all of the instrument's controls in the panel(s) specified by **A**, **B**, and **AB**.

Background Picture

- **Background Picture:** You can load your own picture for the instrument panel background. All panel controls and displays will lie on top of the background picture. You can assign a different background picture to each instrument panel (A and B).
- **Picture Index:** If you choose a background picture that contains multiple smaller pictures, you can select the index for the desired picture (after having set **Num Animations** in the Picture Properties dialog).

Picture Borders

The **Border Top**, **Border Bottom**, **Border Left**, and **Border Right** values determine the number of pixels used for the top, bottom, left, and right borders of the instrument panel(s) specified by **A**, **B**, and **AB** (see above). Due to REAKTOR panel gridding, the Picture Borders values should be set to multiples of 4: 0, 4, 8, 16, etc.

Connection Page

Properties [X]

Instrument - Properties

[Settings] [Info] [Eye] [Connect]

LABEL
Instrument

MIDI IN

DEVICE All [v] **CHANNEL** 1

UPPER NOTE G8	<input type="checkbox"/>	SUSTAIN CTRL 64
LOWER NOTE C-2	<input type="checkbox"/>	HOLD CTRL 66
NOTE SHIFT 0	<input type="checkbox"/>	MORPH CTRL 0

ALL [v] NONE [v]

MIDI OUT

DEVICE All [v] **CHANNEL** 1

OSC

OSC SOURCE no OSC Source [v] **TARGET** no Osc Target [v]

CONNECTIONS

[Empty List Box] [Trash]

[Empty Input Field]

AUTOMATION

☒ Hide Name [IDS [v]]

BASE ID 0 **MAX ID** 10 **MAX ID IN USE** 10

Properties dialog of an instrument, Connection page

MIDI In

- The **Device** drop-down menu specifies which of the available MIDI In devices the instrument can receive messages from. (You make MIDI In devices available to ensembles in the Audio Setup dialog, MIDI page.) Typically, **Device** is set to **All**, enabling the instrument to receive messages from all available MIDI In devices. In some cases, however, you might want to prevent an instrument from receiving messages from a certain MIDI In device.
- **Channel** specifies the MIDI Channel number used by the instrument for MIDI input. The instrument receives only those MIDI messages that are sent on the specified MIDI Channel number.
- **Upper Note** and **Lower Note** specify the range of MIDI In note numbers that the instrument will recognize. Any note numbers outside the range are ignored. This can be used to program a keyboard split.
- **Note Shift** enables all the MIDI In note pitches to be transposed up or down by the specified 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.
- **Sustain Ctrl** specifies the number of the MIDI controller that functions as a sustain pedal (called “hold” or “damper pedal” by some MIDI equipment manufacturers, standard controller number 64). As long as the sustain pedal is turned on, any playing note will be held, even after its key is released. To enable sustain for the instrument, turn on its **Sustain On/Off** option (the box left of **Sustain Ctrl**).
- **Hold Ctrl** specifies the number of the MIDI controller that functions as a hold pedal (called “sostenuto” by some MIDI equipment manufacturers, standard controller number 66). All notes that are playing when hold is turned on will be held even after their key is released until hold is turned off. Notes that are played when hold is already on are not affected. To enable hold for the instrument, turn on its **Hold On/Off** option (the box left of **Hold Ctrl**).
- **Mrph Ctrl** defines the controller number used for the snapshot morph. The snapshot morph is activated by the button left to the controller field.

All and None Menus

- Choose an option from the **All** drop-down menu to transfer it to all the instrument's controls.
- Choose an option from the **None** drop-down menu to remove it from all the instrument's controls.

To learn more about the **All** and **None** menu options, please consult the Panel Controls chapter.

MIDI Out

- The **Device** drop-down menu specifies which of the available MIDI In devices the instrument can send messages to. (You make MIDI In devices available to ensembles in the Audio Setup dialog, MIDI page.)
- **Channel** specifies the MIDI Channel number used by the instrument for MIDI output.

Connection

- The **OSC Source** drop-down menu specifies the OSC computer from which the ensemble receives MIDI data. Only computers which are present in the OSC member list in the **OSC Setup** dialog are available here.
- The **OSC Target** drop-down menu specifies the OSC computer to which the instrument sends MIDI data. Only computers which are present in the OSC member list in the **OSC Setup** dialog are available here.
- The **Connections** box lists the “pathname” of the instrument with respect to the internal structure of the ensemble. For example, if the instrument is named **Synth** and the ensemble is named **Ensemble**, the pathname will be displayed as **Ensemble/Synth**.

16. Primary Macros

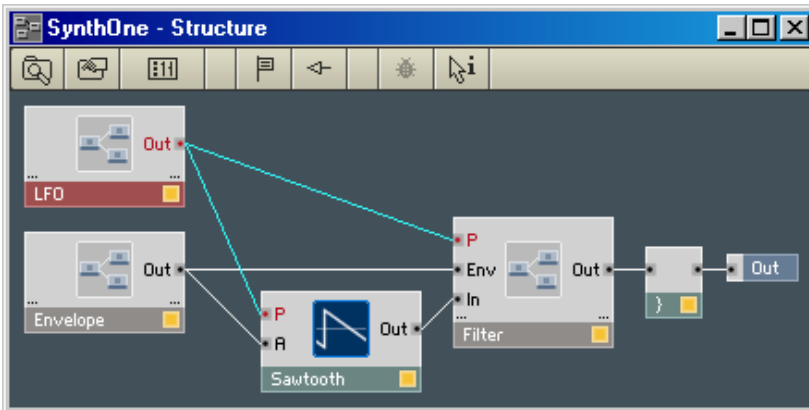
16.1. What is a Primary Macro?

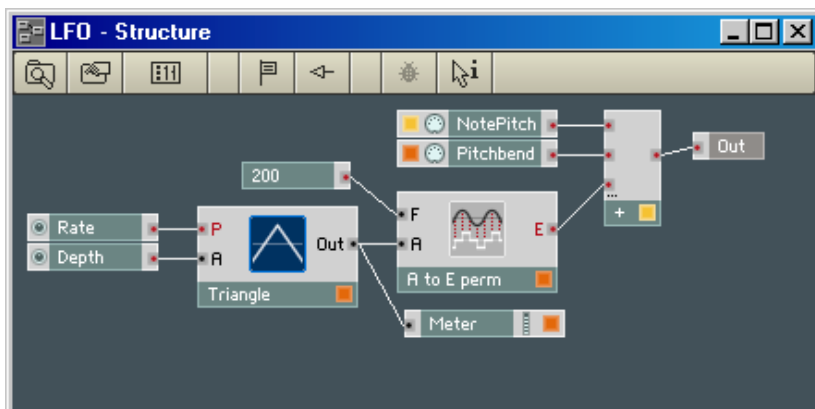
Primary macros have an internal structure just like instruments, but unlike instruments they have no MIDI management, separate panel, or snapshots. Primary macros have a gray label and can be recognized by the picture on their structure icon: 3 modules wired together.



Macro object

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





Example for the integration of a macro into a structure

Note: To avoid wordiness, we will refer to “primary macros” as simply “macros” for the rest of this chapter. Bear in mind that what you read here applies to primary macros only, not to core macros.

16.2. Adding Macros to a Structure

You add macros to a structure by loading them from the REAKTOR system library or from your user content storage area (set in **Preferences** dialog, Directories page).



Use any of these methods to add a macro to a structure:

- XP: Right-click / OS X: Ctrl+click on a blank part of the structure, and use **Macro** from the context menu to find and insert the desired macro.
- Open the Browser (**View->Show Browser** or **F5**). Use the upper **Macro** button to find a system macro, or the lower **Macro** button to find a user macro (from your user content storage area). In the lower pane, drag the desired macro into the structure.
- Use the Browser’s **Disk Navigation** controls (top row) to navigate to the desired macro folder, then drag the macro into the structure.




The system library provides a generous selection of premade macros. If you want to start developing a new macro, you first need to load an empty one (i.e. one that begins with **_New**) from the system library.

When inserting a macro in a structure, you are in effect creating a copy of the macro file that is stored on your disk. The macro copy and the macro file are completely independent. Changes you make to the macro copy will not affect the macro file, and vice-versa. If you want to change the macro file, you must change the copy (in a structure), and then use **Save Macro As...** (from the macro's context menu) to save the changed copy over the existing macro file.

16.3. Ports

There is no fixed arrangement of input and output ports for macros. The type and number of ports in a macro is determined by the user by the insertion of (, ) **terminals** in the instrument structure.

A terminal in the macro's structure appears as a port when you view the macro from its parent structure. For example, if you inserted one input terminal and one output terminal in a macro structure, and then double-clicked in the structure to move up to its parent structure, the macro icon would have an input port (left edge) and an output port (right edge). Thus, the signal enters the macro at its input port, gets processed inside the macro's internal structure, and is then sent back to the parent structure through the macro's output port.

Tip: You can also create macro ports from within the macro's parent. Simply (XP) Ctrl+drag / (OS X): +drag (i.e. hold down the Ctrl/ key while dragging the mouse) a wire from the desired port in the parent structure to the desired edge of the macro icon (left edge to create an input port, right to create an output port). When the new macro port appears, release the mouse button to create it. The macro port will get the name of the macro or module from whose port you Ctrl/+dragged the wire.

16.4. Context Menu

The context menu of a macro contains these entries:

- **Mono**, when on, switches the macro to monophonic operation. For details, see Macro Properties, Function Page, Status below.
- **Mute**, when on, disables the selected macro. For details, see Macro Properties, Function Page, Status below.
- **Cut** removes the selected macro from the structure and stores it temporarily in the clipboard. From there, the macro can be pasted (using the **Paste** command) to a different structure (or another location in the same structure) .
- **Copy** does the same thing as **Cut**, but it does not remove the macro from the structure.
- **Duplicate** creates a copy of the selected macro in the same structure. Choosing **Duplicate** is equivalent to choosing **Copy**, then **Paste**.
- **Delete** deletes the selected macro from the structure.

Tip: To save time, use the keyboard shortcuts for the above four commands: Cut = XP: **Ctrl+X** / OS X: **⌘+X**, Copy = XP: **Ctrl+C** / OS X: **⌘+C**, Duplicate = XP: **Ctrl+D** / OS X: **⌘+D**, Delete = **Del**. Also: Paste = XP: **Ctrl+V** / OS X: **⌘+V**.

- **Save Macro As...** enables the selected macro to be saved to an *.mdl file on your disk. Builders typically use **Save Macro As...** to save new or modified macros to their user content Macros folder.
- **Structure** opens the selected macros' structure in the main structure window. Choosing **Structure** is equivalent to double-clicking on the macros' structure icon.
- **Structure Window** opens the selected macros' structure in a separate (i.e. not the main) structure window. Doing so enables multiple structure windows to be open at the same time. Choosing **Structure Window** is equivalent to Alt+double-clicking on the macros' structure icon.
- **Properties** opens the selected macros' Properties dialog. For details, see Macro Properties below.

16.5. Macro Properties

There are many ways to open a macro's Properties dialog. Use whichever one suits you best:

- Double-click on the macro icon's title bar (not its icon picture!) in a structure window.
- XP: Right-click / OS X: Ctrl+click on the macro icon in a structure window, and select **Properties** from the context menu.
- Select the macro icon in a structure window, and select **View->Show Properties** (or press **F4**).

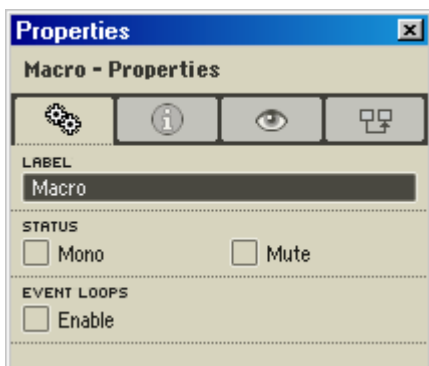
Like all Properties dialogs, a macro's Properties dialog has four pages, each accessible from a picture button on the top of the dialog:

 **Function**,  **Info**,  **Appearance**, and  **Connection**.

Note that its Connection page is empty.

Tip: The contents of the Properties dialog automatically changes to display the values for the currently selected object (ensemble, macro, primary macro, etc.). So, to compare object values, leave the Properties dialog open and toggle your selection between the two objects.

Function Page



Properties dialog of a macro, Function page

Label

The **Label** field specifies the name of the macro; i.e. the name that appears in the macro structure icon title bar and in the macro panel frame (assuming the macro has a frame). You can change this field to (re)name your macro.

Status

- **Mono**, when on, switches the macro to monophonic operation by turning **Mono** on for all the modules inside. Since monophonic mode makes significantly less demands on your CPU, you should always turn **Mono** on, unless the macro must work polyphonically.
- **Mute** mutes (disables) the macro and all objects that lie upstream from it (i.e. that feed into it). In structure view, a macro whose Mute option is turned on has a red M over its status LED and red crosses over its input/output ports.

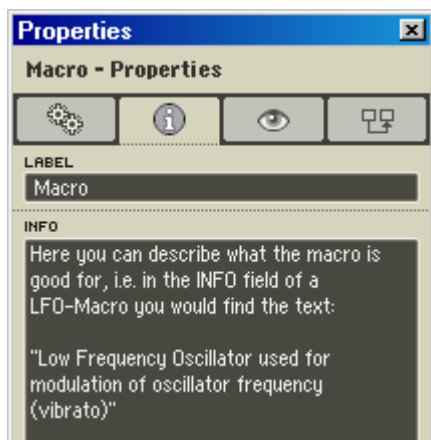
Event Loops

When the **Event Loops** option is enabled, REAKTOR allows event-signal loops to occur within the macro. These loops can lead to stack overflow crashes, which in turn can make the ensemble un-playable and, in some cases, un-openable.

When **Event Loops** is disabled, event-signal loops are prevented from occurring. If a loop is about to occur, REAKTOR displays a message revealing the source of the loop and asking you how to proceed.


We recommend that you disable **Event Loops** to maximize the stability of the macro.

Info Page

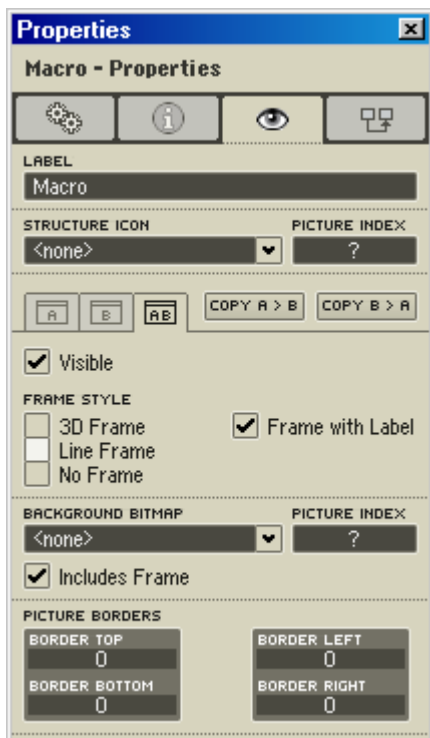


Properties dialog of a macro, Info page

Enter desired information about your macro into the **Info** field of this page.

If  **Show Info** in the Ensemble Panel or Structure toolbar is enabled, your **Info** field text will be displayed in a popup whenever the mouse points to the macro structure icon or panel frame (assuming the macro has a frame).

Appearance Page



Properties

Macro - Properties

LABEL
Macro

STRUCTURE ICON <none> **PICTURE INDEX** ?

Panel Controls
A B AB COPY A > B COPY B > A

☒ Visible

FRAME STYLE
☐ 3D Frame ☒ Frame with Label
☐ Line Frame
☐ No Frame

BACKGROUND BITMAP <none> **PICTURE INDEX** ?

☒ Includes Frame

PICTURE BORDERS

BORDER TOP 0	BORDER LEFT 0
BORDER BOTTOM 0	BORDER RIGHT 0

Properties dialog of a macro, Appearance page

Structure Icon

- **Structure Icon:** Lets you replace the default macro structure icon picture (three modules wired together) with your own picture.
- **Picture Index:** If you choose a structure icon picture that contains multiple smaller pictures, you can select the index for the desired picture (after having set **Num Animations** in the Picture Properties dialog).

Panel Controls

- **A, B, AB:** These determine if changes you make to the appearance of the macro are applied to panel A (**A**), panel B (**B**), or both panels A and B (**AB**). This affects two things: the **Frame Style** options (see below) and

the **Picture Borders** settings (see below). If **A** is enabled, the **Frame Style** options and **Picture Borders** settings will only be applied to panel A of the macro. If **B** is enabled, the options and settings will only be applied to panel B. If **AB** is enabled, the options and settings will be applied to both panels, A and B.

- **Copy A > B, Copy B > A:** Click on one of these buttons to copy the full content and appearance of one panel to the other.

Frame Style

- **3D Frame:** Displays a 3D frame around the macro controls in panel view.
- **Line Frame:** Displays a line frame around the macro controls in panel view.
- **No Frame:** Displays no macro frame in panel view.
- **Frame with Label:** Displays the label (name) of the macro in its panel frame.

Background Bitmap

- **Background Bitmap:** You can load your own picture for the macro panel background. All panel controls and displays will lie on top of the background picture. You can assign a different background picture to each macro panel (A and B).
- **Picture Index:** If you choose a background picture that contains multiple smaller pictures, you can select the index for the desired picture (after having set **Num Animations** in the Picture Properties dialog).
- **Includes Frame:** Displays a frame around the background picture.

Picture Borders

The **Border Top**, **Border Bottom**, **Border Left**, and **Border Right** values determine the number of pixels used for the top, bottom, left, and right borders of the macro panel(s) specified by **A**, **B**, and **AB** (see above). Due to REAKTOR panel gridding, the Picture Borders values should be set to multiples of 4: 0, 4, 8, 16, etc.

17. Primary Structures

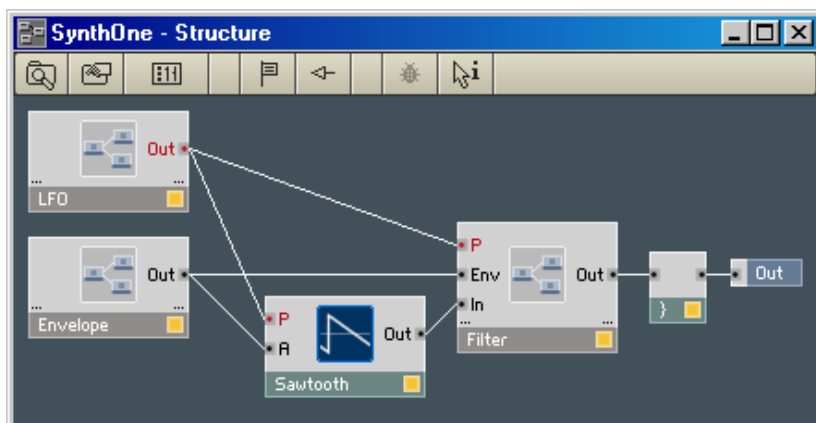
17.1. What is a Primary Structure?

REAKTOR is based on an open concept that allows for the design and realization of any imaginable sound generator. In many respects it is similar to a classic modular synthesizer system. That's why the most important basic building block you deal with in REAKTOR is called a **module**. (Primary module in the primary level; core module in the core.)

A library of primary (and core) modules is built into REAKTOR. These built-in modules provide the basic building blocks for MIDI and audio signal processing. Complex signal processing structures can be created by connecting modules that carry out relatively simple tasks.

The window in which primary modules are placed and interconnected is called a **Primary Structure window**.

Note: To avoid wordiness, we will refer to “primary structures, macros, and modules” as simply “structures, macros, and modules” for the rest of this chapter. Bear in mind that what you read here applies to primary structures, macros, and modules only, not to core structures, macros, and modules.



A Structure window

We strongly recommend that you keep to hierarchical principles when building structures in REAKTOR. The ensemble should (and, in fact, can) contain instruments only. Instruments should contain macros and modules and core cells only (not other instruments). Macros should contain other macros and modules and core cells (not instruments).

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

- Only instruments, not macros or modules, can reside in an ensemble structure window. To this end mixers, which you use to mix signals from several instruments, are available as instruments in the system library.
- During the construction of instruments, group as many functional blocks as possible in the form of macros. One advantage to working this way is that identical elements (e.g., oscillators and envelopes), which are often used more than once in the construction of synthesizers, need only be constructed once and can then be copied as needed. Also, your structures will be very clear, which makes tracking down problems much easier.

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 a **picture icon** (e.g., oscillators have waveform pictures).



The Pulse FM oscillator module

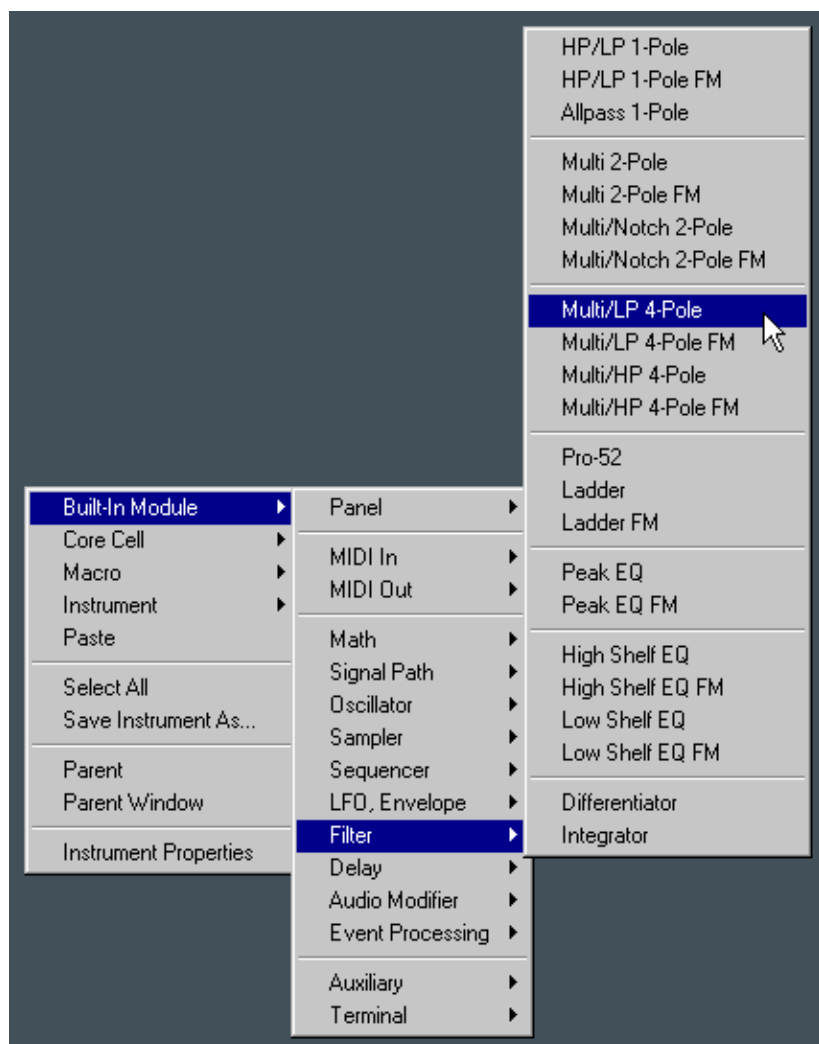
Adding Modules to a Structure

To add a new module to a structure, use the structure window's context menu. The submenu **Built-In Module** lets you select a built-in module from REAKTOR's system library. A popup menu with several levels appears:

First you must select the functional group (e.g., **Filter**) and then choose the

actual module that belongs to the group (e.g., **Multi/LP 4-Pole**). Detailed information about all of REAKTOR's modules can be found in the Module Reference section of this manual.

The module icon will be placed at the point in the structure window where you opened the context menu (by right-clicking or Ctrl+clicking), but you can move it around like any other REAKTOR object.



Menu for inserting a new module

Module Ports

Each REAKTOR module contains one or more ports through which the module can be connected to other modules. The left edge of the module holds the input ports and the right edge holds the output ports.

When any input port is left unconnected, it receives a zero (0) signal. So, connecting no wire to an input port has the same result as connecting a constant source with the value set to 0.

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

- **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 two or more audio signals are to be fed to an audio input, they must first be mixed using an **Add** or **Amp/Mixer** module. If a connection is made to an audio input port that already has a wire attached, the first wire will be deleted as soon 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. Since event processing allows complex manipulation of control messages without continuous calculations, the load on the CPU is reduced. Ports for event signals are labeled with red characters and marked with a small red dot. If two or more event signals are to be fed to an event input, they must first be merged using a **Merge** module. Gate signals are a special type of event signal. An event with a non-zero value turns on the gate. When it is followed with a zero (0) event, the gate is turned off again.

Some modules can be used either for audio or event signals. If you insert such a module (e.g. the **Add** module), it will appear first as an event module (i.e. its ports will be red). As soon as you connect an audio wire to one of its inputs, however, it will convert to an audio module and the CPU load will become substantially 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 section **Panel Editing** and **Panel Operation** for details about working with controls on the panel).

- **Create Constant** automatically creates a Constant module with a suitable value for the port.
- **Mute Port** mutes the port (i.e., sets its value to zero). Muted ports are marked with a red cross.

Module Context Menu

Mono

A module can operate either in monophonic (single-voice) mode or in polyphonic (multiple-voice) mode. In polyphonic mode, 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 LED at the bottom left corner of the module. Monophonic modules have an orange status LED.

For most modules the operating mode can be changed using the entry **Mono** in the context menu or the **Mono** option in the module's Properties dialog (Function page). Unless a module really needs to be run in polyphonic mode, it should always be used in mono mode, since the CPU load increases proportionally to the number of voices used.

Mute

A module can be muted by selecting **Mute** from its context menu or Properties dialog (Function page). Muted modules are recognized by a red cross over the status LED.

A muted module makes no computational demands on your CPU. If a module is not needed temporarily, it should be muted; if it is never used, it should be deleted.

Modules are automatically disabled if their outputs are not connected or are only connected to other disabled or muted modules. The status LEDs of disabled modules are unlit.

This feature is especially useful with switches, because alternative branches of signal processing can be selected but only one will cause 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, Duplicate

- **Cut** removes the selected module from the structure and stores it temporarily in the clipboard. From there, the module can be pasted (using the **Paste** command) to a different structure (or another location in the same structure) .
- **Copy** does the same thing as **Cut**, but it does not remove the module from the structure.
- **Duplicate** creates a copy of the selected module in the same structure. Choosing **Duplicate** is equivalent to choosing **Copy**, then **Paste**.

Delete

Delete deletes the selected module from the structure.

Tip: To save time, use the below keyboard shortcuts for the above four commands:

- Cut: XP: Ctrl+X / OS X: ⌘+X
- Copy: XP: Ctrl+C / OS X: ⌘+C
- Duplicate: XP: Ctrl+D / OS X: ⌘+D
- Delete: Del
- Paste: XP: Ctrl+V / OS X: ⌘+V

Properties

A dialog 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** section of this manual.

17.3. Source Modules

What are Source Modules?

In REAKTOR, **source module** is the name given to a module that outputs a control signal. There are three different kinds of source modules:

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

Control Source Modules

The **Fader**, **Knob** and **Button** modules 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 (**Built-In Module** ⇒ **Panel** ⇒ **Fader / Knob / Button**).
- In the context menu of a module input port select **Create Control**. A control source is created and connected to the input. Type, label, and settings of the control source are configured to suit the input; note that you might have to change these to fix your needs. In many cases you can save a lot of time by using **Create Control** to add control sources.

Control sources and their respective panel elements can be controlled via MIDI in various ways.

Control sources have their own context menu which you open by XP: right-clicking / OS X: Ctrl+clicking on the module. The menu choices include: **MIDI Learn**, **Cut**, **Copy**, **Duplicate**, **Delete** and **Properties**.

MIDI Source Modules

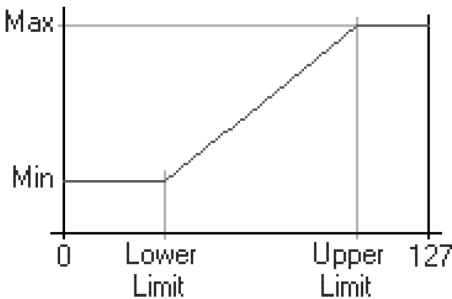
MIDI source modules are used for controlling audio signal processing with MIDI events. For each type of MIDI event there exists a particular kind of source module. The output signal of such a source corresponds to the values transmitted by the particular MIDI events. For example, the **On Vel.** source module outputs a control signal that corresponds to the Note On Velocity message transmitted by MIDI when a MIDI keyboard key is pressed.

MIDI source modules are inserted through the context menu of the structure window by choosing **Built-In Module** ⇒ **MIDI In...**

Range of Values

For control and MIDI source modules the range of the output control signal is scaled to the range between **Min** and **Max** (as set in the module's Properties dialog, Function page) to achieve optimum control of the particular module parameter.

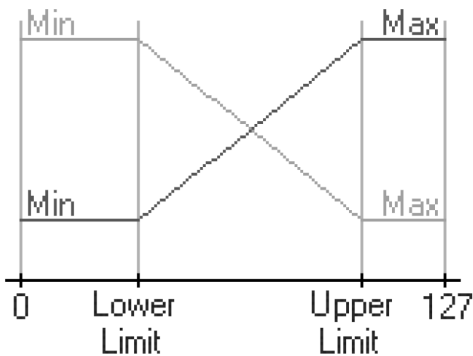
For MIDI source modules the range can also be limited with the Properties **Lower Limit** and **Upper Limit**. The output value of the source module is limited to **Min** for MIDI values below **Lower Limit** and to **Max** for MIDI values above **Upper Limit**. The range between the two limits is interpolated 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 Source Modules

Module and macros with fixed values need to be fed by constant source modules. Set the desired **Value** in the Properties dialog (Function page) of the **Constant** module.

To insert a constant source module, select **Built-In Module** ⇒ **Math** ⇒ **Constant** in the context menu of the structure.

17.4. Switches

Switches are not source modules 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 or macros 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 section of this manual.

The use of switches in a structure can also play a significant part in reducing the load on your CPU. This is because modules or parts of the structure that are not connected to REAKTOR's audio outputs (or to the 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

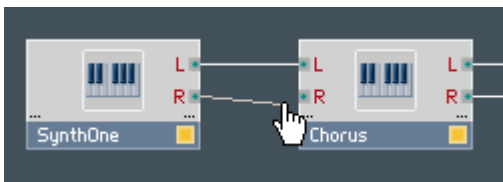
Terminals are very inconspicuous but immensely important modules in REAKTOR structures. They are like the sockets on hardware instruments. Each input or output terminal within a structure appears at the next higher level – i.e., in the instrument or macro – as a port from which connections to other instruments, 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**, **IC Send**, **IC Receive**, **OSC Send** and **OSC Receive**. The normal rules for wiring apply.

Terminals are created using the context menu of a structure window under **Built-In Module** ⇒ **Terminal...**. The **Label** of an In Port or Out Port 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 the following picture) to prevent any possible confusion. You should also provide terminals with a description (Properties dialog, Info page). The terminal label appears as the port label in the parent structure, and the terminal description appears in a popup info box when the mouse points to the port (if **Show Info** is enabled).

17.6. Wires

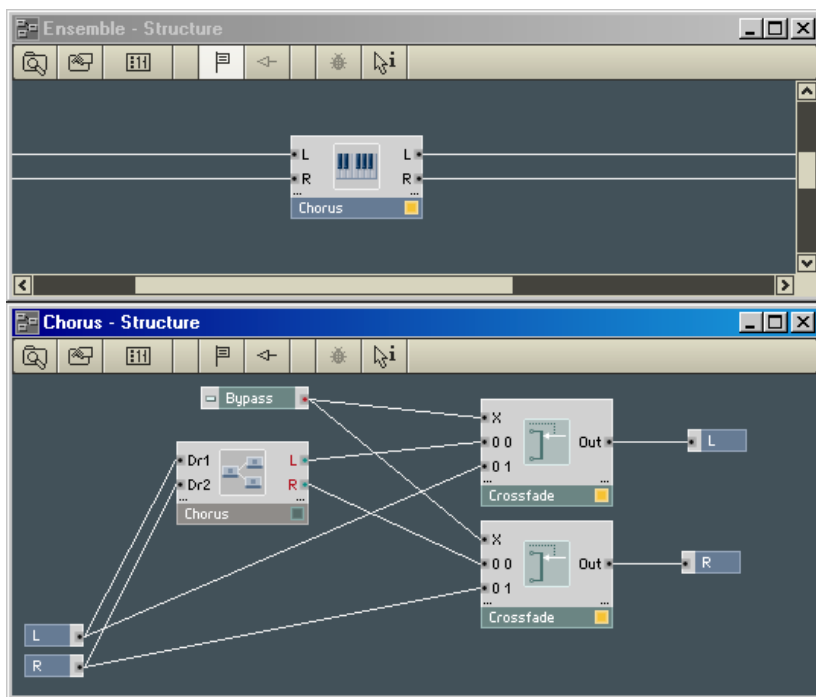
The connection between the ports of two modules or macros, shown as a line, is called a **wire**. Wires transport signals between the modules/macros.



Connecting a wire

Creating

To make a new wire:



17. 5. *Instrument with ports and its structure with terminals*

- Click on one of the two ports to be connected with the left mouse button, drag the mouse pointer to the other port, and then release the mouse button. A wire appears between the ports and the effect on the sound resulting from the change to the structure can be heard immediately.

Deleting

There are two ways to delete a wire:

- Do the same as you would to create a new wire; i.e. drag from one port to the other.
- Drag the mouse from the input port to which the wire is connected to a blank part of the structure.
- Select the wire you want to delete by clicking on it , then press the **Del** key on your computer keyboard.

Rules for Wiring

When wiring modules together, a few **general rules** always apply:

- A wire can only connect an output port to an input port or vice-versa. It can never connect an input port to an input port, or an output port to an output port.
- An output port can feed (be connected to) as many as 40 input ports.
- When no wire is connected to an input port, it receives a zero (0) signal, meaning that the value at the port is 0.

In addition, the following special rules apply:

- An event input port cannot process audio signals. If an event input port is to be fed from an audio output port, the signal must first be converted with an **A to E** module (see the **Module Reference** section of this manual).
- An event output port can be connected to audio input ports as well as event input ports.
- When connecting a monophonic signal to a polyphonic input port, all polyphonic voices receive the same value (from the monophonic signal). For pitch signals this means the voices in effect play in unison.
- A polyphonic output cannot be connected to a monophonic input (a red cross appears on the input port). An **Audio Voice Combiner** module must be used for converting poly to mono.

Displaying Wire Signal Values

While the mouse pointer rests on a wire (and if **Show Info** is on), the value of the signal in the wire is shown in a popup info box.

For event signals, the value of the last event is shown. (If 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 constantly changing, you may need to move the mouse pointer away from the wire to close the popup, 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 with 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. Signal Processing in REAKTOR

REAKTOR distinguishes two kinds of signals: event and audio. Event signals are typically processed at a rate of several hundred times per second, whereas audio signals are processed at the audio sampling rate, which is tens of thousands of times per second. For example, the standard audio sampling rate for Compact Discs is 44,100 times per second (usually written as 44.1 kHz). Having two processing rates conserves CPU load. Both the sample rate and the control rate (which is used by a handful of modules), can be changed from REAKTOR's **Settings** menu.

REAKTOR's audio generating and processing modules process signals at the audio rate. There are a few modules in REAKTOR, such as **Event Smoother**, **LFO**, **Slow Random** and **A to E**, that generate and process event signals at the control rate.



Even Smoother module



LFO module



Slow Random module



A to E module

However, some event modules do not scan continuously for events, but only react when a new event arrives. A new event can be created from inside the structure, by a mouse action (i.e., when you move a panel control with the mouse), by an incoming MIDI message, or even by an audio event. When an audio signal is used to create event signals (e.g., using the **A to E Trig**) an event

output port can even produce a signal which is refreshed at the audio rate.

Event input ports compute all incoming events regardless of their rate. A special case is the **Iteration** module, which can compute even multiple events within one audio sample. Finally, there are hybrid modules that can be configured to process signals at either rate - math modules are a typical example. On those modules, the ports are marked with three different colors to indicate their mode:

- A green dot on a port of a hybrid module indicates that the mode has not yet been set and you can connect either an event or audio signal.
- A red dot on a port of a hybrid module indicates that the mode has been set to **event** by connecting an event cable to the module.
- A black dot on a port of a hybrid module indicates that the mode has been set to **audio** by connecting an audio cable to the module.

Event Signals

Event signals are control messages for changing a value. Typical sources for events are MIDI inputs and panel faders. Event processing allows for complex manipulation of control messages without continuous calculations and thereby reduces the load on the CPU in comparison to the calculation demands of audio signals. Ports for event signals are labeled with a red dot and a red label. To connect more than one event cable to an event input port, place a **Merge** module in front of it. An audio output port cannot be connected directly to an event input port; you must use an **A to E** converter module for this purpose.

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- or negative-valued event, the gate is turned off again.

An event has two properties: the time 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 that operate on Events (e.g. **Add** used as event module) work at the exact moment an event arrives (i.e., event timing keeps perfect pace

with the audio sample rate). Other event modules (**A to E** or **LFO**, for example) operate only at the lower timing resolution determined by the Control Rate (e.g., 200 times per 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 called “depth before breadth”: An event propagates as deep as it can along one track before another wire fanning out from the same port is processed.

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 **Value** module. A complex event processing structure can be connected to its lower (value) input, but it will only pass on this value as an event when a triggering event arrives at the Trig 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 soon 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.

Event Loop Prevention

The **Globally disable event loops** option in the **Preferences** dialog (Options page) and the **Event Loops Enable** options in the ensemble/instrument/macro **Properties** dialogs (Function page) allow the suppression of event signal loops which can lead to stack overflow crashes.

Note: Unprotected **Event Loops** will crash REAKTOR. This is not bug, but by design. This is only avoidable by careful instrument design. The fuse for this case is often the insertion of a **Value** module.

Disabling event loops (by turning on **Globally disable event loops** or turning off **Event Loops Enable**) prevents event-signal loops from occurring in ensembles. If an event loop is about to occur, REAKTOR displays a message revealing the source of the loop and asking you how to proceed.

Event loops can lead to stack overflow crashes, which in turn can make ensembles un-playable and, in some cases, un-openable. If this happens, restart REAKTOR, turn on **Globally disable event loops**, open the problematic ensemble, and trace the source of the event loop with the help of event-loop identification messages. (It can be useful to disable audio to prevent further loops occurring during this process.)

We recommend that you globally disable event loops to maximize the stability of REAKTOR. To ensure backward compatibility, files saved in older versions of REAKTOR have event loops enabled by default.

Note: In most cases, the **Iteration module** can avoid the need for creating event loops. The Iteration module has a limited speed option in its properties, which can avoid audio glitches caused by processing a large number of iterations.

Audio Signals

Audio signals are comparable to sound signals and control voltages in the analog world. The processing of such signals constitutes a permanent load on the CPU. Audio module 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 **Adder** or **Amp/Mixer** module. If a connection is made to an audio input port that already has a wire attached, the first wire will be deleted as soon as the second one is connected.

Activating Audio modules

Since audio modules are a constant drain on the CPU, REAKTOR automatically disables modules (both audio and event) that are not ultimately connected to the audio output. By “ultimately” we mean connected by some series of wires from the modules output to the ensemble’s Audio Out module (of which there is always and only one). You can tell that a module is active by the glowing LED in its lower-right corner.

Some audio modules (lamps for example) can be set to be always active in

their Properties dialog (Function page). In that case, they will always be active (notice that their LED comes on when this property is set for an unconnected module), and they will activate any modules connected to them. Modules with the Always Active option have another special property that doesn't depend on the Always Active option being turned on: when any of their input ports are connected, they will "look back through the signal path" to see if there are any active modules connected to them, and if so, they will become active. Lamps are a good example of the reason for that - they have no outputs to make them active, but you would want them to be active whenever they are connected to an active module (i.e., whenever there's something for them to indicate).

Order of Audio Processing

Unlike event processing, whose order depends on the order in which they were created, audio modules are processed in an order that depends on their position in the signal flow. You can see the processing order of audio modules in the structure by selecting **System->Debug->Show Module Sorting** from the main menu.

The sorting process is relatively straight forward until a feedback loop is encountered. Feedback loops are allowed - they are useful for physical-modeling instruments, for example - but REAKTOR must arbitrarily assign a sorting order for such paths. The first module in a feedback loop is indicated by a vertical blue line at the appropriate port. This indicates an automatically inserted **Unit Delay**. This **Unit Delay** is not visible since it resides inside of the feedbackin module. You can also set the starting point manually by making use of the **Unit Delay** module.

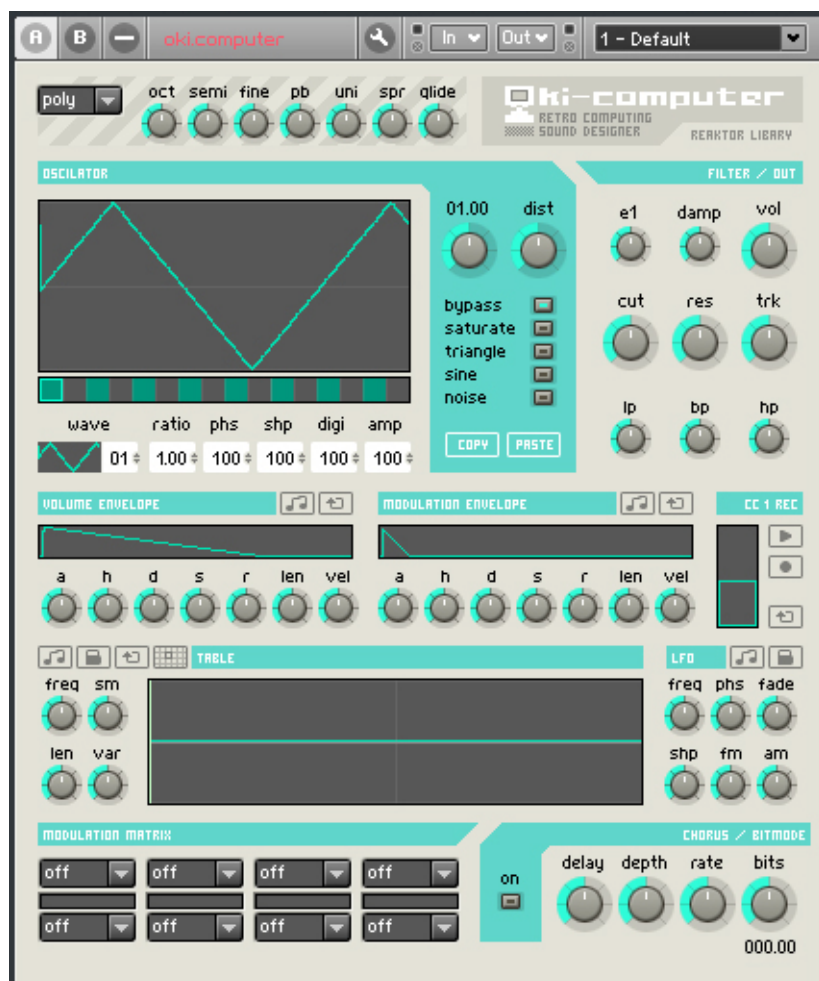
17.8. Context Menu

The context menu of the Structure window has the following entries:

- **Built-In Module** inserts modules into the structure.
- **Core Cell** inserts core cells into the structure.
- **Macro** inserts macros from the library into the structure.
- **Instrument** inserts instruments 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 Windows XP: **Ctrl+V** / OS X: **⌘+V** for pasting, you can specify a point in the structure by clicking on it with the left mouse button first.
- **Select All** selects all the objects in the structure.
- **Save Instrument/Macro As...** saves 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 (.ism or .mdl).
- **Parent** opens the parent structure in the same structure window. For example, if you are in the structure of a macro which resides in an instrument, **Parent** will open the instrument's structure in the same window.
- **Parent Window** opens the parent structure in a separate structure window.
- **Instrument/Macro Properties** opens the Properties dialog of the structure's instrument or macro.

18. Panel Editing

18.1. What Is a Panel?

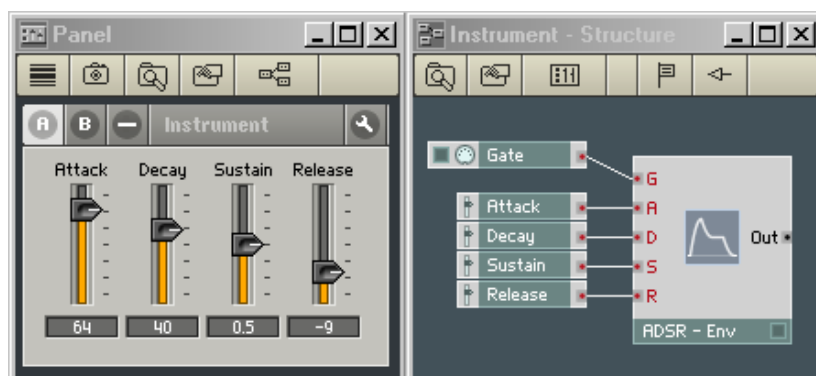


Ensemble Panel window of the OKI Computer ensemble from the REAKTOR 5 factory library

A **panel** is the user interface of an instrument. It corresponds to the front panel of a hardware synthesizer or effects unit, where the various elements (knobs, faders, buttons, meters, etc.) for operating the device are located. Instrument panels are displayed in the Ensemble Panel window.

18.2. What are Panel Controls?

Some REAKTOR modules generate or modify audio signals (oscillators, filters, samplers, saturators, etc.). Others control the signal flow by enabling different values to be sent to module inputs (knobs, faders, buttons, etc.). When these control modules are displayed in an instrument panel, they are called **panel controls**.



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

18.3. Panel Controls

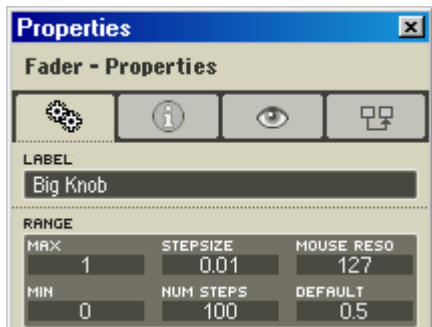
In this section, we'll look at five of REAKTOR's most commonly used panel controls: faders, knobs, buttons, switches, and lists.

Fader and Knob



Different types of faders and knobs

Faders and knobs are panel controls whose settings (i.e. fader-handle and knob-indicator positions) determine the values that their source modules (Fader and Knob) output to other modules in the structure (e.g. the P input of an oscillator, or the A input of a sampler). Their output value range is set by **Min** and **Max** in their Properties dialog (Function page). Their step resolution (the number of increments between Min and Max) is set by **Stepsize**, and their mouse resolution (the distance the mouse must travel to change the knob/fader settings) is set by **Mouse Reso**.



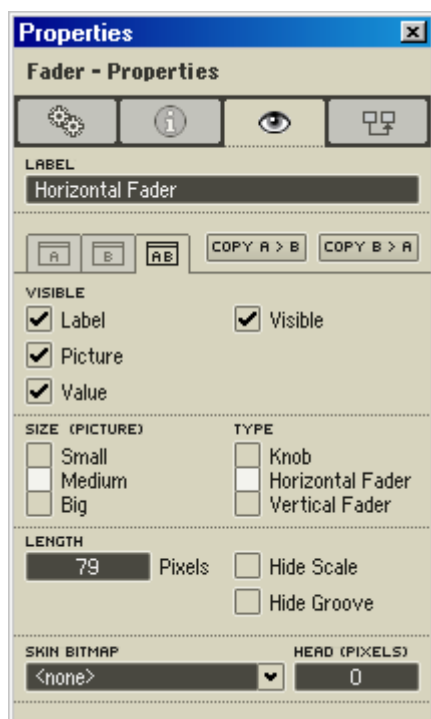
Properties dialog of a knob (Function page)

Tip: If you set **Stepsize** to 0, REAKTOR will automatically change it to a value that yields a total of 127 steps between **Min** and **Max**. This resolution suffices for most controls.

You can change the fader/knob panel setting by dragging over it with your mouse, or by pressing your Up/Down arrow keys (if you click the fader/knob first to select it). You can also use MIDI to change fader/knob settings (see below, MIDI Control).

Tip: Drag your mouse up and down (not sideways!) to change a knob setting.

You can change a fader's/knob's panel appearance in its Properties dialog (Appearance page):



Properties dialog of a fader (Appearance page)

- **Visible (Label, Picture, Value, Visible)** - **Label** shows/hides the label in the panel, **Picture** shows/hides the fader/knob graphical image, and

Value shows/hides the current (output) value. **Visible** shows/hides the entire fader/knob (label, picture, and value).

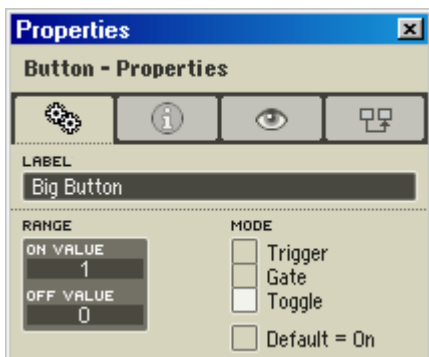
- **Size (Small, Medium, Big)** - determines the size of the fader/knob panel display.
- **Type (Horizontal Fader, Vertical Fader, Knob)** - determines the panel display type. Note that you can display a Fader module as a knob in the panel, and a Knob module as a fader.
- **Length** - the length (or height) of the fader in pixels. This has no effect for knobs.
- **Hide Scale, Hide Groove** (faders only) - shows/hides the fader's scale tickmarks and groove (the slot in which the handle fits).
- **Skin Bitmap, Head** - see below, Panel Control Skins.

Button



Different types of buttons

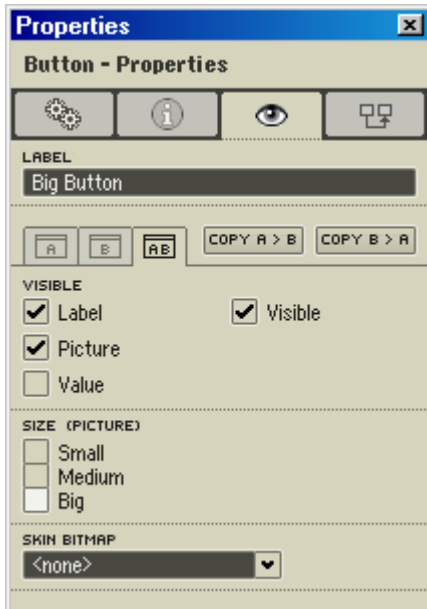
A button is a panel control whose setting (on or off) determines the value that its source module (Button) outputs to other modules in the structure (e.g. the G input of a sampler, or the A input of a clock oscillator). Its output value range is set by **On Value** and **Off Value** in its Properties dialog (Function page).



Properties dialog of a button (Function page)

You turn a button on/off by clicking it with your mouse. You can also use MIDI to turn buttons on/off (see below, MIDI Control).

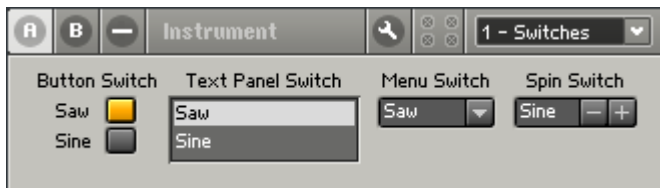
You can change a button's panel appearance in its Properties dialog (Appearance page):



Properties dialog of a button (Appearance page)

- **Visible (Label, Picture, Value, Visible)** - **Label** shows/hides the label in the panel, **Picture** shows/hides the button's graphical image, and **Value** shows/hides the current (output) value. **Visible** shows/hides the entire button (label, picture, and value).
- **Size (Small, Medium, Big)** - determines the size of the button panel display.
- **Skin Bitmap** - see below, **Panel Control** Skins.

Switch



Different types of switches

A switch is a panel control whose setting (selected option) determines which of its source module's input signals is passed to its output port. For example, you might have a switch that receives two input signals, one from a sawtooth oscillator and the other from a sine oscillator. If the Sawtooth option in the switch is turned on, the sawtooth input is passed to the switch output; if the Sine option is on, the sine input is passed to the output.

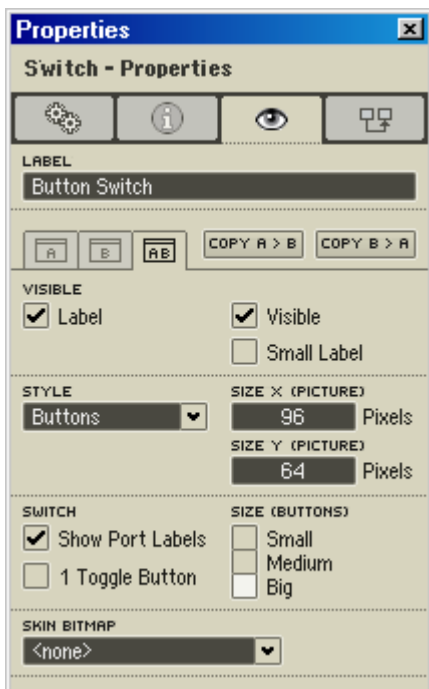


Switch module

A switch can have multiple inputs (as in the sawtooth/sine example) or a single input. In a single-input switch, the switch setting determines whether the input signal is or is not passed to the output.

Along with your mouse, you can use MIDI to change switch settings (see below, **MIDI Control**).

You can change a switch's panel appearance in its Properties dialog (Appearance page):

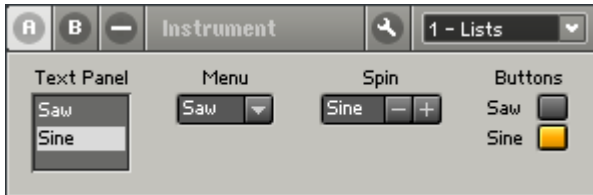


Properties dialog of a switch (Appearance page)



- **Visible (Label, Small Label, Visible)** - **Label** shows/hides the label in the panel, **Small Label** shows/hides a small version of the label. **Visible** shows/hides the entire switch.
- **Style (Buttons, Menu, Text Panel, Spin)** - **Buttons** displays switch options as buttons, **Menu** displays them as text items in a drop-down menu, **Text Panel** displays them as text items in a box, and **Spin** displays them as text items in a menu with +/- navigation buttons.
- **Size X, Size Y** - specify the width and height (in pixels) of a switch whose style is set to Menu, Text Panel, or Spin.
- **Switch (Show Port Labels, 1 Toggle Button)** - **Show Port Labels** shows/hides labels for the switch's buttons. **1 Toggle Button**, when enabled, displays only the first button (i.e. the first input port) of the switch (see Tip below).
- **Size (Small, Medium, Big)** - determines the size of a switch whose style is set to Buttons.
- **Skin Bitmap** - see below, **Panel Control Skins**.

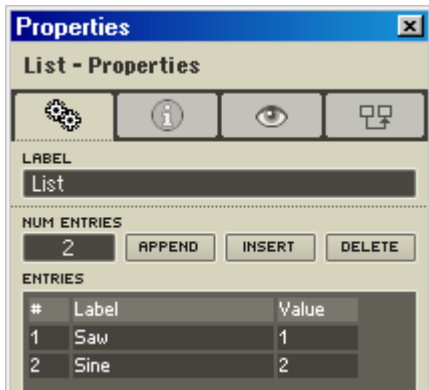
Tip: If you are using a switch to toggle between two states (e.g. on/off, engage/bypass, etc.), you can use the **1 Toggle Button** option to display a single switch button (On, Bypass, etc.) instead of two buttons.

List



Different types of lists

A list is a panel control whose setting (selected option) determines the value that its source module ( List  List) outputs to other modules in the structure. You define its options and their corresponding values in the Entries list box (Properties dialog, Function page):



Properties dialog of a list (Function page)

Num Entries - specifies the number of entries (options) in the list.

- **Append, Insert, Delete** - **Append** appends a new entry to the end of the list, **Insert** inserts a new entry after the selected entry, and **Delete** deletes the selected entry.

- **Entries (#, Label, Value)** - # displays the entry number, **Label** specifies the entry label (i.e. the text that will appear in the list panel control), **Value** specifies the entry value.

Along with your mouse, you can use MIDI to change switch settings (see below, MIDI Control).

You can change a list's panel appearance in its Properties dialog (Appearance page), just as you would change a switch's panel appearance (see above, **Switch**.)

Context Menu

The following context menu options appear when you XP Right-click /OS X: Ctrl.+click) on any of the above panel controls (fader, knob, button, switch, list):



Context menu of a fader, knob, button, switch and list

- **MIDI Learn:** Enables MIDI Learn for the control, which helps you assign an external MIDI control (e.g. a knob on a MIDI keyboard) to a panel control. (For details, see above, **REAKTOR Toolbars**, **Ensemble Toolbar**.)
- **Set to Default:** Sets the control to its default value (as specified in **Properties** dialog, Function page).
- **Show in Structure:** Opens the structure in which the panel control's source module is located.
- **Properties** Opens the control's Properties dialog.

18.4. Panel Control Skins

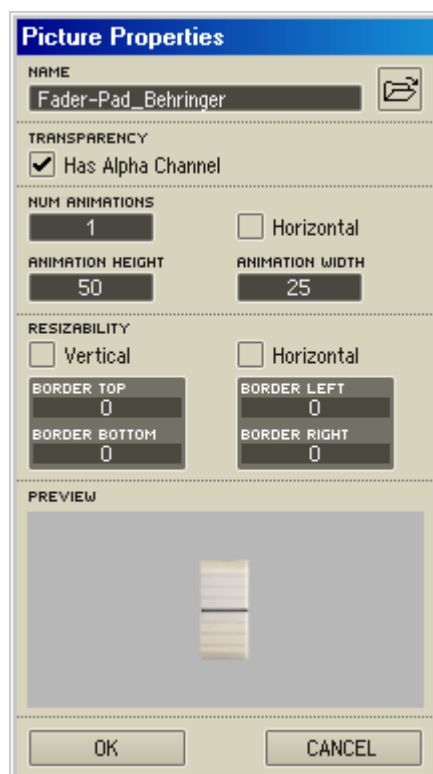
REAKTOR enables you to customize the appearance of several panel controls by applying skins to them: faders, knobs, buttons, lists, switches, Receive modules, lamps, and meters.

Fader Skins

There are two types of fader skins: single-picture skins and animation (multiple-picture) skins.

In a single-picture skin, the picture is used as the handle (not the body) of the fader. If the picture is resizable horizontally or vertically (Picture **Properties** dialog), it is resized to the horizontal or vertical size of the original REAKTOR fader handle. If not, the handle size is the same as the picture size.

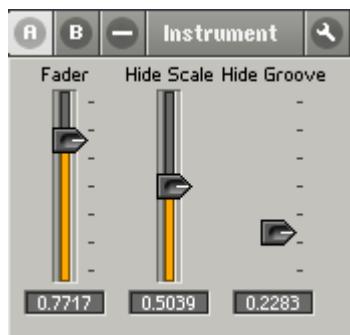
You can use the Head (Pixels) property (**Properties** dialog, Appearance) to add a head to your custom fader handle (see below). Setting Head (Pixels) to 0 places the handle exactly within the fader groove (no head). Setting Head to N (1, 2, 3, etc.) creates an N-pixel-wide handle head.



Picture Properties dialog of a single picture skin bitmap for a fader

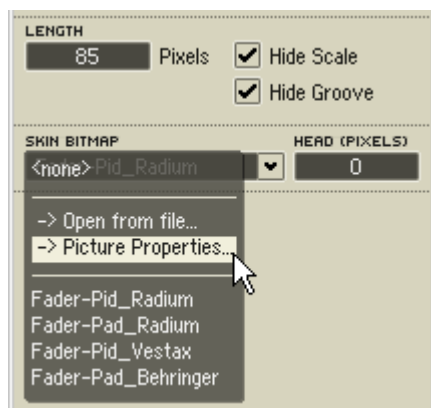
In an animation skin, the entire fader (not just the handle) is replaced by the picture; thus the fader size is determined by the picture size, and the Resiz-

ability (Picture Properties dialog) and Length (Properties dialog, Appearance page) settings are ignored. The number of fader states is equal to the number of animation frames in the picture.



Faders with hidden scale and hidden groove

For all fader skin modes – single-picture, animation, and none – the Hide Scale and Hide Groove options (Properties dialog) hide the fader scale graphics (tickmarks) and groove.



Properties dialog (Appearance page) of a fader

Knob Skins

A knob skin is always treated as an animation. The entire knob is replaced by the animation picture; thus the fader size is determined by the picture size, and the Resizability (Picture Properties dialog) and Length (Properties dialog) settings are ignored. The number of knob states is equal to the number of animation frames in the picture.



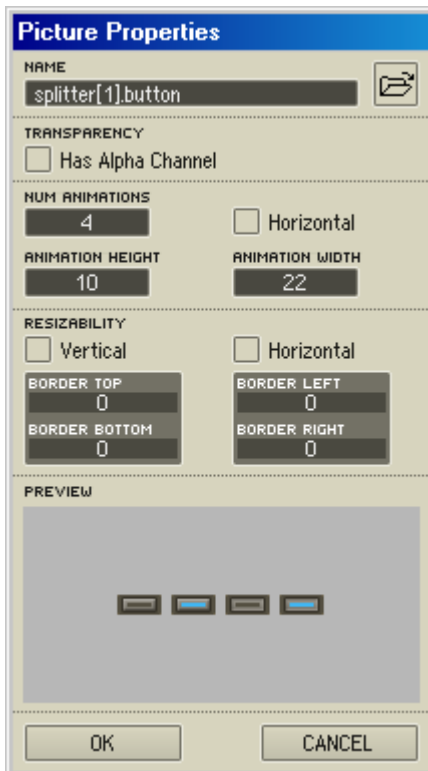
Picture Properties dialog of a skin bitmap for a knob with its animation



Knobs with different skins

Button (List, Switch, Receive) Skins

A button skin uses a four-frame animation picture to define its four states (in this order): Off state Up, On state Up, Off state Down, On state Down. If the picture is resizable horizontally or vertically (Picture Properties dialog), it is resized to the horizontal or vertical size of the original button. If not, the button size is the same as the picture size.



Picture Properties dialog of a skin bitmap for a button with its four animation states

The List, Switch, and Receive modules can all have button-type skins if their style is set to Buttons (Properties dialog, Appearance page).



A list module and a switch module which use skin bitmaps for their button states

Lamp Skins

A lamp skin uses a two-frame animation picture to define its two states (in this order): Off state, On state. If the picture is resizable horizontally or vertically (Picture Properties dialog), it is resized to the horizontal or vertical size of the original lamp (as determined by Size X and Size Y in Properties). If not, the lamp size is the same as the picture size.

Meter Skins

There are two types of meter skins: on/off and animation.

An on/off skin uses a two-frame animation picture to define its two states (in this order): Off state, On state. If the picture is resizable horizontally or vertically (Picture Properties dialog), it is resized to the horizontal or vertical size of the original meter (as determined by Size X Segment and Size Y Segment in Properties). If not, the meter size is the same as the picture size.

An animation skin uses a multiple-frame animation picture to define its states. If the picture is resizable horizontally or vertically (Picture Properties dialog), it is resized to the horizontal or vertical size of the original meter (as determined by Size X Segment and Size Y Segment in Properties). If not, the meter size is the same as the picture size. The number of meter states is equal to the number of animation frames in the picture. The number of segments is determined by the number of animations frames; thus Number of Segments (Properties dialog) is ignored

18.5. Connection Properties of Panel Controls

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

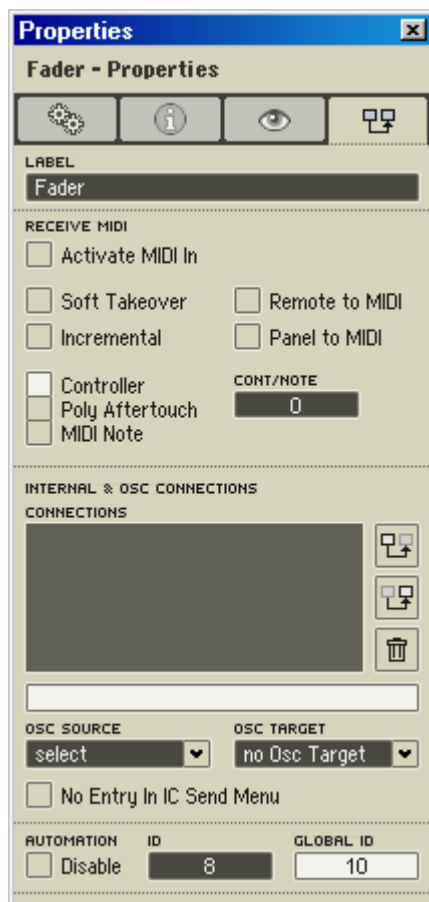
Receive MIDI

Activate MIDI In: When activated, the panel control values 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 incom-

ing value passes the control's current value (either going up or going down). This option 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.



Connection page in the Properties window of a panel control (here: fader)

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 working with 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: These 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 enables 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 the control master and press the  topmost button to the right of the Connections list.
- Select the panel control you want to use as the 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 found 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 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 cannot be edited. This ensures that these two parameters appear consecutively in your host software’s parameter automation list.

18.6. 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 recommend that you carry out these operations only in the structure so that you can keep an eye on the consequences of your actions.

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, panel elements can no longer be moved around. The **Panel Lock** function is set using the context menu of the panel window or simply by clicking on the wrench icon in the Instrument header (a screw icon will appear indicating that the panel is locked down).

19. Panel Operation

19.1. Mouse Control

Fader



To change the fader setting, drag its handle to the desired position.

Knob



To change the knob setting, drag the mouse up and down over it.

Button



You can set a button's operating mode (**Trigger**, **Gate**, or **Toggle**) and its **On Value** and **Off Value** in its **Properties** dialog (Function page).

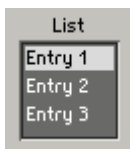


- **Trigger:** Pressing the button generates an event with the specified **On Value**. Releasing the button does not generate an event.

- **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 states; hence the term “toggle”. Pressing it once switches it on and generates an event with the specified **On Value**. Pressing it again switches it off and generates an event with the specified **Off Value**.

When connecting a button to an audio input port, **Trigger** mode behaves the same as **Gate** mode (i.e. the signal returns to its **Off Value** when the button is released).

List



To change the list setting, click on an item to select it.

Switch



Pressing one of the buttons in a switch lets the corresponding input signal pass through the switch, while blocking all other input signals. (To see this, you need to view the switch in its Structure window.) Only one switch button can be active at a time, and therefore only one input signal can pass through at a time.

Drop-Down Menu (Switch and List modules)



In a drop-down menu, you select one item from a group of items. Click (and release) on the menu to display all its items, then click on an item to select

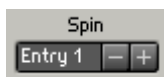
it. Switch and List modules can be displayed as drop-down menus (**Properties** dialog, Appearance page).

Text Panel (Switch and List modules)



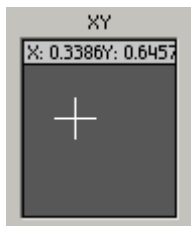
In a text panel, you select one item from a group of items. If there are more items than can fit in the text panel, a scrollbar is displayed. Click on an item to select it. Switch and List modules can be displayed as text panels (**Properties** dialog, Appearance page).

Spin (Switch and List modules)



In a spin control, you select one item from a group of items. To select an item, click the +/- buttons, or click the spin text box and drag your mouse up or down. Switch and List modules can be displayed as spin controls (**Properties** dialog, Appearance page).

XY



XY controls two parameters at once. Click within the XY field (as set by Size X and Size Y in the Properties dialog) and drag up/down to control the Y parameter, or left/right to control the X parameter.

Custom Controls



Splitter is an instrument from the REAKTOR Library

It is possible to create sophisticated custom panel controls in REAKTOR: knobs and faders with beautiful handmade skins, great-looking XY modules that send values to many input ports simultaneously, internal-connection or sequencer driven automation that changes knob/fader settings in real-time. The design and behaviour of custom controls is defined by the instrument creator. Please refer to a specific ensemble's documentation (or, better yet, reverse-engineer the ensemble) to learn more about its custom controls.



GoBox is an Ensemble included in the library

19.2. Using Keys to Change Control Settings

Faders, knobs, and switches can be controlled with keys on a computer keyboard. Note that you must select (click on) the fader, knob, or switch to enable the keys to work.

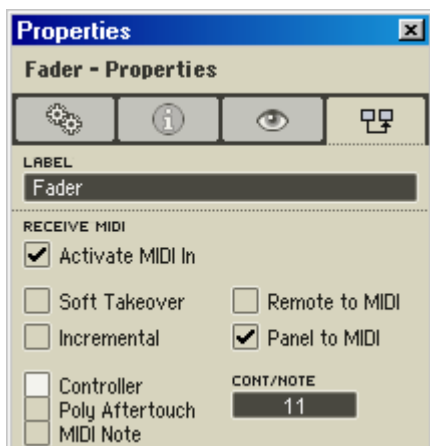
The **↑/↓** and **PgUp/PgDn** keys change fader and knob positions:

Key: Value Change:

- **↑** +Step
- **↓** -Step
- **PgUp** +10 × Step
- **PgDn** -10 × Step

The **↑/↓** keys toggle through a switch's options (buttons, menu items, etc.).

19.3. MIDI Control



Properties dialog of a fader, Connection page

If **Activate MIDI In** is enabled in the **Properties** dialog (Connection page) of a panel control, the control can be operated via MIDI.

- **Controller:** When enabled, MIDI Controller messages (received from an external MIDI device, or internally from within REAKTOR) operate the panel control.

- **Poly Aftertouch:** When enabled, MIDI Poly Aftertouch messages (from an external MIDI device, or from REAKTOR) operate the panel control.
- **MIDI Note:** When enabled, MIDI note messages (from an external MIDI device, or from REAKTOR) operate the panel control. The velocity determines the control position.
- **Cont/Note:** The number of the MIDI controller (if Controller is enabled) or MIDI note (if **Poly Aftertouch** or **MIDI Note** is enabled) whose MIDI messages are used to operate the panel control. You can set the Cont/Note value manually, for **Cont** you can use **MIDI Learn** (see below) to set it automatically.


Fader and knob controls change their positions according to the received MIDI messages.

When using MIDI to operate a button control, you'll find that it turns on only when the received MIDI Controller or Poly Aftertouch message has a value greater than 63. You can also use Note On/Off messages to operate buttons.

When using MIDI Controller or Poly Aftertouch messages to operate a switch or list control, the switch/list selects the option that corresponds to the received MIDI value. The range of possible values (0 to 127) is divided into equal-sized regions according to the number of switch/list options. For example, a switch with four inputs (i.e. four options) would have the following four regions: 0-31, 32-63, 64-95, and 96-127. Note that 0 always selects the lowest switch/list option, and 127 always selects the highest.

MIDI Learn

The **MIDI Learn** function is switched on with the corresponding 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 MIDI-controlled, click on the  **MIDI 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 enable this entry in the **Properties dialog** of controls or MIDI source modules if you want to control them using a MIDI Controller that sends incremental values.

Soft Takeover

When a fader or knob is 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 undesirable. Such jumps can be avoided by enabling **Soft Takeover** in the Properties dialog 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** is enabled in the Properties dialog of a panel control, all control changes (knob movements, button clicks, etc.) are translated to MIDI messages and output by REAKTOR to the MIDI output port(s) specified in the Audio Setup dialog, MIDI page.

When **Remote to MIDI** is enabled, the MIDI events that are received by the control (when **Activate MIDI In** is on) are also sent to the MIDI output port(s). Take care, however, when connecting to a sequencer that sends MIDI messages to REAKTOR and also receives MIDI messages from REAKTOR. In such cases, a feedback loop may result.

19.5. Customized Panels

It is possible to create fully customized panels in REAKTOR. You can design background bitmaps, displays that react to user input, even your own custom controls. (See above, **Custom Controls**.) You can add rectangular bitmaps to the panel, or use alpha-channel transparency to add arbitrarily shaped bitmaps. You can use the **Snap Value** module to save the current setting of a custom control with a snapshot. And so on.

Customized fader

REAKTOR allows you to use 24-bit Bitmap (*.bmp) and 32-bit uncompressed Targa (*.tga) images in many places: as instrument and ensemble panel backgrounds, as instrument and primary macro structure icons, as panel control (e.g. knob, fader, etc.) skins, and in **Multi Display**, **Poly Display**, **Picture**, and **Multi Picture** modules.

The advantage of using a Targa image is that it enables you to add transparency to the image (see below, Transparency).

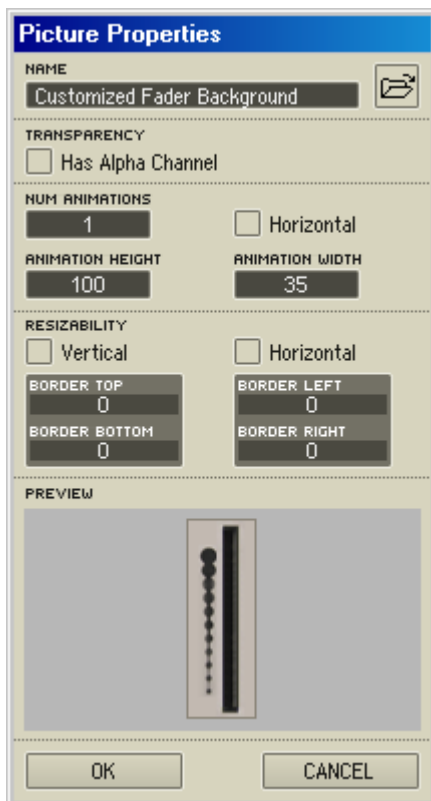
Picture Properties Dialog



Properties dialog (Appearance page) of a fader

You can load a picture using the drop-down **Select Picture** (or **Object Picture**, **Background Picture**, **Structure Icon**) menu in the Properties dialog (Appearance page) of any REAKTOR object that can display pictures: instrument, macro, Picture and Multi Picture modules, etc. To load a picture file from disk, select **Open from File...**; to load a picture from memory, select it by name from the lower half of the menu.

Opening a picture file from disk opens the **Picture Properties** dialog. It is in this dialog box that you make all relevant picture settings.



Picture Properties

Note: All of the pictures that are loaded in an ensemble are available to all the ensemble objects (in their drop-down picture menus) that can display pictures. Using the same picture more than once does not appreciably increase an ensemble's memory needs. So, as long as you work with a relatively small number of picture files, you can use them as much as you want without dire RAM consequences.

Once a picture has been loaded into an object, you can open its **Picture Properties** dialog by selecting **Picture Properties...** from the object's drop-down picture menu (as discussed above). Bear in mind that any changes you make to the picture will apply to all instances of the picture in the ensemble.

The **Picture Properties** dialog has five sections: **Name**, **Transparency**, **Animation**, **Resizeability**, and **Preview**. Let's look at these, one by one.

Name

Name renames the picture for internal use within REAKTOR. (**Name** doesn't change the name of the picture file stored on your hard drive.)

Transparency

Has Alpha Channel enables/disables a transparency mask using the picture's alpha channel. Alpha channels are available for Targa (*.tga) images, but not for Bitmaps (*.bmp). If a Targa image has an alpha channel mask (a task for the creator of the image), enabling **Has Alpha Channel** makes the unmasked portion of the image transparent (invisible). You can use this to display round custom knobs, panel overlays with blank spaces for REAKTOR panel controls, or any other irregular (non-rectangular) image.

Animation

The Animation part of the Picture Properties dialog enables you to break a single multipart image into a number of separate sub-images (frames). Each frame can be displayed by its index number: 0, 1, 2, ..., N-1 (where N is the total number of frames in the animation).

Animation frames are typically equal-sized vertical "slices" of a single very tall image; e.g. 128 instances of a knob image stacked one beside each other and on top of the other, with each successive instance (from left to right and top to bottom) showing the handle a bit closer to the maximum state position of the knob. Display all these frames in order, and you've got an animation of a knob whose handle is moving from its minimum to maximum position.

You can also use horizontal frames, by enabling the **Horizontal** option, but we recommend sticking with vertical frames, because REAKTOR must work a bit harder to process horizontal frames.

- **Num Animations:** Sets the number of animation frames you want to break the picture into.
- **Animation Height:** In vertical mode (i.e. when **Horizontal** is disabled), **Animation Height** sets the height (pixels) of each frame.
- **Animation Width:** In horizontal mode (i.e. when **Horizontal** is enabled), **Animation Width** sets the width (pixels) of each frame.
- **Horizontal:** Toggles between horizontal mode (**Horizontal** enabled) and vertical mode (**Horizontal** disabled).



Picture Properties dialog of a skin bitmap for a knob with its animation

In vertical mode, the **Num Animations** and **Animation Height** values are interdependent. When you set one, the other automatically sets itself accordingly. For example, if your picture is 4000 pixels tall, and you set **Num Animations** to 40, **Animation Height** automatically sets itself to 100 (pixels): $4000 / 40$. Or, if you set Animation Height to 50 (pixels), Num Animations sets itself to 80: $4000 / 50$.

In horizontal mode, the Num Animations and Animation Width values are similarly interdependent.

Resizability

The **Resizability** part of the Picture Properties dialog serves two main purposes. It reduces the size of ensemble files (*.ens) by enabling small pictures to be tiled so that they can fill large panel areas (i.e. instrument panel backgrounds). And it uses scaling to enable pictures (e.g. fader skins) to fit their associated objects (e.g. panel controls).

- **Vertical**: enables vertical tiling and scaling.
- **Horizontal**: enables horizontal tiling and scaling.
- **Border Top, Border Bottom, Border Left, Border Right**: cause a tiled picture to overlap itself in places. You can, for example, create a panel control of which a part does not change in form or color on its X or Y axis (see the example of the knob control below, which uses the **Resizability** option while Horizontal is activated).

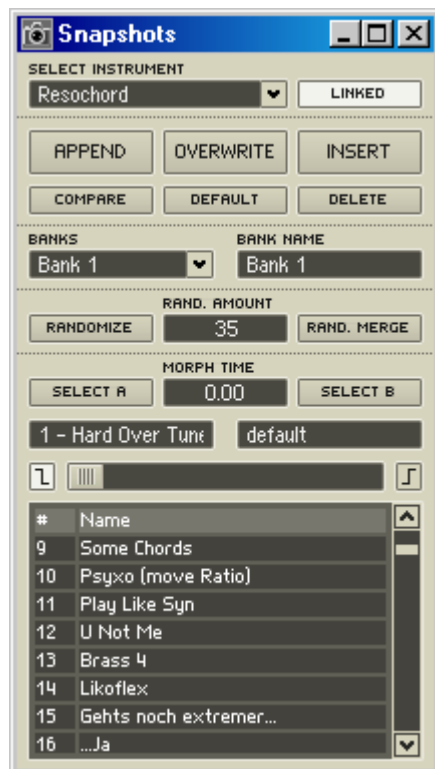


Left: Original picture. Right: Picture enlarged using the Resizability. The text was added using the Text module.

Preview

Preview displays a preview of the picture with the current Picture Properties dialog settings applied to it.

20. Snapshots



The Snapshots window

Snapshots (aka patches, programs, presets) enable you to store and recall an instrument's sounds. When you create a snapshot, the current settings of all the instrument's panel controls (knob/fader positions, list box and switch settings, button states, etc.) and MIDI controllers are stored in the snapshot. When you recall a snapshot, all the instrument's controls are restored to the settings they were in when the snapshot was originally created. Each REAKTOR instrument can store up to 2048 snapshots: 16 banks x 128 snapshots per bank.

Note: Ensembles can have snapshots too. See below, Linking Snapshots.

Control ID Numbers

Every REAKTOR panel control has a unique ID number, as displayed in the **ID For Snapshot Files** field of the control's Properties window (Function page).



ID For Snapshot Files field in an Properties window, Function page

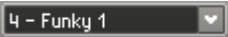

Warning: Do not change these control ID numbers!

REAKTOR lets you change them. But this is not generally a very useful (or smart) thing to do. Snapshots assign specific values to specific control ID numbers. For example, Snapshot1 might assign the value .5 to a knob with the ID number 21, the value .75 to a fader with ID 22, and so on. If you change the knob and fader ID numbers, it will change the values that Snapshot1 assigns to them. Thus your beautifully sculpted snapshots will all be broken.

Tip: If you load a snapshot file into a different instrument than it was created for, the snapshots will assign quasi-random values to the new instrument's controls (provided that the two instruments use the same approximate range of control ID numbers). Experimentalists might use this to coax new sounds out of old instruments.

Recalling Snapshots

There are three different methods you can use to recall an instrument's snapshots:

- You can use your mouse to select snapshots from the  **Snapshots** drop-down menu in the instrument panel header. If you click on the menu, you can use your computer keyboard's Up/Down arrow keys to select previous/next snapshots.
- You can select snapshots from the **Snapshots** window (**View->Show Snapshots**, or **F6** or  **Snapshot Button** in the Ensemble panel toolbar). If you do, make sure that the desired instrument is selected in the **Select Instrument** list box at the top of the Snapshots window.
- You can select snapshots by issuing MIDI Program Change messages

from a MIDI keyboard (or other MIDI controller). For this to work, the **Recall by MIDI** option must be enabled in the instrument's Properties dialog (**Function Page**). The MIDI Program Change message selects a snapshot by its number (1-128): MIDI Program Change 0 selects snapshot number 1, MIDI Program Change 1 selects snapshot 2, and so on.

Tip: REAKTOR builders, you can use the Snapshot module to recall, store, randomize, and morph snapshots.

Linking Snapshots

By default, snapshots are stored and recalled independently for each instrument. For example, say you have an ensemble that contains two instruments, Inst1 and Inst2. Selecting a new Inst1 snapshot does not select a new Inst2 snapshot, and vice versa.

Sometimes this is just what you want. Other times, you might want to select snapshots for multiple instruments at the same time. You can do this by linking snapshots:

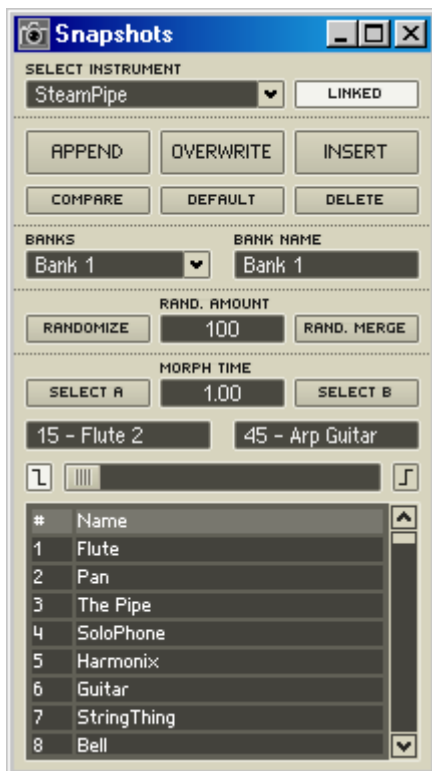
1. Use the Panelset bar to display the ensemble panel and all instrument panels in the Ensemble Panel window.
2. Enable each instrument's **Recall by Parent** option (**Properties** dialog, **Function page**).
3. Select an existing ensemble snapshot, or create a new one and select it.
4. Select the desired snapshot for each instrument.
5. Save the ensemble snapshot. (For help, see below, Managing Snapshots.) From now on, when you select this ensemble snapshot, it will cause all the instruments to select the snapshots you specified for them in step 4.

Managing Snapshots

You can manage (create, save, delete, etc.) snapshots in the Snapshots window. To open this window:

- Select **View->Show Snapshots** from the main menu.
- Or press **F6**.
- Or XP: Right-click /OS X: Ctrl.+click) on an instrument panel header and select **Snapshots** from the context menu.

- Press the  **Snapshot Button** in the Ensemble panel toolbar.



The Snapshots window

In the **Select Instrument** drop-down menu, you select the instrument whose snapshots you want to manage. You can also select the ensemble to manage ensemble snapshots.

If the **Linked** option is enabled, the Snapshots window automatically shows the snapshots of the instrument that is currently selected in the Ensemble Panel window. If **Linked** is off, you must use the **Select Instrument** menu to manually choose which instrument's snapshots to display. Most experienced users work with **Linked** on.

Below the **Select Instrument** menu is a set of six buttons for managing snapshots:

- **Append** saves the current instrument/ensemble settings as a snapshot to the first empty slot in the snapshots list. If the current snapshot bank is

full, **Append** saves the snapshot to the first empty slot in the next bank. If there are no more empty snapshot slots, **Append** does nothing. (Each instrument/ensemble can have a total of 2048 snapshots: 16 banks x 128 snapshots per bank.)

- **Overwrite** replaces the selected snapshot with the current instrument/ensemble settings. Note that when you overwrite a snapshot, you lose its original settings.
- **Insert** inserts the current instrument/ensemble settings as a new snapshot directly after the selected snapshot. Note that this can cause snapshots to move from the current bank to the next bank.

Important: You need to click the **Append**, **Overwrite**, and **Insert** buttons *twice* for them to work correctly. The first click lights the button and places a blinking cursor in the appended, overwritten, or inserted snapshot, giving you the opportunity to type a name for it. The second click un-lights the button and saves the appended, overwritten, or inserted snapshot. Remember to click twice! If you forget the second click, you might end up doing something very different than you intended.

- **Compare** compares the current instrument/ensemble settings with the original settings of the selected snapshot. (See Comparing, below.)
- **Default** changes the current instrument/ensemble settings to their default values (as specified in each instrument/ensemble control's Properties dialog, Function page). Note that clicking **Default** changes the selected snapshot's settings, but does not save the changed snapshot. To do this, you must use **Overwrite**.
- **Delete** deletes the selected snapshot(s). Note that **Delete** creates holes (empty slots) in the snapshots list; you can use the Banks menu's Sort command to remove these holes. (See below, Banks Menu.)

Renaming and Copying Snapshots

To rename an existing snapshot in the Snapshots window:

- Double-click on the snapshot, type the desired name, and click on the **Enter** key to save the renamed snapshot.
- Or select the snapshot, click the **Overwrite** button, type the desired name, and then click **Overwrite** a second time to save the renamed snapshot.

To copy an existing snapshot in the Snapshots window:

- Select the snapshot, click the **Append** button, rename the appended snapshot if desired, and then click **Append** a second time to save the

appended snapshot. Note that this copies the snapshot to the first empty slot in the snapshots list.

- Or select the snapshot, click the **Insert** button, rename the inserted snapshot if desired, and then click **Insert** a second time to save the inserted snapshot. Note that this copies the snapshot to a new slot below the originally selected slot in the snapshots list.

Comparing Snapshots

The **Compare** button is used for two main tasks:

- To compare a snapshot with a modified version of the same snapshot. Listening back and forth between the original and modified versions can help you create better snapshots.
- To compare two different snapshots.

The theory behind Compare is simple. The modified (or different) snapshot is stored in the Compare buffer, and the Compare button is used to toggle between the original snapshot and the modified (or different) snapshot.

To compare a snapshot with a modified version of the same snapshot:

1. Select a snapshot in the Snapshots window.
2. Make sure the **Compare** button is off (unlit).
3. Modify the snapshot control settings as desired.
4. Click twice on the **Compare** button; the first click turns it on (lit), the second off (unlit). The modified snapshot is now stored in the Compare buffer.
5. Use the **Compare** button to toggle between the original and modified snapshot versions.
6. Repeat steps 2-5 for further modifications.

To compare two different snapshots:

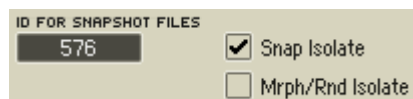
1. Select a snapshot in the Snapshots window.
2. Select another snapshot. The first snapshot is now stored in the Compare buffer.
3. Use the **Compare** button to toggle between the two snapshots.

Tip: If you are modifying a snapshot and accidentally select a different snapshot, you can recover your modifications by clicking on Compare right away (before you make any changes to the new snapshot)

Snap Isolate

When you select a snapshot, the panel controls and MIDI controllers jump to their positions as stored in the snapshot. In some cases, you might not want this to happen. For example, you might create a sequencer with a BPM (tempo) knob whose settings you want to be snapshot-independent.

To prevent a panel control or MIDI controller from jumping to its snapshot-designated position, enable the **Snap Isolate** option in its Properties dialog (Function page).



Snap Isolate in an Properties window, Function page

Banks Menu



Banks menu in the Snapshots window

The Banks menu is divided into two parts. The upper part is used for selecting banks (of snapshots). The lower part is used for managing (creating, sorting, cloning, etc.) banks.

- To select an existing bank, click on its name in the upper part of the Banks menu. The new bank's snapshots will be displayed in the snapshots list.

- To rename an existing bank, select it, and type the desired name in the Bank Name field (to the right of the Banks menu).

The lower part of the Banks menu offers these commands:

- **New** creates a new, empty bank and saves it to the first free slot in the banks list. For example, if an instrument had a Bank 1 and a Bank 3, **New** would create a Bank 2; if it had a Bank 1, 2, and 3, **New** would create a Bank 4; and so on. Each instrument/ensemble can have up to 16 banks (with up to 128 snapshots per bank).
- **Sort** sorts the selected bank's snapshots by number, and removes all <empty> slots (holes) from the snapshots list.
- **Init** initializes all snapshot. This erases all snapshots!
- **Clone** creates a copy of the selected bank and saves it to the first free slot in the banks list.
- **Save** saves the selected bank's snapshots to a snapshot file (*.ssf).
- **Load** loads the snapshots from a snapshot file (*.ssf) into the selected bank. You will be given the option to write these new snapshots over the bank's current snapshots (thus deleting the current snapshots), or to append the new snapshots to the end of the bank's snapshots.
- **Delete** deletes the selected bank.

Tip: If you delete a bank by mistake, don't panic! Simply use REAKTOR's **Undo** command to un-delete it.

Randomizing Snapshots

The Snapshots window provides several controls that you can use to add a degree of randomization to your snapshots:



Randomization row in Snapshots window

- Clicking on the **Randomize** button randomizes all of the selected instrument's panel controls, except those whose **Random Isolate** option is enabled (Properties dialog, Function page). (See below, Tip.)
- The value in the **Rand. Amount** field (0-100, which corresponds to 0%-100%) determines the maximum amount of randomization that the **Randomize** button can deliver. Clicking on **Randomize** can change a

control's current setting up to **+/- Rand. Amount %** of the control's range. For example, if a knob with a range of -1 to 1 is set to its middle point (0), and **Rand. Amount** is set to 25 (25%), clicking on **Randomize** can change the knob's value to anywhere from -.5 to .5 (0 +/- (25% of 2)). If the knob is set to -.5, and **Rand. Amount** is set to 50 (50%), clicking on **Randomize** can change the knob's value to anywhere from -1 to .5 (-.5 +/- (50% of 2)). Note that a control can never be randomized to a value beyond its Min/Max range.

- The **Rand. Merge** button works in conjunction with the **Select A** and **Select B** buttons that are used to select snapshots A and B for morphing (see below, Snapshot Morphing). Clicking on **Rand. Merge** spawns a “child” snapshot whose panel-control values are all randomly placed between their values in snapshot A and snapshot B. The degree of randomness is determined by **Rand. Amount**. If, for example, **Rand. Amount** is set to 50 when you click **Rand. Merge**, the child snapshot's control values will all be exactly halfway between their values in snapshot A and snapshot B. If **Rand. Amount** is set to 100, the child snapshot's control values can all end up anywhere between their values in snapshot A and snapshot B. And so on.

Tip: To make a panel control immune to randomization, enable its **Random Isolate** option (Properties dialog, Function page); the **Randomize** button will have no effect on the control. You can use this technique, in conjunction with the **Rand. Amount** value, to limit the amount of randomization an instrument receives.

Morphing between Snapshots

The Snapshots window provides a flexible set of controls for morphing between snapshots; i.e. changing an instrument's panel-control settings gradually (over a period of 0-60 seconds) from their values in one snapshot to their values in another snapshot.





The morphing section in the Snapshots window

Here's how you do it:

1. Set your desired morphing time in seconds (0-60) in the **Morph Time** field. This is how long it will take the controls to morph (move) from their current settings to their new settings.
2. Click the **Select A** button to turn it on (lit), and then select your desired snapshot A from the snapshots list.
3. Click the **Select B** button to turn it on, and select snapshot B.
4. Now you're ready to morph: Move the horizontal **Morph** slider to a new position (full left = 100% snapshot A, full right = 100% snapshot B, midway = 50% A and 50% B, and so on). The instrument's controls will move from their current settings to the settings specified by the new **Morph** slider position over the number of seconds specified in the **Morph Time** field.

Note: Shorter Morph Time values decrease the delay between changing the Morph slider position and having the panel controls complete their morphs. Longer Morph Time values increase this delay.

Gradual, incremental change between two states (i.e. two snapshots) is the basis of morphing. Because button and switch settings cannot be changed gradually, REAKTOR does not let you morph them. Therefore, before you begin to morph, you must decide whether to use the button/switch settings from snapshot A or snapshot B. Here's how:

- To use the snapshot A button/switch settings, click the  button to the left of the Morph slider to turn it on (lit).
- To use the snapshot B settings, click the  button to the right of the Morph slider to turn it on (lit).

21. Sampling and Resynthesis

In this chapter, you will learn all about REAKTOR's very powerful sampling and resynthesis capabilities.

21.1. Sample Management

Sample Files and RAM

REAKTOR's sampler modules can use mono or stereo .wav or .aif/.aiff sample files of any sample rate and bit depth. If a sample file contains any loop or keyboard-allocation information, REAKTOR finds and processes it.

Before a sampler module can use a sample file, the entire file needs to be loaded into actual – not virtual (see below) – RAM memory. Regardless of the bit depth of the sample file (8-bit, 16-bit, etc.), REAKTOR converts the audio to 32-bit for internal processing. One minute of stereo 32-bit audio at a typical REAKTOR sample rate of 44100 uses 20 MB of RAM. Thus large sample files can use huge amounts of RAM.

Systems that use virtual RAM (the default in Windows, but not in OS X) can allocate a great deal more RAM than is actually available; hence the term “virtual”. In such systems, REAKTOR does not display an error message when you load a sample file which is so large that it requires more actual RAM than is available. Instead, an error message appears later, informing you that the CPU is overloaded.

This error message appears because REAKTOR cannot access the sample audio 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 artifact caused by interruptions of the audio data stream. This situation, particularly undesirable during live performances, can only be avoided by adding more RAM to your system.

Multiple Use of Identical Samples

If one sample is used by several sampler modules in an ensemble, all the modules access the same sample as it is stored in RAM. This means that you can load the same sample in as many sampler modules as you like without increasing the RAM usage by any appreciable amount. Similarly, the number of polyphonic voices used by a sampler module is irrelevant as far as RAM is concerned.

REAKTOR identifies a sample using the pathname (directory and name) of the sample file from which the sample was loaded. When a new sample is loaded, REAKTOR searches through the list of already loaded sample files. If it finds one with the same pathname, REAKTOR reuses it instead of loading the new sample file.

Missing Samples

By default, REAKTOR saves the pathnames of the samples that you load into a sampler module, not the actual sample files. If any of these sample files are deleted, renamed, or moved after the ensemble has been saved, they will not be available when the ensemble is reopened.

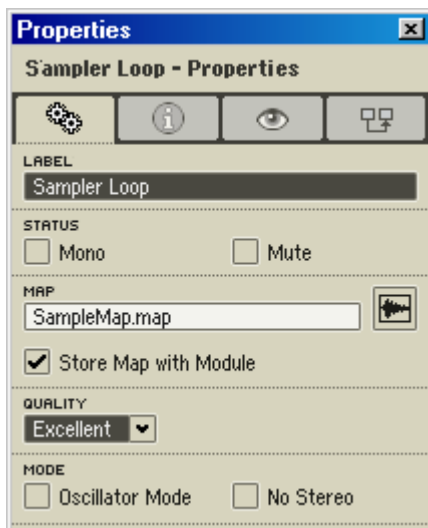
If this happens, you will get an error message upon opening the ensemble informing you that the sample files are missing. These samples are labeled as **missing** in the File column of the Sample Map Editor. To relocate or replace missing samples, select the sample in the Sample Map Editor, and click on the **Replace** button.

Storing Samples with Modules

You can avoid the potential problem of missing samples by turning on **Store Map with Module** in the sampler module's Properties dialog (Function page). Doing so saves a copy of all the sample files loaded in the module's sample map with the module (and, by extension, with the ensemble). When you reopen the ensemble, REAKTOR will load the module's sample files directly from the ensemble file, rather than loading the original files from their individual locations on your hard disk.

The advantage of storing samples with a sampler module: you can be sure that these samples will still be there when you open the ensemble, or when another user, on a different computer, opens the ensemble. The disadvantage: depending on the sample files' sizes, you might end up with a very large ensemble file size.

Tip: A common mistake is to save a sampler ensemble without having turned on **Store Map with Module** in the sampler module(s), and then to share this ensemble with other REAKTOR users (e.g. in the REAKTOR User Library). When another user opens the ensemble, the sample files will be missing. Simple fix: Remember to turn on **Store Map with Module** in all sampler modules before saving an ensemble!



Properties dialog of a sampler module, Function page

Sample Analysis

Some sampler modules (e.g. Grain Resynth, Grain Pitch Former, Beat Loop) perform real-time resynthesis of sample files. When you load a sample in a resynthesis module, REAKTOR analyzes the sample file while it is loading. This analysis takes about as long as the duration of the sample. To prevent having to repeat the same analysis every time this file is loaded in a resynthesis module, REAKTOR displays a message dialog asking if you want to save the analysis data into the sample file. Read this message carefully before deciding what to do; it explains the potential danger of embedding analysis data into your sample file. Note that REAKTOR can only save analysis data into a sample file which is not write-protected.

Note: Once REAKTOR has saved analysis data into a sample file, it assumes that this file has been fully analyzed. This can cause problems. For example, if you modify an analyzed sample file, then load it into a resynthesis module, REAKTOR thinks that the file is still fully analyzed (though it isn't, because you modified it). To get around this, simply rename an analyzed sample file when you modify it. Then, when you load it into a resynthesis module, REAKTOR will re-analyze it.

Sample-Editors

REAKTOR does not have its own internal sample editor. To edit sample files, you have to use an external audio editor (e.g. Windows XP: Sound Forge, WaveLab, Audition, GoldWave etc; OS X: Peak, Spark XL, Audacity etc.). To facilitate this process, REAKTOR enables you to open a sample file in your chosen editor from within the Sample Map Editor window. To do this:

1. Tell REAKTOR where to find your sample editor by entering its pathname in the Preferences dialog: **System** ⇒ **Preferences** ⇒ **Directories: External Sample Editor**.
2. Select the sample file in the Sample Map Editor.
3. Select **Edit** from the drop-down Edit Sample List box to load the sample file in your external sample editor.
4. Edit your sample file as desired, and then save it.
5. Select the sample in the Sample Map Editor, and select **Reload** from the Edit Sample List box to reload the modified sample file.

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

21.2. Sample Maps

One common use of sample maps is to simulate the sound of acoustic instruments. This is usually done by assigning multiple samples (rather than a single sample) from the instrument to the instrument's pitch range. For example, rather than assigning a single clarinet sample to its entire three-octave range, you might assign a dozen samples to the range: one for the lowest three pitches in the range, another for the next higher three pitches, and so on, all the way to the highest three pitches. The further you stretch a sample (i.e. the more pitches you assign it to), the less natural the simulation will sound. So, to make an instrumental simulation sound natural, you need to use an appropriate number of samples.

Another use of sample maps is to trigger multiple samples with one key. Following are explanations for both uses.

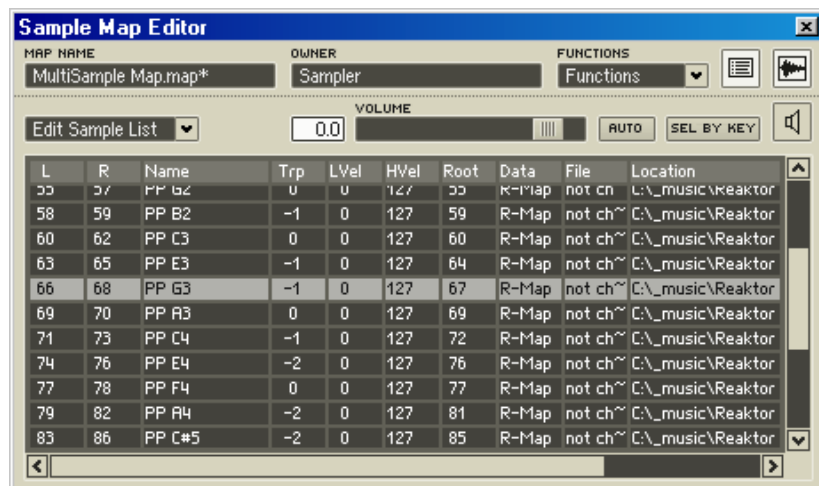
Multi-Sampling

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 assigned to multiple keys (pitches), the further it is transposed from the root key of the original sample, the less natural it sounds. This happens because the frequency spectrum (overtones) of the transposed sample does not match the spectrum of the acoustic instrument. That is why a human voice transposed an octave higher sounds so unnatural and silly. Samplers try to overcome this limitation by using multi-sampling, wherein each sample is only transposed a small amount from its root key.

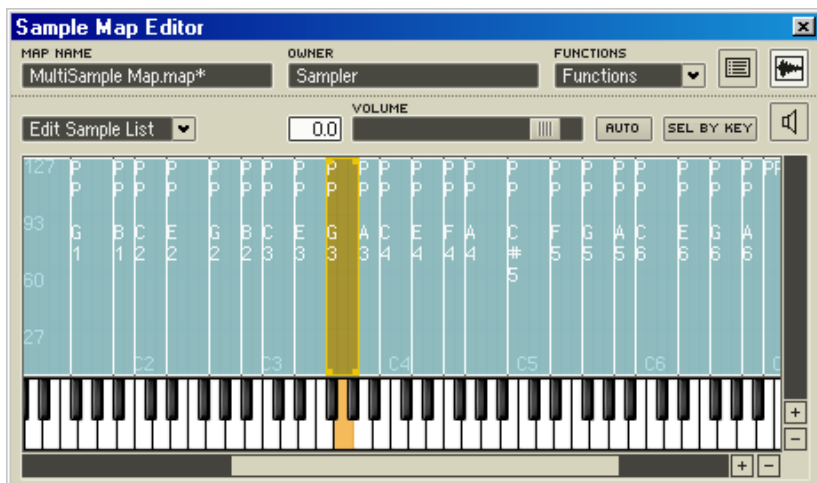
Assuming that a sample map contains multiple samples of the original sound, at least three parameters must be set (in the Sample Map Editor) to control each sample in the map:

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



Sample Map Editor showing a list of multiple samples

The optimum result is achieved when the Root is kept in the middle of the sample key zone, and the amount of transposition is kept to a minimum.



Part of the Sample Map Editor with a sample map containing multiple samples. The orange key shows the Root key position of the selected sample zone.

Every sampler module has a **P**(itch) input, which is used to select a sample from the map and to control the pitch 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 specified sample and pitch.

The sample module also has a **Sel**(ect) input. If connected, **Sel** overrides the sample selection of the **P** input, which then only sets the pitch. 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 (e.g., soft and hard). Then the dynamically different samples are triggered by different key velocities, resulting in a more expressive representation of the original sound.

Drum-Maps

As in Multi-Sampling, a "Drum-Map" uses the **Root Key** and left and right splits to determine the position of the samples in the map. Sometimes the **Root Key** of the original sample is not used or gets overly transposed. With Drum-Maps, it is possible to use any range of keys to a sample (these keys do not have to be anywhere near actual **Root Key** of the drum sample).

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 filename 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. Sample Map Editor


You use the **Sample Map Editor** to load, save, edit, map, and loop the sample files that are used in REAKTOR's various sampler modules: Sampler, Grain Resynth, Beat Loop, Sample Lookup, etc. Like the Properties dialog, the Sample Map Editor window can be kept open while you work, and its contents 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.

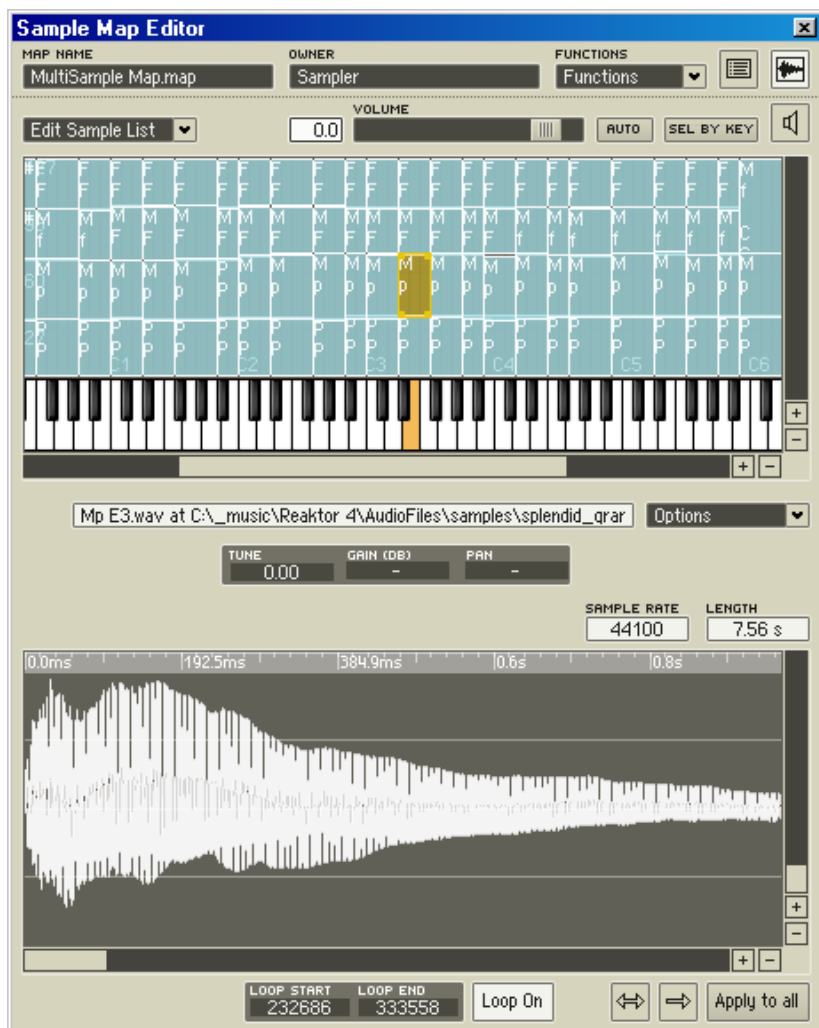
Opening the Sample Map Editor

There are four different ways to open the Sample Map Editor. These first two ways open the Sample Map Editor with no sample files loaded in it:

- Select **View->Show Map Editor** from the main menu.
- Or press the **F7** function key.

These next two ways are preferable, since they open the Sample Map Editor with the selected sampler module's samples loaded in it:

- Click on the  **Show Map Editor** button (waveform icon) in the sampler module's Properties dialog (Function page).
- Or Windows XP: Right-click / OS X: Ctrl.-click) on a sampler module's panel display and select **Open Map Editor** from the context menu.

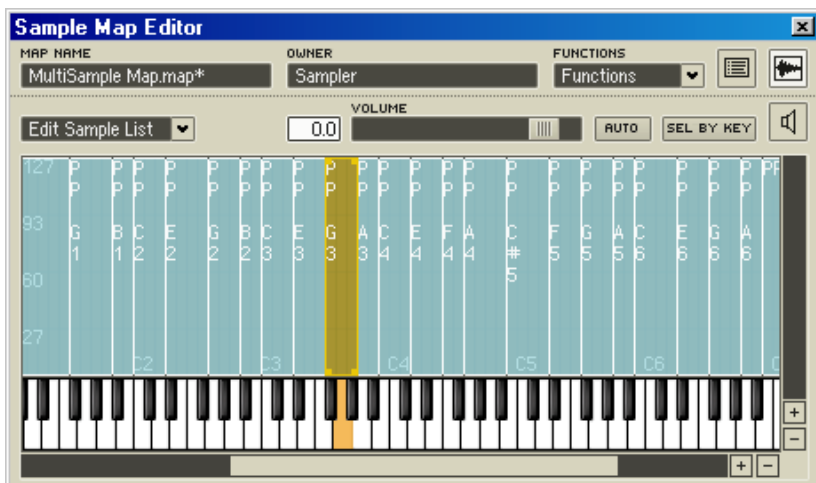


Sample Map Editor with a sample map containing multiple samples

Sample Map Editor Components

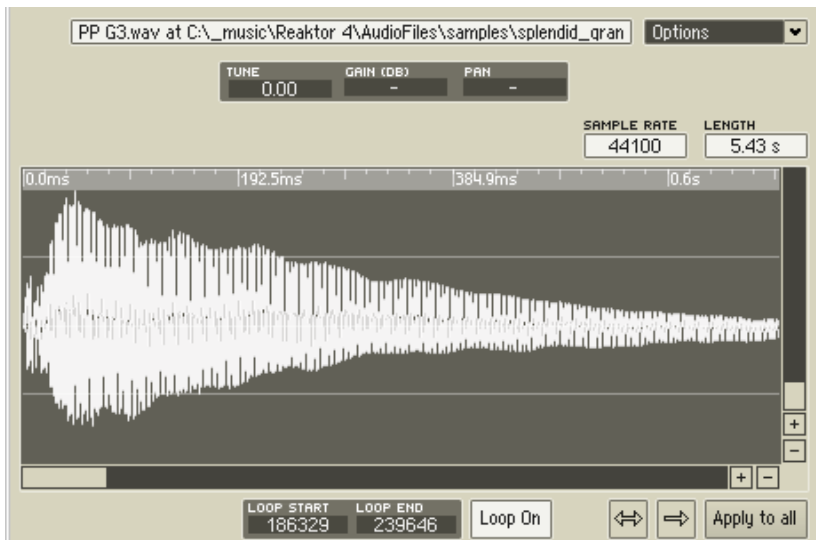
The Sample Map Editor window has two components:

- **Sample Mapper** (upper pane) - used for loading and arranging samples in the sample map.



The Sample Mapper section in the upper pane of the Sample Map Editor window

- **Loop Editor** (lower pane) - used for editing sample loops.



The Loop Editor section in the lower pane of the Sample Map Editor window

Let's take a closer look at both of these components.

Sample Mapper

The Sample Mapper contains these controls:

- **Map Name** field: Displays the name of the currently loaded sample map file. If no map file is loaded, “untitled Map” is displayed.
- **Owner** field: Displays the name of the sampler module whose map is displayed.
- **Functions** list box: Contains two commands, Remap to Single Keys and Set transpose to Null for All. For details, see Functions List Box below.



- **Map View** button (list icon): The Sample Mapper can display samples in list (text) or keyboard (graphic) view. You use the **Map View** button to toggle between these two views. For details, see Map List View and Map Keyboard View below.



- **Show/Hide Loop Editor** button (waveform icon): Shows and hides the Loop Editor. For details, see Loop Editor below.
- **Edit Sample List** box: Contains sample-editing commands. For details, see Edit Sample List Box below.

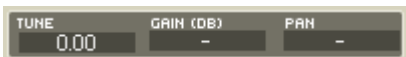


- **Sel by Key** button: When this option is enabled, incoming MIDI notes will cause the sample selection to change accordingly.



- **Auditioning** controls: Contains a set of controls for auditioning sample files. For details, see Auditioning Sample Files below.

- **Sample Name** field: Displays the pathname of the currently selected sample file.
- **Options** list box: Contains three commands, Auto-Move RootKey, Ignore RootKey When Loading From File, and Single Key Mode. For details, see Options List Box below.



- **Tune, Gain, Pan** fields: Display the tuning (in cents), the gain (in dB), and the pan position of the selected sample.



- The **Sample Rate** and **Length** displays show the

sampling rate for the selected sample and its length in milliseconds. These values cannot be edited.

Edit Sample List Box

The Edit Sample List box contains the following commands:

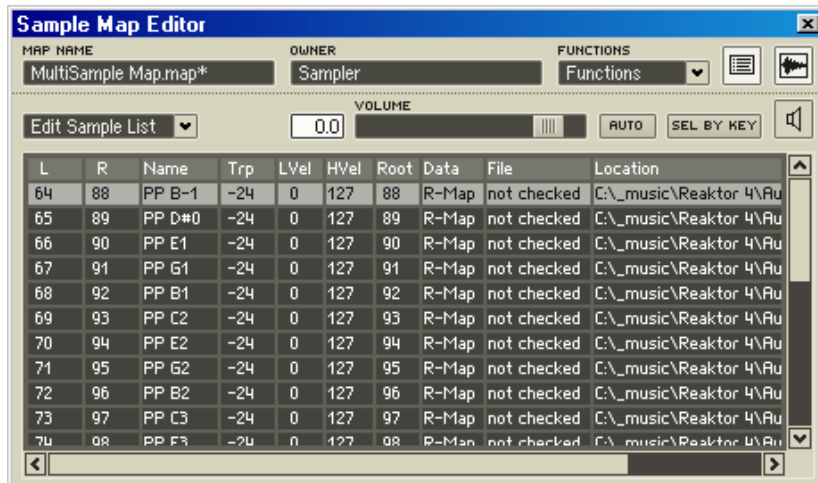
- **Add:** Adds a sample to the sample map.
- **Replace:** Replaces the selected sample in the map.
- **Save:** Saves the selected sample.
- **Delete:** Deletes the selected sample from the map. Note that this does not delete the sample from your hard disk.
- **Edit:** Opens the selected sample in the external audio editor specified in the Preferences dialog (Directories).
- **Update:** Reloads the currently selected sample. This is typically used to reload a sample file after having edited it.

Note that if no samples are selected, the **Delete**, **Edit**, and **Update** commands are applied to the entire map.

- **Load Map:** Loads a sample map file.
- **Save Map:** Saves the current sample map file. When you save a map file, REAKTOR asks if you would like to save the audio data along with the map. If you say **Yes**, copies of the sample files will be saved with the map; this results in a larger map file, but it also ensures that the samples will all be there if you load the map file in another sampler module. If you say **No**, pathnames of the sample files will be saved with the map; this results in a smaller map file, but it runs the risk of having missing samples if you load the map file in another sampler module.
- **Akai Import:** Opens the Akai Import window for importing sample maps from Akai CDs (S 1000-3000 format). See below for details.

Tip: To use the keyboard to cycle between samples in a map, press Tab (to cycle forward) and Shift + Tab (to cycle backward).

Map List View



The Sample Mapper section in the upper pane of the Sample Map Editor window, map list view

In Map List view, the sample map list (upper pane in the Sample Map Editor) displays ten columns of data for each sample. You can sort the list by the values in any column by clicking the column label: once to sort up, again to sort down.

You can edit the data values in all of these columns, except where noted:

- **L**: Left end of the key zone; i.e. the lowest MIDI note number that will play the sample. See below, **Trp**.
- **R**: Right end of the key zone; i.e. the highest MIDI note number that will play the sample.

L and **R** define a contiguous set of MIDI notes (keys on a MIDI keyboard) that will play the sample. This is clearly displayed in Map Keyboard mode (see below).

- **Name**: Name of the sample file (without the file extension). You cannot edit this value.
- **Trp**: Sample transpose value; i.e. the number of semitones the sample's **Root** value must be lowered or raised to reach its **L** value. **Trp = L - Root**. If you edit the **Trp** value, the **Root** value is automatically adjusted, but the **L** value stays the same.
- **LVel**: Low end of the velocity range; i.e. the lowest velocity that will play

the sample.

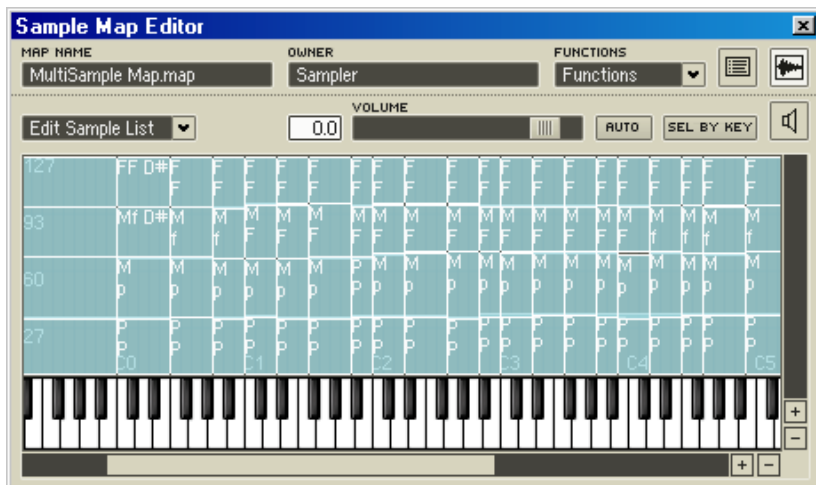
- **HVel**: High end of the velocity range; i.e. the highest velocity that will play the sample.

LVel and **HVel** define a range of velocities that will play the sample. You can use **LVel** and **HVel** to enable the same MIDI note to play different samples, depending on the note's velocity (volume).

- **Root**: The MIDI note number that will play the sample at its original (intended) pitch. (See above, **Trp.**)
- **Data**: You cannot edit this value.
- **File**: You cannot edit this value.
- **Location**: The location on your hard drive where the file is located. You cannot edit this value.

Note: When the key or velocity ranges of different samples overlap, you can see which sample is given priority in Map Keyboard view. In general, the most recently added sample always gets highest priority.

Map Keyboard View



The Sample Mapper section in the upper pane of the Sample Map Editor window, keyboard view

In Map Keyboard view, the sample map keyboard display (upper pane in the

Sample Map Editor window) contains a two-dimensional region representing each sample in the map and a keyboard graphic along the bottom. The position, as well as the horizontal and vertical size, determine which MIDI note events cause a 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 regions.

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

Functions List Box

Remap to Single Keys

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

Set Transpose to Null for All

Set Transpose To Null For All 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 left-most key in each region play the sample at its natural pitch. You can best see its effect by using it in the List view.

Options List Box



Below the sample map display is a text box for displaying the name and location of the selected sample. The options drop-down menu to the right of the text box has 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.

Auditioning Sample Files




The Sample Map Editor enables you to audition (pre-listen to) a sample file, once it has been loaded:

1. Select a loaded sample to audition.
2. Click on the **Play** button (speaker icon) to start/stop auditioning. Use the Volume fader to set the volume.
3. If you turn **Auto** on, auditioning starts automatically when you select a sample.

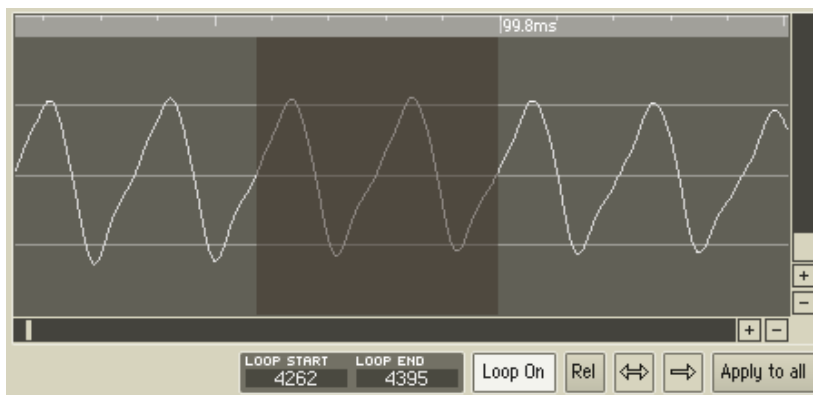
Loop Editor

The **Loop Editor** (located in the lower half of the Sample Map Editor) enables you to set and edit loops for individual samples. (If the Loop Editor window is

not displayed, click the  **Show/Hide Loop Editor** button (waveform icon) in the upper-right corner of the Sample Map Editor.) The reddish-brown area in the sample waveform display specifies the portion of the sample that will be looped.

To change the loop start and end points, drag the start/end edges as desired. To move the entire loop, drag it (by the middle) as desired. Note that you might need to change the zoom level of the waveform display to see the entire loop. To do this, use the Zoom In/Out buttons in the lower-right corner of the display.

Note that the loop start and end points are displayed in sample numbers in the Loop Start and Loop End boxes below the waveform. But you cannot change the start/end points by editing these boxes.



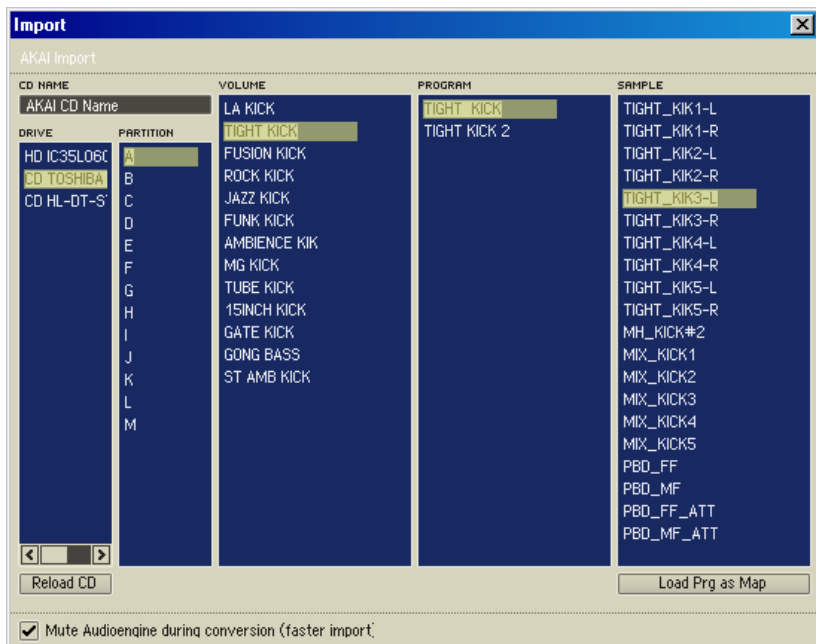
The Loop Editor section in the lower pane of the Sample Map Editor window

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

- **Loop On:** Enables/disables looping for the sample.
- **Rel:** Enables/disables Loop in Release mode. When this mode is enabled, samples continue looping after the gate signal has turned off (i.e. during the release stage of an ADSR envelope).
- **<->:** Enables/disables Alternating Loop mode. When this mode is enabled, the sample loop reverses direction at each end: The loop plays forward, then backward, then forward, then backward, and so on. This often produces a smoother loop with less of a hitch at its looping point.
- **->:** Enables/disables Reverse Playback mode. When this mode is enabled (i.e. when the arrow points to the left), the entire sample (not just the loop) plays backward.
- **Apply to All:** Applies all the current loop settings (i.e. of the four buttons above) to all samples in the map.

21.4. Akai Import

You can import Akai files from the Sample Map Editor or from a sampler module's panel (not structure) context menu. Select **Akai Import** to open the Akai Import window that shows the contents of an Akai-formatted CD loaded in your CD drive. If the CD contents don't appear, click the **Reload CD** button.



Akai Import window

The Akai Import window displays all of the Akai CD's partitions, volumes, programs, and samples, and enables you to convert selected Akai samples and programs to REAKTOR Map files (*.map). Converted files are stored in the folder specified by **Imported Files (Akai)** in the REAKTOR Preferences dialog (Directories page).

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

- Sample data
- Loop points
- Root key
- Pan

The following information is not preserved during conversion:

- Filter settings
- Envelope settings
- Velocity splits
- Gain

It is also possible to load Akai samples and programs directly into a sampler module by clicking the **Load Prg as Map** button (in the Akai Import window). You can save these samples as a Map file by selecting **Save Map** from the Edit Sample List box in the Sample Map Editor. Or you can save them with the ensemble by turning on **Store Map with Module** in the module's Properties dialog (Function page).

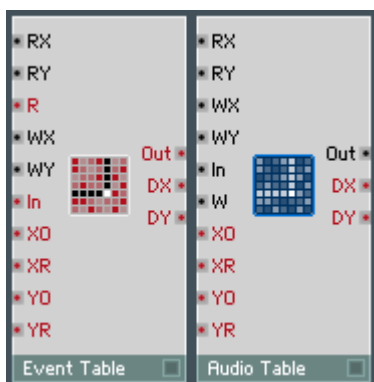
To speed up importing, turn the **Mute Audioengine During Conversion** option on. Note that doing this will stop all REAKTOR audio output until the importing is finished.

22. Table Modules

The Table modules allow for very flexible handling of events and audio data. 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 window which is identical for both kinds of module. Some general properties which are already described for many other modules are not covered again here.



The Event Table module and the Audio Table module

Function Page

Properties

Event Table - Properties

LABEL
Event Table

STATUS
☐ Mono ☐ Mute

INTERPOLATION
None

CLIP/WRAP X,Y
Wrap

☒ Backup Data With Module

FILE
<empty>

CLIENTS
-

X SIZE
empty

Y SIZE
empty

Set

VALUE

MAX	STEPSIZE	DEFAULT
0	0	0
MIN	NUM STEPS	
0	0	

DISPLAY UNITS
▼

X UNITS
▼

Y UNITS
▼

SAMPLES/SEC	BPM	TICKS/BEAT
0	0.00	0
SAMPLES/TICK	NUM. OF BEATS	
0	0	

Properties dialog of a Event Table module, 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. Fractional

values at the **RX** input are used to compute smooth transitions between table cells.

- **Y:** Interpolation between values is only used on the Y-axis. Fractional values at the **RY** input are used to compute smooth transitions 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.

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 were connected in 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 to create 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 (Y always equals 0).

You can save the data from a table as a file for reuse 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 the **Apply** button is pressed.

Note: If you reduce the number of cells, then any data contained in the removed cells will also be deleted.

Value

Min, **Max**, **Stepsize** and **Num Steps** work just like they do 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 for 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. The sample rate of a loaded Wav or Aiff file will automatically appear 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

Properties [X]

Audio Table - Properties

Icons: [Settings] [Info] [Eye] [Copy]

LABEL
Audio Table

VISIBLE IN VIEW A AND B

☒ Picture ☒ H Scroll Bar ☒ V Scroll Bar

SIZE X
384 Pixels

SIZE Y
128 Pixels

GRAPH
Line

VIEW PARAMETERS

☒ X Auto Fit ☒ Y Auto Fit ☒ Value Auto Fit

ALIGNMENT

X: [Slider] Y: [Slider]

GRID

Grid	Step	Sz1	Sz2	Sz3	Sz4
X	-	-	-	-	-
Y	-	-	-	-	-
Value	-	-	-	-	-

ENABLE GRID

☐ X ☐ Y ☐ Value

Buttons: [A] [B] [AB] [COPY A > B] [COPY B > A]

VISIBLE

☒ Label ☒ Value ☒ Visible ☐ Small Label/Value

Properties dialog of a Event Table module, Appearance page

Visible in View A and B

The settings in this area are global module settings and always apply 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 pixels.

Graph Format

This drop-down 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.
- **Line:** Values are drawn with a horizontal line and vertical lines are also drawn to connect the values.
- **Bar:** Values are drawn with a horizontal line, vertical lines are drawn to connect and the area underneath is filled with color.
- **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-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 it 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 that indicated by **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 that indicated by **Y Size** in the **Table** tab. **In display modes other than 2D, Auto Fit** has no effect.
- **Y Alignment:** When **Auto Fit** is off **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 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 that position. The units of the displayed values depend on the selected units in the Properties box under the **Table** tab.
- **Small Label/Value:** Tick this checkbox to set a smaller display size for the module label and value status bar in the panel.

22.2. Context Menu

When you (Windows XP) click the right mouse button / (OS X) hold Ctrl. and click the mouse button on a table graph, a special context menu for editing in the graph becomes visible. For ease and speed of navigation, we recommend becoming familiar with the keyboard shortcuts for editing operations. 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, set the value with which to draw by selecting the menu option **Set 2D Draw Value...** or pick an existing value by Ctrl-mouse clicking 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 cannot be changed 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 window.
- **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 command.

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 six graph display modes above in the description of the **Appearance** tab in the Properties window.

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' **Grid** tab. If 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 vertically symmetric axis in the middle of the selection.
- **Mirror Y:** In 2D mode, swaps the data between top and bottom using a horizontally symmetric axis in the middle of the selection.
- **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 to 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 window. This is the normally used mode. Unselect this option to draw at a finer resolution (regardless of 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 Windows XP: Ctrl+Right-click / OS X: \mathcal{H} +click in draw mode.

Cut, Copy, Paste

- **Copy** (Windows XP: **Ctrl+C** / OSX: \mathcal{H} +**C**): Copies the selected data into the copy buffer.
- **Cut** (Windows XP: **Ctrl+X** / OSX: \mathcal{H} +**X**): Cuts the selected data and puts it in the copy buffer.
- **Paste** (Windows XP: **Ctrl+V** / OSX: \mathcal{H} +**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 or right, or in all directions in 2D-mode. When everything is selected, you can drag around the whole graph.

Add

By holding the Windows XP: Ctrl key / OS X: \mathcal{H} 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 which way 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 (Windows XP: Ctrl+C / OS X: \mathcal{H} +C) and paste (Windows XP: CTRL+V / OS X: \mathcal{H} +V) apply to the selected data inside the table graph instead of the module itself. So when the panel is locked, you will move the data in the module instead of the module's representation on the panel.

23. “Classic Modular” Macro Collection

The Classic Modular macro collection is an assortment of high-level REAKTOR building blocks that 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. For instance, the Classic Modular macros include sampling, granular sampling, and sophisticated sequencers, 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, and the Analogic Filter Box were built in part out of elementary Classic Modular macros. Take some time to listen to these 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 inputs and outputs. That means that all output signals can be connected to any input without down or up scaling. This is the main difference from REAKTOR's other modules and macros in which the value range can vary depending on what type of unit a parameter controls (e.g., semitones, decibels, or milliseconds). For instance the pitch (P) input of an oscillator module expects values in the range of 0 to 127, while the amplitude (A) input expects values between 0 and 1.

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

The only exception to the rule 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) for controlling this parameter directly along with one or two modulation amount controls. For each modulation amount control there is a corresponding input with a similar name for connecting modulation sources such as LFOs and envelopes. The incoming 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 (such as 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 (which can be found in the “Auxiliary” folder of REAKTOR’s modules).

Some file names contain “Event” or “Audio”. The purpose of these macros is to scale, mix, invert and switch between modulation signals (such as LFOs and envelopes), and they are available in event and audio versions. Some names contain “Stereo” or “Mono”. The purpose of these macros is to mix, amplify, filter or distort ‘normal’ audio signals (in contrast to modulation signals), and they are available in stereo and mono versions.

The “Classic Modular” macros can be used to build polyphonic structures. This is one big advantage over analog modular systems, which 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 (which can be found in the “Auxiliary” folder of REAKTOR’s modules). Please note that all display modules such as meters, number displays, oscilloscopes, LEDs, 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, we recommend turning on the built-in peak detector with the “Peak” switch. Then the overall amplitude envelope will be 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, and volume.

Notes - Monophonic

Receives MIDI note messages and passes their pitch, gate and note on velocity values to the corresponding outputs in monophonic manner (i.e., 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 dialog 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 (i.e., the notes are assigned to different voices). The key range can be set in the Properties dialog 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 dialog 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 dialog 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 dialog 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 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. 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 region.

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 exceeds 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 dialog 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 determines 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 dialog 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 ($8 \cdot 2$) sequences are used as copy/paste and undo buffers the number of sequences must be at least 24 ($8 \cdot 3$).

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 lets 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 produces 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 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).

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 “PWM” (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.

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.

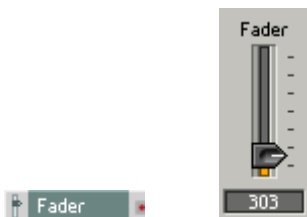
Dynamic ports management

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 in 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 processes 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 managment 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 do not recognize the panel element anymore and ignores it. Only modify this number if you know exactly what you are doing.

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.



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.

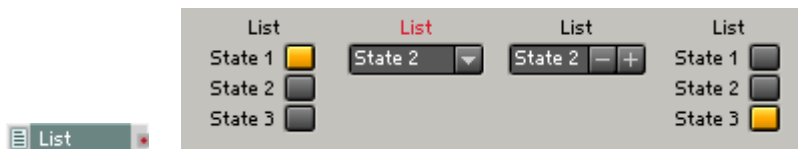
The button can be displayed in three sizes: **Small**, **Medium** and **Big**.

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.

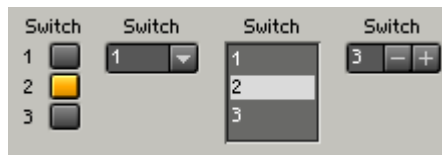
The **Size X** and **Size Y** fields are controlling the display size of the control element in the panel.

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 on the right side of the list entry panel display.

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.

LevelMeter

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

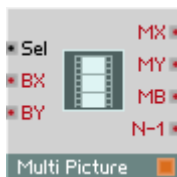


Allows loading a panel decoration bitmap from a picture file in TGA format (File extension *.tga). The bitmap module has no in- or outputs. The display size will be set to the size of the image. It is possible to modify the visible frame in pixels under Appearance/Size in the properties. The image cannot 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 light gray background and a darker gray border. The word "Text" is written in a small, black, sans-serif font in the center.

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.

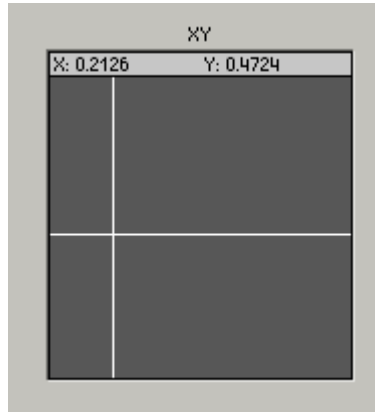
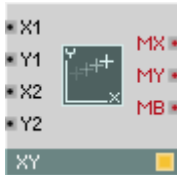
Multi Text

Panel

A small rectangular icon with a light gray background and a darker gray border. The text "Multi Text" is written in a small, black, sans-serif font. To the right of the text is a small square icon containing a document symbol.

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.



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.

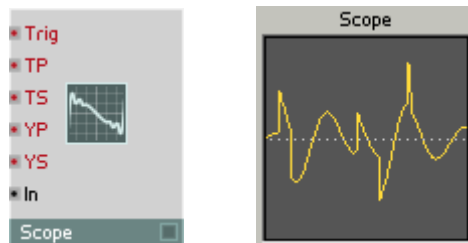
Everything drawn in the display fades out more or less slowly, depending on the value set for **Fade Time** in the Properties (maximum value is 99).

- **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.

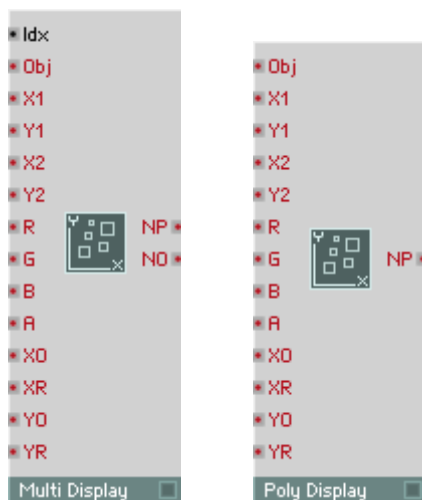
You would typically connect an **A to E Trig** module to the **Trg** input for triggering the Scope from an audio signal.

- **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.

Multi Display and Poly Display

Panel



Both the Multi Display and Poly Display modules enable the display and manipulation of multiple graphical objects (crosses, bars, pictures, animations, etc.). A range of parameters (including type, position, size and color) can be defined individually for each graphical object.

The major difference between two modules is that for the Multi Display module, the number of graphical objects is specified by the Number of Objects field in the properties, whereas for the Poly Display module, the number of graphical objects is specified by the number of voices in the instrument.

The advantage of Multi Display is that it can display any number of graphical objects, regardless of the number of polyphonic voices. Each object can then be addressed independently using the Idx input. In contrast, the Poly Display could be considered easier to program, as it doesn't require any index-pro-

cessing code, and all objects can be addressed simultaneously by virtue of parallel polyphonic processing. However, the number of objects is limited to the number of voices of the instrument.

The properties include a variety of options for customisation of the display, including background color and picture.

When used in conjunction with the Mouse Area and Snap Value Array modules, the Multi Display and Poly Display allow sophisticated custom interface elements to be constructed (sequencers for example).

All Multi Display inputs accept monophonic event signals. Idx also accepts monophonic audio signals.

- **Idx:** Index of the graphical element to address. The index is 1-based; i.e. the ID of the first element is 1 (not 0). Fractional values are rounded to the nearest integer. Where Idx values are outside the range of 1 to N, the behaviour depends on the Index Behaviour option in the properties. The stacking order of graphical objects is determined by their Idx values. Objects with lower Idx's appear on top of objects with higher Idx's. It is essential to set the Idx value before transmitting events to the other ports..
- **Obj:** Graphical object type, or picture frame selection.
 - 4: Cross.
 - 3: Line from X1, Y1 to X1, Y1 of the next object of the same type.
 - 2: Line from X1, Y1 to X2, Y2.
 - 1: Bar.
 - 0: Rectangle.
 - 1 ... NP: Picture index that specifies a single picture (1) or a series of pictures in an animation (1, 2, 3, ..., NP), where NP is the total number of pictures in the animation.

Fractional values are rounded to the nearest integer.

If Ignore Index Obj (in Properties) is on, all graphical objects are set to the same type.

- **X1, Y1:** Coordinates for the first corner of the graphical object area, or if the 'Center to X1, Y1' option is selected in the properties,

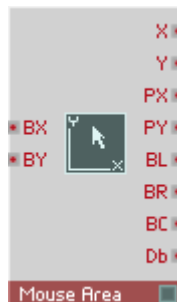
then these values define the coordinates of the centre of the object.

- **X2, Y2:** Coordinates for the second corner of the graphical object area.
- **R, G, B:** These inputs define the amount of Red, Green and Blue components of the object colour (range 0 to 1). If Ignore Index RGB (in Properties) is on, all graphical objects are set to the same color.
- **A:** Object transparency (0=fully transparent, 1=fully opaque).
- **XO, YO:** Lowest visible X and Y values, (i.e. for scrolling). Inputs to these ports override entries to the X Origin and Y Origin fields in the properties.
- **XR, YR:** Horizontal/vertical view range, (i.e. for zooming). Inputs to these ports override entries to the X Range and Y Range fields in the properties.
- **NP:** Output reporting the number of pictures in the animation.
- **NO:** Output reporting the number of graphical objects.

The Poly Display inputs and outputs are the same as those of Multi Display, except that Poly Display has no Idx input and no NO output. All Poly Display inputs accept polyphonic event signals, except for XO, XR, YO and YR, which accept monophonic event signals.

Mouse Area

Panel



The Mouse Area module detects and outputs mouse actions, including button clicks, mouse drags, and changes in position. The Mouse Area can have its own outline and fill color, and can therefore be used as a panel interface

element by itself. However, it most typically used as an invisible (transparent) overlay on top of other modules, especially the Multi Display and Poly Display modules, in order to build highly customised interface elements.

The Mouse Area X and Y outputs can either operate in absolute or incremental mode (as selected in the properties). When Incremental Mode is enabled, the X and Y positions are adjusted incrementally by multiple drags actions, in the same way in which the built-in REAKTOR knobs and faders work. Specifically, when the user begins a new drag, the X and Y outputs transmit values relative to where the previous drag ended, as opposed to the absolute position of the mouse cursor. The BX and BY inputs are only effective in incremental mode, and are used to set the incremental 'base' for subsequent drags (overriding the last mouse drag). Typically, these are connected to Snap Value modules to ensure that the incremental base is set according to the last mouse movement saved in a snapshot.

In the properties, Outline Style specifies the outline appearance of the Mouse Area box. If 'Rectangle' is selected, then a 1-pixel outline is drawn around the Mouse Area box, whereas if 'Bar' is selected the area is filled with solid colour.

The Active State option specifies the action that causes the box outline to change from 'inactive' to 'active' state. Choose from either 'Selection' (the active state occurs whenever the Mouse Area box is selected), or the Left/Right/Center button options (active state occurs whenever the relevant mouse button is pressed). The Outline Color properties allow the color and transparency settings to be defined independently for both inactive and active states.

Also on the properties are values for X and Y offset, which allow the position of the Mouse Area on the instrument panel to be offset from the REAKTOR 4 x 4 grid.

BX: Input to set the base value for incremental changes at the X output. Legal values include those within the Range X, as defined in the properties.

BY: Input to set the base value for incremental changes at the Y output. Legal values include those within the Range Y, as defined in the properties.

Note that BX and BY inputs only influence the X and Y outputs when Incremental Mode option is enabled in the properties.

X: Output for the horizontal mouse position, scaled and limited to the Range X values (in the properties). The X output only reports mouse positions that are within the Mouse Area box (as defined by Size X and Size Y in the

properties).

Y: Output for the horizontal mouse position, scaled and limited to the Range Y values (in the properties). The Y output only reports mouse positions that are within the Mouse Area box (as defined by Size X and Size Y in the properties).

PX: Horizontal mouse position in pixels relative to the X origin line (left edge of the Mouse Area box). Moving the mouse to the left of the X origin line outputs negative pixel values, moving it to the right outputs positive values. In contrast to the X output, which is limited to movements within the Mouse Area box, PX reports mouse positions all the way to the left and right edges of the screen.

PY: Vertical mouse position in pixels relative to the Y origin line (left edge of the Mouse Area box). Moving the mouse above the Y origin line outputs negative pixel values, moving it below outputs positive values. In contrast to the Y output, which is limited to movements within the Mouse Area box, PY reports mouse positions all the way to the left and right edges of the screen.

BL: Left mouse button state: 1 when pressed, or 0 when not pressed.

BR: Right mouse button state: 1 when pressed, or 0 when not pressed.

BC: Center mouse button state: 1 when pressed, or 0 when not pressed.

Db: Outputs a single event with value 1 every time a double-click occurs. The Db value at remains 1.0 after a double-click, so you must test for Db events, not static values.

Stacked Macro

Panel



Stacked Macros enables multiple graphical objects to share the same area of the instrument panel. Which objects are displayed within the area at any one time can then be controlled using the Panel Index module.

After inserting a Stacked Macro into your structure, place it in the correct position on the panel, and set the desired size in the properties. Then insert 2 or more macros (normal macros, not additional Stacked Macros) inside the Stacked Macro, along with a Panel Index module. Only one of the 'normal'

macros (along with any panel elements inside it) will be visible at any one time. Which macro is visible depends on the input value to the Panel Index module. (To discover the index number of a macro, right click on that macro on the instrument panel - the index number will be displayed on the context menu).

IC Send

Terminal



Transmits monophonic event signals to any module capable of receiving IC (Internal Connection) transmission. Such modules include IC Receive modules, but also various panel elements (such as knobs and switches). As internal connections work globally (i.e. at the ensemble level), this module can be used to make wireless connections between different instruments within the ensemble.

The IC Send module has a panel display allowing connections to be configured from the instrument panel. All modules in the ensemble capable of receiving IC transmission will appear here, except those with the 'No Entry in IC Menu' properties option enabled. Connections can also be established in the properties dialog.

IC Receive

Terminal



Receives and outputs monophonic event signals to modules which connected via the Internal Connection (IC) protocol. Typically the IC Receive module is used with IC Send modules, but can be connected to any module capable of connection via IC (such as knobs and switches).

IC connections can be established in the properties, and when connecting to IC Send modules, connections can also be established using the IC Send panel interface.

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**. 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 **Sync 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).

Channel Message

MIDI In



Monophonic source for receiving MIDI channel messages from an external MIDI device (keyboard or sequencer etc.), or internally from other instruments

within the ensemble. The output order of the ports (see below) is strictly defined, from top to bottom. This ensures that the type (e.g. control change) and source (e.g. controller 7 on channel 1) of the message is always reported before the actual value.

St: Output port which reports the type of message received.

- 0 = Note Off
- 1 = Note On
- 2 = Poly Aftertouch
- 3 = Control Change
- 4 = Program Change
- 5 = Channel Aftertouch
- 6 = Pitchbend

Ch: Number of the MIDI channel (1-16).

Nr: Number of a Note, Control Change, or Program Change (0-127).

Val: Velocity of a Note, pressure of Aftertouch, or value of Control Change and Pitchbend. With external MIDI, the values are 7-bit quantized (14-bit for Pitchbend). With internal MIDI, the values are unlimited 32-bit floating-points. All values are scaled according to the Min and Max settings in the properties, which is 0 to 1 by default. Another common setting is 0 to 127, which can aid interpretability of values in some situations (Program Change for example).

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 out-

put. 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.



The value at the **Pos** input is sent with an event at the **Trig** input as MIDI song position.

- **Trig**: 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.



Monophonic source for sending MIDI channel messages to an external MIDI device (sequencer etc.), or internally to other instruments within the ensemble. Channel Messages are transmitted every time an event arrives at the **St** port. Thus, the desired message value and destination must be correctly defined with appropriate values at the other inputs before events arrive at the **St** input. All inputs accept monophonic signals only.

St: Event input defining the type of channel message to be sent. A channel message is transmitted everytime an event arrives at this input.

- 0 = Note Off
- 1 = Note On
- 2 = Poly Aftertouch
- 3 = Control Change
- 4 = Program Change
- 5 = Channel Aftertouch
- 6 = Pitchbend

- Ch:** Audio input for the number of the MIDI channel (1-16). Fractional values arriving at Ch are rounded up to the nearest integer; e.g. a value of 0.5 would be rounded up to 1.
- Nr:** Audio input for the number of the Note, Control Change, or Program Change (0-127).
- Val:** Audio input for the velocity of a Note, pressure of Aftertouch, or value of Control Change and Pitchbend.

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.

Subtract

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** = **In** * **In** = **In²**).

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

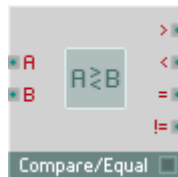


Comparing Logic Function. Compares the two input values and sets the two outputs according to the result of the comparison.

- **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



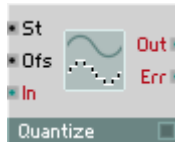
Comparing Logic Function. Compares the two input values and sets the four outputs according to the result of the comparison.

- **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^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.

Sine

Math



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 2π (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.

Sine/Cos

Math



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 $\text{Max}+1$ is the same as 0, $\text{Max}+2$ 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



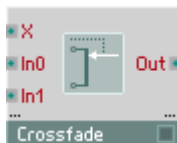
Relay. The upper input is connected to the output if **Ctl** > 0. Otherwise the lower input is connected to the output. If the lower input is not present a zero signal is produced at the output.

The module has 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-port 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 ($\text{Out} = (1 - X) \text{In1} + X \text{In2}$).

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 ($\text{Out} = \text{In1} \cdot 10^{\text{Lv1}/20} + \text{In2} \cdot 10^{\text{Lv2}/20}$ 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 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.

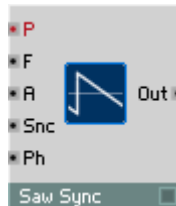
- **P:** Logarithmic event input for controlling the pitch (oscillator fre-

quency). 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 goes from 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 sawtooth waveform.



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.



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 ($69 = 440$ Hz).
- **A:** Audio input for controlling the amplitude. The output signal moves between $+A$ and $-A$.
- **Out:** Audio signal output for the triangle waveform.

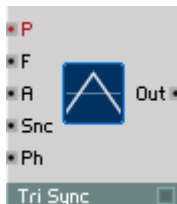
Tri FM

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 ($69 = 440$ 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.



Oscillator for symmetric triangle waveform with synchronization, logarithmic pitch control, linear frequency modulation (FM) and linear amplitude modulation.

Whenever the synchronization signal goes from 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.



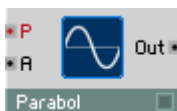
Oscillator for variable triangle and parabolic (near to sine) signals, with logarithmic pitch (P) control and linear amplitude (A) modulation. The symmetry is adjustable by (W). Setting the symmetry (with W) allows a range of wave-

forms 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



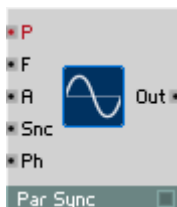
Oscillator for parabolic waveform with logarithmic pitch control, linear fre-

quency 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 ($69 = 440$ 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 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 goes from 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 parabolic waveform.



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 ($69 = 440$ 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.



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.

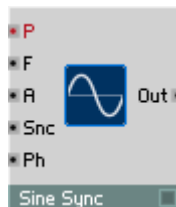
- **P:** Logarithmic event input for controlling the pitch (oscillator frequency). Value in semitones ($69 = 440$ Hz).
- **A:** Audio input for controlling the amplitude. The output signal moves between $+A$ and $-A$.
- **Out:** Audio signal output for the sine waveform.



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.



Oscillator for pure sine waveform with synchronization, logarithmic pitch control, linear frequency modulation (FM) and linear amplitude modulation.

Whenever the synchronization signal goes from 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



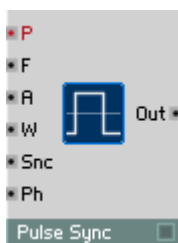
Oscillator for pulse wave with logarithmic pitch control, pulse width modulation (PWM) and linear amplitude modulation.

- **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. $Lo:Hi = (1 + W) / (1 - W)$.
- **Out**: Audio signal output for the pulse waveform.



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 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. $Lo:Hi = (1 + W) / (1 - W)$.
- **Out:** Audio signal output for the pulse waveform.



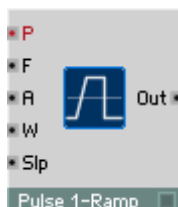
Oscillator for pulse wave with synchronization, logarithmic pitch control, linear frequency modulation (FM), pulse width modulation (PWM) and linear amplitude modulation.

Whenever the synchronization signal goes from 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 $-A$.
- **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 . 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



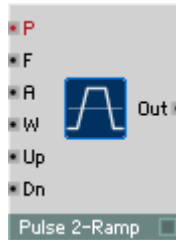
Oscillator for rectangular trapezoid waveform with logarithmic pitch control, linear frequency modulation (FM), pulse width modulation (PWM) and linear amplitude modulation. The waveform is a mixture of pulse and sawtooth: The rising edge of a pulse wave can be flattened and its slope adjusted. The falling edge is vertical.

- **P:** Logarithmic event input for controlling the pitch (oscillator frequency). Value in semitones ($69 = 440$ 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 .

- **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 ($69 = 440 \text{ Hz}$).
- **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.
- **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



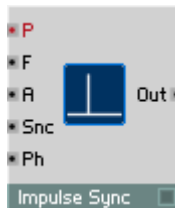
Oscillator for impulse train signal with logarithmic pitch control (P) and linear amplitude modulation (A).

- **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.



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.



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 goes from zero to positive values (rising edge) the phase of the oscillator is reset to a position specified by the phase input.

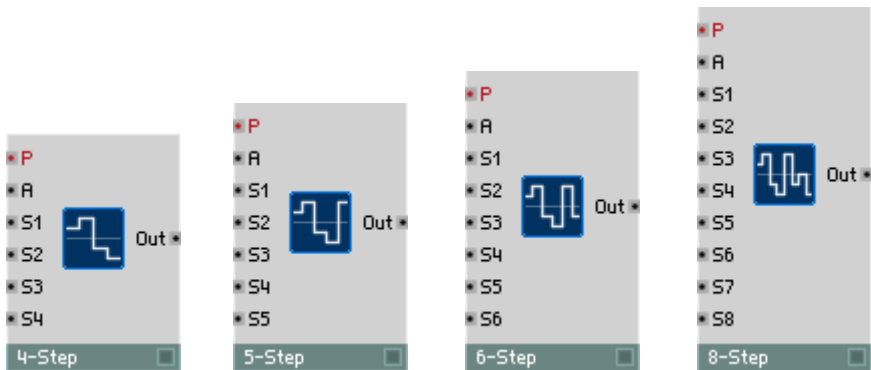
- **P:** Logarithmic event input for controlling the pitch (oscillator frequency). Value in semitones ($69 = 440$ Hz).
- **F:** Audio input for linear frequency modulation. Value in Hz. **P** and **F** together determine the oscillator frequency.
- **A:** Audio input for controlling the amplitude.
- **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.

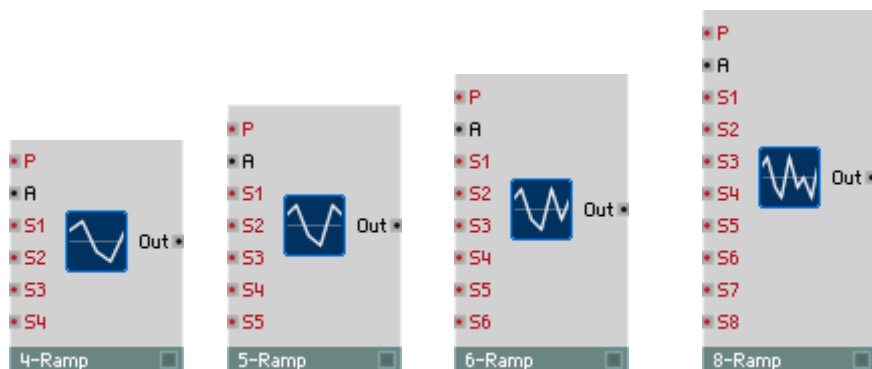
8-Step

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

Oscillator

Like 4-Ramp but with 5 ramps.

6-Ramp

Oscillator

Like 4-Ramp but with 6 ramps.

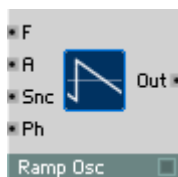
8-Ramp

Oscillator

Like 4-Ramp but with 8 ramps.

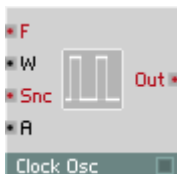
Ramp

Oscillator



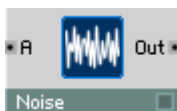
Oscillator to produce a ramp waveform, typically used as a control signal, for example to use an Audio Table module as a waveform oscillator. The signal ramps up from 0 to A and then resets instantly.

- **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.



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 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 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 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.



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

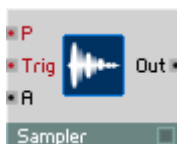
etc. If the loop length sets the vibration of a waveform, the position of the loop logically sets the choice of waveforms from the WaveSet. **Sampler Loop** uses the **LS** input to control the loop start point. In waveform mode the values at this input are quantized, so that glitch free playback between 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 examples. **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 re-started 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 begin-

ning 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.

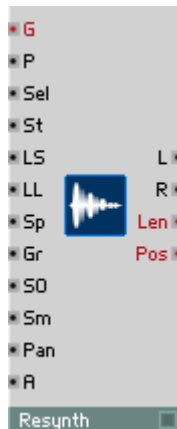
If the **No Stereo** option is activated (you can set this in the properties dialog box), stereo samples will be treated as mono samples, i.e. the same signal is sent to the **L** and **R** outputs. If this is the case, **Sampler Loop** only uses the left channel of stereo samples. This option has been 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.

Grain Resynth



Sampler

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 resynthesis 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.

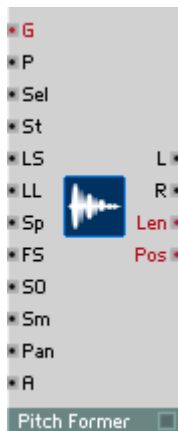
If the **Loop in Release** option is deactivated, loop playback will fade or fade out completely upon a **Gate** event occurring. Depending on the direction set, the sample will run to its beginning or end and will then fade out. If the **Loop in Release** option is activated or if loop playback is switched 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**).



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.



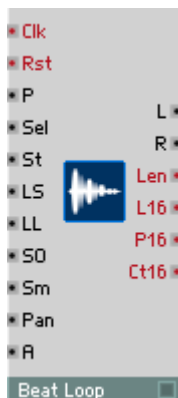
Stereo-multi-sample granular-synthesizer with independent control over pitch **P**, pitch-slide **PS**, sample selection **Sel**, sample-position **Pos** and length **Len** of each grain. The envelope of each grain can be controlled with attack **Att** and decay **Dec**.

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 control. 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.
- **Dist:** Audio input for setting the delta time before starting the next grain in ms. Typ range: [5...100]. Default: 20.
- **DisJ:** 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.



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.

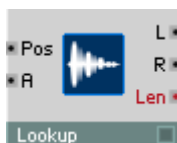
All of **Beat Loop**'s inputs, except **C** and **Rst**, are designed as audio inputs. During sample playback, changes to values take effect at intervals of sixteenths of a note value.

- **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**.

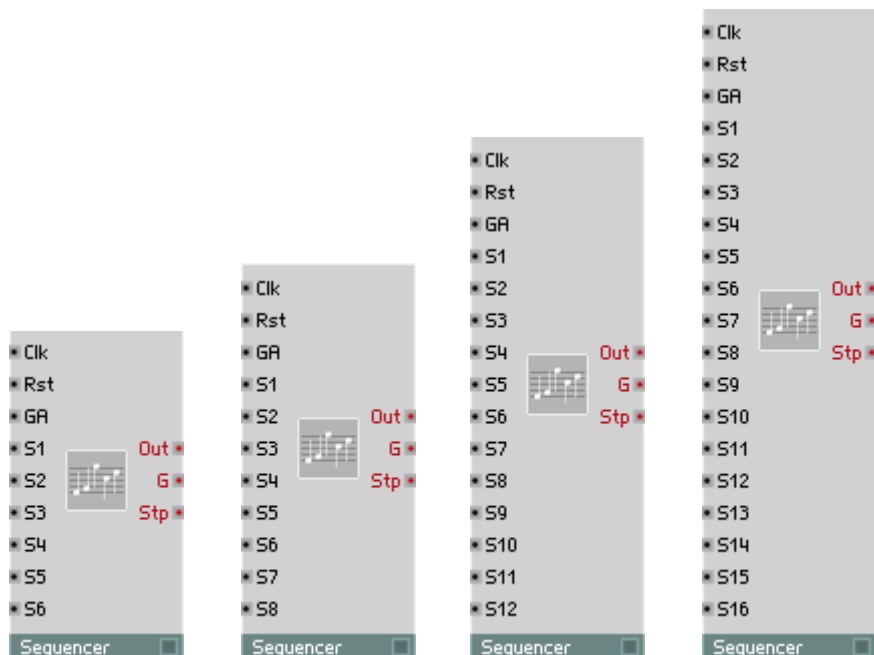
The properties dialog box allows you to select one of three levels of quality that is to be used for interpolation during transposed sample playback (**Poor**, **Good**, **Excellent**). If set to **Poor**, the sample values are not interpolated during output. Higher playback quality is achieved at the 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



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 ... (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 ... 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.

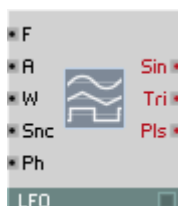
LFO, Envelope

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.

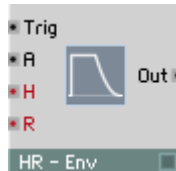
- **T:** Audio input for triggering the envelope on a rising edge (value goes from zero in positive direction).
- **A:** Audio input for the output value during the hold time. The input is sampled at the triggering instant after which the value is held at the output.
- **H:** Logarithmic event input 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 goes from 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 $\text{dB}_{1\text{ms}}$).
- **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.



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 goes from 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 $\text{dB}_{1\text{ms}}$).
- **Out:** Audio output for the envelope signal.



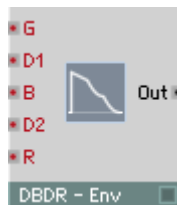
Envelope generator with decay-release characteristic. When the envelope is triggered with a gate event the output value jumps to the amplitude value of the gate signal, after which the output decays exponentially with the decay time back to zero. A gate event with amplitude zero (at note off) sets the 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.



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.



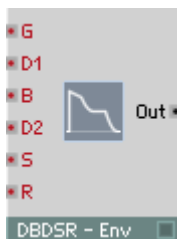
Envelope generator with decay-breakpoint-release characteristic. When the envelope is triggered with a gate event the output value jumps to the amplitude value of the gate signal, after which the output decays exponentially with the decay-1 time until it reaches the breakpoint level (multiplied by the 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 dB_{1ms}).
- **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 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.

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.

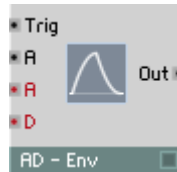
- **G:** Event input for the gate signal that triggers the envelope. The amplitude of the gate signal determines the initial output value.
- **D1:** Logarithmic event input for controlling the first decay time. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (i.e. in dB_{1ms}).
- **B:** Event input for controlling 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.

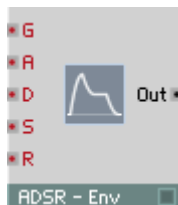
ADR-Env

LFO, Envelope



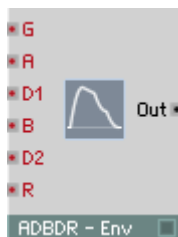
Envelope generator with attack-decay-release characteristic. When the envelope is triggered with a gate event the output value rises during the attack time with a linear slope to the amplitude value of the gate signal, after which it decays with the decay time to zero. When a gate event with amplitude 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.



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 dB_{1ms}).
- **D:** Logarithmic event input for controlling the 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 (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 dB_{1ms}).
- **Out:** Audio output for the envelope signal.

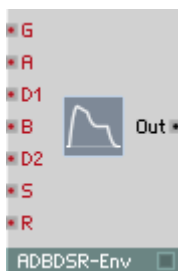


Envelope generator with attack-decay-breakpoint-decay-release characteristic. When the envelope is triggered with a gate event the output value rises during the attack time with a linear slope to the amplitude value of the gate signal, after which it decays exponentially with the decay-1 time until it 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



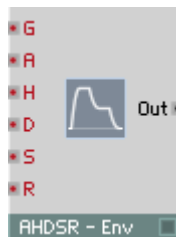
Envelope generator with attack-decay-breakpoint-decay-sustain-release characteristic. When the envelope is triggered with a gate event the output value rises during the attack time with a linear slope to the amplitude value of the gate signal, after which it decays according to the first decay time with 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 $\text{dB}_{1\text{ms}}$).
- **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 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 $\text{dB}_{1\text{ms}}$).
- **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 $\text{dB}_{1\text{ms}}$).
- **Out:** Audio output for the envelope signal.

AHDSR - Env

LFO, Envelope



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.

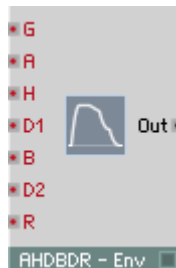
- **G:** Event input for the gate signal that triggers the envelope. The

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



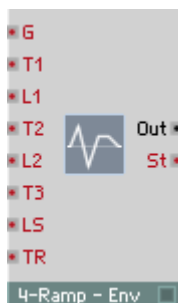
Envelope generator with attack-hold-decay-breakpoint-decay-release characteristic. When the envelope is triggered with a gate event the output value rises during the attack time with a linear slope to the amplitude value of the gate signal and stays there until the hold time has passed, after which it 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.

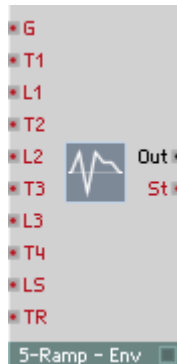
- **G:** Event input for the gate signal that triggers the envelope. The amplitude of the gate signal combines with all the level values to determine the actual levels reached.
- **T1:** Logarithmic event input for controlling the time for the first stage. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (i.e. in $\text{dB}_{1\text{ms}}$).
- **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 $\text{dB}_{1\text{ms}}$).
- **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 $\text{dB}_{1\text{ms}}$).
- **St**: 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 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 dB_{1ms}).
- **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 dB_{1ms}).
- **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 dB_{1ms}).
- **St:** 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-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 $\text{dB}_{1\text{ms}}$).
- **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 $\text{dB}_{1\text{ms}}$).

- **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 dB_{lms}).
- **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 dB_{lms}).
- **St:** 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.

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.

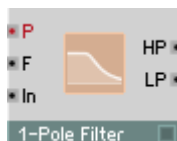
HP/LP 1-Pole

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



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 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



First order allpass filter. This type of filter has a flat amplitude response, but the phase shift between input and output grows from 0 degrees at low frequencies to -180 degrees at high frequencies. At the corner frequency controlled by the P input the phase is shifted for -90 degrees.

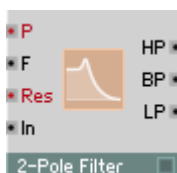
- **P:** Logarithmic control input for the corner frequency, where the phase shift is -90 degrees. Scale: 1 Semitone per unit, 43 = G1 = 98 Hz, 84 = C5 = 1047 Hz.
- **In:** Input for the signal to be allpass filtered (phase-shifted).
- **Out:** Output for the allpass filtered (phased-shifted signal).



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 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] $\cong 1 / (2 - 2 \text{ Res})$.
- **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



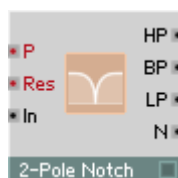
2-pole filter with high pass and low pass outputs (12 dB/octave falloff), band pass (above and below each 6 dB/octave falloff), variable resonance, logarithmic and linear control of cutoff frequency.

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 ($69 = 440 \text{ Hz}$).
- **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/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] $\cong 1 / (2 - 2 \text{ Res})$.
- **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] $\cong 1 / (2 - 2 \text{ Res})$.
- **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

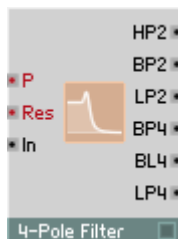


2-pole filter with band reject (notch) output, high pass and low pass outputs (12 dB/octave falloff), band pass (above and below each 6 dB/octave falloff), variable resonance, logarithmic and linear control of cutoff frequency.

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.

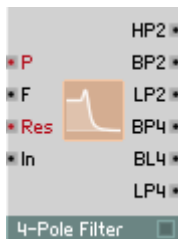
- **P:** Logarithmic event input for controlling the cutoff frequency. Value in semitones (69 = 440 Hz).
- **F:** Audio input for linear control of the cutoff frequency. Value in Hz. **P** and **F** together determine the oscillator frequency.
- **Res:** Event input for controlling the filter's resonance/damping. 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] $\cong 1 / (2 - 2 \text{ Res})$.
- **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



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.

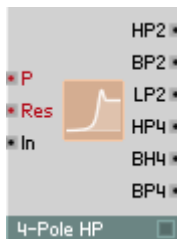
- **P:** Logarithmic event input for controlling the cutoff frequency. Value in semitones ($69 = 440$ Hz).
- **Res:** Event input for controlling the filter's resonance/damping. Range of values 0 (maximal damping, no resonance) to 1 (no damping, maximal resonance, self oscillation!).
- **In:** Audio 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



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.

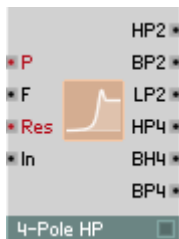
- **P:** Logarithmic event input for controlling the cutoff frequency. Value in semitones (69 = 440 Hz).
- **F:** Audio input for linear control of the cutoff frequency. Value in Hz. **P** and **F** together determine the oscillator frequency.
- **Res:** Event input for controlling the filter's resonance/damping. 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



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



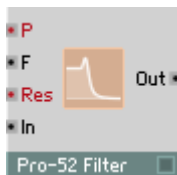
4-pole filter with high pass (24 dB/oct falloff), band pass (12/12 dB/oct) and band/high pass (18/6 dB/oct) outputs, 2-pole high and low pass (12 dB/oct) and band pass (6/6 dB/oct) outputs, variable resonance, logarithmic and linear control of cutoff frequency.

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



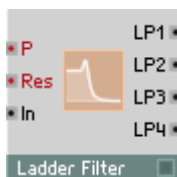
Filter taken from the Pro-52 virtual analog synthesizer. It is a 4-pole low pass filter (24 dB/oct falloff) with variable resonance and logarithmic and linear control of cutoff frequency.

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**. See next module description.

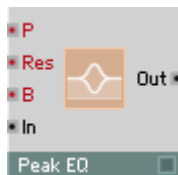


Filter modelled on the classic ladder circuit patented by Bob Moog. It is a 4-pole filter with different low pass outputs: 24 dB/oct, 18 dB/oct, 12 dB/oct and 6 dB/oct falloff. It also has variable resonance and logarithmic and linear control of cutoff frequency.

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.

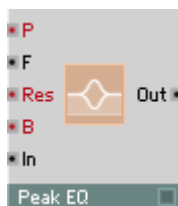
The filter also has options for selecting the quality of the simulation: **Standard**, **High** and **Excellent**. The effect is particularly noticeable when distortion is turned on. Of course, the higher quality settings come at the price of increased CPU usage.

- **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



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 ($69 = 440$ 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] $\cong 1 / (2 - 2 \text{ Res})$.
- **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



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 ($69 = 440$ 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] $\approx 1 / (2 - 2 \text{ Res})$.
- **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



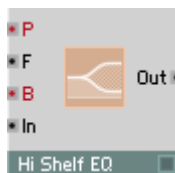
Parametric equalizer with high shelving characteristic, adjustable boost/cut, band width and logarithmic frequency control.

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 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 ($69 = 440 \text{ Hz}$).
- **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.

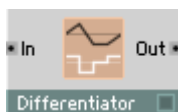
- **P:** Logarithmic event input for controlling the cutoff frequency. Value in semitones ($69 = 440 \text{ Hz}$).
- **B:** Event input for controlling boost/cut in dB. At value **B** = 0 the signal remains unchanged.
- **In:** Audio signal input for the signal to be equalized
- **Out:** Audio output for the equalized signal



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



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 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.

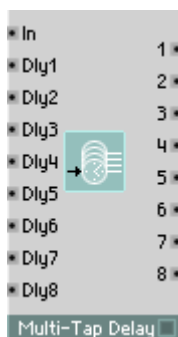
- It behaves like a REAKTOR 3 **Event Delay** if you connect an event signal to the In-input.

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.



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.



Delays audio signal by one sample period ($1/\text{sample rate}$). 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

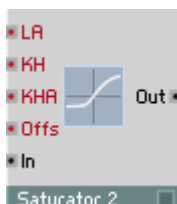


Distortion with a smoothly rounded input-output curve for soft transition to saturation. The output value is limited to ± 2 (reached for input values bigger than ± 4). Very small input values are not changed.

- **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

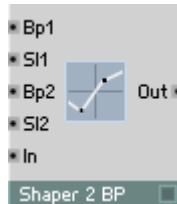


Signal shaper with piecewise linear input/output curve and one breakpoint. The slope of the curve above the breakpoint can be adjusted. Signal values below the breakpoint are passed unchanged.

- **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.

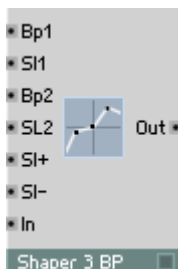
Shaper 2 BP

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
- **Sl1:** 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.
- **Sl2:** 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.



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



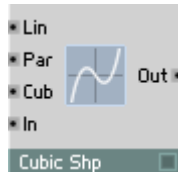
Signal shaper with parabolic (2nd order polynomial) input/output curve.

The linear and square parts of the signal can be adjusted (**Out = Lin In + Par In²**).

- **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

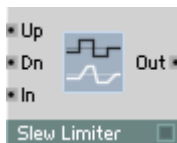
Audio Modifier



Signal shaper with cubic parabolic (3rd order polynomial) input/output curve.

The linear, square and cubic parts of the signal can be adjusted (**Out = Lin In + Par In² + Cub In³**).

- **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 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



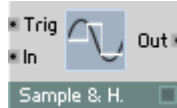
Detector for the peak amplitude. The input signal is rectified and smoothed with an adjustable release time. The attack time is zero.

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 $\text{dB}_{1\text{ms}}$).
- **In:** Audio input for signal to be detected
- **Out:** Audio output for the signal

Sample & Hold

Audio Modifier



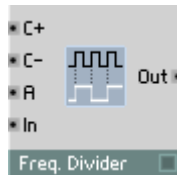
Sample & Hold with clock input.

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

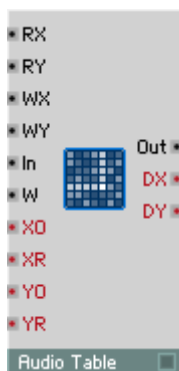
Audio Modifier



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.



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 165.

- **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

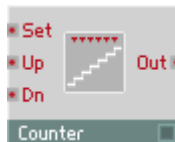


Accumulator (sum) for event values. The value of each event at the input is added to the total value stored in the module. The value of the updated sum is sent as an event at the output.

- **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.

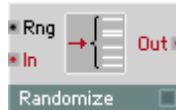
Counter

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.



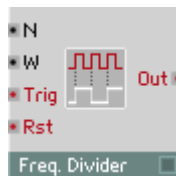
Event randomizer with adjustable spread.

The arriving events are modified with a random positive or negative offset in the given range (**Out** = **In** + X, where $-\mathbf{Rng} \leq X \leq \mathbf{Rng}$).

- **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**.

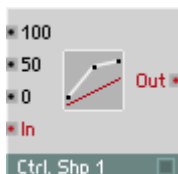
- **N:** Control input for the number of input events per output period. ($N < 2$: no division, $2 \dots 2.99$: division by 2, $3 \dots 3.99$: division by 3, etc.).
- **W:** Control input for the pulse width of the output signal. Range of values: $0 \dots 1$ (0% ... 100%). **W** = 0 : the gate signal 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

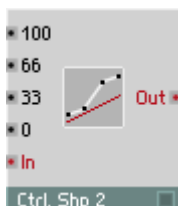


Event signal shaper with piecewise linear input/output curve and one fixed breakpoint. The output produced by linear interpolation between the given points.

- **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



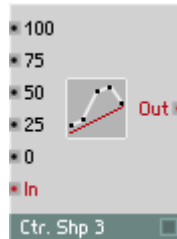
Event signal shaper with piecewise linear input/output curve and two fixed breakpoints. The output produced by linear interpolation between the given points.

- **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

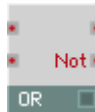


Logic gate for event signals. The output is the logic AND of the two input values, i.e. the output is 1 when both inputs are positive, otherwise the output is 0.

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.

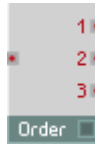
Logic NOT

Event Processing



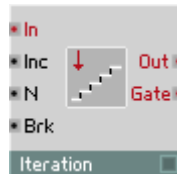
Logic gate for event signals. The output is the logic complement of the input value, i.e. the output is 0 when the input is positive, otherwise the output is 1.

The input treats values of zero and below as the logic False state, values above zero are logic True.



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.



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 unstable 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



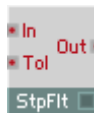
Merger for event signals. When more than one wire is connected at the input, the output value is that of the last received input event, irrespective of which wire it came from.

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 $\text{Max}+1$ is the same as 0, $\text{Max}+2$ 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.



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.

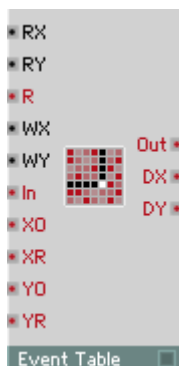
- **In:** Polyphonic input for the events whose distance in time is to be measured.
- **F:** Polyphonic event output for the frequency of input events in Hz.
- **T:** Polyphonic event output for the time between events in milliseconds.



Hold 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 165.

- **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.
- **In:** Event input for writing into the table 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 **In**.
- **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 tapedecks 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 hddisk you can use the **Relaod** 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 hddisk** in the properties the tapedeck module starts working in hddisk mode. Audio files are written to and read from hddisk 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

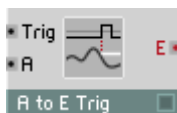
Auxiliary



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 goes from 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



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



Audio to gate event converter with gate amplitude input. When the trigger signal goes from 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.

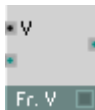
- **T:** Audio input for triggering the gate signal
- **A:** Audio input for controlling the amplitude of the gate events
- **Out:** Event output for the gate event signal.



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.

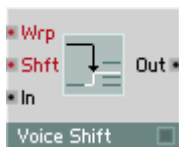
This is useful for making polyphonic sequencers, for example.

- **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.



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.



The Voice Shift module is used to rearrange the polyphonic input values, so that the output values are remapped across the voices as required. For example, you could use Voice Shift to remap the values of voices 1, 2, 3, 4 to output voices 2, 3, 4, 1, or to 3, 4, 2, 1, and so on. If multiple input voices are shifted to the same output voice, they are summed. For example, if input voices 1 and 2 are both shifted to output voice 3, voice 3 will consist of: voice 1 signal + voice 2 signal.

Wrp: Monophonic event value that turns voice-shift wrap on and off (off by default). When wrap is off (ie. $Wrp \leq 0$), voices shifted to invalid voice numbers (i.e. less than 1 or greater than the number of polyphonic voices) are discarded. When wrap is on (i.e. $Wrp > 0$), voices shifted to invalid voice numbers are wrapped around to valid voice numbers using modulo math. For example, +1 shifting in a 3-voice instrument will cause voice 3 to be remapped to voice 1.

Sh: Polyphonic event value that controls the voice shifting. Positive Sh values shift voices up (e.g. $Sh = 1$ would shift input voice 1 to output voice 2), and negative Sh values shift voices down ($Sh = -2$ would shift voice 3 to voice 1). Since Sh is a polyphonic input, each voice can be shifted by its own individual offset. The default Sh value is 1.

In: Input for the polyphonic signal to be voice-shifted.

Out: Output for the polyphonic voice-shifted signal.



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

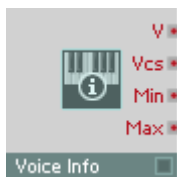


Source for the current Tempo measured in Beats per Second. To get the BPM value, multiply by 60.

- **Out**: Event output for the current tempo in Beats per Second (Hz).

Voice Info

Auxiliary



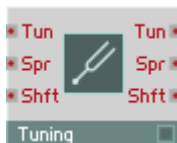
Voice Info module.

- **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

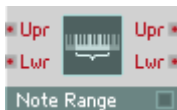


Source for information about the system: Sampling rate (in samples/sec), control rate (in Hz) and CPU load (in %).

- **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.

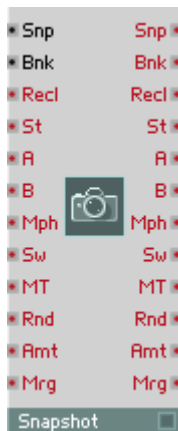
MIDI Channel Info

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.



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 at the right side 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.
- **Bank**: 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.
- **Morph**: 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 ... 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.
- **Morph**: 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

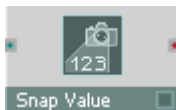
Auxiliary



Polyphonic event source for a constant value used to spread out parameters for unison voices.

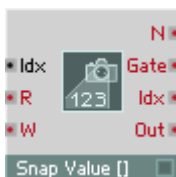
In unison mode, the different voices that play the same note need to have their parameters spread out a little to make them slightly different so that their sum results in a sound that is fatter, and not just louder. For note pitch this happens automatically and is controlled by the **Unison Spread** 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.



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.



Stores and recalls arrays of floating-point (fractional) values to/from the edit buffer and snapshots.

A single Snap Value Array can hold 1-40 arrays, and each array can contain any number of elements, limited only by the availability of system memory. All arrays contain the same number of elements. Both the number of arrays and the number of elements per-array are defined in the properties.

Memory for the Snap Value Array module is allocated dynamically. That is, creating additional snapshots causes additional memory to be allocated, and deleting snapshots causes memory to be released. This keeps memory requirements as low as possible.

When the Snap Isolate option is enabled in the properties, only the most recent value written to each array element is stored (and recalled during ensemble initialisation), and no data is stored or recalled in response to snapshot operations.

A typical application of the Snap Value Array is storing sequencer data in

snapshots. For this purpose, it is often used in conjunction with Event Tables or Multi Display modules.

All Snap Value Array inputs accept monophonic signals only.

Idx: Index of the array element to address (for both read and write operations). Arrays are 1-based; i.e. the index of the first element is 1 (not 0). Fractional values are rounded to the nearest integer. Where Idx values are outside the range of 1 to N, the behaviour depends on the Index Behaviour option in the properties.

R: When an event (of any value) arrives at the R (read) input, the array element specified by the Idx input is read, and it's value is transmitted to the output port. In other words, each event at the R input propagates a single event at the array output port. Multiple arrays are addressed in parallel by the same Idx index. Thus, events arriving at the R input propagate output events at every array output port (of the value of the element selected by Idx). It is essential to set the Idx value before events arrive at the R input.

W: When an event arrives at W, its value is written to element [Idx] of the array, overwriting any data that was previously there. The Snap Value Array provides a separate W input port for each array it contains (1 array by default, but more can be created in the properties). Each port can be renamed as appropriate. When the Events Thru option is enabled in the properties, events arriving at the W inputs are passed to the corresponding Out outputs.

N: The N output reports the number of elements in each array.

Gate: Gate sends an event with value 1 before the array output ports send events, and then sends an event of value 0 afterwards.

Idx: Output reporting the number of the element currently being read or written to (in response to events arriving at the R and W ports, or from snapshot operations, such as recall and morph). With respect to read operations, the Idx number will be reported before the value event is transmitted at the Out outputs.

Out: The value of the currently accessed array element (as defined by Idx) are transmitted here in response to events at the R port and snapshot operations (recall, morph etc.). When the Self-Iteration option is enabled in the properties, all array elements are output in serial fashion (i.e. the first element, then the second element, then the third etc) when any of the following operations occur: initialization, activation, snap recall, randomize, random merge, and morph. Self-Iteration enables the updat-

ing of a complete set of data with snap operations.

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.

IC Send

Terminal

Transmits monophonic event signals to any module capable of receiving IC (Internal Connection) transmission. Such modules include IC Receive modules, but also various panel elements (such as knobs and switches). As internal connections work globally (i.e. at the ensemble level), this module can be used to make wireless connections between different instruments within the ensemble.

The IC Send module has a panel display allowing connections to be configured from the instrument panel. All modules in the ensemble capable of receiving IC transmission will appear here, except those with the 'No Entry in IC Menu' properties option enabled. Connections can also be established in the properties dialog.

IC Receive

Terminal

Receives and outputs monophonic event signals to modules which connected via the Internal Connection (IC) protocol. Typically the IC Receive module is used with IC Send modules, but can be connected to any module capable of connection via IC (such as knobs and switches).

IC connections can be established in the properties, and when connecting to IC Send modules, connections can also be established using the IC Send panel interface.

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

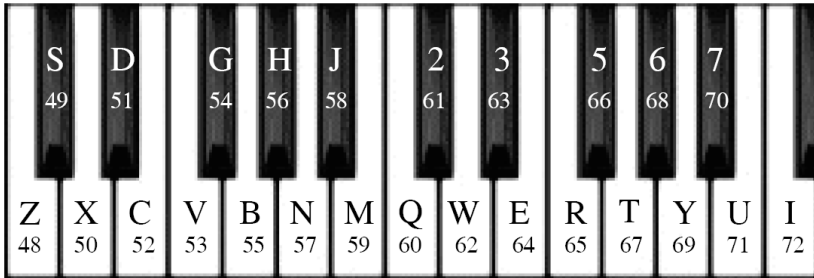
Terminal



See OSC Send module.

Appendix

Transposing Incoming MIDI Notes



- Holding down the **Shift** key raises all incoming MIDI notes by two octaves (+24).
- Holding down WindowsXP: **Ctrl + Shift** / OS X: **APPLE + Shift** lowers all incoming MIDI notes by two octaves (-24).

Index

Symbole

? Menu	114
1/96 Clock	137, 306
12-Step	363
16-Step	363
16-Step Sequencer Plus Bassline	69
1 / Square Root	316
2D Table	
Default	150
Max	150
Min	150
4-Ramp	340, 377
4-Step	339
5-Ramp	378
6-Ramp	380
6-Step	362
8-Step	363
Show Event Initialization Order	101

A

About	114
Accessing Files	121
Accumulator	416
ADBDR - Env	373
ADBDSR-Env	374
Add	309
Adding Instruments	138
Adding Modules	165
ADSR - Env	372, 373, 375, 376
AD - Env	371
Akai Import	238
Allpass 1-Pole	383
Alpha Channel	209
Always on top	102
Amp	321

Animation	208, 209
Height	209
Width	209
Appearance Page	134, 148, 162
Appendix	450
Append snapshot	215
Arccos	317
Arcsin	317
Arctan	318
Arrange Icons	113
Arranging the Panel	82
AR - Env	371
ASIO	37
AU	37
Audio-In Module	65, 127
Audio-Out Module	65, 128
Audio + MIDI Settings	39, 101
Audio files path	106
Audio Interface	35, 39
Audio In level meter	116
Audio Modifier	405
Audio Out level meter	116
Audio signals	179
Audio Smoother	437
Audio Table	414
Audio Units	37, 50
Plug-in Installation	34
Audio Voice Combiner	431
Auditioning Files	124
Auditioning Sample Files	236
Authorization	22
Key	22
Automatic	
Layout	98
Panel Layout	135
Voice Reduction	104
Automation ID editing	44
Auto Save Configuration	45
Auto Voice Reduction	145
Auxiliary	428

Available in Panelsets	150	Close	113
a * b + c	311	Close All Structures	112
A to E	433	Color, edit Instrument	149
A to E (Perm)	434	Colors, edit Ensemble	135
A to E (Trig)	433	Color Scheme	135, 149
A to Gate	434	Compare	313
B		Compare/Equal	313
Background		Comparing Snapshots	217
Bitmap	151, 163	Complete button	26
Bitmap, Macro	163	Connection	154, 198
Backup Data with Module	242	Connection Page	135, 152
Bank	218	Connection Properties Controls	196
Basic Operation	90	Constant	309
Batch Processing	95	Constant Sources	172
Beatloop	224	Cont/Note	198
Beat Loop	359	Context Menu	91, 139, 191
Bi-Pulse	337	Context Menu, Macro	158
Bi-Saw	325	Context Menu, structure	181
BMP	209	Control	
Bookmark	119	ID Numbers	213
jump to	119	Rate	97, 133
BPM by Snapshot	132	Skins	191
Browser	120	Controller	198, 299, 304
Browser and Snapshots		Controls	151
always on top	102	Copy	96, 140
Button	186, 200, 280	Copying Snapshots	216
C		Core Audio	37
Cascade	112	Counter	416
Ch. Aftertouch	300, 305	CPU	
Choose Color	135, 149	Load Display	116
Chopper	408	Usage	103
Classic Modular	253	Create backup file before saving	102
Clipper	406	Create Constant	168
Clock		Create Control	167
Start	98	Creating Instruments	138
Stop	98	Creating Macros	156
Clock Oscillator	342	Creating wires	173
		Crossfade	320
		Ctrl. Shaper	418, 419
		Cubase	48

Custom-made control	203
Customized fader	207
Customized panels	206
Cut.....	96, 140

D

DBDR - Env.....	369
Debug	99
Delay	398
Delete.....	96, 140
Deleting wires.....	174
Differentiator	396
Diffuser Delay.....	400
Digital Performer	52
Directories	104
DirectSound	37
Disable Automation	199
Display	254
Distributor.....	320
Divide x/y.....	311
DIY	72
Double-click	90
Draw/Select/Control Mode	248
Drop-down Menu	201
Drum-Maps	227
DR - Env.....	368
DSR - Env	369
Duplicate	96, 140
DXi 2.....	37, 54
Plug-in Setup.....	32
Dynamic ports	276
D - Env	368

E

Editing Panels	199
Edit Menu	95
Edit Sample List Box.....	232
Ensemble.....	125
is not Found	46

Panel Toolbar.....	115, 117
Properties	130
Structure.....	127
Window.....	129
Envelope.....	365
Event	
Initialization Order	101
Loops	102, 132, 147
Loops Enable.....	178
Loop Prevention	178
Processing	178
Signals.....	177
Event Processing.....	416
Event Smoother	437
Event Table	426
Event V.C.	
All	432
Max.....	432
Min	433
Exit	95
Expon. (A).....	314
Expon. (F).....	314
External Sample Editor	106
External Sync	98, 137

F

Factory Content path.....	105
Fader.....	184, 200, 277
File	
Menu	93
missing	223
Files.....	121
Fill Out Form button.....	29
Filter	382
Filter Envelope.....	88
First Steps	61
FM Overdrive.....	67
Frame Style.....	163
Frequency Divider	413, 417
From Voice.....	435

Functions List Box	235
Function page.....	130, 142
Function page; Macro	159

G

Garage Band	53
Gate	201, 280, 298
Geiger.....	343
Globally disable event loops	102, 178
Global Auto Save	47
Global Pitch	132
Grain Cloud	357
Grain Cloud Delay	402
Grain Delay	401
Graph	
BG	150
Fill	150
Line.....	150
Grid.....	150, 247

H

Header	115
Hierarchy of Reaktor.....	125
High Shelf EQ.....	394
High Shelf EQ FM	394
Hints	175
Hold	425
Hold Ctrl.....	153
HP/LP 1-Pole	382
HP/LP 1-Pole FM.....	383
HR - Env.....	367
Hybrid modules	275
H - Env.....	366

I

IC	
Receive.....	173
Send	173

IC Receive.....	296, 448
IC Send	296, 448
Identical Samples	222
Ignore Tempo Change	94, 99
Import MIDI File	94
Impulse	337
Impulse FM.....	338
Impulse Sync	338
Incremental.....	197, 206
Indicator	150
Info Page	134, 148, 161
Initialization Order.....	101
Insert snapshot.....	216
Installation under MacOS X.....	33
Installation under Windows XP....	31
Instrument	138
Header.....	115, 140
Object.....	138
Properties	142
Integrator	397
Interface	39
Internal Sample Rate.....	133
Invert	310
In Port.....	173, 446
Iteration	421
Iteration module	177

K

Key	
Control.....	204
Tracking.....	87
Knob	200, 201, 279

L

Label.....	131
Instrument	143
Ladder Filter	391
Ladder Filter FM	392
Lamp.....	283

Latency	38	Master Tune	132
Legacy Mode	133	Master Tune/Level.....	438
Level	132	Math	309
Level Lamp	284	Max.....	172, 184
Level Meter	286	Automation ID	199
LFO.....	63, 365	Unison V	145
Linking Snapshots.....	214	Maximum processor usage	104
List of Open Windows	113	Measure CPU Usage.....	99
Load Data into Table	249	Menu.....	93
Local Identifier	59	Merge	423
Local IP Address.....	59	Meter	285
Local Port	59	MIDI.....	41
Lock/Unlock Panel	82, 141	Activity lamps	141
Lock Voices	144	Clock Out	98
Logic	50	Control.....	204
Logic AND	419	Data Types	204
Logic EXOR.....	420	File.....	94
Logic NOT.....	420	In Device.....	136, 137, 153, 154
Logic OR.....	420	In lamp	117
Log (A)	315	Learn.....	97, 118, 205
Log (F)	315	Note	198
Loop		Out.....	206
Editor	236	Out Device	154
MIDI File.....	99	Out lamp.....	117
loop length.....	344	Panel	196
Lower Limit	172	Sources	170
Lower Note Limit	153	MIDI Channel Info.....	440
Low Shelf EQ	395	MIDI In.....	297
Low Shelf EQ FM	396	MIDI Out.....	304
M		Min	172, 184
Macro.....	155	Unison V	145
Collection	253	Minimize	113, 141
Properties	159	Mirror 1 Level.....	407
Main toolbar	115	Mirror 2 Levels	407
Managing Snapshots	214	Missing Samples.....	223
Map		Mixer	321
Keyboard View	234	MME	37
List View	233	Mod. Clipper	407
Master Level.....	132	Module	165
		Adding.....	165

Context Menu	168	Noise.....	342
Ports	167	Note Pitch.....	297
Properties	169	Note Pitch/Gate.....	304
Sorting.....	100	Note Range Info.....	440
Modulo	312	Note Shift	153
Mono.....	168	Nuendo.....	49
Morph, snapshot	136	Number of Undos.....	103
Morphing	220	Num Animations	
Mouse Area	293 150, 151, 162, 163, 209	
Mouse Functions.....	90		
Moving Objects.....	90		
Mrph Ctrl	153	O	
MRPH Time	221	Off Velocity	299
Multi-Ramp	340	on/off switch	424
Multi-Sampling	226	On Velocity.....	299
Multi-Sine	332	Open	93, 115
Multi-Step.....	339	Open File button.....	28
Multi-Tap Delay.....	399	Open Sound Control	57
Multi/HP 4-Pole.....	389	Options.....	101
Multi/HP 4-Pole FM	390	Options List Box	235
Multi/LP 4-Pole	387	Order.....	421
Multi/LP 4-Pole FM.....	388	of Audio Processing.....	180
Multi/Notch 2-Pole.....	385	of Event Processing	178
Multi/Notch 2-Pole FM	386	OSC	57
Multiplex 16.....	363	Client.....	60
multiplier	310	Ensemble	137
Multiply	310	Identification	59
Multi 2-Pole	384	Member List	60
Multi 2-Pole FM.....	384	Monitor	60
Multi Display	291	Receive	173
Multi Picture	287	Send	173
Multi Text.....	288	Settings	101, 154
Mute	139, 158, 168	Source	198
All	131	Synchronization	59
audio during Core compilation	103	System Setup	58
Port.....	168	Target	198
		Oscillator	323
N		oscillator mode	344
Networked Computers	57	OSC Receive	449
New Ensemble.....	93	OSC Send	449
		Output	

Device	40
Latency	40
Out Port	173, 446
Overwrite snapshot	216

P

Padecho	64
Panel	182
Color	135
Controls	151, 162, 183, 184
Editing	182
Operation	200
to MIDI	197
Panner	320
Parabol	328
Parametric equalizer	393, 394
Par FM	328
Par PWM	330
Par Sync	329
Paste	96
Peak Detector	412
Peak EQ	393
Peak EQ FM	393
PgUp/PgDn	204
Picture	286
Borders	151, 163
Index	150, 151, 162
Properties	207
Pitchbend	297, 304
Pitch Former	224
Playerbox	107
Play MIDI File	94, 99
Plug-in	35, 43
Size Functions	47
Poly Aftertouch	198, 300
Poly Display	291
Ports	139, 157, 167
Power x y	315
Pre-listening	124
Preferences	101

Preview	208
Primary Macros	155
Primary Structures	164
Pro-52 Filter	391
Product Authorization	22
Program Change	301, 306
Properties	140
Pro Tools	55
Pulse	333
Pulse 1-ramp	335
Pulse 2-ramp	336
Pulse FM	334
Pulse Sync	334

Q

Quantize	314
QWERTY	450

R

Ramp	377
Random	343
Randomize	219
Randomizer	417
Randomizing Snapshots	219
Range of Values	171
REAKTOR 4 Legacy Mode	133
Reaktor as Plug-in	43
Reassign	146
Recall	
by MIDI	132, 147, 214
by Parent	146
Recalling Snapshots	213
Receive	173, 446
Channel	136, 137, 153, 154
MIDI	196
Recent Ensembles	95
Reciprocal	311
Recorderbox	108
Recorder Settings	108

Rect./Sign.....	312	Sample Rate.....	40, 97, 116, 133
Rectifier.....	312	Sampling and Resynthesis	222
Redo	96	Saturator.....	405
Register Now	24	Saturator 2.....	405
Registration support	30	Save	115
Registration Tool	22	Ensemble	93
relay	424	Ensemble As	93
Relay 1,2	319	Instrument As.....	140, 158
Reload last ensemble		Registration File	27
at start-up.....	102, 103	Saving	83
Remap to single key	235	Maps	228
Remote to MIDI	198	Sawtooth	323
Renaming Snapshots	216	Saw FM	323
Reset All Tool Window Positions	111	Saw Pulse	325
Resizeability	208, 211	Saw Sync	324
Restore ... with Ensemble	102	Scanner	319
Resynth	224	Scope.....	290
Right mouse button.....	91	Screen-sets	111, 112
Root Key	227	Sel. Note Gate	298
Router 1,2	424	Sel. Poly AT.....	300, 305
Router 1->M.....	424	Selectionng Objects	90
Router M->1.....	423	Selector	319
Routing.....	41	Select A.....	221
RTAS.....	37, 39	Select All	97
Plug-in installation	33, 34	Select Picture.....	207
Run/Stop Audio	99, 116	Send	173, 446
S		Separator	422
Sampler.....	345	Sequencer.....	362
Sampler FM	346	Settings Menu	97
Sampler Loop	347	Set Protected	98
Samples	222	Set Random	443
Analysis	224	Set transpose to zero.....	235
Editor	225	Set Unprotected	98
Loop Player	71	Shaper 1 BP	409
Management.....	222	Shaper 2 BP	409
Maps.....	225	Shaper 3 BP	410
Map Editor	228	Shaper Cubic.....	411
Sample & Hold	413	Shaper Parabolic.....	411
Sample Lookup.....	361	Show	
		Hints	118, 119

Module Sorting	100	Square Root	316
Value	186	Stacked Macro	295
Show/Hide		Stand-alone	39
Browser	111	Application	35
Hints	106	Start/Restart Clock	117
Panel	111	Start/Stop	301, 306
Playerbox	107	Step	184
Properties	111	Stepsize	172
Recorderbox	108	Step Filter	423
Sample Map Editor	111	Stereo Amp	322
Snapshots	111	Stereo Pan	321
Toolbox	106	Store by Parent	146
Signal Path	319	Store Map with Ensemble	223, 239
Signal Processing	176	Structure	164
Sine	316, 330	Icon	150
Sine/Cos	317	Icon, Macro	162
Sine FM	331	Toolbar	115, 119
Sine Sync	331	Subtract	310
Single Delay	398	Sustain Control	153
Single Trig. Gate	298	Switch	187, 190, 201, 282
Skin	191	Switches	172
Skins, button	194	Sync Clock	137, 302
Skins, fader	192	SynthOne	62
Slew Limiter	412	System ID	22
Slow Random	366	System Info	439
Snapshot	141, 212, 441	System Menu	99
ID	213	System Requirements	31, 33
Master for Plug-in	132		
Morphing	220	T	
Snap Isolate	218	Table	
Snap Value	444	Control Mode	248
Soft Takeover	196, 206	Draw Mode	248
Solo	139	Files path	106
Sonar	54	Modules	240
Song Pos	302, 307	Select Mode	248
Soundcard	39	Tapedeck 1-Ch	428
Sound installations	57	Tapedeck 2-Ch	431
Sources	170	Targa	209
Source Modules	170	TCP/IP	57
Spin Control	202	Tempo	118

Tempo Info.....	438
Terminal	446
Terminals.....	139, 157, 173
Text.....	288
Text box.....	202
TGA.....	209
Tile	
Horizontally	113
Vertically.....	113
Timer	425
Toggle	201, 280
Toolbar	115
toring Samples with Modules....	223
Total Recall	44
To Voice.....	435
Transparency	208, 209
Transposing.....	450
Tri/Par Symm	327
Triangle.....	326
Trigger.....	200, 280
Tri FM	326
Tri Sync	327
Tune.....	132
Instrument	144
Tuning Info.....	439
Tutorial Ensembles.....	62

U

Undo.....	95, 103, 116
Unison.....	141
Mode.....	145
Spread.....	144
Unison Spread.....	443
Unit Delay.....	404
Upper Limit.....	172
Note	153
User Content path.....	105

V

Value.....	422
Value Grid	247
View	106
Visible	151, 163
Voices.....	141, 144
Allocation	144
Assign	146
Combiner	139
Steal mode.....	146
Voice & MIDI	
Slave To	144
Voice Info.....	438
Voice Shift	436
VST	37
Plug-in Installation	32

W

What is the	
Product Authorization?.....	22
Wire	74
Wires.....	173
Wires, creating.....	90
workspace.....	125

X

XY.....	288
XY Field.....	202