

VERSION 2.0

FOR

MAC™ OS

WINDOWS® 95/98

WINDOWS NT®



Reaktor Series

The Fusion of Synthesis, Sampling and Effects.

This manual was brought to you courtesy of Lord Twerk

WE GENERATE THE FUTURE OF SOUND



Contents

1	Introduction	1
1.1	What is  REAKTOR ?	1
1.2	What is  GENERATOR ?	1
1.3	What is  TRANSFORMATOR ?	1
1.4	Product Specific Texts.....	2
1.5	Getting Started.....	2
1.6	Support	2
2	Installation under Windows	4
2.1	System Requirements and Recommendations	4
2.1.1	Hardware	4
2.1.2	Software	4
2.2	Software Installation	4
2.2.1	Updating an Old Installation	4
2.2.2	Installing the REAKTOR SERIES Software	5
2.2.3	Installation with and without the Key File	5
2.2.4	Installed Folders, Files and Shortcuts	5
2.2.5	Unlocking the Installation.....	6
2.2.6	Serial Number and Registration.....	6
2.2.7	Copy Protection with the Installation CD.....	6
2.3	Soundcard Settings	7
2.3.1	Devices	7
2.3.2	Latency	8
2.4	MIDI Interfaces	9
2.4.1	Using REAKTOR SERIES with a Software Sequencer	10
2.5	Uninstalling the Software	11
3	Installation under Mac OS.....	12
3.1	System Requirements and Recommendations	12
3.1.1	Hardware	12
3.1.2	Software.....	12
3.2	Software Installation	12
3.2.1	Installing the REAKTOR SERIES Software	12
3.2.2	Installation with and without the Key File	13
3.2.3	Unlocking the Installation.....	13
3.2.4	Serial Number and Registration.....	13

3.2.5	Copy Protection with the Installation CD.....	13
3.3	Audio Output Properties	13
3.3.1	Latency	14
3.4	MIDI Input.....	14
3.4.1	Using REAKTOR SERIES with a Software Sequencer	14
3.5	Uninstalling the Software	14
4	First Steps in  GENERATOR	15
4.1	Opening and Playing Example Ensembles	15
4.1.1	Synth1	16
4.1.2	Padecho.....	17
4.1.3	FM Overdrive	18
4.1.4	16-Step-Sequencer plus Bassline.....	19
4.1.5	Sample Loop Player	20
4.2	Your First DIY Synthesizer	21
4.2.1	Preparation	21
4.2.2	Choice of Components.....	22
4.2.3	Wiring.....	24
4.2.4	Arranging the Panel.....	25
4.2.5	Saving	25
4.2.6	A Touch of Luxury	25
4.3	Your First Self-Made Structure	26
4.3.1	Building the Basic Structure	26
4.3.2	How Does It All Work?	27
4.3.3	Adding a Resonant Filter	27
4.3.4	Filter's Function.....	28
4.3.5	Adding Key Tracking	28
4.3.6	Adding a Filter Envelope	28
4.3.7	Recording the Sound.....	29
4.3.8	Variations	30
5	The  TRANSFORMATOR Tour.....	31
5.1	Practicing with Diletant.....	31
5.2	Percussion with Impaktor	35
5.3	Playing instruments with Simulant.....	39
5.4	FM and WaveSets with Stimulant and Vibrator.....	41
5.5	Playing Samples like WaveSets using Diktaphon	44
5.6	A tour through samples with Loopo, Plasma and Kompressor	44
5.7	4Dex.....	48

6	Basic Operation	50
6.1	Mouse.....	50
6.2	Context Menus.....	51
6.3	Key Commands	51
6.4	Windows.....	51
7	Menus.....	53
7.1	File Menu	53
7.1.1	New Ensemble	53
7.1.2	Open... ..	53
7.1.3	Save Ensemble	53
7.1.4	Save Ensemble As... ..	53
7.1.5	Save Window As... ..	53
7.1.6	Load Audio.....	54
7.1.7	Reload Audio	54
7.1.8	Export Audio... ..	54
7.1.9	Overwrite Audio	54
7.1.10	Batch Analyse.....	54
7.1.11	List of Recently Opened Ensemble Files.....	55
7.1.12	Exit	55
7.2	Edit Menu.....	55
7.2.1	Undo.....	55
7.2.2	Redo.....	55
7.2.3	Cut	55
7.2.4	Copy.....	56
7.2.5	Paste.....	56
7.2.6	Delete.....	56
7.2.7	Select All	56
7.2.8	Solo	56
7.2.9	Mute	56
7.2.10	Mono.....	57
7.2.11	Properties.....	57
7.3	Insert-Menu	57
7.4	Settings-Menu	57
7.4.1	Sample Rate	57
7.4.2	Control Rate.....	57
7.4.3	External Sync.....	57
7.4.4	Clock Start	58
7.4.5	Clock Stop.....	58
7.5	System Menu	58

7.5.1	Run/Stop Audio.....	58
7.5.2	Measure CPU Usage	58
7.5.3	Audio Port.....	59
7.5.4	Audio Settings.....	59
7.5.5	MIDI Settings.....	59
7.6	Preferences	60
7.6.1	Files.....	60
7.6.2	Processor Usage	60
7.6.3	Appearance	60
7.6.4	Directories.....	61
7.7	Window	61
7.7.1	Cascade.....	61
7.7.2	Tile Horizontally	61
7.7.3	Tile Vertically.....	61
7.7.4	Arrange Icons	61
7.7.5	Store Snapshots.....	61
7.7.6	Learn.....	61
7.7.7	Lock.....	62
7.7.8	Properties.....	62
7.7.9	Goto Panel	62
7.7.10	Goto Structure	62
7.7.11	Goto Parent	62
7.7.12	Show / Hide Toolbar	62
7.7.13	Show / Hide Hints.....	62
7.7.14	Show Ensemble.....	62
7.7.15	Close All Panels.....	62
7.7.16	Close All Structures	62
7.7.17	All Panels to Front	63
7.7.18	List of Open Windows	63
7.8	Help Menu.....	63
7.8.1	Help.....	63
7.8.2	About.....	63
7.9	Context Menu	63
8	Toolbars	64
8.1	Ensemble Toolbar	64
8.2	Instrument Toolbar.....	65
9	Ensemble	67
9.1	Ensemble Structure	67
9.1.1	Audio-In Module.....	68

9.1.2	Audio-Out Module.....	68
9.2	Ensemble Panel.....	68
10	Instruments	70
10.1	What is an Instrument?	70
10.2	Creating Instruments	70
10.3	Ports.....	70
10.4	Context Menu.....	71
10.5	Properties	71
10.5.1	Label and Info	71
10.5.2	Number of Voices	71
10.5.3	MIDI-Parameters	73
11	Macros.....	75
11.1	What is a Macro?	75
11.2	Creating Macros	75
11.3	Ports.....	76
11.4	Context Menu.....	76
12	Structures	77
12.1	What is a Structure?	77
12.2	Modules	78
12.2.1	Creating	78
12.2.2	Ports.....	79
12.2.3	Mono.....	80
12.2.4	Mute	80
12.2.5	Cut, Copy & Paste.....	81
12.2.6	Delete.....	81
12.2.7	Properties.....	81
12.3	Sources.....	81
12.3.1	What are Sources?.....	81
12.3.2	Control Sources.....	82
12.3.3	MIDI Sources	82
12.3.4	Range of Values	82
12.3.5	Step.....	83
12.3.6	Constant Sources.....	84
12.4	Switches	84
12.5	Terminals.....	84
12.6	Wires	85

12.6.1	Creating	85
12.6.2	Deleting.....	86
12.6.3	Rules for Wiring	86
12.6.4	Hints.....	86
12.7	Context Menu.....	87
13	Panel Editing	88
13.1	What is a Panel?	88
13.2	What are Controls?	88
13.3	Panel Controls	88
13.3.1	Faders.....	88
13.3.2	Knob.....	89
13.3.3	Button.....	89
13.3.4	Switch	89
13.4	Editing the Panels	90
14	Panel Operation	91
14.1	Mouse Control	91
14.1.1	Fader.....	91
14.1.2	Knob.....	91
14.1.3	Button.....	91
14.1.4	Switch	91
14.2	Key Control	91
14.3	MIDI Control	92
14.3.1	MIDI Data Types.....	92
14.3.2	MIDI Learn	92
14.3.3	Incremental	93
14.3.4	Soft Takeover	93
14.4	MIDI Out.....	93
14.5	Snapshots	94
14.5.1	What are Snapshots?.....	94
14.5.2	Recalling	94
14.5.3	Storing.....	94
14.5.4	Deleting.....	95
14.5.5	Copying and Renaming.....	95
14.5.6	Snap Isolate.....	96
15	Sampling and Re-Synthesis in ② TRANSFORMATOR	97
15.1	Sample Management	97
15.1.1	Sound Files and Samples	97

15.1.2	Multiple Use of Identical Samples	98
15.1.3	Missing Samples	98
15.1.4	Sample-Backup in the Module	98
15.1.5	Sample-Analysis.....	98
15.1.6	Sample-Editors	99
15.2	Sample-Maps	99
15.2.1	Multi-Sampling.....	99
15.2.2	Drum-Maps.....	100
15.3	Sampler Properties Dialog window	100
15.3.1	The Sample Map Field	102
15.3.2	The Selected Sample Field	102
15.3.3	The Module field	103
15.3.4	The Quality Field.....	103
15.3.5	The Direction Field.....	104
15.3.6	The Loop Field.....	104
16	Appendix.....	105
16.1	Trouble Shooting.....	105
16.1.1	Possible Problems.....	105
16.1.2	If All Else Fails.....	109
16.2	Audiowerk8	110
16.2.1	Driver Installation	110
16.2.2	Settings	111
16.2.3	Operation	111
16.2.4	Driver De-Installation	111
16.3	Watching the CPU Load	112
16.4	Key Commands	113
16.4.1	Musical Note Keys.....	114
16.5	4Control MIDI Unit	115
17	Glossary of Synthesizer Terms.....	116
18	Index	133

1 Introduction

1.1 What is REAKTOR ?

REAKTOR is a piece of software that turns your computer and soundcard into a powerful synthesizer and audio processing system. Its completely modular structure ensures that no limits are imposed on your imagination in the creation of electronic musical instruments and sound effects. From the simulation of relatively simple, analog synthesizers as well as large, complex modular systems, through sample players and FM synthesis up to exotic methods such as delay-line resonance or granular synthesis – REAKTOR will show itself to be capable of fulfilling all your desires.

REAKTOR's ability to take on many different forms is the first major difference compared to hardware synthesizers and is its major advantage. "How do you want to sound today" would certainly be a good slogan for REAKTOR.

But even if the construction of synthesizers is not amongst your preferred activities, you have still made a good choice with REAKTOR. Together with the program you get a bunch of ready-made instruments that allow you to dedicate yourself without much ado to your favorite occupation: making music.

And finally: You never know what will happen. Because of the way it's structured, REAKTOR always lets you take a look "behind the scenes" – another difference to hardware solutions where opening the cabinet rarely leads to an increased understanding of the internal workings, it just invalidates your warranty. It could be that you will find these insights so fascinating that after some time you, who maybe always believed yourself to be someone with the proverbial two left hands, find yourself unexpectedly a member of the guild of synthesizer designers.

1.2 What is GENERATOR ?

GENERATOR is a modular synthesizer. It has all the power that results from REAKTOR's flexible modular structure, but it comes without TRANSFORMATOR's sample playback and resynthesis functions.

1.3 What is TRANSFORMATOR ?

TRANSFORMATOR is a modular sampler. It also has all the power that results from REAKTOR's flexible modular structure, but it comes without GENERATOR's large range of audio oscillators which are used for building synthesizers.

1.4 Product Specific Texts

This User's Guide covers all three products in the REAKTOR SERIES, that is GENERATOR, TRANSFORMATOR and REAKTOR.

If you own REAKTOR, then you have all the features described in this guide, so everything always applies. If you own one of the other two, however, some parts of the text may not apply. They are marked like this:

 Text passages that are marked with the GENERATOR icon only apply to GENERATOR and REAKTOR, but not to TRANSFORMATOR. When talking about features that are not in TRANSFORMATOR, the product is called GENERATOR, but REAKTOR is also meant.

 Text passages that are marked with the TRANSFORMATOR icon only apply to TRANSFORMATOR and REAKTOR, but not to GENERATOR. When talking about features that are not in GENERATOR, the product is called TRANSFORMATOR, but REAKTOR is also meant.

The rest of the text applies to all three, and the product is called REAKTOR SERIES or simply REAKTOR.

Similarly, depending on whether you are running the software on a Macintosh or on a Windows PC, different parts of the text may apply:

 Text passages that are marked with the Windows icon, only apply when using the software with Windows 95, Windows 98 or Windows NT.

 Text passages that are marked with the MacOS icon, only apply when using the software with MacOS.

1.5 Getting Started

One more word about this guide: We definitely recommend that after installation you first consult the chapter "First Steps", even if you are already an experienced synthesist. There you will learn much about how you can work quickly, efficiently and economically with REAKTOR and your initial attempts at creating something will be that much easier.

1.6 Support

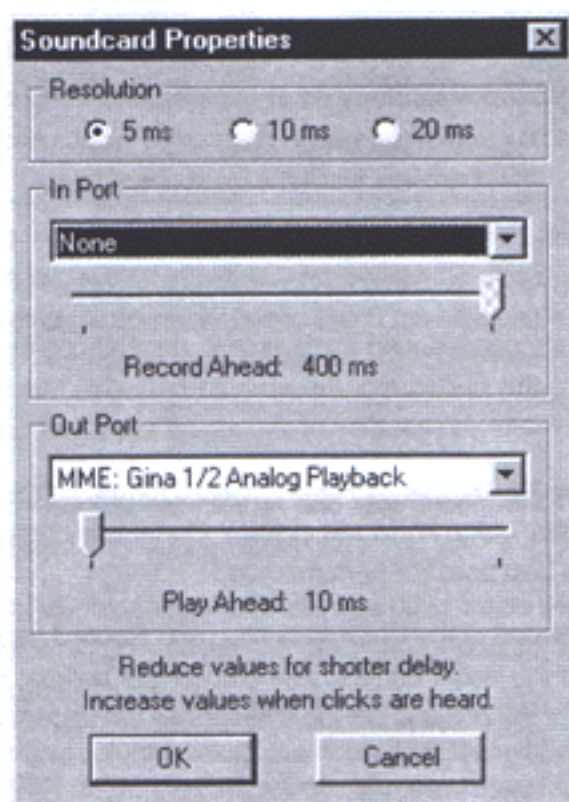
NATIVE INSTRUMENTS is a young, independent company founded with a vision – to bring innovative, cutting edge software to musicians and empower them to make new sounds using the latest technology.

2.3 Soundcard Settings

The REAKTOR SERIES software needs a soundcard for playing the sounds it produces. In addition, the soundcard's inputs can be used to process external audio signals live (REAKTOR SERIES as effects unit) or to sample them.

To use a standard soundcard as the audio interface for the software no extra drivers are needed. The REAKTOR SERIES software makes use of the standard WaveAudio or DirectX drivers installed with the cards.

However, some adjustment of the REAKTOR SERIES software's parameters is necessary to tune it to the hardware and achieve optimum performance. Choose **Soundcard** in the **System⇒Audio Port** menu and open the appropriate settings dialog window by selecting **System⇒Audio Settings....** You can also open this dialog window by double-clicking on the **Audio In** or the **Audio Out** module.



Audio Settings dialog window for standard soundcards

2.3.1 Devices

If more than one soundcard (or driver port) is installed, the selection boxes **In Port** and **Out Port** allow the choice of which soundcard (or which driver port) is to be used by the REAKTOR

SERIES software. If the audio inputs of a soundcard are to be used, it must support 16-bit full duplex operation. Also, the In Port and Out Port that the software uses must be on the same card. Otherwise smooth operation cannot be guaranteed.

In the list of available ports for **Out Port** the available devices are marked **MME:** or **DirectSound:**. The latest DirectSound drivers for your soundcard are probably better optimized with regard to latency (delay) in the audio output than the earlier MME (WaveOut) drivers and normally perform better. We recommend that you try out all the available drivers, see what latency can be achieved with each and then use the one that performs best.

Do not use emulated DirectSound drivers (which are usually marked "emulated") because these are actually MME drivers made to look like DirectSound and will perform worse than all other options.

Precondition for using DirectSound is that the Windows extension DirectX 5.0 (or later) is installed on your PC. Unfortunately, there are no DirectX drivers available yet for the **In Port**.

Note: The **low latency** technology employed in REAKTOR SERIES software makes high demands on **soundcard drivers**. Many drivers, especially **older** ones, are not able to cope and cause system crashes or incorrect operation particularly when using **audio input**. Please make sure you have the very **latest drivers** available for your soundcard.

2.3.2 Latency

The delay (latency) that occurs during audio output depends on the size of the audio buffer that the software passes to the soundcard. For smooth operation this buffer must have a minimum length which depends mainly on the type of soundcard and driver used.

If you are installing the REAKTOR SERIES software for the first time you can skip the settings described below and first get to know the system. Come back here later to optimize the latency so that you will get the best possible performance.

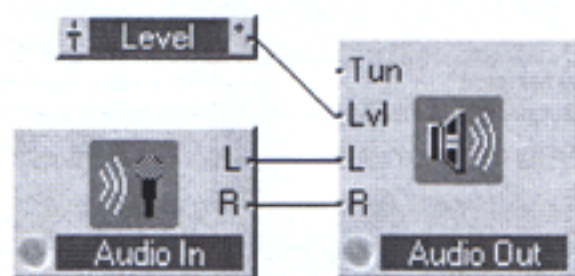
With the sliders **Play ahead** of the Out Port section and **Record ahead** of the In Port section you can adjust the size of the audio buffers. The smaller the buffers, the faster is the response of REAKTOR. However, if the buffer is set too small, clicks will appear in the audio output.

Important: To get good **performance** from the REAKTOR SERIES software you must optimize the **Play ahead** by hand every time you change your **soundcard** or update the soundcard driver.

You can optimize the buffer length for sound output on your system like this: Open an ensemble and play it while at the same time moving the slider for **Play ahead** in the **Soundcard Properties** dialog window. Move the slider to the left to reduce **Play ahead** until you start to hear clicks in the sound output. Now move it back to the right a bit to find the point where the clicks disappear. Now you have found the minimum output buffer size for your card.

When using MME drivers, the sound will break up when **Play ahead** is too small, with DirectSound drivers on the other hand, there will only be one glitch after which the effective latency suddenly becomes very large (about 1 second).

You can find the optimum position for the **Record ahead** slider in a similar way. First draw a wire from the **Audio-In** module to the **Audio-Out** module in the ensemble structure window, connect an audio source (e.g. CD player) to the soundcard input and listen to the soundcard output as usual. Now move the slide for **Record ahead** to the left until clicking starts. Then slowly move it back to the right until the sound become clean. You have now found the optimum for the audio inputs of your soundcard.



Connection for optimizing **Record ahead**

The **Timing Resolution** for arriving MIDI events can be switched between 5 ms, 10 ms and 20 ms. A setting of 5 ms delivers the most precise relative timing (smallest variation in delay). You should only change the setting to 10 ms or 20 ms if at 5 ms noticeably more clicking or stuttering occurs. However, at the highest resolution (5 ms) the CPU load caused by the driver is somewhat higher. If timing accuracy is not so important but CPU overload is causing problems, the resolution may be decreased.

When the soundcard is configured well, the **In** and **Out** level meters in the Ensemble Toolbar appear in green/red, otherwise they appear gray.

By the way, the number of voices and the sample rate used inside the software don't have any affect on either latency or timing resolution, but they do, of course, affect overall performance through the CPU load caused.

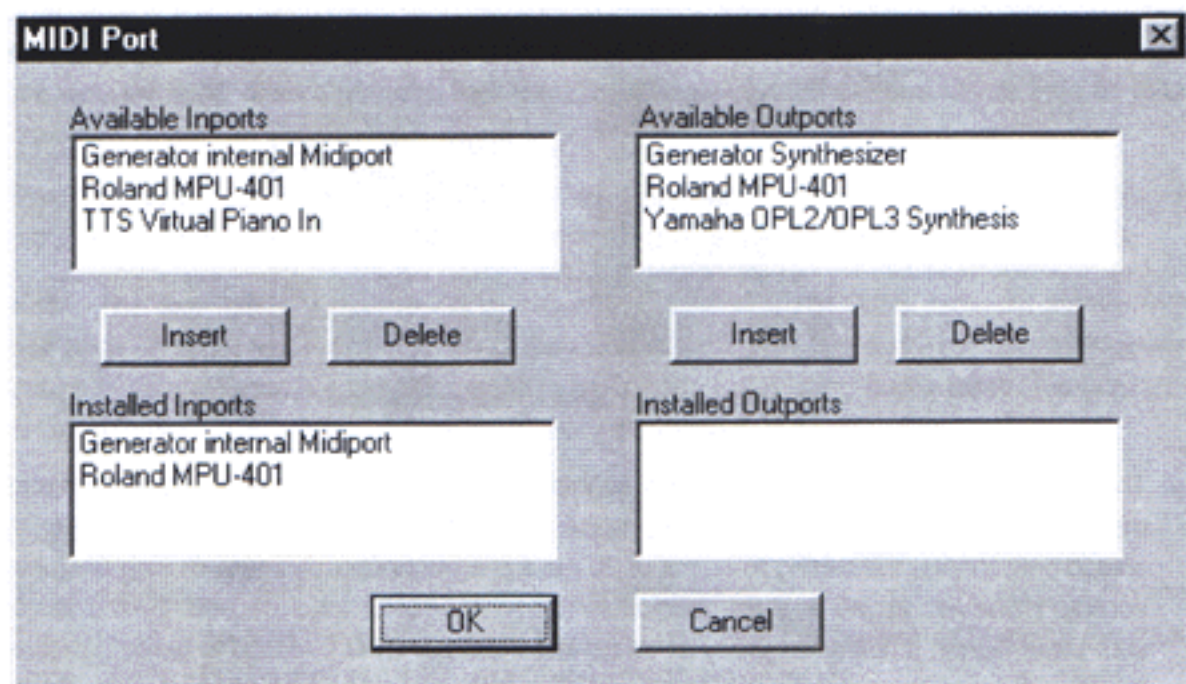
Please note that the input and output levels of the soundcard depend on the settings of the mixer on the card. You can control this device using the Windows accessory **Volume Control**, the **Multimedia Properties** of the **Control Panel** or a mixer program, delivered with the soundcard.

2.4 MIDI Interfaces

The MIDI Ports through which the REAKTOR SERIES software is to communicate with the rest of the world are selected in the **MIDI Port** dialog window, which is opened via the menu entry **System⇒MIDI Settings....** All the existing MIDI ports in your PC which are installed under

Windows appear here and can be chosen for use with the REAKTOR SERIES software. For details see section 7.5.5.

If a MIDI input port has already been opened by another program, the software will not be able to use it and it will not appear under **Available Inputs**. In this case free up the port in the other program or make sure that the REAKTOR SERIES software starts up first. Conversely, an input has to be removed from the list of **Installed Inputs** before another program has access to it.



MIDI Port dialog window

2.4.1 Using REAKTOR SERIES with a Software Sequencer

You can control the REAKTOR SERIES software with another piece of **MIDI** software such as a sequencer, running on the same computer. The driver **genmidi.dll** that provides the internal MIDI input and output ports which are required for this has already been copied to your computer and registered as a MIDI driver during the installation procedure under Windows 95/98. This driver does not work with Windows NT, however, where you will need a special MIDI loop-back device designed for NT (e.g. MIDI Yoke).

genmidi.dll provides the MIDI out-port **Generator Synthesizer** for use by other MIDI programs such as your sequencer. MIDI events routed to this port are passed to the REAKTOR SERIES software inside the computer. You can find this interface as **Generator internal Midiport** in the list of **Available Inputs**.

If you want to disable this internal connection to other MIDI programs, simply remove the entry **Generator internal Midiport** for the list of **Installed Inports** in the **MIDI Port** dialog window (**System⇒MIDI Settings...**) by selecting it and pressing the **Delete** button.

If you want to connect the REAKTOR SERIES software's MIDI output to the input of your sequencer you will need another driver (for example the free Hubi's LoopBack device) to forward MIDI events inside the computer.

2.5 Uninstalling the Software

If you want to remove the software installation from your computer completely, we recommend the following procedure:

- Open **Start⇒Settings⇒Control Panel⇒Add/Remove Programs**
- On the page **Install/Uninstall** choose the REAKTOR SERIES software from the list of installed programs.
- Click on **Add/Remove** and confirm deletion by clicking on **Yes**.

You should also remove the internal MIDI driver. Proceed like this:

- Start the program **sysedit** (**Start⇒Run...⇒sysedit**).
- Click on the window **C:\Windows\System.ini** and look for the section labeled **[drivers]**.
- In the section **[drivers]** remove the line **midix=genmidi.dll**, where the **X** in **midix** stands for the number occupied by **genmidi.dll**.
- If entries with higher numbers exist, change them so that the result is again a continuous sequence
- Save the changed version of **C:\Windows\System.ini**.

6 Basic Operation

The REAKTOR SERIES user interface follows the conventions of the operating system, so it is easy to get used to the software for someone who has already worked with  MacOS or  Windows 95/98. Nevertheless we want to explain some particular characteristics and draw your attention to some features that may be new to you.

6.1 Mouse

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

- **Selection** of an object by clicking on it with the left mouse button. Selected objects are recognized by their label field which is colored red. If you want to select several objects, hold down the **Ctrl** key ( Windows) or the **Shift** key ( MacOS) on the computer keyboard while clicking on the desired objects one after the other. Alternatively you can click with the left mouse button on a blank part of the window and open a frame by dragging with the button pressed. All objects within the frame are selected.
- **Moving** an object is achieved by clicking the left mouse button on it and keeping the button pressed while moving the mouse pointer, and with it the object, to the desired location. To move several objects together, first select all the desired objects, and then move one of the objects as above. All of them will move together according to the mouse movement and all the wiring remains and follows like rubber bands. On releasing the mouse button the modules are aligned on a grid and then remain at the new position. The grid helps to ensure a tidy appearance.
- **Wires** are drawn by clicking with the left mouse button on the out-port whose signal is to be transmitted. Then you move the mouse pointer, and with it the wire, to the desired in-port of another module that is to receive the signal. There you perform another click with the left button - the connection is now established. The wiring operation can also be carried out in the other direction (in-port to out-port) or by dragging with the left mouse button held - the result is the same.
- **Double-click** with the left mouse button on an object (or on the background of windows) and one of various actions is performed, depending on the object. The object's context menu shows which action it is that is executed on a double-click by listing the corresponding menu entry in bold type
- **Right mouse button** ( Windows) or **ctrl** key + mouse button ( MacOS) opens the context menu that belongs to the object (or window) on which the button was clicked. Context menus play a very important part in the REAKTOR SERIES operation which is why the next section is dedicated to explaining them in detail.

6.2 Context Menus

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

Objects which have context menus are:

- Modules in the structure
- Controls in the panel
- Input and output ports of modules
- Structure windows (background)
- Panel windows (background)

6.3 Key Commands

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

6.4 Windows

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

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

- As many windows as you like can be open and displayed on the screen at any time.
- Opening windows and jumping to other windows is performed through either the **Window** menu, a context menu or a keyboard shortcut.
- **Goto Structure** leads to the window that shows the internal wiring of the module. Keyboard shortcut, from the panel: **Ctrl + B** (Windows), **⌘ + B** (MacOS).
- **Goto Panel** opens the window with the control elements for the Instrument. Keyboard shortcut, from the panel: **Ctrl + B** (Windows), **⌘ + B** (MacOS).
- **Goto Parent** shows the structure window that is above the current window in the structural hierarchy. Keyboard shortcut: **Ctrl + P** (Windows), **⌘ + P** (MacOS).
- All open windows are shown in a list at the bottom of the **Window** menu. A window can be brought to the front by selecting its entry in the menu.

- You can move, size, minimize and close the windows just like you are used to from other programs. If a window is too small to display its entire contents, you will find the usual scrollbars at its right and bottom edges and you can use them to move around the window contents.

The following applies under  Windows:

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

7 Menus

The commands for operating the REAKTOR SERIES system are to be found, apart from the various context menus (see section 6.2), in the menus in the menu bar of the main window. The program's global functions controlled from there are described below.

7.1 File Menu

7.1.1 New Ensemble

The menu item **New Ensemble** creates an empty **Ensemble**. It consists of only the **Audio-In** and the **Audio-Out** modules. The empty ensemble is loaded from the file **New.ens** in the installation's **Bin** folder. If you would prefer to have a different structure as the basis for new ensembles, you can replace **New.ens** with another ensemble, i.e. save the desired ensemble with the name **New.ens** in the folder **Bin**.

7.1.2 Open...

By selecting **Open...** you can load one of three kinds of files:

- **Ensemble** (filename extension **.ens**).
- **Instrument** (filename extension **.ism**).
- **Macro** (filename extension **.mdl**)

7.1.3 Save Ensemble

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

7.1.4 Save Ensemble As...

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

7.1.5 Save Window As...

The instrument or macro in the active window can be saved here. The same function is also available as **Save As...** in the context menu of the macro or instrument, respectively.

7.1.6 Load Audio...

An audio file can be loaded to a selected Sampler or Tapedeck module from this menu. The same function is also available as **Load Audio...** in the context menu of the Sampler or Tapedeck, respectively.

7.1.7 Reload Audio

An audio file which has already been loaded by a Sampler or Tapedeck module can be reloaded here, e.g. if it has been changed using an editor. The same function is also available as **Reload Audio...** in the context menu of the Sampler or Tapedeck, respectively.

7.1.8 Export Audio...

From this menu, an audio file can be saved from a selected Tapedeck module. The same function is also available as **Export Audio...** in the context menu of the Tapedeck.

7.1.9 Overwrite Audio

An audio file can be saved from a selected Tapedeck module, overwriting a previously saved audio file, e.g. if the Tapedeck now contains a new recording. The same function is also available as **Overwrite Audio...** in the context menu of the Tapedeck.

7.1.10 Batch Analyse...

This function can do the analysis of sound files, which is necessary for some modules in TRANSFORMATOR / REAKTOR, on whole sound libraries. Usually there is some waiting time during the import of sound files which have not been analyzed before. This wait can be avoided if all the sound files of a library are analyzed in one go using **Batch Analyse...** This time consuming process does not require any user interaction and can be done while you are away or overnight.

A standard File Open Dialog opens, prompting you to specify, a sound file from which the batch process should start. **Batch Analyse...** will analyze this sound file and all other sound files with *write access* in the same directory and all its subdirectories. After finishing the batch analysis, the program tells you the numbers of files found and the number of files analyzed.

Windows: Sound files, which were copied from CD-ROM to hard disk are write protected by default. If you want to save yourself the trouble to remove the write protection for each individual file in all of the subdirectories, you can use the DOS-command **attrib**. Let's assume that you want to remove the write protection in the directory **d:\soundfiles** including all subdirectories. With **Start→Programs→MS-DOS-Prompt** you descend to your computer's basement. Type **cd d:\soundfiles** and then **attrib -r d:\wavefiles* /S**.

Caution: When storing the analysis data, soundfiles will be overwritten by TRANSFORMATOR / REAKTOR. Even though the sound waveform data itself does not get changed during the analysis, some auxiliary information, which may be included in the sound files, will get lost if TRANSFORMATOR / REAKTOR does not know what to do with it! This additional data may be required by other programs working with these files. If such problems should occur, please tell us. If in doubt, you should only let TRANSFORMATOR / REAKTOR work on sound files of which you have backup copies.

7.1.11 List of Recently Opened Ensemble Files

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

7.1.12 Exit .

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

7.2 Edit Menu

7.2.1 Undo

Selecting **Undo** from the menu or pressing **Ctrl + Z** reverses the effect of the last editing operation carried out in any of the structures.

The number of steps through which operations can be reversed is determined by the settings in the Preferences (see section 7.6). Undo can also be disabled in the Preferences.

7.2.2 Redo

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

7.2.3 Cut .

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

7.2.4 Copy

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

7.2.5 Paste

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

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

7.2.6 Delete

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

7.2.7 Select All

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

7.2.8 Solo .

The **Solo** function directly connects the output of the selected instrument to the audio output. All instruments which lie upstream of the selected instrument in the structure remain active. All other instruments in the ensemble are muted.

An example: A synthesizer signal is processed by a chorus effect. The chorus effect is switched to **Solo**. Then the synthesizer with the chorus effect is connected with the output, while all other instruments in the ensemble are muted.

This function is equivalent to the entry **Solo** in the context menu of the instrument.

7.2.9 Mute

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

7.2.10 Mono

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

7.2.11 Properties

This opens the **Properties** dialog of the selected object. The same function is also available as **Properties** in the context menu of the selected modules or panel elements.

7.3 Insert-Menu

This menu is used to insert elementary modules or panel elements. It is equivalent to the menu behind **Insert** in the context menu of the panel window or **Modules** in the context menu of the structure window (see chapter 6.2).

7.4 Settings-Menu

7.4.1 Sample Rate

This menu item sets the internal sample rate for audio signals. With higher sample rates you can achieve better sound quality, but the CPU load rises proportionately. If the internal sample rate is different from the soundcard's sample rate (44100 Hz), the Audio In and Audio Out modules will do the conversion. The sample rate can also be adjusted using the selector in the Ensemble Toolbar.

7.4.2 Control Rate

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

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

7.4.3 External Sync

Toggles between the internally generated clock and the external clock (received via MIDI) for all **Sync Clock** and **1/96 Clock** source modules. Control of the **Start/Stop** source by MIDI-

Start/Stop is also activated. When **External Sync** is activated, the tempo cannot be adjusted in the Master Clock Tempo field on the toolbar. It shows **ext** instead of the BPM value.

7.4.4 Clock Start

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

7.4.5 Clock Stop

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

7.5 System Menu

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

7.5.1 Run/Stop Audio

With this menu item the audio computations can be started (**Run Audio**) and stopped (**Stop Audio**). Effectively this is the main on/off switch for the REAKTOR SERIES software. The same function is delivered by a button in the ensemble toolbar and by the **0** in the keypad.

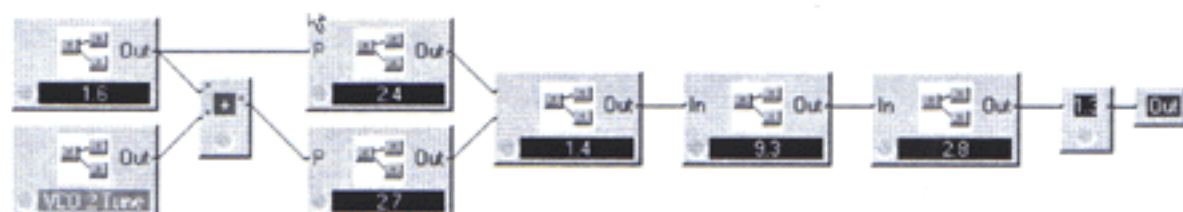
7.5.2 Measure CPU Usage

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

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

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

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



CPU Usage Display (The macro bottom left and the adder only process events)

7.5.3 Audio Port

Windows: This menu item is used to select the soundcard whose outputs are to be used to play the generated sounds (represented by the **Audio-Out** module) and whose inputs (represented by the **Audio-In** module) supply signals for processing by the REAKTOR SERIES software. Cards that are not installed are shown in gray in the menu, those that are installed are shown in black.

7.5.4 Audio Settings...

Windows: This menu item opens a dialog window in which settings for optimizing the performance of your soundcard are made. Detailed instructions for standard soundcards are found in chapter 2.2.7. If you have an EMAGIC Audiowerk8 installed in your computer, please consult chapter 16.2 of this guide.

7.5.5 MIDI Settings...

This menu item allows you to choose which of the MIDI inputs and outputs installed on your computer are to be used by the software.

To select one or more inputs from which the REAKTOR SERIES software should receive data, choose the appropriate in-port from the list of **Available Ports** and insert it into the list of **Installed Ports** by pressing **Insert**. If more than one input is installed, they will be active in parallel. Use **Delete** to remove a selected port from the list. The changes become valid on clicking **OK**.

If you install the driver of a MIDI interface or a soundcard as an Inport to the REAKTOR SERIES software, you can control the software from an external MIDI instrument, e.g. a master keyboard, connected to this input.

If you want to control REAKTOR from a MIDI program (e.g. a software sequencer) running on the same computer, install **Generator internal Midiport** as one of the Inports.

To select one or more outputs to which the software should send data, choose the appropriate out-port from the list of **Available Ports** and insert it into the list of **Installed Ports** by pressing

Insert. If more than one output is installed, the same data will be sent to all. Use **Delete** to remove a selected port from the list. The changes become valid on clicking **OK**.

7.6 Preferences

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

7.6.1 Files

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**. Like this, you can always return to the penultimate version of a file if something should go wrong (by renaming to remove the extension **.bak**).

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

If you activate **Undo Enable**, undo information will be stored before all major editing operations. This gives you the option to reverse the effects of editing operation later if required.

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

If REAKTOR should terminate abnormally (e.g. because of a crash or power failure), the stored undo information allows it to restore the ensemble when the program is next started. If REAKTOR detects this state on startup, it will ask you whether you want to restore the ensemble to the way it was just before the abnormal termination.

7.6.2 Processor Usage

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

7.6.3 Appearance

The color for the background of the structure window can be adjusted here. Clicking on **Change** opens a color palette. With **Reset** you return to the original color.

7.6.4 Directories

Here you can set the paths to the two folders from which macros and instruments are loaded using the context menu. After installing the software, the paths are initially set pointing to the supplied library. If you want to build your own library, you can change these paths to point to the folders in which you keep your library.

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

7.7 Window

7.7.1 Cascade

Gives all windows a standard size and places them in an overlapping arrangement.

7.7.2 Tile Horizontally

Divides the main window horizontally and arranges all windows from top to bottom.

7.7.3 Tile Vertically

Divides the main window vertically and arranges all windows from left to right.

7.7.4 Arrange Icons

Arranges the icons of minimized windows in a row at the lower edge of the main window.

7.7.5 Store Snapshots

Opens a dialog for storing the current settings of an instrument's panel as a snapshot (see also section 15.5).

7.7.6 Learn

Activates the MIDI-Learn mode for the panel of an instrument (see also section 8.2). This mode is automatically deactivated after a MIDI-controller message is received. There is a corresponding button in the instrument toolbar.

7.7.7 Lock

Switches the panel of an instrument into lock mode to prevent inadvertant movements of the panel elements. There is a corresponding button in the instrument toolbar.

7.7.8 Properties

Opens a dialog for setting the properties of the instrument or macro in the selected window. This function corresponds to the entry **Properties** in the context menu of the background of the active window.

7.7.9 Goto Panel

Opens the window with the corresponding panel or brings it to the front.

7.7.10 Goto Structure

Opens the window with the corresponding structure or brings it to the front.

7.7.11 Goto Parent

Opens the window which is one level higher in the hierarchy or brings it to the front.

7.7.12 Show / Hide Toolbar

Here you can choose whether you want to see the toolbar or not.

7.7.13 Show / Hide Hints

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

7.7.14 Show Ensemble

Opens the structure window of the ensemble.

7.7.15 Close All Panels

Closes all panel windows.

7.7.16 Close All Structures

Closes all structure windows.

7.7.17 All Panels to Front

Brings all panel windows to the front.

7.7.18 List of Open Windows

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

7.8 Help Menu

7.8.1 Help

The menu item **Help** starts REAKTOR's online help system.

7.8.2 About

The menu item **About** opens REAKTOR's info window. In its lower part you find the version number of the software and the serial number of your REAKTOR license.

7.9 Context Menu

The context menu of the ensemble (accessed with a right mouse-click on an empty part of the ensemble window) is exactly the same as the context menu of an instrument's structure window. Please see chapter 12.7 for details.

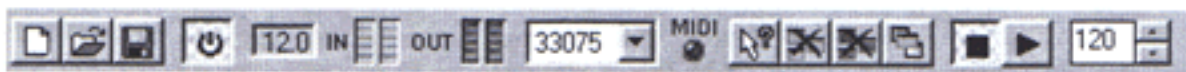
8 Toolbars

■ **MacOS:** The **Toolbar** appears at the top left of the screen. The top half is the **Ensemble Toolbar**, the bottom half is the **Instrument Toolbar**.

■ **Windows:** The **Toolbars** appears at the top of the main window. Above is the **Ensemble Toolbar**, below that the **Instrument Toolbar**. The Toolbars can be "floated" by dragging them by the edge away from their original position. They then appear as windows floating above or beside the main window.

8.1 Ensemble Toolbar

The Ensemble Toolbar contains controls and indicators for the current operational state of the REAKTOR SERIES system.



The Ensemble Toolbar

The Toolbar has the following displays and controls (from left to right):

- With the three buttons **New**, **Open** and **Save** you create a new ensemble, open an existing ensemble, or store the current ensemble with all changes.
- The **main switch** is used to start and stop all audio processing in REAKTOR. It has the same function as the entry **Run/Stop Audio** in the **System** menu. Audio processing can be stopped to reduce the CPU load when no sounds are being played. Switching audio on and off also causes an initialization (reset) of all audio processes.
- The **CPU Load Display** shows the CPU time (in percent) used by the audio process which generates the sound. In case of overload the display shows **Over**. The value that can be reached in smooth, undisturbed operation is normally between 60% and 80%, in any case well below 100%. The reason is that the transfer of audio data to the soundcard, MIDI and event processing, and graphic display also take up some CPU time. Furthermore, enough CPU time must be left over for the operating system and possibly other programs (e.g. a sequencer) to function. You can find the limit for your computer by raising the number of voices for an ensemble until the CPU overload message appears.
- The **In** level meter is used to display the level of the audio input. If it appears gray, no driver is opened as audio input.
- The **Out** level meter is used to display the level of the audio output. If it appears gray, no driver is opened as audio input.

- The selector for the **Sample Rate** displays REAKTOR's internal sample rate, which can be changed by selecting from the list. Which sample rates are available depends on the installed soundcard.
- The global **MIDI** lamp lights up blue whenever REAKTOR receives a MIDI event from one of the installed MIDI in-ports.
- The **Show Hints** button (icon with arrow and question mark) activates hints.
- The **Close All Panels** button (icon with a cross above three light blue windows) closes all panels.
- The **Close All Structures** button (icon with cross above three dark windows) closes all structure windows.
- The **All Panels to Front** button (icon with three panels) raises all open panel windows to the front.
- The **Stop** button stops the Master Clock
- The **Start** button starts the Master Clock
- The **Tempo** selector adjusts the rate of the internal Master Clock in beats per minute (BPM)

8.2 Instrument Toolbar

The Instrument-Toolbar is used to adjust and control an instrument.

The settings apply to the last selected instrument. The instrument can be chosen by selecting the instrument module in the ensemble window, or by selecting its structure window, its panel window, or the structure window of a contained macro.



The Instrument Toolbar

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

- The first element is a list for the selection of the instrument which is to be controlled by the Toolbar. The display will also be updated after the selection of an instrument in the windows.
- The **Panel** button opens the panel or raises the panel of the active window to the front.
- The **Structure** button opens the structure or raises the structure of the active window to the front.
- With the **Info** button, the Instrument Info of the active window is displayed.
- The **Properties** button opens the Instrument Properties.
- The **Solo** button directly connects the output of the selected instrument to the audio output. All instruments which lie upstream of the selected instrument in the structure remain active.

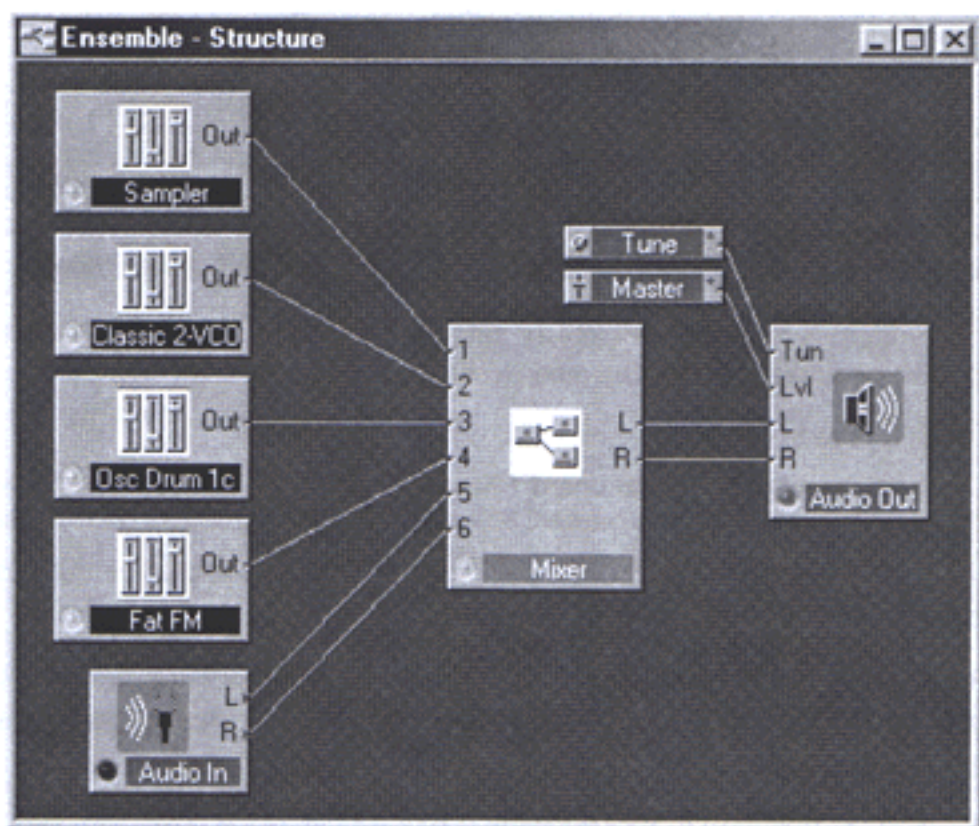
All other instruments in the ensemble are muted. An example: A synthesizer signal is processed by a chorus effect. The chorus effect is switched to **Solo**. Then the synthesizer with the chorus effect is connected with the output, while all other instruments in the ensemble are muted.

- The **Mute** button mutes the selected instrument. All instruments upstream of the selected one are also muted.
- With this selector you can choose a snapshot to recall on the selected instrument.
- With the **Store, Delete, Rename Snapshot...** button (camera as icon) you open the dialog for snapshot management which is described in section 14.5.
- The **Recall Last Snapshot** button recalls the last selected snapshot, discarding all changes.
- The selector for the number of voices displays and changes the number of voices of the selected instrument. The number of voices can also be adjusted in the instrument properties.
- The next selector determines the (maximum) number of unison voices per note. The unison effect is enabled by selecting a value greater than one. The unison voices are detuned by a value which is set with **Unison Spread** in the instrument properties. The minimum number of unison voices per note (**Min Unison Voices**) can be set there, as well.
- The **MIDI Learn** button allows you to easily assign a panel control to a MIDI-controller. You select the panel control, activate the **MIDI Learn** button, and then operate the MIDI Controller, e.g. the modulation wheel. In the **Properties** you can cancel this assignment by deselecting **Remote**.
- The **Instrument MIDI Activity** lamp shows the reception of MIDI events by the selected instruments.
- The **Lock** button protects all panel elements against accidental movement.

9 Ensemble

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

9.1 Ensemble Structure



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

The ensemble's structure window gives a bird's eye view of the entire environment, which consists of a number of *instruments* as well as the audio inputs and outputs representing the soundcard

9.1.1 Audio-In Module

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

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

- **Mute** is used to disable the module.
- **Properties** opens the **Soundcard Properties** dialog window just like **System⇒Audio Settings....**

The **Audio-In** module has the two audio out-ports **L** and **R**. These correspond to the left and right inputs of the selected soundcard.

9.1.2 Audio-Out Module

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

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

- **Mute** is used to disable the module.
- **Properties** opens the **Soundcard Properties** dialog window just like **System⇒Audio Settings....**

The **Audio-Out** module has two event ports for control signals and some audio ports, whose number depends on the chosen soundcard.

The event input **Tun** is used to set the master tuning of the ensemble. At a value of zero, standard tuning is used with the note a3 (MIDI note 69) at 440 Hz. A value of ± 1 changes the tuning by ± 1 semitone or ± 100 cents.

The event input **Lvl** is used to set the master output volume of the ensemble in dB, i.e. negative values reduce the level, positive values increase it.

9.2 Ensemble Panel

Since the ensemble can combine several instruments, its panel can display the controls of several instruments. For this purpose, it is divided into sub-panels. A sub-panel with the label **Ensemble** is always visible. It contains controls of the ensemble's top structure level. For each instrument in the ensemble there is an additional sub-panel with the name of the instrument visible in the title bar.



Ensemble-panel with the sub-panels of six instruments

You can use the switch **Visible in Ensemble** in the **Properties** of the panel elements to determine whether you can see them in the ensemble panel. A second switch, **Visible in Instrument**, allows you to remove a panel element from the panel window of the instrument, as well.

The default setting for instrument's controls is that they are only visible in the panel window of the instrument, since you will typically arrange only a selection of them in the sub-panel in the ensemble panel. An empty sub-panel appears in the ensemble panel as just a title bar with the name of the instrument. Like this you get an overview of all the instruments in the ensemble. You can select one of them for the toolbar by clicking on it with the mouse or you can open its panel window with a double click.

10 Instruments

10.1 What is an Instrument?

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



Instrument Module

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

10.2 Creating Instruments

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

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

10.3 Ports

There is no fixed arrangement of ports for instruments. The type and number of ports in an instrument is determined by the user by the insertion of so-called **Terminals** (see chapter 12.5) in the instrument structure.

The connections into and out of an instrument through the terminals are always **mono**, they cannot be polyphonic. This is necessary because the number of voices inside the instrument will normally be different from that outside - only mono signals will work in any configuration. If the

instrument is polyphonic, a **Voice Combiner** module needs to be inserted before the output port(s) to make sure that the signal leaving the instrument is mono.

10.4 Context Menu

The context menu of an instrument contains eight entries:

- **Mute** disables the instrument.
- **Cut** removes the instrument from its current position. It is copied to the clipboard from where it can be inserted at another place, even in another window, using the **Paste** command.
- **Copy** marks the instrument for copying. A copy of the instrument is stored in the clipboard from where it can be inserted at another place, even in another window, using the **Paste** command.
- **Delete** removes the instrument and everything it contains.
- **Save As...** allows saving the instrument. The file location as well as the filename can be specified. Instrument files are given the filename extension **.ism**.
- **Panel** opens the instrument's panel window. See chapter 13 of this guide for details about the panel.
- **Structure** opens the instrument's structure window to display its internal wiring. See chapter 12 of this guide for details about creating and editing structures.
- **Properties** opens a window with various information about and settings for the instrument. Details are given in the following section.

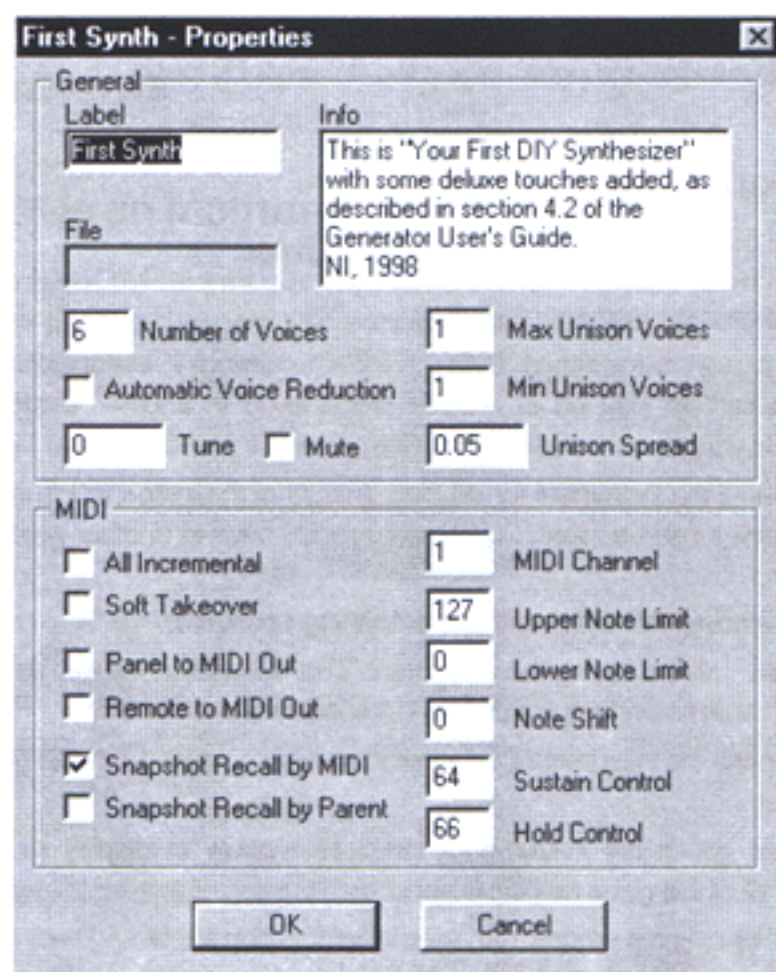
10.5 Properties

10.5.1 Label and Info

In the **Label** field, the instrument can be given a name which appears on the instrument module in the ensemble. A description of the instrument can be entered under **Info**. The text will appear as a hint while the mouse pointer rests on the instrument module in the structure, if hints are switched on. The field labeled **File** shows the name of the file from which the instrument was loaded.

10.5.2 Number of Voices

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



Properties window of an Instrument

If **Automatic Voice Reduction** is activated, REAKTOR can automatically reduce the number of voices when the total CPU load (set in Preferences under **Processor Usage**) exceeds a certain limit. Like this, the polyphony can be adjusted according to the available processing power.

You can use **Tune** to change the instrument's pitch with respect to the tuning set in the ensemble. The value is given in units of semitones. Positive values raise the pitch, negative values lower it. A typical application would be to detune two instruments to get a fatter sound. A good value for this would be **Tune** = 0.05 (5 cents).

With **Mute** the instrument can be deactivated. It then uses no CPU power and is marked with a red cross over its status lamp in the **Ensemble** structure window.

Unison mode is activated if you set **Max Unison Voices** to a value greater than 1. This means that more than one voice plays the same note and the voices are detuned slightly to get a rich, fat sound. **Max Unison Voices** determines the number of voices that are assigned to a newly triggered note when enough voices are still available.

Min Unison Voices is used to set the number of voices assigned to a new note when not enough voices are available. It can never be greater than **Max Unison Voices**.

The parameter **Unison Spread** determines the degree of detuning between the voices sounding in unison. The value is given in semitone steps. A typical value would be 0.05 (5 cents), which would mean that each of the voices playing the same note is detuned by 5 cents relative to the next voice.

10.5.3 MIDI-Parameters

- When **All Incremental** is switched on here in the instrument properties, **Incremental** will be activated in the properties of all the controls belonging to the instrument. When **All Incremental** is switched off in the instrument properties, it will be deactivated in all the contained controls. This is used to put the controls in incremental controller mode, which is necessary to operate REAKTOR with the optional MIDI control unit **4Control** by NATIVE INSTRUMENTS. The **Incremental** mode is detected automaticall when using the **MIDI Learn** function. See section 14.3.3. For details about the 4Control please see section 16.5.
- When **Soft Takeover** is switched on, **Soft Takeover** will be activated for all the controls in the instrument. When **Soft Takeover** is switched off in the instrument properties, it will be deactivated in all controls. This is used to prevent jumps when controlling REAKTOR from external controllers. Please see section 14.3.4 for details.
- When **Panel to MIDI Out** is switched on here in the instrument properties, **Panel to MIDI Out** will be activated in the properties of all the controls belonging to the instrument. When **Panel to MIDI Out** is switched off in the instrument properties, it will be deactivated in all the contained controls. This is used to control whether MIDI events are generated and output by REAKTOR when the controls in the panel are moved. Please see section 14.4 for details.
- When **Remote to MIDI Out** is switched on, **Remote to MIDI Out** will be activated in the properties of all the controls belonging to the instrument. When **Remote to MIDI Out** is switched off in the instrument properties, it will be deactivated in all the contained controls. This is used to control whether MIDI events are output by REAKTOR when the controls are moved by MIDI remote control. Please see section 14.4 for details.
- When **Snapshot Recall by MIDI** is activated, any received MIDI Program Change messages will cause the snapshot with the appropriate number to be called up if it exists. Like this you can quickly recall a snapshot in REAKTOR from your MIDI controller (master keyboard) by issuing a MIDI Program Change command with the appropriate number.
- When **Snapshot Recall by Parent** is activated, the instrument will change its snapshot when the instrument above it in the hierarchy (usually the Ensemble) changes its snapshot. Each snapshot of the parent is linked to one snapshot of the child instrument. The connection is made when a snapshot is stored in the parent, storing the number of the child's currently selected snapshot.

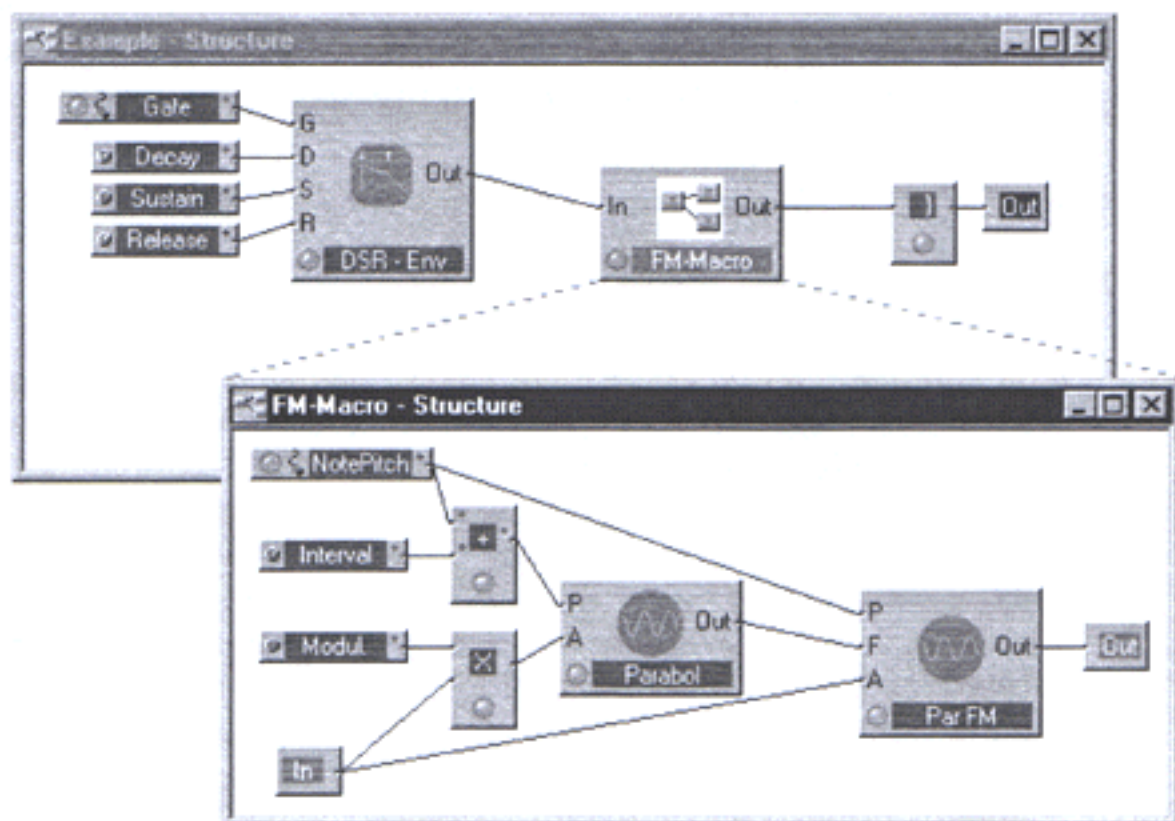
- With **MIDI Channel** the number of the channel (1...16) which is used by the instrument for receiving and sending MIDI data is set. The instrument only processes MIDI data received on the selected channel.
- With **Upper Note Limit** and **Lower Note Limit** the range of MIDI note numbers that the instrument recognizes can be limited. Any note numbers outside the given range are ignored. This can be used for example to program a keyboard split.
- With **Note Shift** all the received MIDI note information is transposed by the given number of semitones. For example, if you want to transpose the whole instrument down by one octave you have to enter the value -12 here.
- With **Sustain Control** the number of the MIDI controller (foot pedal) which works as sustain pedal (called "hold" or "damper pedal" by some MIDI equipment manufacturers, standard controller no: 64) is set. As long as sustain is on (controller value 64 or more) any playing note will be held even after its key is released.
- With **Hold Control** the number of the MIDI controller (foot pedal) which works as hold pedal (called "sostenuto" by some MIDI equipment manufacturers, standard controller no: 66) is set. All notes that are already playing when hold is switched on will be held even after their key is released - until hold is switched off. Keys that are pressed when hold is already on are not affected.

11 Macros

11.1 What is a Macro?

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

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



Example for the integration of a macro into a structure.

11.2 Creating Macros

Macros are added into a Structure (often an Instrument) by loading them from the library which comes with the REAKTOR software. In the folder **Macros** of the library there are numerous

ready-made utility and specialized components. If you want to start developing a new macro you first need to load an empty one from the library (**Macros\New**).

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

11.3 Ports

There is no fixed arrangement of ports for macros, rather the type and number of ports in a macro is determined by the user by the insertion of **Terminals** in the structure. A terminal in the macro's structure appears as a port on the macro's module representation at the next higher level (**Parent**) in the hierarchy. In this way wires connected to the macro's inputs feed it signals which are processed in its internal structure and then passed back to the parent structure through the macro's outputs. For details see chapter 12.5.

11.4 Context Menu

The context menu of a macro contains eight entries:

- **Mute** disables the macro.
- **Mono** switches the macro to monophonic operation by switching all the modules inside to mono. You should always use this function unless the module has to work polyphonically because in mono mode there is significantly less load on the CPU.
- **Cut** removes the macro from its current position. It is copied to the clipboard from where it can be inserted at another place, even in another window, using the **Paste** command.
- **Copy** marks the macro for copying. A copy of the macro is stored in the clipboard from where it can be inserted at another place, even in another window, using the **Paste** command.
- **Delete** removes the macro and everything it contains.
- **Save As...** allows saving the macro. The file location as well as the filename can be specified. Macro files are given the filename extension **.mdl**.
- **Structure** opens the macro's structure window to display its internal wiring. See chapter 12 of this guide for details about creating and editing structures.
- **Properties** opens a window with information about the macro.

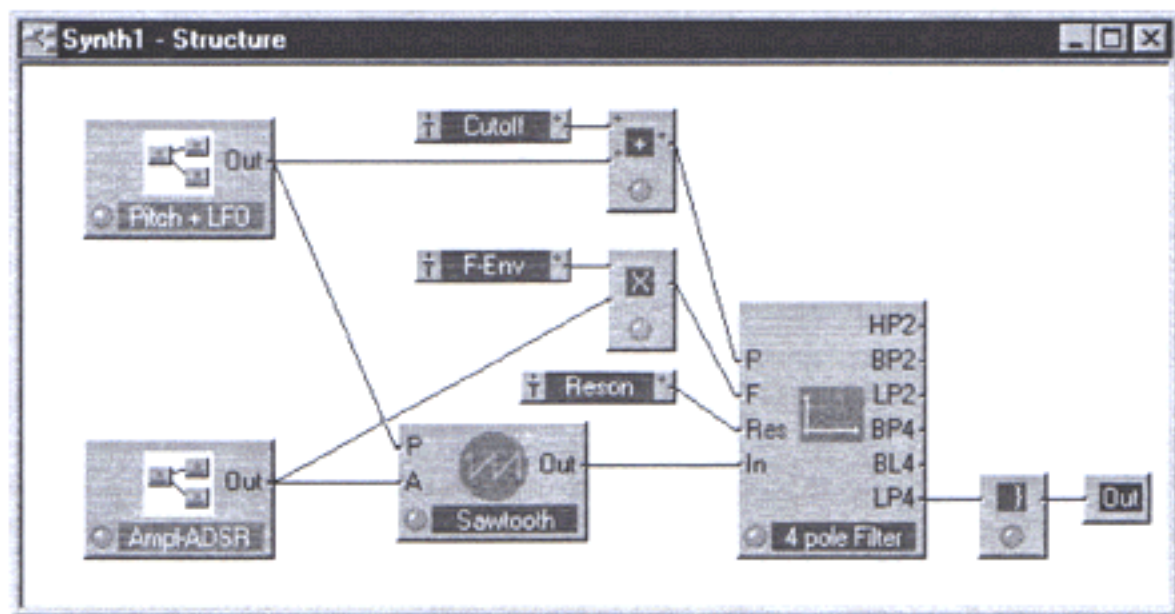
12 Structures

12.1 What is a Structure?

REAKTOR is based on an open concept that allows the design and realization of any imaginable sound generator. In many respects it is similar to the classic modular synthesizer systems. That's why the most important basic building block you deal with in REAKTOR is the **module**.

A library of elementary modules is built into REAKTOR. These provides the basic building blocks for MIDI and audio signal processing. Complex signal processing structures can be created by connecting modules which carry out relatively simple tasks.

A window, where modules are placed and interconnected is called **Structure**



A structure window

When building structures in REAKTOR, keeping to certain hierarchical principles is highly recommended, even if REAKTOR doesn't force these on you but basically gives you complete freedom. For example, it is possible to make a structure using just elementary modules at the ensemble level, i.e. the highest level in REAKTOR. But there are at least two good reasons to avoid this method of construction: firstly, you very quickly loose sight of where you are going, and secondly, the number of elements in a structure is limited (to 100 in the current version).

On the other hand, if you take to heart the following recommendations during the conception and realization of your structures, an optimum of clarity and ergonomics will result even with the most complex layouts.

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

12.2 Modules

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



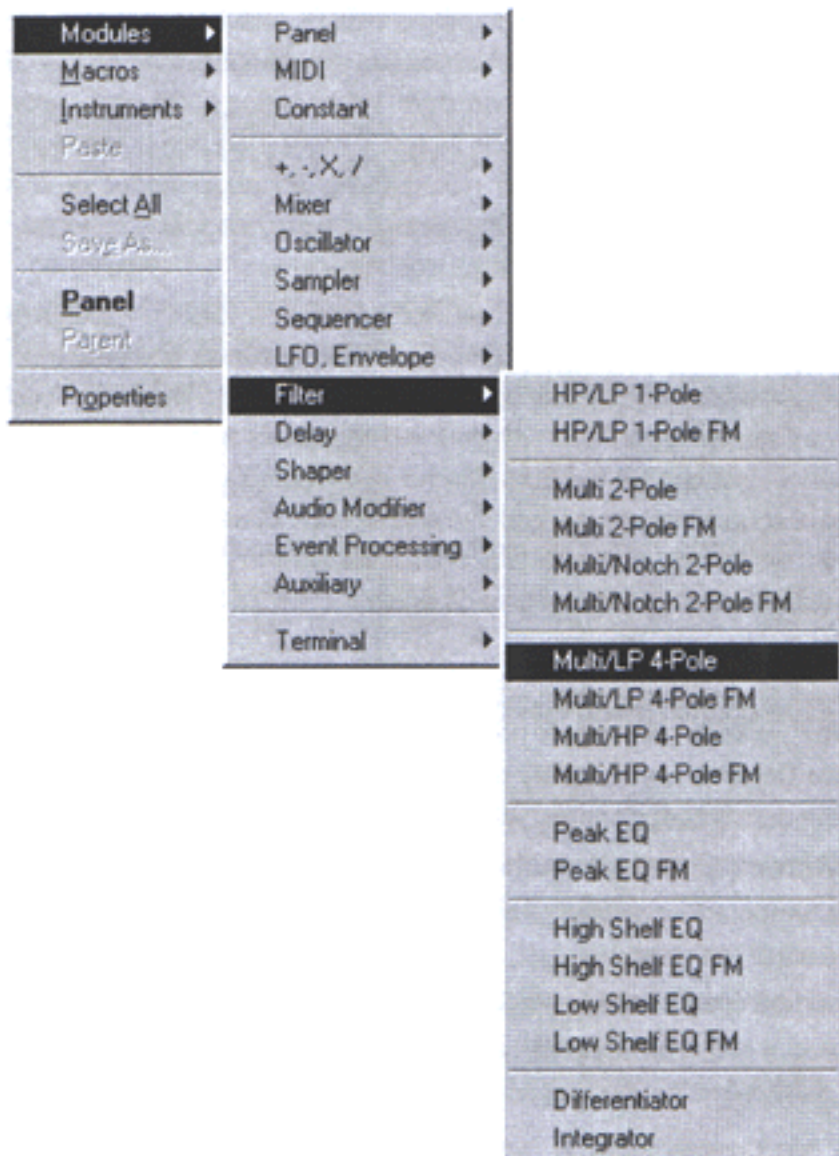
The pulse oscillator module

12.2.1 Creating

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

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

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



Menu for inserting a new module

12.2.2 Ports

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

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

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

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

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

- **Create Control** automatically creates a suitable panel controller for the port. (see section 13.3 for details about working with controllers on the panel).
- **Wire/Unwire** creates a connection from this port to another port.
- **Mute** temporarily deactivates the port, i.e. sets its value to zero. Muted ports are marked with a small red cross.
- **Properties** opens a dialog window with information about the port.

12.2.3 Mono

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

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

12.2.4 Mute

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

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

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

This feature is especially useful when using switches, because alternative branches of signal processing can be selected but only one causes CPU load at any one time. 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 don't cause any unnecessary load on the CPU.

12.2.5 Cut, Copy & Paste

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

- **Copy** marks the module for copying. A copy of the module is stored in the clipboard from where it can be inserted at another place, even in another window, using the **Paste** command.
- You can also use the key combinations **Ctrl + X** (Cut), **Ctrl + C** (Copy) and **Ctrl + V** (Paste). When using the keyboard shortcut **Ctrl + V** for pasting, you can specify a point in the structure by clicking there with the left mouse button first.

12.2.6 Delete

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

12.2.7 Properties

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

12.3 Sources

12.3.1 What are Sources?

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

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

12.3.2 Control Sources

The module types **Fader**, **Knob** and **Button** are the control sources. There are two ways to insert them into a structure:

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

Control sources and their respective panel elements can be controlled via MIDI in various ways. Please see section 14.3 for details about this topic.

12.3.3 MIDI Sources

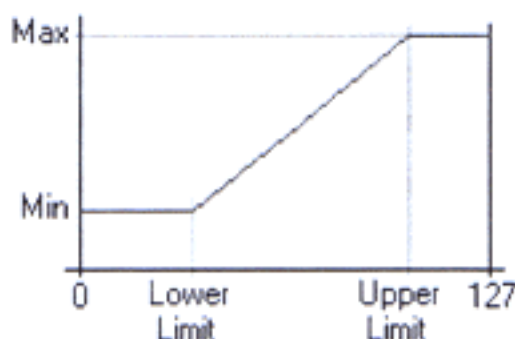
MIDI source modules are used for controlling the audio signal processing with MIDI events. For each type of MIDI event there exists a particular kind of source module. The output signal of such a source corresponds to the values transmitted by the particular MIDI events. The **OnVelocity** source for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

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

12.3.4 Range of Values

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

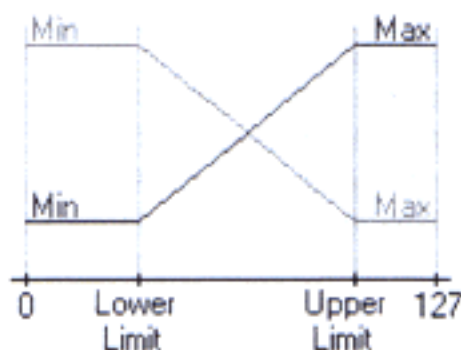
For MIDI sources the range can also be limited with the properties **Lower Limit** and **Upper Limit**. The output value of the source is limited to **Min** for MIDI values below **Lower Limit** and to **Max** for MIDI values above **Upper Limit**. The range between the two limits is interpolated linearly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



Scaling and limiting

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

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



Crossfade

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

12.3.5 Step

The range of values in source modules normally has a resolution of 128 steps. In many modules (particularly Fader and Knob) the parameter **Step** 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 **Step** value of 12.

12.3.6 Constant Sources

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

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

12.4 Switches

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

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

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

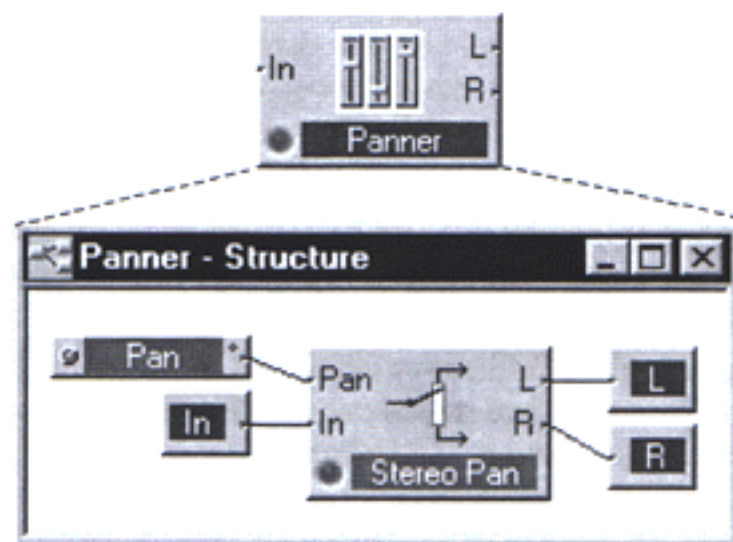
12.5 Terminals

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

According to the different kinds of module ports, four different types of terminal ports are available: **Audio In**, **Audio Out**, **Event In**, **Event Out**. The normal rules for wiring apply (see section 12.6.3), e.g. an **Audio In** terminal can only be connected to an audio in-port of a module.

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

the next higher level (**Parent**), and the description is shown as the hint for the port when the mouse pointer rests on it (provided **Show Hints** is enabled).



Instrument with ports and its structure with terminals

12.6 Wires

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



Connecting a wire

12.6.1 Creating

There are two ways to make a new wire:

- Click on one of the two ports to be connected with the left mouse button, move the mouse pointer to the other port and click this one also. A visible connection (that's the wire) appears between the ports and the effect on the sound resulting from the change to the structure can be heard immediately. The wiring operation is aborted if you click the left mouse button somewhere other than on a valid port, e.g. on an empty part of the window.

- Click on the entry **Wire/Unwire** in the context menu of the port. Then move the mouse pointer to the other port and click the left mouse button on it. Again, the wiring operation is aborted if you click the left mouse button somewhere other than on a valid port, e.g. on an empty part of the window.

12.6.2 Deleting

There are two ways to delete a wire:

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

12.6.3 Rules for Wiring

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

- A wire can only connect an out-port to an in-port.
- An out-port can be connect to up to 16 in-ports.
- When no wire is connected to an in-port, it receives a zero signal.

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

- An **event in-port** cannot process **audio** signals. If an event in-port is to be fed from an audio out-port, the signal must first be converted with an **A to E** module (see Module Reference).
- An **event in-port** can be fed from up to 16 **event out-ports**. If there is more than one connection, the event signals are merged so that it is always the last received event that determines the value.
- An **audio in-port** can only be fed from a single out-port.
- An **event out-port** can be connected to **audio in-ports** as well as **event in-ports**.
- When connecting a **mono** output to a **poly** input all voices receive the same signal. For pitch signals this means the voices in effect play in unison.
- A **poly** output cannot be connected to a **mono** input (a red cross appears on the in-port). A **Voice Combiner** module must be used for converting poly to mono.

12.6.4 Hints

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

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

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

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

12.7 Context Menu

The context menu of the structure window has nine entries:

- **Modules** is for inserting elementary modules into the structure.
- **Macros** is for inserting macros from the library into the structure.
- **Instruments** is for inserting instruments from the library into the structure.
- **Paste** inserts a previously cut or copied object into the structure at the point where the context menu was opened. When using the keyboard shortcut **Ctrl + V** for pasting, you can specify a point in the structure by clicking there with the left mouse button first.
- **Select All** selects all the objects in the structure.
- **Save As...** is for saving the structure to a file with a new name. Depending on the type of structure (instrument or macro) the correct filename extension will be appended.
- **Panel** opens the panel window which belongs to the structure. For macros, the panel command opens the panel window of the instrument or ensemble of which the macro is a part.
- **Parent** opens the structure of the hierarchical level above. For example, if you are in the structure of a macro which is part of an instrument, the parent command will open the instrument's structure.
- **Properties** opens a dialog window with settings and a description of the instrument or macro to which the current structure belongs. See section 10.5 for details about the properties of instruments.

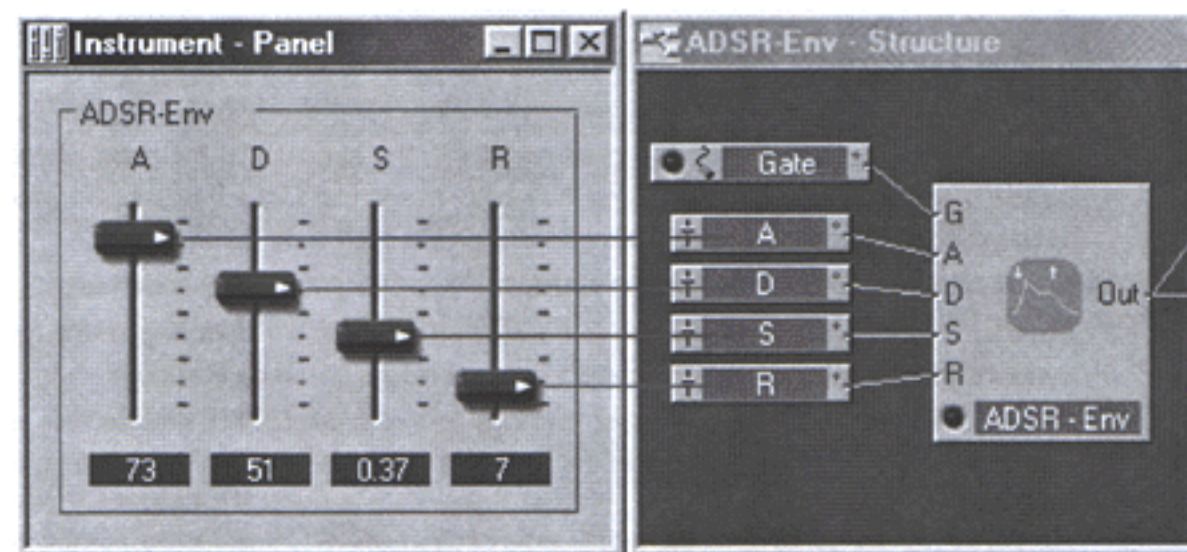
13 Panel Editing

13.1 What is a Panel?

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

13.2 What are Controls?

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



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

13.3 Panel Controls

13.3.1 Faders

Faders are linear sliding controllers in the panel, whose position determines the control value of the corresponding source module. The range of output values of the fader is set with the values

Min and **Max** in the module properties. The resolution can be set with the property **Step** (when **Step** is zero there are 128 steps).

Besides moving a Fader with the mouse, it can also be remote controlled with MIDI (see section 14.3).

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

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

13.3.2 Knob

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

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

13.3.3 Button

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

Besides operating the Button with the mouse, it can also be remote controlled with MIDI (see section 14.3).

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

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

13.3.4 Switch

Switches are used to activate and deactivate different signal paths by choosing one of the switch inputs to connect to the output. There are separate types of switch modules for audio and event signals. Both kinds of switches are available with varying number of inputs.

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

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

13.4 Editing the Panels

Just like the modules in a structure, the controls in the panel can be edited with **Cut**, **Copy**, **Paste** and **Delete**. However, with all these operations you should remember that they 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, therefore, to carry out these operations only in the structure window because only there can you keep control of the outcome.

Moving controls in the panel, on the other hand, has no effect on the structure. Simply click the left mouse button on the label of the control you want to move and drag it with held mouse button to the desired position. Once all controls have been arranged it is best to activate the **Lock** function (through the context menu of the panel window or by clicking on the button with the padlock icon in the Instrument Toolbar). When **Lock** is enabled it is no longer possible to move and panel elements.

14 Panel Operation

14.1 Mouse Control

14.1.1 Fader

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

14.1.2 Knob

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

14.1.3 Button

There is a choice of three operating modes for buttons which can be chosen in the properties dialog window:

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

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

14.1.4 Switch

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

14.2 Key Control

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

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

Key	Value Change
•	+Step
•	-Step
• <i>PgUp</i>	+10 × Step
• <i>PgDn</i>	-10 × Step

A Switch changes the active input on pressing the keys /.

The control element to which the keys apply can be selected with the keys /.

14.3 MIDI Control

14.3.1 MIDI Data Types

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

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

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

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

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

14.3.2 MIDI Learn

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

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

The REAKTOR SERIES software automatically detects whether the controller data comes from a standard MIDI controller or from an incremental controller like the 4Control made by NATIVE INSTRUMENTS. The panel element's **Incremental** mode switch is set accordingly. In the rare case that the **MIDI Learn** function choses the wrong mode, simply repeat the operation or set the correct mode in the properties of the panel element manually.

14.3.3 Incremental

You need to activate this entry in the **Properties** of controls or MIDI source modules if you want to use the optional **4Control** MIDI Unit by NATIVE INSTRUMENTS to set the value. The 4Control sends relative position values for forward or backward movement of the knobs (63 = 1 step back, 65 = 1 step forward). In **Incremental** mode this values get translated to absolute values (0 ... 127). For details about using REAKTOR with the 4Control please see section 16.5.

14.3.4 Soft Takeover

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

14.4 MIDI Out

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

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

14.5 Snapshots

14.5.1 What are Snapshots?

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

14.5.2 Recalling

Snapshots are recalled using the selection list in the instrument toolbar of the main window. This presumes that the window of the wanted instrument is active. You open the list of the snapshots by clicking the left mouse button on the box marked , then you can choose a snapshot to recall. Just click on it and it becomes active. The snapshot that was last recalled is always shown in the toolbar. You can also recall snapshots by moving around the list with the cursor keys  .

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

It is also possible to link snapshot in different layers of the hierarchy. When recalling a snapshot in the ensemble, for example, the instruments in the ensemble will also change snapshot if they have **Snapshot Recall by Parent** activated in their respective properties. The snapshot that the instrument reverts to is the same number that it was set to at the time that the ensemble's snapshot was generated.

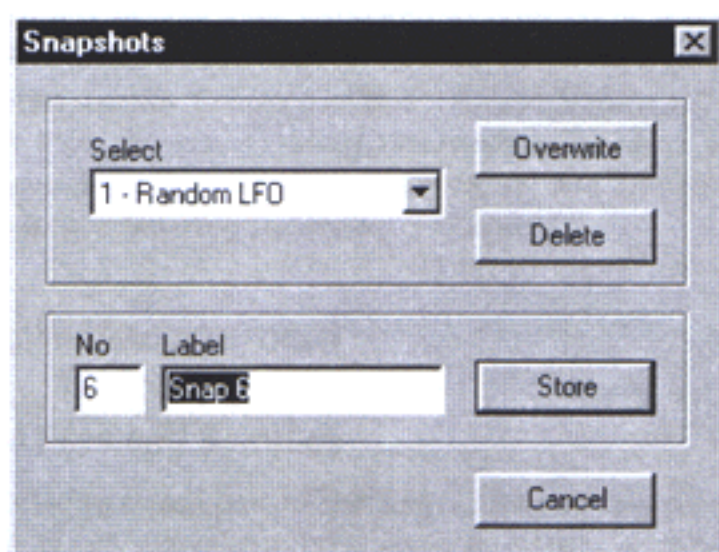
The last recalled snapshot can be recalled from memory once more by pressing the **Reload** button in the instrument toolbar (to the right of the button with the camera icon). This is very useful if you want to discard all the changes you have made since the first recall.

14.5.3 Storing

To store a snapshot, left-click with the mouse on the **Store** button (which has an icon representing a camera) in the instrument toolbar. The **Snapshots** dialog window opens up.

With **No** you can assign a memory position (and MIDI Program Change number) to the snapshot. With **Label** you give the snapshot a name. Clicking on **Store** then saves the snapshot and closes the dialog window.

You can also replace the snapshot shown under **Select** (by default the current snapshot) with the current control settings by clicking on **Overwrite**. The snapshot is stored with the new settings but under the old name at the old location.



The dialog window used to store, delete, rename and copy snapshots.

14.5.4 Deleting

To delete a snapshot, left-click with the mouse on the **Store** button in the instrument toolbar. The **Snapshots** dialog window opens up.

Now choose the snapshot to be deleted in the **Select** list and click on **Delete** to remove it from the list.

14.5.5 Copying and Renaming

To copy or rename a snapshot you first have to recall it using the list in the instrument toolbar. Next open the **Snapshots** dialog window by pressing the button with the camera icon. Then choose the current snapshot in the **Select** list to copy its number and label to the fields at the bottom.

To **rename** the snapshot, give it a new name under **Label** and press **Store** to overwrite the old position with the new name.

To **copy** the snapshot, enter a new number under **No** and press **Store** to copy it to that location with its old name.

14.5.6 Snap Isolate

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

15 Sampling and Re-Synthesis in ⓧ TRANSFORMATOR

Since TRANSFORMATOR / REAKTOR is a modular system, it does not require a fixed structure for organizing samples. Chapter 5 entitled "TRANSFORMATOR Tour" uses examples from the library to show you how to create instruments as powerful as you like, just by interconnecting modules in complex ways. However, it also demonstrates how simple instruments can be constructed using the simplest of means.

15.1 Sample Management

15.1.1 Sound Files and Samples

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

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

Instead you will get an error message later: TRANSFORMATOR will inform you that the CPU is overloaded and processing will stop. This error message is displayed because the software cannot access the sample data on the hard drive in time (after first having tried to access it in RAM and not having found it there). Frequently the message is preceded by an unintentional granular sound effect caused by the more or less regular interruptions of the audio data stream. This situation, particularly undesirable during live performances, can only be avoided through the addition of more main memory. Under Windows, if latency is not an issue, the susceptibility of the system to such problems can be reduced by use of large buffer settings (**Play Ahead**) (See section 2.3.2). Under MacOS, virtual memory should be turned off, also for reasons of MIDI timing (See section 3.3.1).

15.1.2 Multiple Use of Identical Samples

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

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

15.1.3 Missing Samples

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

In this situation you will get an error message after having opened the macro / instrument / ensemble. The "missing" samples are labeled in the properties dialog window as **Missing**. Via the **Replace** button the File-Open dialog window is reached, with which it is possible to localize or replace missing sound files.

15.1.4 Sample-Backup in the Module

The loss of samples can be avoided if the **Backup Sound with Module** option is activated, which can be accessed via the sampler module's properties dialog window. Sampler modules, which make use of this option, save a copy of the sample file with the module file. In most cases the increase in the file size of macro / instrument / ensemble files will be considerable, but your work can be reconstructed whatever the circumstances.

Whenever a module containing a Sample-backup is loaded it will attempt to find the original sound file first. The Backup-files are only used if the original files cannot be located. These "recovered" samples are labeled in the properties dialog window of the relevant sample modules as **Recovered**.

15.1.5 Sample-Analysis

Some modules (**Sample Resynth**, **Sample Pitch Former**) perform real-time re-synthesis. For this purpose, after loading, the sample will be analyzed. This process takes approximately as long as the duration of the sample. To avoid repeating analysis TRANSFORMATOR tries to save the analysis data with the sound file. This of course is only possible if the sound file is not

write-protected. Also keep in mind that sound files from CD-ROMs should be copied to hard disk. If a sound file has been changed it is possible to re-analyze it.

15.1.6 Sample-Editors

TRANSFORMATOR does not have its own sample editor as there are many sample editors available on the market. In the preferences menu, the edit button can be set to launch your preferred sample editor (**System⇨Preferences⇨Directories: Sample-Editor Application**). It is also possible to edit the sound files saved using the Sample-backup option.

Naturally sound files changed in the sample editor have to be **saved** before they can be re-loaded into TRANSFORMATOR. Use **Reload** in the properties dialog window and in the context menu of the sample module.

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

15.2 Sample-Maps

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

15.2.1 Multi-Sampling

The repetitive nature of samples makes it difficult to simulate acoustic instruments. With TRANSFORMATOR it is possible to vary the sample each time it is triggered. Usually when a sample is transposed over multiple keys, the further it is transposed from the root key of the original sample the greater the loss of sound naturalness. This loss occurs because of the spectral aspects of transposed samples do not change according to the spectral aspect changes occurring naturally in acoustic instruments. Samplers try to overcome this limitation by using multiple sample techniques, therefore only transposing the root key of the original sample by a smaller amount.

Assuming a sample-map contains multiple samples of the original sound at least two parameters have to be set to control each sample in the map.

- The **Root Key** sets pitch of the untransposed sample.
- The **Left Split** sets the starting point of the key zone. The end of the zone ("Right Split") does not have to be set unless wanted or imposed by the Left Split of the next sample.

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

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

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

Sample Maps in TRANSFORMATOR can also be used to trigger multiple samples with only one key. This is a technique used to further reflect an instrument's dynamics. Velocity is used to achieve this via MIDI. Usually an instrument is sampled playing the same note with different dynamics. Then the dynamically different samples can be triggered using different velocities, resulting in a more expressive representation of the original sound. Chapter 5 TRANSFORMATOR Tour contains appropriate examples.

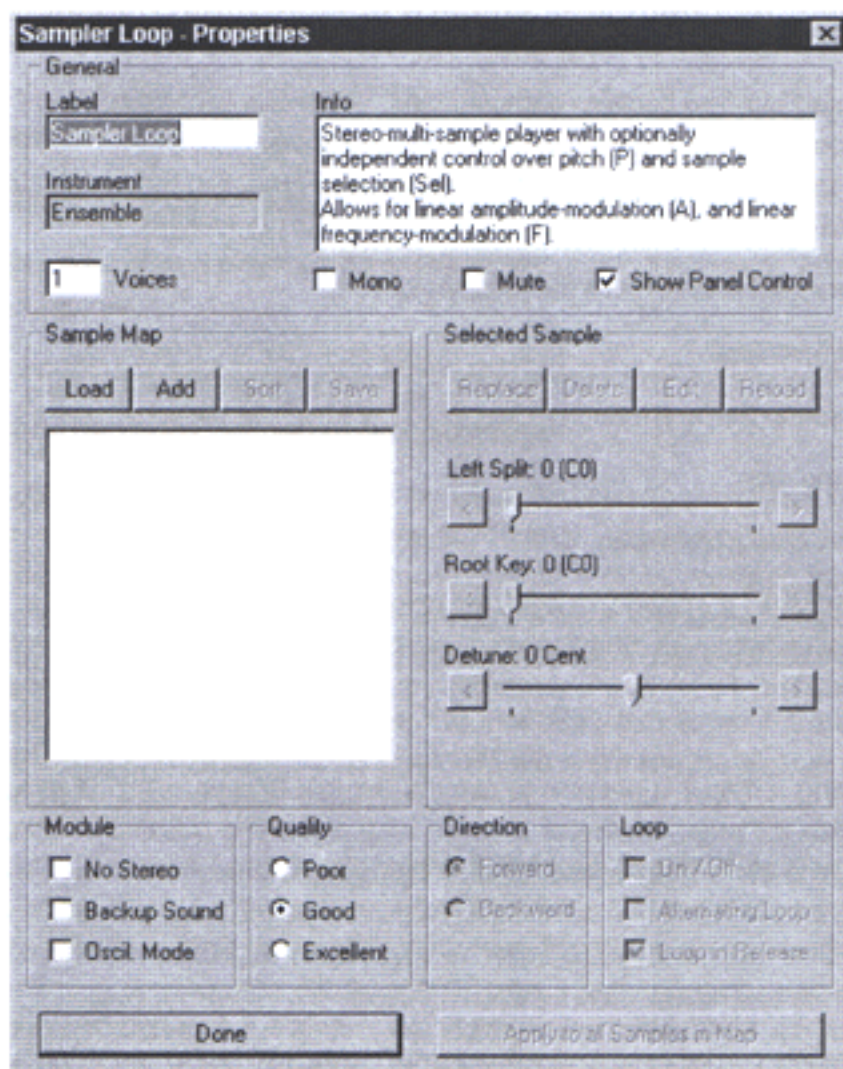
15.2.2 Drum-Maps

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

As described above the **Sel**-input is used to overwrite 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.

15.3 Sampler Properties Dialog window

The sampler and re-sampler modules in TRANSFORMATOR use the properties dialog window to organize Sampler-maps and to set the module's parameters. The easiest way to open the properties dialog window is by double-clicking the module or its icon in the toolbar. The properties dialog window varies very little between modules. For example the Sampler Loop looks like this:



Sampler Loop module properties dialog window

The properties dialog window is split into three areas:

- The **General** field in the top part of the dialog window is used to set general module parameters. All REAKTOR SERIES modules have this field in their properties dialog window.
- The **Sample Map** field including the **Module** and **Quality** fields below it are used to set the parameters for the whole module, independent of the individual samples. Use the **Done** button to close the dialog window.
- The **Selected Sample** field including the **Direction** and **Loop** fields below it are used to set sample specific parameters. These settings relate to the selected sample shown in bold in the **Sample map** field and only affect this module. The **Apply to all Samples in Map** button copies the settings from the **Module** and **Quality** fields and applies them to all samples in the sample map.

15.3.1 The Sample Map Field

The **Sample Map** field contains sample maps, which can be selected from the list by clicking with the mouse or using the up and down keys on the computer's keyboard. This list contains the names of sound files and its location. Every entry is preceded by the left **Split-point**, which is represented as text (C0..G10 etc) and also in MIDI note numbers (0..127).

The **Sample Map** field contains the following options:

- **Load** map files
- **Add** sound files to a map
- **Sort** the map
- **Save** the map file (.map)

When a sound file is **added** to a map, the sample's root key information sets the **Root Key** and the **Left Split** parameters. Should the sample not contain any root key information, a value of 60 (C3) will be used, if this value is unavailable the sample will be placed at the next available note number.

When **saving** a map it is possible to use the Sample Backup option, if used this option saves samples with the map. Whenever a sound map containing a Sample-backup is **loaded** it will attempt to find the original sound file first. By saving the sound files with the sound map you assure that all the sound files are included within the sound map, but the original sound files will remain at their original location, therefore resulting in duplicates of the sound files. It is also possible to transfer sound maps between modules.

A single sample can be used to create a sample map. The properties dialog window of a module does not have to be opened to **load** samples / maps, this task can also be achieved with the **Import Sound** option in the **context menu**.

15.3.2 The Selected Sample Field

To the right of the **Sample Map** field is the **Selected Sample** field, which displays the parameters of the selected sample. The parameters can be changed using the scrollbar or the up and down keys. Which parameters are used depends on the module type and the map mode. All sample parameters are saved with the maps.

The **Selected Sample** field also contains buttons to:

- **Replace** the selected sample,
- **Delete** the selected sample from a map,
- **Edit** to open the selected sample in a sample editor,

- **Reload** the selected sample.

15.3.3 The Module field

Here a maximum of three parameters can be set:

- **No Stereo** stops stereo playback (Only the left channel is then used). Activation reduces the processor load.
- **Backup Sound** activates the Backup Sound with Module option (see 15.1.4)
- **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 the 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 to 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, these waveforms are called 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. Like this, WaveSet Synthesis is now available for the first time in combination with FM synthesis.

15.3.4 The Quality Field

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 it is the noise that may improve the musical "Quality" of a sample.

15.3.5 The Direction Field

Sets the playback direction (**Forward** or **Backward**) of the sample.

15.3.6 The Loop Field

A maximum of three Loop parameters can be set here:

- **On / Off** activates the Sample-Loop
- **Alternating Loop** activates alternating Loop playback. The sample will be played alternatively forwards and backwards within the Loop.
- **Loop in Release** keeps the Loop playback in the release phase (only available for modules with a **G**-input).

16.4 Key Commands

On the Macintosh, the  key is used instead of **Ctrl**.

Ctrl + N	<u>N</u>ew Ensemble = Open an empty ensemble New.ens
Ctrl + O	<u>O</u>pen... an ensemble, instrument, or macro
Ctrl + S	<u>S</u>ave Ensemble = Save the current ensemble, overwriting the file of the ensemble which was loaded last
Ctrl + E	<u>S</u>ave <u>W</u>indow <u>A</u>s... Save the macro or instrument in the selected window
Ctrl + Z	<u>U</u>ndo last edit
Ctrl + Y	<u>R</u>edo last undo
Ctrl + X	<u>C</u>ut selected modules or panel elements
Ctrl + C	<u>C</u>opy selected modules or panel elements
Ctrl + V	<u>P</u>aste modules or panel elements from the clipboard at the point specified with a left mouse-click
Ctrl + A	<u>S</u>elect <u>A</u>ll modules or panel elements in the active window
Ctrl + M	<u>M</u>ute on/off for selected modules
0 in keypad	<u>R</u>un/<u>S</u>top the audio process, toggle between on and off
Ctrl + U	<u>M</u>easure <u>C</u>PU <u>U</u>sage = Measure contributions to CPU load
Ctrl + T	<u>S</u>tore <u>S</u>napshot = Store, delete, or rename a snapshot
Ctrl + I	<u>M</u>IDI-<u>L</u>earn -function activate/deactivate
Ctrl + L	<u>L</u>ock panel elements in position
Alt + Return Alt + Enter	<u>P</u>roperties of the instrument or macro whose panel is shown
Ctrl + B Return	<u>P</u>anel of the instrument, whose structure window is selected <u>S</u>tructure of the instrument, whose panel window is selected
Ctrl + P	<u>P</u>arent of the macro or instrument, whose structure window is selected

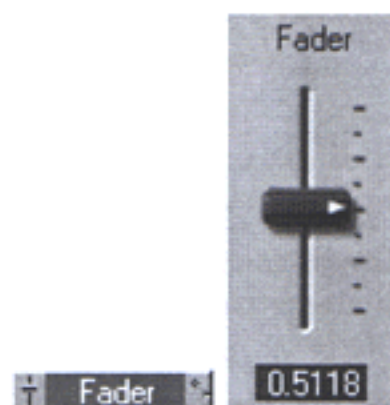
Contents

1	Panel.....	1
2	MIDI.....	10
3	MIDI Out	15
4	Constant	17
5	+, −, X, /	18
6	Mixer	24
7	Oscillators 	28
7.1	Multistep	44
8	Samplers 	47
9	Sequencer.....	60
10	LFO, Envelope	64
11	Filter.....	77
12	Delay.....	90
13	Shaper	93
14	Modifier	100
15	Event Processing	107
16	Auxiliary	114
17	Conversion Tables for Control Values	122
17.1	Control Value / Time.....	122
17.2	Pitch / Frequency.....	123
18	Index	124

1 Panel

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.

Range

The output value corresponds to the position of the knob (resolution: 128 steps) between the limits **Min** and **Max** set in the properties dialog window. With the parameter **Step**, the display and output values can be quantized to coarser steps, e.g. to display less decimal digits.

MIDI

When **Remote** is activated, the fader movement can be controlled by MIDI events. You can choose between MIDI Controller and Poly Aftertouch of a note and you can select the number of the controller or note.

If **Panel to MIDI-Out** is enabled, REAKTOR will send MIDI events whenever the fader in the panel is moved. **Remote to MIDI-Out** determines whether MIDI events are also output by REAKTOR when the fader is moved by received MIDI events (**Remote**). Take care, though, that when connecting to a sequencer which sends MIDI data to REAKTOR and at the same time receives MIDI from it, a feedback loop may result.

The selections **Controller** and **Poly Aftertouch** are used to set the fader to receive and/or send either MIDI Controller or Poly Aftertouch (Key Pressure) messages. **Controller/Note No.** is used to display and set the number of the controller or note that is assigned to the fader.

Activate **Incremental** if the fader is to be controlled from a 4Control MIDI unit or another device which operates in incremental mode.

If **Soft Takeover** is enabled, the fader will not move on receiving MIDI data until the input value goes past the fader's current value. This is to prevent any sudden jumps in the value when the position of the software fader does not match that of the hardware controller, e.g. after snapshot recall.

If the **Snap Isolate** switch is activated, the fader will not respond to snapshot recall.

Panel Appearance

The label will only be shown above the fader in the panel if **Show Label** is activated in the properties. Likewise, the currently set value will only be shown as a number below the fader if **Show Value** is activated.

If **Small Design** is activated in the properties, the fader will be shown in the panel at half the normal size.

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.

Knob

Panel



Just like the Fader but with a different appearance in the panel.

Button

Panel



With the push button control in the panel you select the value which is output (as event and audio) by the corresponding module in the structure. The output values in the on and off state

are set with **On Value** and **Off Value** in the button's properties. The signal is mono – when connected to a poly input all voices receive the same value.

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.

MIDI

When **Remote** is activated, the button movement can be controlled by MIDI events. You can choose between MIDI Controller, Poly Aftertouch and Note events and you can select the number of the controller or note. A controller or aftertouch value between 64 and 127 or a Note On event corresponds to clicking the mouse on the button, a value between 0 and 63 or a Note Off event corresponds to letting go.

If **Panel to MIDI-Out** is enabled, REAKTOR will send MIDI events whenever the button in the panel is operated. **Remote to MIDI-Out** determines whether MIDI events are also output by REAKTOR when the button is operated by received MIDI events (**Remote**). Take care, though, that when connecting to a sequencer which sends MIDI data to REAKTOR and at the same time receives MIDI from it, a feedback loop may result.

The selections **Controller**, **Poly Aftertouch** and **Note On/Off** are used to set the button to receive and/or send either MIDI Controller, Poly Aftertouch (Key Pressure) or Note On/Off messages. **Controller/Note No.** is used to display and set the number of the controller or note that is assigned to the button.

If the **Snap Isolate** switch is activated, the button will not respond to snapshot recall.

Panel Appearance

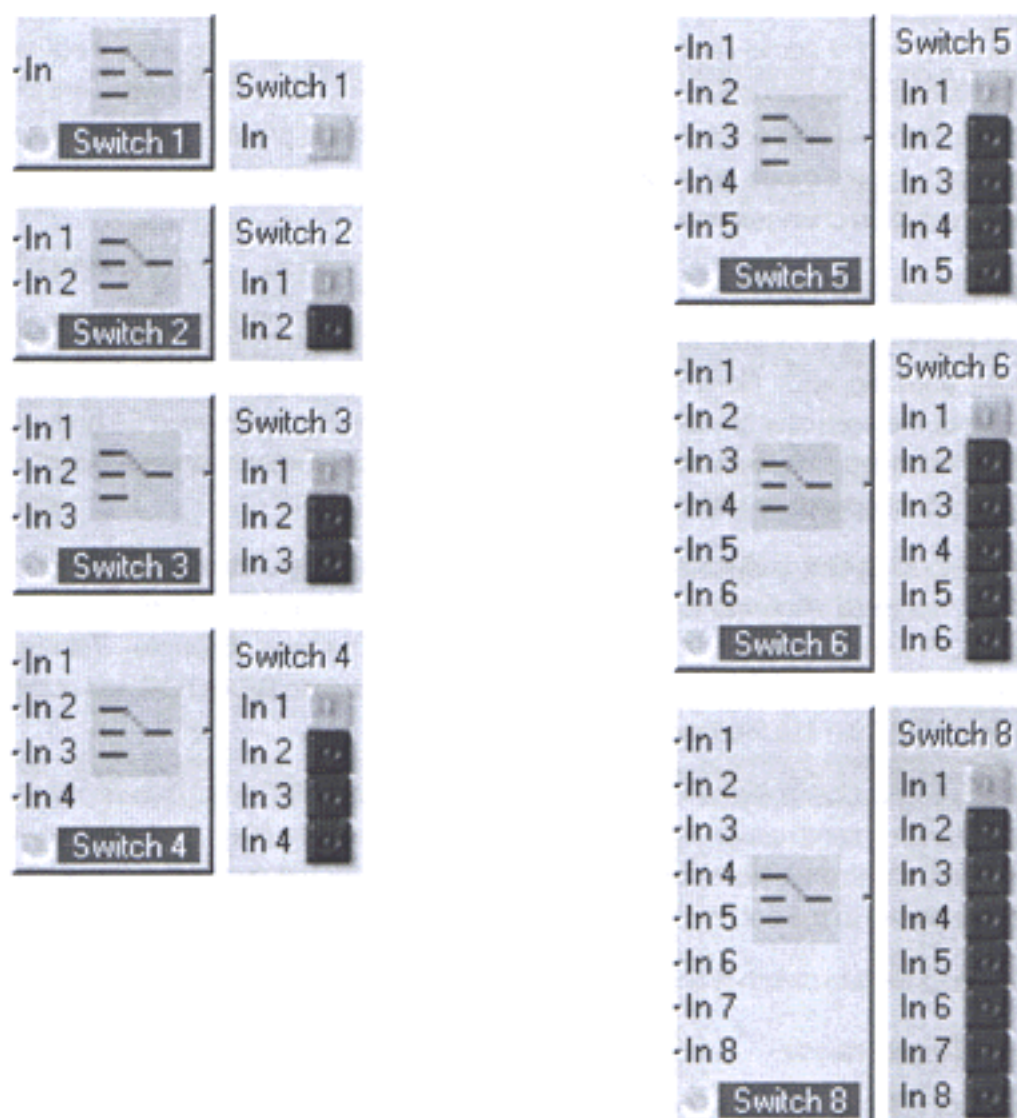
The label will only be shown above the button in the panel if **Show Label** is activated in the properties. Likewise, the currently set value will only be shown as a number below the button if **Show Value** is activated.

If **Small Design** is activated in the properties, the button will be shown in the panel at half the normal size.

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.

Audio 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. 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.

When **Remote** is activated, the switch movement can be controlled by MIDI events. You can choose between MIDI Controller and Poly Aftertouch events and you can select the number of the controller or note. The range of values for the MIDI events is divided into regions according to

the number of possible positions for the switch. The value zero sets the switch to its lowest position.

If **Panel to MIDI-Out** is enabled, REAKTOR will send MIDI events whenever the switch in the panel is operated. **Remote to MIDI-Out** determines whether MIDI events are also output by REAKTOR when the switch is operated by received MIDI events (**Remote**). Take care, though, that when connecting to a sequencer which sends MIDI data to REAKTOR and at the same time receives MIDI from it, a feedback loop may result.

The label will only be shown above the button in the panel if **Show Label** is activated in the properties. Also, if **Small Design** is activated, no labels will be shown to the left of the buttons. For the Switch 2, only one button (the upper button) is shown when **Small Design** is selected: When the button is on input 1 is connected, when the button is off input 2 is connected.

Audio Switch 1

Panel

On/Off Switch for one audio signal.

Audio Switch 2

Panel

Switch for two audio inputs to one output.

Audio Switch 3

Panel

Like Audio Switch 2 but with 3 inputs.

Audio Switch 4

Panel

Like Audio Switch 2 but with 4 inputs.

Audio Switch 5

Panel

Like Audio Switch 2 but with 5 inputs.

Audio Switch 6

Panel

Like Audio Switch 2 but with 6 inputs.

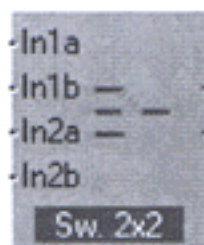
Audio Switch 8

Panel

Like Audio Switch 2 but with 8 inputs.

Audio Switch 2x2

Panel



Double switch with two audio outputs, each of which has two possible audio inputs. When output **a** is connected to input **In1a** then output **b** is connected to input **In1b**. Otherwise output **a** is connected to input **In2a** and output **b** is connected to input **In2b**.

Event Switch 1

Panel

On/Off Switch for one event signal.

Event Switch 2

Panel

Switch for two event inputs to one output.

Event Switch 3

Panel

Like Event Switch 2 but with 3 inputs.

Event Switch 4

Panel

Like Event Switch 2 but with 4 inputs.

Event Switch 5

Panel

Like Event Switch 2 but with 5 inputs.

Event Switch 6

Panel

Like Event Switch 2 but with 6 inputs.

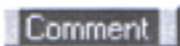
Event Switch 8

Panel

Like Event Switch 2 but with 8 inputs.

Comment

Panel


 Comment

The Comment 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.

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** 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**.

The color of the lamp (red, green, blue or yellow) can be chosen in the properties.

The label will only be shown above the lamp in the panel if **Show Label** is activated in the properties.

Level Lamp

Panel



Indicator lamp for a monophonic signal with logarithmic settings.

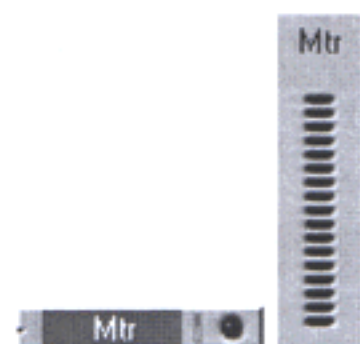
The lamp in the panel lights as long as the input level is within the range set with **Min** and **Max** (in dB) in the properties, i.e. the lamp is on when the signal's value is larger than **Min** and less than or equal to **Max**.

The color of the lamp (red, green, blue or yellow) can be chosen in the properties.

The label will only be shown above the lamp in the panel if **Show Label** is activated in the properties.

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 label will only be shown above the meter in the panel if **Show Label** is activated in the properties.

Level Meter

Panel



Level indicator for a monophonic audio signal.

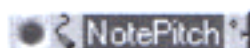
The amplitude of the connected audio signal is displayed on a logarithmic scale. The displayed range is set in dB with **Min** and **Max** in the properties

The label will only be shown above the level meter in the panel if **Show Label** is activated in the properties.

2 MIDI

Note Pitch

MIDI



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



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



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



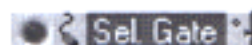
Monophonic event source for MIDI Note On and Note Off events with single trigger characteristic.

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



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



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



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



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.

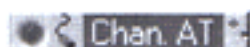
The numbers of some standard MIDI controllers are:

- Modulation Wheel 1
- Breath Controller 2
- Volume 7
- Panpot 10
- Sustain Switch 64
- Portamento Switch 65
- Hold Switch (Sostenuto) 66

See your keyboard's MIDI implementation chart to find out which controllers it can transmit.

Ch. Aftertouch

MIDI



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



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



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



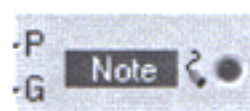
Monophonic event source for MIDI Program Change events. A MIDI event sets the module's output to the value determined by the transmitted program number. The range of the output value is set with **Min** and **Max** in the properties dialog window. The resolution is 128 steps.

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.

3 MIDI Out

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.

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.

Controller

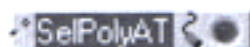
MIDI Out



Converts a monophonic event signal to MIDI Controller events. Each event generates a MIDI message at the MIDI port which REAKTOR uses for output. The number of the controller is set in the properties dialog window with **Controller No** and the range of the input value is set with **Min** and **Max**. The output resolution is 128 steps. An event with value equal to **Min** sets the MIDI Controller value to 0, an event with value equal to **Max** sets the MIDI Controller value to 127.

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.

4 Constant

Constant

Constant

1

Monophonic event 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.

5 +, -, X, /

This section contains modules for simple arithmetic applied to events and audio sample values.

Audio Invert

+, -, X, /

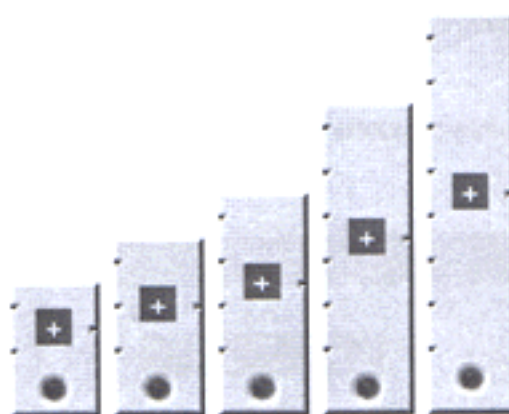


Inverter for an audio signal. The output signal is the inverse of the input signal ($\text{Out} = -\text{In}$).

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.

Audio Add

+, -, X, /



Audio Add 2

+, -, X, /

Adder for two audio signals. The output signal is the sum of the two input signals ($\text{Out} = \text{In1} + \text{In2}$).

Can also be used as a simple two-channel mixer where the level of both channels is fixed at 0 dB.

Audio Add 3

+, -, X, /

Like Audio Add 2 but with 3 inputs.

Audio Add 4

+, -, X, /

Like Audio Add 2 but with 4 inputs.

Audio Add 6

+, -, X, /

Like Audio Add 2 but with 6 inputs.

Audio Add 8

+, -, X, /

Like Audio Add 2 but with 8 inputs.

Event Invert

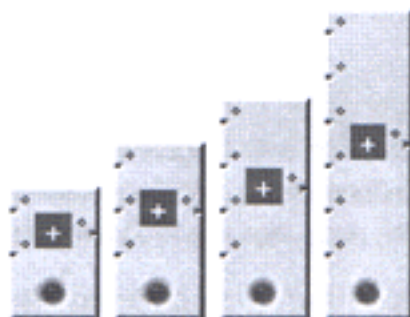
+, -, X, /



Inverter for an event signal. For each event at the input an event with the inverse value is output ($\text{Out} = -\text{In}$).

Event Add

+, -, X, /



Event Add 2

+, -, X_n /

Adder for two event signals. When an event is received at an input, an event is output whose value is the sum of the current (last received) input values ($Out = In1 + In2$).

Event Add 3

+, -, X_n /

Like Event Add 2 but with 3 inputs.

Event Add 4

+, -, X_n /

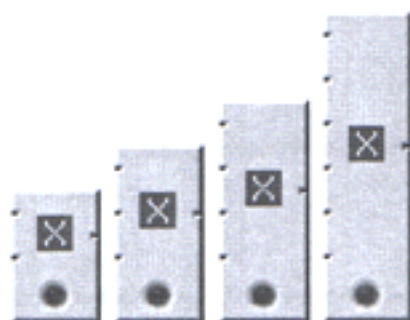
Like Event Add 2 but with 4 inputs.

Event Add 6

+, -, X_n /

Like Event Add 2 but with 6 inputs.

Audio Mult

+, -, X_n /

Audio Mult 2

+, -, X_n /

Multiplier for two audio signals. The output signal is the product of the two input signals ($Out = In1 \cdot In2$). 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 an input is not connected (multiplication with zero) the output value is always zero.

Can also be used to compute the square of a signal when the same signal is fed to both inputs ($\text{Out} = \text{In} \cdot \text{In} = \text{In}^2$).

When two different sounds are connected to the inputs, the output is the ring modulation of the two sounds.

Audio Mult 3

+, -, X, /

Multiplier for three audio signals. The output signal is the product of the three input signals ($\text{Out} = \text{In1} \cdot \text{In2} \cdot \text{In3}$).

When an input is not connected (multiplication with zero) the output value is always zero.

Audio Mult 4

+, -, X, /

Like Audio Mult 3 but with 4 inputs.

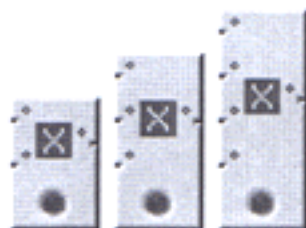
Audio Mult 6

+, -, X, /

Like Audio Mult 3 but with 6 inputs.

Event Mult

+, -, X, /



Event Mult 2

+, -, X, /

Multiplier for two event signals. When an event is received at an input, an event is output whose value is the product of the current (last received) input values ($\text{Out} = \text{In1} \cdot \text{In2}$).

When an input is not connected (multiplication with zero) the output value is always zero.

Can also be used to compute the square of a signal when the same signal is fed to both inputs ($\text{Out} = \text{In} \cdot \text{In} = \text{In}^2$).

Event Mult 3

+, -, X, /

Like Event Mult 2 but with 3 inputs

Event Mult 4

+, -, X, /

Like Event Mult 2 but with 4 inputs

Audio 1/x

+, -, X, /



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).

Audio x/y

+, -, X, /



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 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.

Event $1/x$

+, -, X, /



The output is one divided by the input.

Exception: If the input signal is zero, the output is set to zero.

Event x/y

+, -, X, /



The output is the first input divided by the second input.

Exception: If the second input signal is zero, the output is set to zero.

6 Mixer

Amp

Mixer

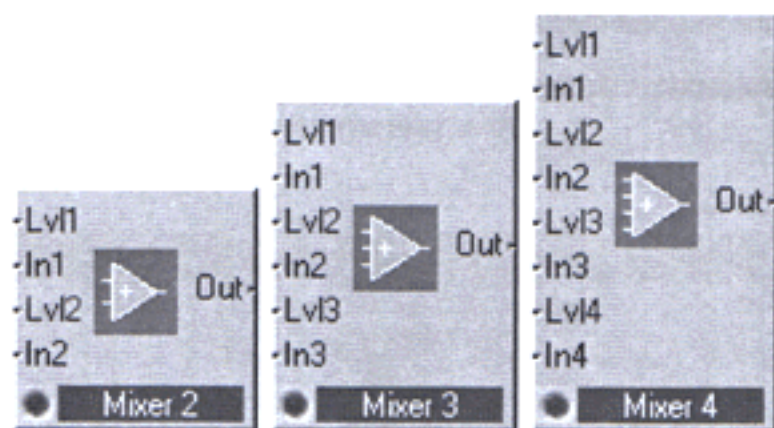


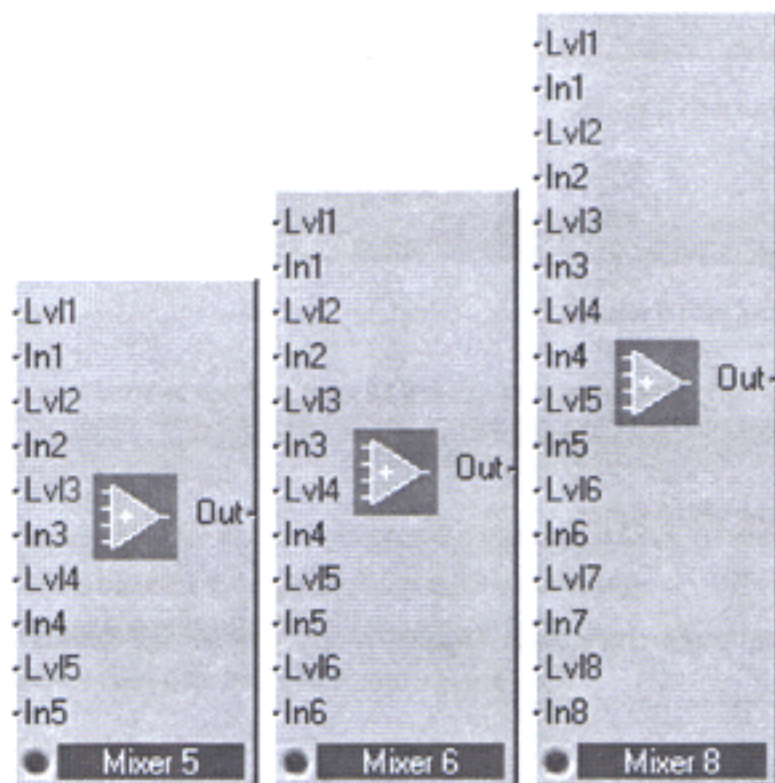
Amplifier for an audio signal. The input signal is amplified (or attenuated) by the set amount (in dB). At 0 dB the output signal is the same as the input signal.

- **Lvl:** Logarithmic event input for controlling the gain, value in dB. When the value is negative, the output signal is smaller than the input signal.
- **In:** Audio signal input
- **Out:** Audio signal output for the amplified signal ($\text{Out} = \text{In} \cdot 10^{\frac{\text{Lvl}}{20}}$).

Mixer

Mixer





Mixer 2

Mixer

Mixer for two audio signals. The two input signals are amplified (or attenuated) by their respective amounts and then summed to form the output.

- **Lvl1, Lvl2:** Logarithmic event input for controlling the gain, value in dB.
- **In1, In2:** Audio signal inputs
- **Out:** Audio signal output for the mixed signal ($\text{Out} = \text{In1} \cdot 10^{\text{Lvl1}/20} + \text{In2} \cdot 10^{\text{Lvl2}/20}$).

Mixer 3

Mixer

Like Mixer 2 but with 3 inputs.

Mixer 4

Mixer

Like Mixer 2 but with 4 inputs.

Mixer 5

Mixer

Like Mixer 2 but with 5 inputs.

Mixer 6

Mixer

Like Mixer 2 but with 6 inputs.

Mixer 8

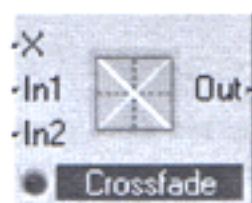
Mixer

Like Mixer 2 but with 8 inputs.

If a mixer with more channels is needed, simply use several smaller mixers and add their outputs.

Crossfade

Mixer

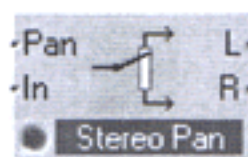


Crossfade module for two audio signals. The output signal is the weighted sum (linear interpolation) of the two input signals. The ratio of their respective amounts in the output is adjustable.

- **X:** Audio control input for the mixing ratio. Value between 0 (**Out** = **In1**) and 1 (**Out** = **In2**).
- **In1, In2:** Inputs for the two audio signals to be mixed.
- **Out:** Audio signal output for the mixed signal ($\text{Out} = (1 - X) \text{In1} + X \text{In2}$).

Stereo Pan

Mixer



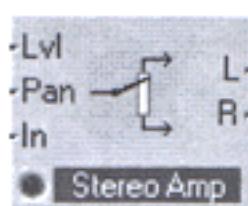
Left-right panner.

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.

- **Pan:** Audio control input for the left-right position. Value -1 = left, 0 = center, 1 = right
- **In:** Audio signal input for the signal to be positioned in stereo.
- **L:** Audio signal output for the left channel signal
- **R:** Audio signal output for the right channel signal

Stereo Amp

Mixer



Amplifier for an audio signal with integrated left-right panner. The input signal is amplified (or attenuated) by the set amount (in dB). 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 amplified input value, i.e. the input is split across the two outputs at a variable ratio.

- **Lvl:** Logarithmic event input for controlling the gain, value in dB. When the value is negative, the output signal is smaller than the input signal.
- **Pan:** Event control input for the left-right position. Value -1 = left, 0 = center, 1 = right
- **In:** Audio 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

7 Oscillators

All of GENERATOR's oscillators can run at any frequency, from 0 Hz (standstill) through the entire audio range right up to the limit set by the sample rate. Like this they are all equally suited for use as LFOs or for producing sound waves. When using an oscillator as an LFO to modulate another module's input which is of the type that only accepts events (e.g. P as opposed to F) you need to insert an A to E (perm) module for conversion.

Sawtooth

Oscillator



 Oscillator for sawtooth waveform with logarithmic pitch control and linear amplitude modulation.

- **P:** Logarithmic event input for controlling the pitch (oscillator frequency). Value in semitones ($69 = 440 \text{ Hz}$, $f_{\text{osc}} = 2^{(P-69)/12} \cdot 440\text{Hz}$).
- **A:** Audio input for controlling the amplitude. The output signal moves between $+A$ and $-A$.
- **Out:** Audio signal output for the sawtooth waveform.

Saw FM

Oscillator



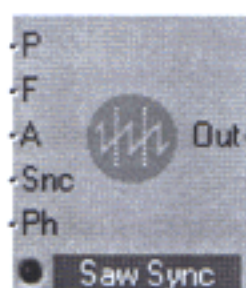
 Oscillator for sawtooth waveform with logarithmic pitch control, linear frequency modulation (FM) and linear amplitude modulation.

- **P:** Logarithmic event input for controlling the pitch (oscillator frequency). Value in semitones ($69 = 440 \text{ Hz}$).

- **F:** Audio input for linear frequency modulation. Value in Hz. **P** and **F** together determine the oscillator frequency: $f_{osc} = F + 2^{(P-69)/12} \cdot 440\text{Hz}$
- **A:** Audio input for controlling the amplitude. The output signal moves between **+A** and **-A**.
- **Out:** Audio signal output for the sawtooth waveform.

Saw Sync

Oscillator



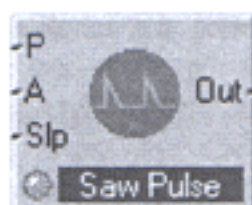
- ⊗ Oscillator for sawtooth waveform with synchronization, logarithmic pitch control, linear frequency modulation (FM) and linear amplitude modulation.

Whenever the synchronization signal leaves zero to positive values (rising edge) the phase of the oscillator is reset to a position specified by the phase input.

- **P:** Logarithmic event input for controlling the pitch (oscillator frequency). Value in semitones (69 = 440 Hz).
- **F:** Audio input for linear frequency modulation. Value in Hz. **P** and **F** together determine the oscillator frequency: $f_{osc} = F + 2^{(P-69)/12} \cdot 440\text{Hz}$
- **A:** Audio input for controlling the amplitude. The output signal moves between **+A** and **-A**.
- **Snc:** Audio input for controlling synchronization of the waveform. A rising edge restarts the oscillator.
- **Ph:** Input for specifying the phase (position in the waveform) to which the oscillator is reset when synchronization occurs.
- **Out:** Audio signal output for the sawtooth waveform.

Saw Pulse

Oscillator



Oscillator for variable sawtooth/needle-pulse waveform with logarithmic pitch control and linear amplitude modulation. The slope of the ramp is variable and with it the shape of the waveform can be changed from a normal sawtooth to a short triangular pulse.

- **P:** Logarithmic event input for controlling the pitch (oscillator frequency). Value in semitones ($69 = 440 \text{ Hz}$).
- **A:** Audio input for controlling the amplitude. The output signal moves between $+A$ and $-A$.
- **Slp:** Audio input for controlling the slope of the waveform. A value of 0 produces a normal sawtooth, a larger value shortens the ramp to a short spike (needle).
- **Out:** Audio signal output for the sawtooth/pulse waveform.

Bi-Saw

Oscillator



Oscillator for bipolar sawtooth waveform with zero-phase. With logarithmic pitch control, pulse width modulation (PWM) and linear amplitude modulation.

- **P:** Logarithmic event input for controlling the pitch (oscillator frequency). Value in semitones ($69 = 440 \text{ Hz}$, $f_{\text{osc}} = 2^{(P-69)/12} \cdot 440 \text{ Hz}$).
- **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 saw-wave at $W = 0$, larger W results in a shorter pulse and a longer zero-phase.
- **Out:** Audio signal output for the bipolar sawtooth waveform with zero-phase.

Triangle

Oscillator



⊗ Oscillator for symmetric triangle waveform with logarithmic pitch control and linear amplitude modulation.

- **P:** Logarithmic event input for controlling the pitch (oscillator frequency). Value in semitones ($69 = 440 \text{ Hz}$, $f_{\text{osc}} = 2^{(P-69)/12} \cdot 440\text{Hz}$).
- **A:** Audio input for controlling the amplitude. The output signal moves between $+A$ and $-A$.
- **Out:** Audio signal output for the triangle waveform.

Tri FM

Oscillator

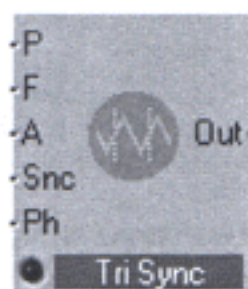


⊗ Oscillator for symmetric triangle waveform with logarithmic pitch control, linear frequency modulation (FM) and linear amplitude modulation.

- **P:** Logarithmic event input for controlling the pitch (oscillator frequency). Value in semitones ($69 = 440 \text{ Hz}$).
- **F:** Audio input for linear frequency modulation. Value in Hz. **P** and **F** together determine the oscillator frequency: $f_{\text{osc}} = F + 2^{(P-69)/12} \cdot 440\text{Hz}$
- **A:** Audio input for controlling the amplitude. The output signal moves between $+A$ and $-A$.
- **Out:** Audio signal output for the triangle waveform.

Tri Sync

Oscillator



Oscillator for symmetric triangle waveform with synchronization, logarithmic pitch control, linear frequency modulation (FM) and linear amplitude modulation.

Whenever the synchronization signal leaves zero to positive values (rising edge) the phase of the oscillator is reset to a position specified by the phase input.

- **P:** Logarithmic event input for controlling the pitch (oscillator frequency). Value in semitones (69 = 440 Hz).
- **F:** Audio input for linear frequency modulation. Value in Hz. **P** and **F** together determine the oscillator frequency: $f_{\text{osc}} = F + 2^{(P-69)/12} \cdot 440\text{Hz}$
- **A:** Audio input for controlling the amplitude. The output signal moves between **+A** and **-A**.
- **Snc:** Audio input for controlling synchronization of the waveform. A rising edge restarts the oscillator.
- **Ph:** Input for specifying the phase (position in the waveform) to which the oscillator is reset when synchronization occurs.
- **Out:** Audio signal output for the triangle waveform.

Tri/Par Symm

Oscillator



Oscillator for variable triangle and parabolic (near to sine) signals, with logarithmic pitch (**P**) control and linear amplitude (**A**) modulation. The symmetry is adjustable by (**W**). Setting the symmetry (with **W**) allows to produce a range of waveforms from symmetric with few harmonics to asymmetric with a broader spectrum.

- **P:** Logarithmic event input for controlling the pitch (oscillator frequency). Value in semitones ($69 = 440 \text{ Hz}$).
- **A:** Audio input for controlling the amplitude. The output signal moves between $+A$ and $-A$.
- **W:** Event input for controlling the symmetry of the waveform. A value of -1 produces 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 ($69 = 440 \text{ Hz}$, $f_{\text{osc}} = 2^{(P-69)/12} \cdot 440\text{Hz}$).
- **A:** Audio input for controlling the amplitude. The output signal moves between $+A$ and $-A$.
- **Out:** Audio signal output for the parabolic waveform.

Par FM

Oscillator



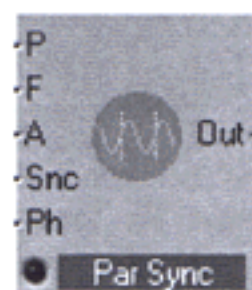
⊗ Oscillator for parabolic waveform with logarithmic pitch control, linear frequency modulation (FM) and linear amplitude modulation. The oscillator sounds like a sine wave with some added

odd numbered overtones at very low level. In many cases it can be used as replacement for a sine generator with less computational load.

- **P:** Logarithmic event input for controlling the pitch (oscillator frequency). Value in semitones ($69 = 440 \text{ Hz}$).
- **F:** Audio input for linear frequency modulation. Value in Hz. **P** and **F** together determine the oscillator frequency: $f_{\text{osc}} = F + 2^{(P-69)/12} \cdot 440\text{Hz}$
- **A:** Audio input for controlling the amplitude. The output signal moves between **+A** and **-A**.
- **Out:** Audio signal output for the parabolic waveform.

Par Sync

Oscillator



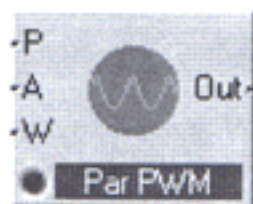
Oscillator for parabolic waveform with synchronization, logarithmic pitch control, linear frequency modulation (FM) and linear amplitude modulation.

Whenever the synchronization signal leaves zero to positive values (rising edge) the phase of the oscillator is reset to a position specified by the phase input.

- **P:** Logarithmic event input for controlling the pitch (oscillator frequency). Value in semitones ($69 = 440 \text{ Hz}$).
- **F:** Audio input for linear frequency modulation. Value in Hz. **P** and **F** together determine the oscillator frequency: $f_{\text{osc}} = F + 2^{(P-69)/12} \cdot 440\text{Hz}$
- **A:** Audio input for controlling the amplitude. The output signal moves between **+A** and **-A**.
- **Snc:** Audio input for controlling synchronization of the waveform. A rising edge restarts the oscillator.
- **Ph:** Input for specifying the phase (position in the waveform) to which the oscillator is reset when synchronization occurs.
- **Out:** Audio signal output for the parabolic waveform.

Par PWM

Oscillator



⊗ Oscillator for variable parabolic waveform with logarithmic pitch control and linear amplitude modulation. The ratio between the length of the upper and lower parts of the curve can be controlled, and with it the wave-form 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 downwards, 0 a normal symmetric parabolic wave and $+1$ parabolas open upwards.
- **Out:** Audio signal output for the variable parabolic waveform.

Sine

Oscillator



⊗ Oscillator for pure sine waveform with logarithmic pitch control and linear amplitude modulation.

If the sine tone does not need to be completely pure, the parabolic oscillator makes a good replacement with less computational load.

- **P:** Logarithmic event input for controlling the pitch (oscillator frequency). Value in semitones ($69 = 440$ Hz).
- **A:** Audio input for controlling the amplitude. The output signal moves between $+A$ and $-A$.

- **Out:** Audio signal output for the sine waveform.

Sine FM

Oscillator



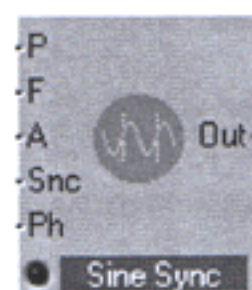
- Oscillator for pure sine waveform with logarithmic pitch control, linear frequency modulation (FM) and linear amplitude modulation.

If the sine tone does not need to be completely pure, the parabolic oscillator makes a good replacement with less computational load.

- **P:** Logarithmic event input for controlling the pitch (oscillator frequency). Value in semitones ($69 = 440 \text{ Hz}$).
- **F:** Audio input for linear frequency modulation. Value in Hz. **P** and **F** together determine the oscillator frequency: $f_{\text{osc}} = F + 2^{(P-69)/12} \cdot 440\text{Hz}$
- **A:** Audio input for controlling the amplitude. The output signal moves between $+A$ and $-A$.
- **Out:** Audio signal output for the sine waveform.

Sine Sync

Oscillator



- Oscillator for pure sine waveform with synchronization, logarithmic pitch control, linear frequency modulation (FM) and linear amplitude modulation.

Whenever the synchronization signal leaves zero to positive values (rising edge) the phase of the oscillator is reset to a position specified by the phase input.

If the sine tone does not need to be completely pure, the parabolic oscillator makes a good replacement with less computational load.

- **P:** Logarithmic event input for controlling the pitch (oscillator frequency). Value in semitones ($69 = 440 \text{ Hz}$).
- **F:** Audio input for linear frequency modulation. Value in Hz. **P** and **F** together determine the oscillator frequency: $f_{\text{osc}} = F + 2^{(P-69)/12} \cdot 440\text{Hz}$
- **A:** Audio input for controlling the amplitude. The output signal moves between **+A** and **-A**.
- **Sync:** 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.

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 ($69 = 440 \text{ Hz}$).
- **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.

Pulse FM

Oscillator

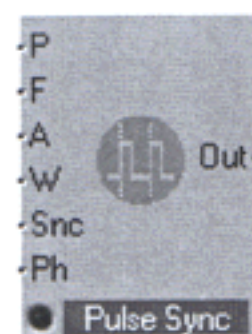


⊗ Oscillator for pulse wave with logarithmic pitch control, linear frequency modulation (FM), pulse width modulation (PWM) and linear amplitude modulation.

- **P:** Logarithmic event input for controlling the pitch (oscillator frequency). Value in semitones (69 = 440 Hz).
- **F:** Audio input for linear frequency modulation. Value in Hz. **P** and **F** together determine the oscillator frequency: $f_{osc} = F + 2^{(P-69)/12} \cdot 440\text{Hz}$
- **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.

Pulse Sync

Oscillator



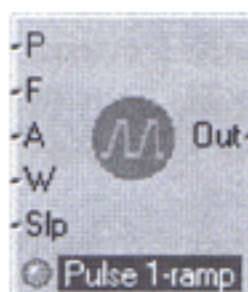
⊗ Oscillator for pulse wave with synchronization, logarithmic pitch control, linear frequency modulation (FM), pulse width modulation (PWM) and linear amplitude modulation.

Whenever the synchronization signal leaves zero to positive values (rising edge) the phase of the oscillator is reset to a position specified by the phase input.

- **P:** Logarithmic event input for controlling the pitch (oscillator frequency). Value in semitones (69 = 440 Hz).
- **F:** Audio input for linear frequency modulation. Value in Hz. **P** and **F** together determine the oscillator frequency: $f_{osc} = F + 2^{(P-69)/12} \cdot 440\text{Hz}$
- **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 = $(1 + W) / (1 - W)$.
- **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



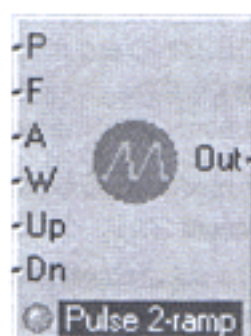
⦿ Oscillator for rectangular trapezoid waveform with logarithmic pitch control, linear frequency modulation (FM), pulse width modulation (PWM) and linear amplitude modulation. The waveform is a mixture of pulse and sawtooth: The rising edge of a pulse wave can be flattened and its slope adjusted. The 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: $f_{osc} = F + 2^{(P-69)/12} \cdot 440\text{Hz}$
- **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 .

- **Slp:** Audio input for controlling the slope of the rising edge of the waveform. When **Slp** is zero (or when the input is not connected) the waveform does not rise and the output is always zero, i.e. there is no sound. Typical range of values: 1 ... 20
- **Out:** Audio signal output for the trapezoid waveform.

Pulse 2-ramp

Oscillator

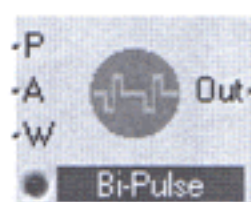


Oscillator for trapezoid waveform with logarithmic pitch control, linear frequency modulation (FM), pulse width modulation (PWM) and linear amplitude modulation. The waveform is a mixture of pulse, sawtooth and triangle: Both edges of a pulse wave can be flattened and their slope adjusted.

- **P:** Logarithmic event input for controlling the pitch (oscillator frequency). Value in semitones (69 = 440 Hz).
- **F:** Audio input for linear frequency modulation. Value in Hz. **P** and **F** together determine the oscillator frequency: $f_{osc} = F + 2^{(P-69)/12} \cdot 440\text{Hz}$
- **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.
- **Up:** Audio input for controlling the slope of the rising edge of the waveform.
- **Dn:** Audio input for controlling the slope of the falling edge of the waveform.
- **Out:** Audio signal output for the trapezoid waveform.

Bi-Pulse

Oscillator



⊙ Oscillator for bipolar pulse wave with zero-phase. With logarithmic pitch control, pulse width modulation (PWM) and linear amplitude modulation.

- **P:** Logarithmic event input for controlling the pitch (oscillator frequency). Value in semitones ($69 = 440$ Hz).
- **A:** Audio input for controlling the amplitude. The output signal moves between $+A$ and $-A$.
- **W:** Audio input for controlling the pulse width (PWM). Range of values is 0 to 1. Symmetric waveform (square wave) 50:50 at $W = 0$, larger W results in a shorter pulse and longer zero-phase.
- **Out:** Audio signal output for the bipolar pulse waveform.

Impulse

Oscillator

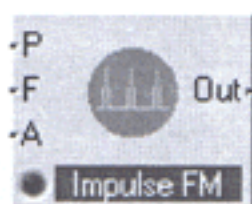


⊙ Oscillator for impulse train signal with logarithmic pitch control (P) and linear amplitude modulation (A).

- **P:** Logarithmic event input for controlling the pitch (oscillator frequency). Value in semitones ($69 = 440$ Hz).
- **A:** Audio input for controlling the amplitude.
- **Out:** Audio signal output for the impulse waveform.

Impulse FM

Oscillator

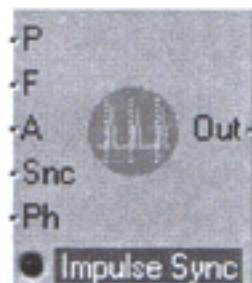


⊗ Oscillator for impulse train signal with logarithmic pitch control (P), linear frequency modulation (F) and linear amplitude modulation (A).

- **P:** Logarithmic event input for controlling the pitch (oscillator frequency). Value in semitones ($69 = 440$ Hz).
- **F:** Audio input for linear frequency modulation. Value in Hz. **P** and **F** together determine the oscillator frequency: $f_{osc} = F + 2^{(P-69)/12} \cdot 440\text{Hz}$
- **A:** Audio input for controlling the amplitude.
- **Out:** Audio signal output for the impulse waveform.

Impulse Sync

Oscillator



⊗ Oscillator for synchronizable impulse train signal with logarithmic pitch control (P), linear frequency modulation (F), linear amplitude modulation (A) and Sync input (Snc).

Whenever the synchronization signal leaves zero to positive values (rising edge) the phase of the oscillator is reset to a position specified by the phase input.

- **P:** Logarithmic event input for controlling the pitch (oscillator frequency). Value in semitones ($69 = 440$ Hz).
- **F:** Audio input for linear frequency modulation. Value in Hz. **P** and **F** together determine the oscillator frequency: $f_{osc} = F + 2^{(P-69)/12} \cdot 440\text{Hz}$
- **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.

Noise

Oscillator



⊙ Noise generator. Produces white noise, i.e. a random signal containing equal amounts of all possible frequency components. The signal consists of only two values, $A/2$ and $-A/2$, but which appear in a random sequence.

- **A:** Audio input for controlling the amplitude of the output signal.
- **Out:** Audio signal output for the noise waveform.

Random

Oscillator



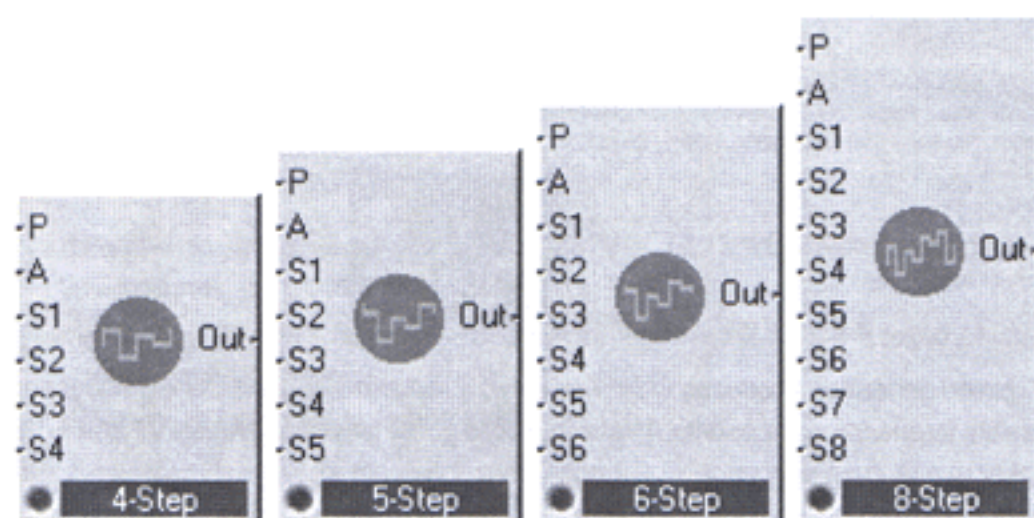
⊙ Random value generator. Produces step waveform where the level of each step is random in the given interval with equal distribution. It works like a noise generator with uniform distribution followed by a sample & hold circuit clocked at a regular frequency.

- **P:** Logarithmic event input for controlling the step rate (sample & hold frequency). Value in semitones ($69 = 440$ 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.

7.1 Multistep

Multi-Step

Multistep



4-Step

Multistep

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

Multistep

Like 4-Step but with 5 steps.

6-Step

Multistep

Like 4-Step but with 6 steps.

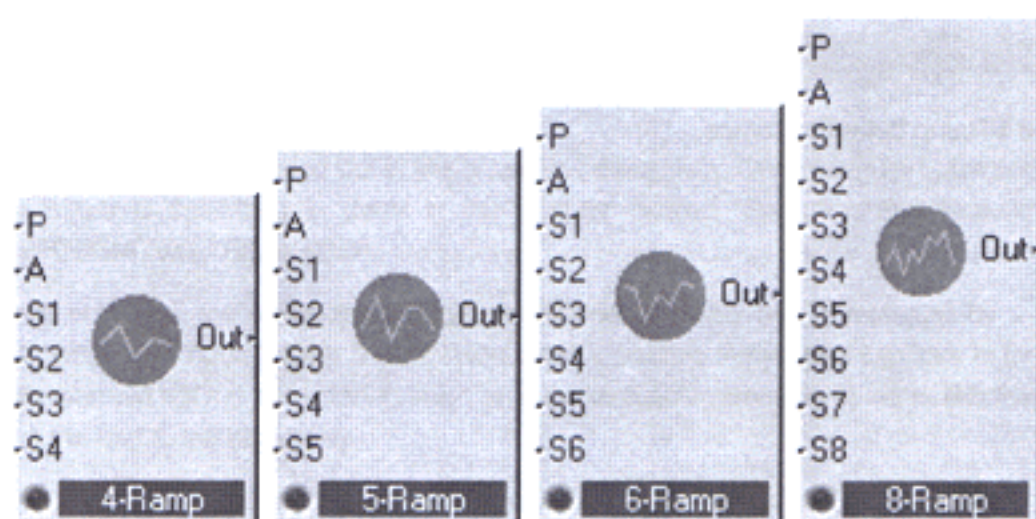
8-Step

Multistep

Like 4-Step but with 8 steps.

Multi-Ramp

Multistep



4-Ramp

Multistep

⊗ 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 ($69 = 440 \text{ Hz}$).
- **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

Multistep

Like 4-Ramp but with 5 ramps.

6-Ramp

Multistep

Like 4-Ramp but with 6 ramps.

8-Ramp

Multistep

Like 4-Ramp but with 8 ramps.

8 Samplers ⓘ

Sampler

Sampler



ⓘ This is a player used for the polyphonic and transposed playback of a sample or a sample map.

Sample management is carried out in the properties dialog box. Organizing samples and editing sample maps is described in detail in the chapter entitled "Sampling and Resynthesis in TRANSFORMATOR / REAKTOR".

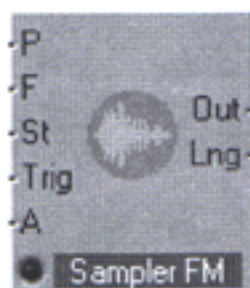
If **loop** is activated, the whole sample is repeated continuously and is restarted by a positive event at the trigger input. If **loop** is deactivated and a positive trigger event arrives at the trigger input, the sample will run from start to finish or, if the direction parameter is set to **Backward**, will run from the end to the beginning.

If **Oscillator Mode** is active, the playback rate will be adjusted to suit the length of the current sample in order to produce the correct pitch when operating as a waveform oscillator.

- **P:** Logarithmic control input for the playback rate (pitch) and for selecting a sample from the loaded sample map. If **P** is equal to the **Root Key** of the current sample, the sample will be played back at its original pitch.
- **Trig:** A positive event at this input triggers the sample to be played back from the beginning again.
- **A:** Control input for the output amplitude.
- **Out:** The sample player's output.

Sampler FM

Sampler



Ⓢ This is a player used for the polyphonic and transposed playback of a sample or a sample map. The **F** input and the starting point control input (**St**) allow you to manipulate sample playback.

Sample management is carried out in the properties dialog box. Organizing samples and editing sample maps is described in detail in the chapter entitled "Sampling and Resynthesis in TRANSFORMATOR / REAKTOR".

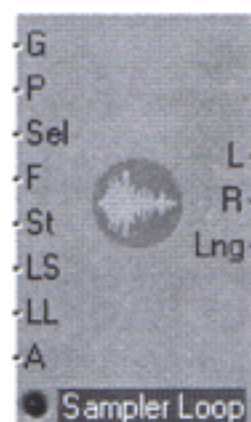
If **loop** is activated, the whole sample is repeated continuously; a positive event at the trigger input will make playback jump back to the starting point that has been set via the **St** input. If **loop** is deactivated and a positive trigger event arrives at the trigger input, the sample will run 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 GENERATOR. 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 properties dialog box. Organizing samples and editing sample maps is described in detail in the chapter entitled "Sampling and Resynthesis in TRANSFORMATOR / REAKTOR".

After a positive **Gate** event, sample playback starts from the starting point (configurable via the **St** input). If loop is deactivated, the sample will run once from the starting point to the end or, if the direction parameter is set to **Backward**, will run from the starting point to the beginning. If loop is activated, the sample will be continuously repeated within the loop range as soon as playback enters this range. The loop range is preset using data from the sound file from which the sample was loaded. If the sound file does not contain any loop data, the beginning of the sample is taken to be the beginning of the loop and the sample length is taken as the loop length. The loop data from the sound file are the default settings. These defaults are always used if the Loop Start (**LS**) or Loop Length (**LL**) inputs are not connected to other modules.

X1 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.

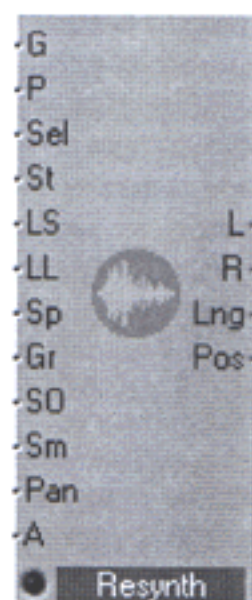
If the **No Stereo** option is activated (you can set this in the properties dialog box), stereo samples will be treated as mono samples, i.e. the same signal is sent to the **L** and **R** outputs. If this is the case, **Sampler Loop** only uses the left channel of stereo samples. This option has been 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 **GENERATOR**, **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 **GENERATOR**.
- **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.

Sample 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 TRANSFORMATOR / 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 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 Resynth** uses a pointer at the current sample position. However, the signal at the outputs is not simply the sample amplitude occurring at the pointer position. What actually happens is that the output signal is generated by a synthesizer inside the module. This synthesizer *resynthesizes* the signal at the pointer position. The pitch of the signal generated by this synthesizer is independent of the pointer's speed. Even if the pointer is not moving at all, the synthesizer continues to produce a sound. While the pitch of the output signal is determined, as is usual, over the **P** input, the **Sp** input determines the pointer speed.

Of course, the slowed down or "frozen" signal does not always correspond to what you might have imagined it to be. What does a hammer hitting a nail sound like if its is slowed down to infinity? The resynthesis algorithm used by **Sample Resynth** is designed in such a way that a broad range of sounds can be processed in a (musically-speaking) sensible, subtle or drastic manner. The algorithm can be adjusted by configuring the **G** (Granularity) and **Sm** (Smoothness) parameters. This means that you can manually adjust it to suit the sample and to produce drastic, strange sound effects. In the properties dialog box you can activate the **Signal-Informed Granulation** option separately for each sample in a sample map. If this option is switched on, the resynthesize algorithm in **Resynth** takes information on the sample into account. This information was collected during the analysis that was carried out when the sample was loaded for the first time. What this means is that the algorithm reacts to the characteristics of the audio material. If this option is deactivated, the results produced are generally quite "electronic".

Sample management is carried out in the properties dialog box. Organizing samples and editing sample maps is described in detail in the chapter entitled "Sampling and Resynthesis in TRANSFORMATOR / REAKTOR".

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.

X2, (same as X1 ;-D) 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.

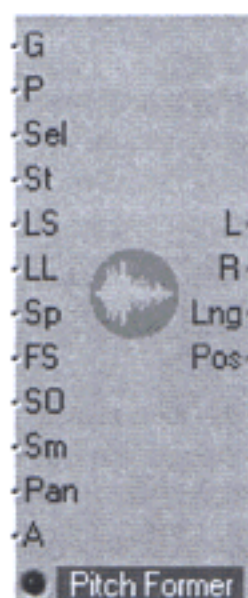
If the **No Stereo** option is activated (you can set this in the properties dialog box), stereo samples will be treated as mono samples, i.e. the same signal is sent to the **L** and **R** outputs. In this case, **Resynth** only uses the left channel of stereo samples. This option has been made available because mono playback requires less processing capacity.

All of **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**).

Sample Pitch Former

Sampler



Ⓢ This is a real-time resynthesizer for the polyphonic playback of mono or stereo samples and sample maps or WaveSets. **Sample Pitch Former** is a WaveSet synthesizer in which not only WaveSets can be loaded but also any samples you like. **Sample Pitch Former** removes the pitch development from a sample and gives the sample any new pitch you wish. Besides independent real-time control over the playback speed, **Sample Pitch Former** also allows you to carry out a spectral transposition, i.e. a pitch-independent transposition of the timbre.

"Standard" samplers like the familiar hardware samplers or the **Sampler**, **Sampler FM** and **Sampler Loop** modules in TRANSFORMATOR / 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 TRANSFORMATOR / REAKTOR achieve a *relative* pitch change by *transposing* a sample, **Sample Pitch Former** forces an *absolute and definite pitch* that you wish onto a sample. **Sample Pitch Former** can therefore be used like a GENERATOR 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 less sound alterations the more uniquely 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 TRANSFORMATOR / 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 properties dialog box. Organizing samples and editing sample maps is described in detail in the chapter entitled "Sampling and Resynthesis in TRANSFORMATOR / REAKTOR".

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 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** only uses the left channel of stereo samples. This option has been made available because mono playback requires less processing capacity.

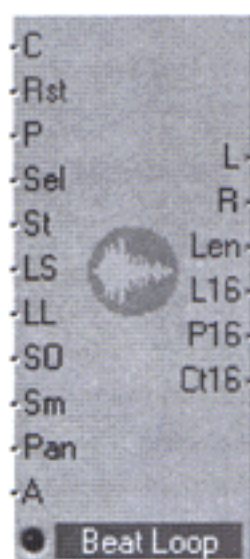
All of **Sample Pitch Former's** inputs, except **Gate**, are designed as audio inputs. The values at the inputs are applied when samples are (re-) triggered (i.e. during positive **Gate** events). During sample playback, changes to values take effect at intervals of the fundamental pitch period (you 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.

Beat Loop

Sampler



Ⓢ This is a real-time resynthesizer for the synchronized playback of beat-loop samples. The transposing of beat-loops can be set via the **P** input independent of playback speed. **Beat Loop** synchronizes itself to a 96th note clock source (24 ppq) that is connected to the **C** input or, if the **C** input is not connected, it can sync with the global TRANSFORMATOR / REAKTOR clock. Therefore, by default all **Beat Loops** in TRANSFORMATOR / REAKTOR run in sync with one another independent of the sample's internal speed. Furthermore, **Beat Loop** also allows you to easily link internal TRANSFORMATOR / REAKTOR sequencer modules and MIDI clocks to rhythmic sample material.

Sample management is carried out in the properties dialog box. Organizing samples and editing sample maps is described in detail in the chapter entitled "Sampling and Resynthesis in TRANSFORMATOR / REAKTOR".

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** only uses 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 TRANSFORMATOR / 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.

- **Lng**: 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.

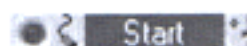
The properties dialog box allows you to select one of three levels of quality that is to be used for interpolation during transposed sample playback (**Poor**, **Good**, **Excellent**). If set to **Poor**, the sample values are not interpolated during output. Higher playback quality is achieved at the 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.

9 Sequencer

Start/Stop

Sequencer



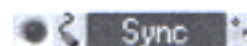
Source for start and stop events for the synchronization with external MIDI devices or the internal master clock.

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.

Sync Clock

Sequencer



Source for a clock signal which is derived from an external MIDI Clock or the internal master clock.

The output of the **Synch Clock** module is a monophonic gate signal. The value can be set with **Output Value** in the properties. At each beat, the gate jumps to the respective value and back to zero after a certain time interval. The module is activated and deactivated by start-stop events.

The rate of the clock signal (quarter, eighth, sixteenth notes, etc.) can be adjusted in properties as **Rate**.

The duration of the gate signal (eighth, sixteenth note, etc.) can be adjusted in properties as **Duration**.

1/96 Clock

Sequencer



Source of a clock signal which corresponds to the external MIDI Clock or the internal master clock.

For each 96th note an event is sent at the output. The value can be set as **Output Value** in the properties.

In contrast to the **Sync Clock** source, the events of the **1/96 Clock** have to be processed by an **Event Freq. Divider** (see section 15). This has the advantage that you can experiment with arbitrary division factors.

Clock

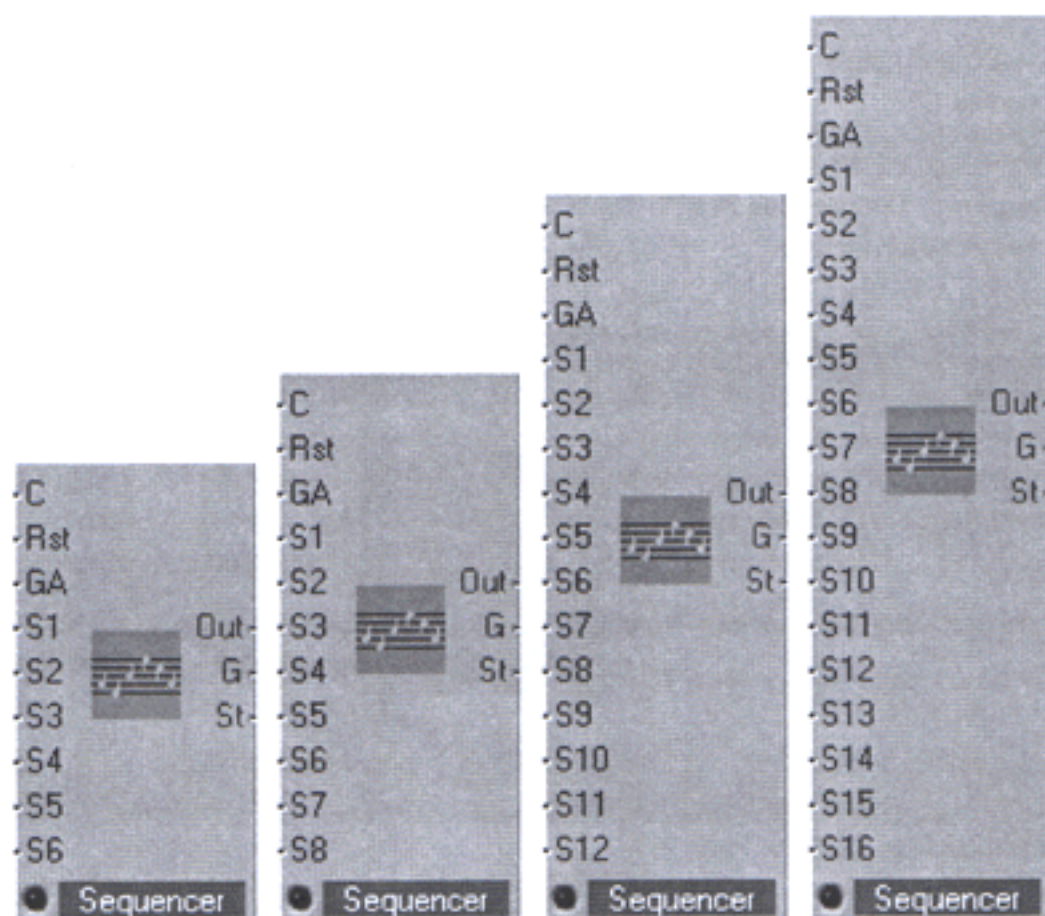
Sequencer



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's 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.
- **Out:** Event output for the clock signal, alternating between on-value and zero.

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.

Demux

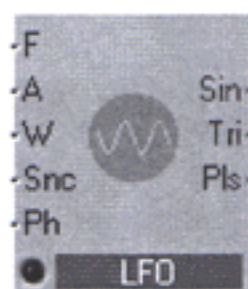
Sequencer

16-input selector switch with **Pos**-input for direct step selection.

10 LFO, Envelope

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's $\frac{3}{4}$ 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).
- **Par:** Event output for the parabolic waveshape signal.
- **Tri:** Event output for the triangle waveshape signal.
- **Pls:** Event output for the pulse waveshape signal.

Slow Random

LFO, Envelope



Low Frequency Oscillator which produces a random step waveform.

- **F:** Control input for the step frequency in Hz.
- **A:** Control input for the amplitude. The output value is somewhere in the range $-A$ to $+A$.
- **Out:** Event output for the random signal.

H - Env

LFO, Envelope



Envelope generator with hold characteristic. When the envelope is triggered the output value jumps to the current value of the amplitude input and stays there until the hold time has passed, after which the output jumps back to zero. The envelope can be triggered at any time, including retriggering during the hold time.

- **T:** Audio input for triggering the envelope on a rising edge (value leaves zero in positive direction).
- **A:** Audio input for the output value during the hold time. The input is sampled at the triggering instant after which the value is held at the output.
- **H:** Logarithmic event input for controlling the hold time. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (i.e. in $\text{dB}_{1\text{ms}}$).
- **Out:** Audio output for the envelope signal.

HR - Env

LFO, Envelope



Envelope generator with hold-release characteristic. When the envelope is triggered the output value jumps to the current value of the amplitude input and stays there until the hold time has passed, after which the output decays exponentially with the release time back to zero.

- **T:** Audio input for triggering the envelope on a rising edge (value leaves zero in positive direction).
- **A:** Audio input for the output value during the hold time. The input is sampled at the triggering instant after which the value is held at the output.
- **H:** Logarithmic event input for controlling the hold time. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (i.e. in $\text{dB}_{1\text{ms}}$).
- **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.

D - Env

LFO, Envelope



Envelope generator with decay characteristic. When the envelope is triggered the output value jumps to the current value of the amplitude input, after which the output decays exponentially with the decay time back to zero.

- **T:** Audio input for triggering the envelope on a rising edge (value leaves zero in positive direction).
- **A:** Audio input for the initial output value. The input is sampled at the triggering instant.
- **D:** Logarithmic event input for controlling the decay time. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (i.e. in $\text{dB}_{1\text{ms}}$).

- **Out:** Audio output for the envelope signal.

DR - Env

LFO, Envelope



Envelope generator with decay-release characteristic. When the envelope is triggered with a gate event the output value jumps to the amplitude value of the gate signal, after which the output decays exponentially with the decay time back to zero. A gate event with amplitude zero (at note off) sets the decaying time to the release time value which is normally set to be shorter.

- **G:** Event input for the gate signal that triggers the envelope. The amplitude of the gate signal determines the initial output value.
- **D:** Logarithmic event input for controlling the decay time. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (i.e. in $\text{dB}_{1\text{ms}}$).
- **R:** Logarithmic event input for controlling the release time. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (i.e. in $\text{dB}_{1\text{ms}}$).
- **Out:** Audio output for the envelope signal.

DSR - Env

LFO, Envelope



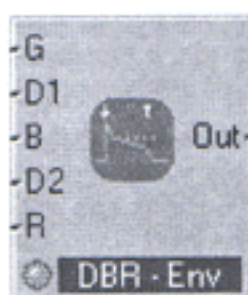
Envelope generator with decay-sustain-release characteristic. When the envelope is triggered with a gate event the output value jumps to the amplitude value of the gate signal, after which the output decays exponentially with the decay time to the sustain level (multiplied by the amplitude). After a gate event with amplitude zero (at note off) the output decays exponentially with the release time back to zero.

- **G:** Event input for the gate signal that triggers the envelope. The amplitude of the gate signal determines the initial output value.

- **D:** Logarithmic event input for controlling the decay time. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (i.e. in $\text{dB}_{1\text{ms}}$).
- **S:** Event input for controlling the sustain level. Typical range of values is: 0 (decay to zero) to 1 (hold at initial level), but sustain can be greater than 1 or even less than 0.
- **R:** Logarithmic event input for controlling the release time. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (i.e. in $\text{dB}_{1\text{ms}}$).
- **Out:** Audio output for the envelope signal.

DBR - Env

LFO, Envelope

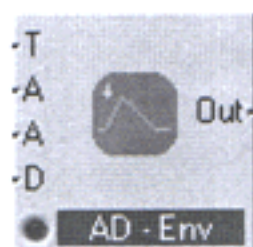


Envelope generator with decay-breakpoint-release characteristic. When the envelope is triggered with a gate event the output value jumps to the amplitude value of the gate signal, after which the output decays exponentially with the decay-1 time until it reaches the breakpoint level (multiplied by the amplitude). Next it continues decaying exponentially with the decay-2 time back to zero. A gate event with amplitude zero (at note off) sets the decaying time to the release time value which is normally set to be shorter.

- **G:** Event input for the gate signal that triggers the envelope. The amplitude of the gate signal determines the initial output value.
- **D1:** Logarithmic event input for controlling the first decay time. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (i.e. in $\text{dB}_{1\text{ms}}$).
- **B:** Event input for controlling the breakpoint level. Range of values: 0 (never use decay-2 time) to 1 (immediately use decay-2 time).
- **D2:** Logarithmic event input for controlling the second decay time. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (i.e. in $\text{dB}_{1\text{ms}}$).
- **R:** Logarithmic event input for controlling the release time. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (i.e. in $\text{dB}_{1\text{ms}}$).
- **Out:** Audio output for the envelope signal.

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.
- **Att:** Control input for the attack time of the envelope. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (i.e. in dB_{1ms}).
- **Dec:** Control input for the decay time of the envelope. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (i.e. in dB_{1ms}).
- **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 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.

ADSR - Env

LFO, Envelope

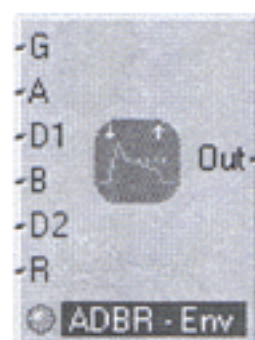


Envelope generator with the classic attack-decay-sustain-release characteristic. When the envelope is triggered with a gate event the output value rises during the attack time with a linear slope to the amplitude value of the gate signal, after which it decays exponentially with the decay time to the sustain level (multiplied by the amplitude). The output is held at the sustain level until a gate event with amplitude zero (at note off) arrives, after which the output decays exponentially with the release time back to zero.

- **G:** Event input for the gate signal that triggers the envelope. The amplitude of the gate signal determines the maximum output value.
- **A:** Logarithmic event input for controlling the attack time. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (i.e. in $\text{dB}_{1\text{ms}}$).
- **D:** Logarithmic event input for controlling the decay time. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (i.e. in $\text{dB}_{1\text{ms}}$).
- **S:** Event input for controlling the sustain level. Typical range of values is: 0 (decay to zero) to 1 (hold at end of attack), but sustain can be greater than 1 or even less than 0.
- **R:** Logarithmic event input for controlling the release time. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (i.e. in $\text{dB}_{1\text{ms}}$).
- **Out:** Audio output for the envelope signal.

ADBDR - Env

LFO, Envelope

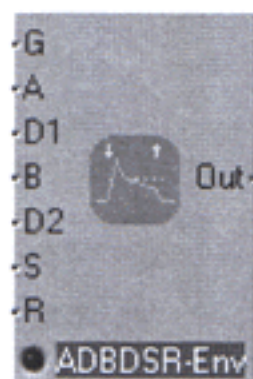


Envelope generator with attack-decay-breakpoint-decay-release characteristic. When the envelope is triggered with a gate event the output value rises during the attack time with a linear slope to the amplitude value of the gate signal, after which it decays exponentially with the decay-1 time until it reaches the breakpoint level (multiplied by the amplitude). Next it continues decaying exponentially with the decay-2 time back to zero. A gate event with amplitude zero (at note off) sets the decaying time to the release time value which is normally set to be shorter.

- **G:** Event input for the gate signal that triggers the envelope. The amplitude of the gate signal determines the maximum output value.
- **A:** Logarithmic event input for controlling the attack time. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (i.e. in dB_{rms}).
- **D1:** Logarithmic event input for controlling the first decay time. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (i.e. in dB_{rms}).
- **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_{rms}).
- **R:** Logarithmic event input for controlling the release time. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (i.e. in dB_{rms}).
- **Out:** Audio output for the envelope signal.

ADBDSR-Env

LFO, Envelope

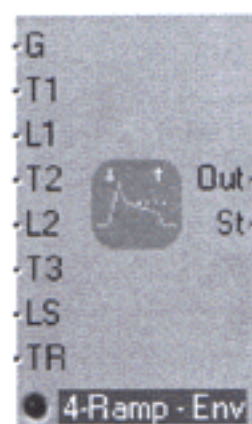


Envelope generator with attack-decay-breakpoint-decay-sustain-release characteristic. When the envelope is triggered with a gate event the output value rises during the attack time with a linear slope to the amplitude value of the gate signal, after which it decays according to the first decay time with a linear slope to the breakpoint level. From there it continues with the second decay time to the sustain level (the levels are multiplied by the amplitude). The output is held at the sustain level until a gate event with amplitude zero (at note off) arrives, after which the output decays exponentially with the release time back to zero.

- **G:** Event input for the gate signal that triggers the envelope. The amplitude of the gate signal determines the maximum output value.
- **A:** Logarithmic event input for controlling the attack time. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (i.e. in dB_{time}).
- **D1:** Logarithmic event input for controlling the first decay time. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (i.e. in dB_{time}).
- **B:** Event input for controlling the break point level. Typical range of values is: 0 to 1.
- **D2:** Logarithmic event input for controlling the second decay time. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (i.e. in dB_{time}).
- **S:** Event input for controlling the sustain level. Typical range of values is: 0 to 1.
- **R:** Logarithmic event input for controlling the release time. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (i.e. in dB_{time}).
- **Out:** Audio output for the envelope signal.

4-Ramp

LFO, Envelope

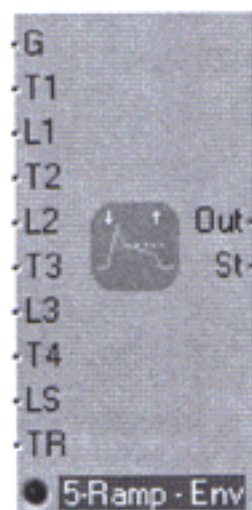


4-stage envelope generator with linear slopes. When the envelope is triggered with a gate event the output value rises to the first level, reaching it within the time set for the first stage. Then it continues to the second level within the time set for the second stage, and so on. The third level is the sustain level, at which the output is held until a gate event with amplitude zero (at note off) arrives, after which the output returns to zero within the last time-period.

- **G:** Event input for the gate signal that triggers the envelope. The amplitude of the gate signal combines with all the level values to determine the actual levels reached.
- **T1:** Logarithmic event input for controlling the time for the first stage. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (i.e. in dB_{time}).
- **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_{time}).
- **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_{time}).
- **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 dB_{time}).
- **Stg:** Event output for the current stage in which the envelope is (1, 2 ...). After the end of the release stage and before a new trigger, the value is 0. With appropriate processing this value in can be used to chain other envelopes, or for the envelope to retrigger itself.
- **Out:** Audio output for the envelope signal.

5-Ramp

LFO, Envelope



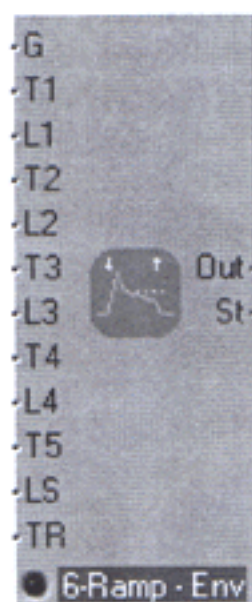
5-stage envelope generator with linear slopes. When the envelope is triggered with a gate event the output value rises to the first level, reaching it within the time set for the first stage. Then it continues to the second level within the time set for the second stage, and so on. The fourth level is the sustain level, at which the output is held until a gate event with amplitude zero (at note off) arrives, after which the output returns to zero within the last time-period.

- **G:** Event input for the gate signal that triggers the envelope. The amplitude of the gate signal combines with all the level values to determine the actual levels reached.
- **T1:** Logarithmic event input for controlling the time for the first stage. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (i.e. in $\text{dB}_{1\text{ms}}$).
- **L1:** Input for controlling the level which is reached at the end of the first stage.
- **T2:** Logarithmic event input for controlling the time for the second stage. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (i.e. in $\text{dB}_{1\text{ms}}$).
- **L2:** Input for controlling the level which is reached at the end of the second stage.
- **T3:** Logarithmic event input for controlling the time for the third stage. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (i.e. in $\text{dB}_{1\text{ms}}$).
- **L3:** Input for controlling the level which is reached at the end of the third stage.
- **T4:** Logarithmic event input for controlling the time for the fourth stage. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (i.e. in $\text{dB}_{1\text{ms}}$).
- **LS:** Input for controlling the level which is reached at the end of the fourth stage. This is the sustain level.
- **TR:** Logarithmic event input for controlling the time for the last stage which is the release. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (i.e. in $\text{dB}_{1\text{ms}}$).

- **Stg:** Event output for the current stage in which the envelope is (1, 2 ...). After the end of the release stage and before a new trigger, the value is 0. With appropriate processing this value in can be used to chain other envelopes, or for the envelope to retrigger itself.
- **Out:** Audio output for the envelope signal.

6-Ramp

LFO, Envelope



6-stage envelope generator with linear slopes. When the envelope is triggered with a gate event the output value rises to the first level, reaching it within the time set for the first stage. Then it continues to the second level within the time set for the second stage, and so on. The fifth level is the sustain level, at which the output is held until a gate event with amplitude zero (at note off) arrives, after which the output returns to zero within the last time-period.

- **G:** Event input for the gate signal that triggers the envelope. The amplitude of the gate signal combines with all the level values to determine the actual levels reached.
- **T1:** Logarithmic event input for controlling the time for the first stage. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (i.e. in dB_{1ms}).
- **L1:** Input for controlling the level which is reached at the end of the first stage.
- **T2:** Logarithmic event input for controlling the time for the second stage. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (i.e. in dB_{1ms}).
- **L2:** Input for controlling the level which is reached at the end of the second stage.
- **T3:** Logarithmic event input for controlling the time for the third stage. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (i.e. in 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 $\text{dB}_{1\text{ms}}$).
- **L4:** Input for controlling the level which is reached at the end of the fourth stage.
- **T5:** Logarithmic event input for controlling the time for the fifth stage. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (i.e. in $\text{dB}_{1\text{ms}}$).
- **LS:** Input for controlling the level which is reached at the end of the fifth stage. This is the sustain level.
- **TR:** Logarithmic event input for controlling the time for the last stage which is the release. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (i.e. in $\text{dB}_{1\text{ms}}$).
- **Stg:** Event output for the current stage in which the envelope is (1, 2 ...). After the end of the release stage and before a new trigger, the value is 0. With appropriate processing this value in can be used to chain other envelopes, or for the envelope to retrigger itself.
- **Out:** Audio output for the envelope signal.

11 Filter

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. Like this they are all equally suited for use 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 turned on with the switches **Visible in Instr.** and **Visible in Ens.** under **Panel Appearance** in the module's properties. These turn on the display in the instrument panel or the ensemble panel respectively. The size of the graph can be set in the properties with **Pixel in X** and **Pixel in Y**. The graph's frequency axis is logarithmic and ranges from 10 Hz to 20 kHz.

HP/LP 1-Pole

Filter

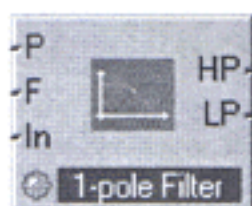


1-pole filter with high pass and low pass outputs (6 dB/octave falloff) and logarithmic control of cutoff frequency.

- **P:** Logarithmic event input for controlling the cutoff frequency. Value in semitones (69 = 440 Hz).
- **In:** Audio signal input for the signal to be filtered
- **HP:** Audio output for the high pass filtered signal
- **LP:** Audio output for the low pass filtered signal

HP/LP 1-Pole FM

Filter

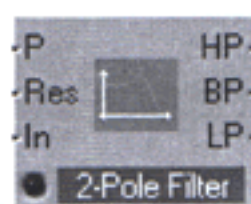


1-pole filter with high pass and low pass outputs (6 dB/octave falloff), logarithmic and linear control of cutoff frequency.

- **P:** Logarithmic event input for controlling the cutoff frequency. Value in semitones (69 = 440 Hz).
- **F:** Audio input for linear control of the cutoff frequency. Value in Hz. **P** and **F** together determine the oscillator frequency: $f_0 = F + 2^{(P-69)/12} \cdot 440\text{Hz}$
- **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

Multi 2-Pole

Filter



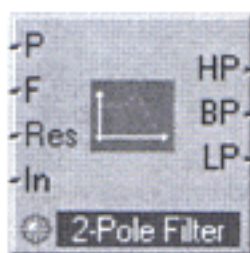
2-pole filter with high pass and low pass outputs (12 dB/octave falloff), band pass (above and below each 6 dB/octave falloff), variable resonance, and logarithmic control of cutoff frequency.

The pass band gain is always 1 (0 dB), whereas the gain at the cutoff frequency increases with the resonance. Caution: Very large amplitudes are generated when **Res** is almost 1.

- **P:** Logarithmic event input for controlling the cutoff frequency. Value in semitones (69 = 440 Hz).
- **Res:** Event input for controlling the filter's resonance/damping. Range of values 0 (maximal damping, no resonance) to 1 (no damping, maximal resonance, self oscillation!). At high resonance the filter's Q-factor = frequency [Hz] / bandwidth [Hz] = $1 / (2 - 2 \text{ Res})$.
- **In:** Audio signal input for the signal to be filtered
- **HP:** Audio output for the high pass filtered signal
- **BP:** Audio output for the band pass filtered signal
- **LP:** Audio output for the low pass filtered signal

Multi 2-Pole FM

Filter



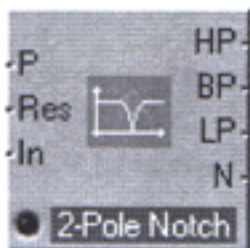
2-pole filter with high pass and low pass outputs (12 dB/octave falloff), band pass (above and below each 6 dB/octave falloff), variable resonance, logarithmic and linear control of cutoff frequency.

The pass band gain is always 1 (0 dB), whereas the gain at the cutoff frequency increases with the resonance. Caution: Very large amplitudes are generated when **Res** is almost 1.

- **P**: Logarithmic event input for controlling the cutoff frequency. Value in semitones (69 = 440 Hz).
- **F**: Audio input for linear control of the cutoff frequency. Value in Hz. **P** and **F** together determine the oscillator frequency: $f_0 = F + 2^{(P-69)/12} \cdot 440\text{Hz}$
- **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] $\approx 1 / (2 - 2 \text{ Res})$.
- **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 Hz).
- **Res:** Event input for controlling the filter's resonance/damping. Range of values 0 (maximal damping, no resonance) to 1 (no damping, maximal resonance, self oscillation!). At high resonance the filter's Q-factor = frequency [Hz] / bandwidth [Hz] = $1 / (2 - 2 \text{ Res})$.
- **In:** Audio signal input for the signal to be filtered
- **HP:** Audio output for the high pass filtered signal
- **BP:** Audio output for the band pass filtered signal
- **LP:** Audio output for the low pass filtered signal
- **N:** Audio output for the band reject (notch) filtered signal

Multi/Notch 2-Pole FM

Filter



2-pole filter with band reject (notch) output, high pass and low pass outputs (12 dB/octave falloff), band pass (above and below each 6 dB/octave falloff), variable resonance, logarithmic and linear control of cutoff frequency.

The gain at the cutoff frequency is always 1 (0 dB), whereas the pass band gain decreases with increasing resonance.

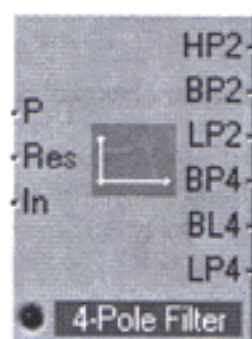
At the notch filter output the selected frequency is completely removed.

- **P:** Logarithmic event input for controlling the cutoff frequency. Value in semitones (69 = 440 Hz).

- **F:** Audio input for linear control of the cutoff frequency. Value in Hz. **P** and **F** together determine the oscillator frequency: $f_0 = F + 2^{(P-69)/12} \cdot 440\text{Hz}$
- **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] $\approx 1 / (2 - 2 \text{ Res})$.
- **In:** Audio signal input for the signal to be filtered
- **HP:** Audio output for the high pass filtered signal
- **BP:** Audio output for the band pass filtered signal
- **LP:** Audio output for the low pass filtered signal
- **N:** Audio output for the band reject (notch) filtered signal

Multi/LP 4-Pole

Filter



4-pole filter with low pass (24 dB/oct falloff), band pass (12/12 dB/oct) and band/low pass (6/18 dB/oct) outputs, 2-pole high and low pass (12 dB/oct) and band pass (6/6 dB/oct) outputs, variable resonance and logarithmic control of cutoff frequency.

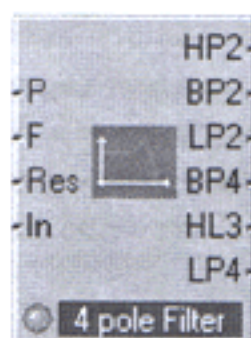
The pass band gain is always 1 (0 dB), whereas the gain at the cutoff frequency increases with the resonance. Caution: Very large amplitudes are generated when **Res** is almost 1.

- **P:** Logarithmic event input for controlling the cutoff frequency. Value in semitones (69 = 440 Hz).
- **Res:** Event input for controlling the filter's resonance/damping. Range of values 0 (maximal damping, no resonance) to 1 (no damping, maximal resonance, self oscillation!).
- **In:** Audio signal input for the signal to be filtered
- **HP2:** Audio output for the 2-pole high pass filtered signal
- **BP2:** Audio output for the 2-pole band pass filtered signal
- **LP2:** Audio output for the 2-pole low pass filtered signal
- **BP4:** Audio output for the 4-pole band pass filtered signal

- **BL4:** Audio output for the 4-pole band/low pass filtered signal
- **LP4:** Audio output for the 4-pole low pass filtered signal

Multi/LP 4-Pole FM

Filter



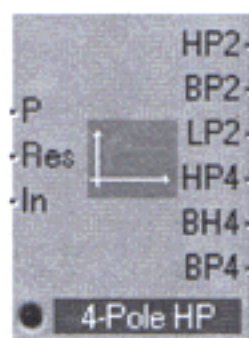
4-pole filter with low pass (24 dB/oct falloff), band pass (12/12 dB/oct) and band/low pass (6/18 dB/oct) outputs, 2-pole high and low pass (12 dB/oct) and band pass (6/6 dB/oct) outputs, variable resonance, logarithmic and linear control of cutoff frequency.

The pass band gain is always 1 (0 dB), whereas the gain at the cutoff frequency increases with the resonance. Caution: Very large amplitudes are generated when **Res** is almost 1.

- **P:** Logarithmic event input for controlling the cutoff frequency. Value in semitones (69 = 440 Hz).
- **F:** Audio input for linear control of the cutoff frequency. Value in Hz. **P** and **F** together determine the oscillator frequency: $f_0 = F + 2^{(P-69)/12} \cdot 440\text{Hz}$
- **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!).
- **In:** Audio signal input for the signal to be filtered
- **HP2:** Audio output for the 2-pole high pass filtered signal
- **BP2:** Audio output for the 2-pole band pass filtered signal
- **LP2:** Audio output for the 2-pole low pass filtered signal
- **BP4:** Audio output for the 4-pole band pass filtered signal
- **BL4:** Audio output for the 4-pole band/low pass filtered signal
- **LP4:** Audio output for the 4-pole low pass filtered signal

Multi/HP 4-Pole

Filter



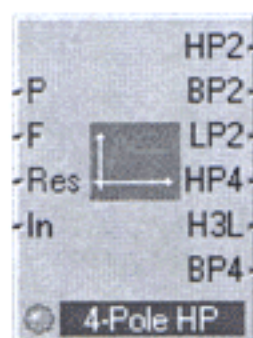
4-pole filter with high pass (24 dB/oct falloff), band pass (12/12 dB/oct) and band/high pass (18/6 dB/oct) outputs, 2-pole high and low pass (12 dB/oct) and band pass (6/6 dB/oct) outputs, variable resonance and logarithmic control of cutoff frequency.

The pass band gain is always 1 (0 dB), whereas the gain at the cutoff frequency increases with the resonance. Caution: Very large amplitudes are generated when **Res** is almost 1.

- **P**: Logarithmic event input for controlling the cutoff frequency. Value in semitones (69 = 440 Hz).
- **Res**: Event input for controlling the filter's resonance/damping. Range of values 0 (maximal damping, no resonance) to 1 (no damping, maximal resonance, self oscillation!).
- **In**: Audio signal input for the signal to be filtered
- **HP2**: Audio output for the 2-pole high pass filtered signal
- **BP2**: Audio output for the 2-pole band pass filtered signal
- **LP2**: Audio output for the 2-pole low pass filtered signal
- **HP4**: Audio output for the 4-pole high pass filtered signal
- **BH4**: Audio output for the 4-pole band/high pass filtered signal
- **BP4**: Audio output for the 4-pole band pass filtered signal

Multi/HP 4-Pole FM

Filter



4-pole filter with high pass (24 dB/oct falloff), band pass (12/12 dB/oct) and band/high pass (18/6 dB/oct) outputs, 2-pole high and low pass (12 dB/oct) and band pass (6/6 dB/oct) outputs, variable resonance, logarithmic and linear control of cutoff frequency.

The pass band gain is always 1 (0 dB), whereas the gain at the cutoff frequency increases with the resonance. Caution: Very large amplitudes are generated when **Res** is almost 1.

- **P**: Logarithmic event input for controlling the cutoff frequency. Value in semitones (69 = 440 Hz).
- **F**: Audio input for linear control of the cutoff frequency. Value in Hz. **P** and **F** together determine the oscillator frequency: $f_0 = F + 2^{(P-69)/12} \cdot 440\text{Hz}$
- **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!).
- **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

Peak EQ

Filter

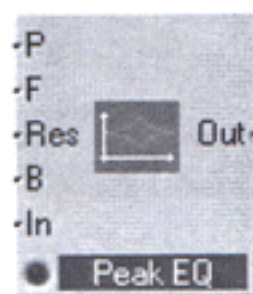


Parametric equalizer with adjustable boost/cut, band width and logarithmic frequency control. With the Peak EQ a specific frequency in the signal and a narrow or wide band of frequencies around it can be amplified or attenuated. More distant frequencies are not affected.

- **P:** Logarithmic event input for controlling the cutoff frequency. Value in semitones ($69 = 440 \text{ Hz}$).
- **Res:** Event input for controlling the filter's resonance (Q-factor). Range of values 0 (minimal resonance, maximal band width) to 1 (maximal resonance, minimal band width). At high resonance the filter's Q-factor = frequency [Hz] / bandwidth [Hz] $\approx 1 / (2 - 2 \text{ Res})$.
- **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

Peak EQ FM

Filter



Parametric equalizer with adjustable boost/cut, band width and logarithmic and linear frequency control. With the Peak EQ a specific frequency in the signal and a narrow or wide band of frequencies around it can be amplified or attenuated. More distant frequencies are not affected.

- **P:** Logarithmic event input for controlling the cutoff frequency. Value in semitones ($69 = 440 \text{ Hz}$).

- **F:** Audio input for linear control of the cutoff frequency. Value in Hz. **P** and **F** together determine the oscillator frequency; $f_0 = F + 2^{(P-69)/12} \cdot 440\text{Hz}$
- **Res:** Event input for controlling the filter's resonance (Q-factor). Range of values 0 (minimal resonance, maximal band width) to 1 (maximal resonance, minimal band width). At high resonance the filter's Q-factor = frequency [Hz] / bandwidth [Hz] $\approx 1 / (2 - 2 \text{ Res})$.
- **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



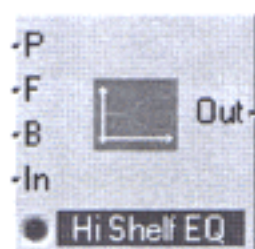
Parametric equalizer with high shelving characteristic, adjustable boost/cut, band width and logarithmic frequency control.

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



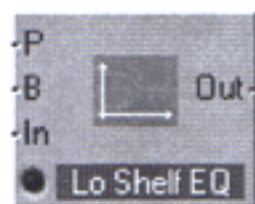
Parametric equalizer with high shelving characteristic, adjustable boost/cut, band width and logarithmic and linear frequency control.

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 ($69 = 440$ Hz).
- **F:** Audio input for linear control of the cutoff frequency. Value in Hz. **P** and **F** together determine the oscillator frequency: $f_0 = F + 2^{(P-69)/12} \cdot 440\text{Hz}$
- **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

Low Shelf EQ

Filter



Parametric equalizer with low shelving characteristic, adjustable boost/cut, band width and logarithmic frequency control.

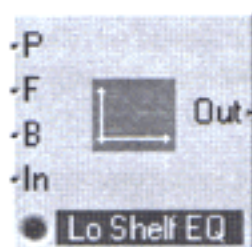
The level of frequencies below the corner frequency is raised or reduced by the set amount. High frequencies are not affected.

- **P:** Logarithmic event input for controlling the cutoff frequency. Value in semitones ($69 = 440$ Hz).

- **B:** Event input for controlling boost/cut in dB. At value **B** = 0 the signal remains unchanged.
- **In:** Audio signal input for the signal to be equalized
- **Out:** Audio output for the equalized signal

Low Shelf EQ FM

Filter



Parametric equalizer with low shelving characteristic, adjustable boost/cut, band width and logarithmic and linear frequency control.

The level of frequencies below the corner frequency is raised or reduced by the set amount. High frequencies are not affected.

- **P:** Logarithmic event input for controlling the cutoff frequency. Value in semitones (69 = 440 Hz).
- **F:** Audio input for linear control of the cutoff frequency. Value in Hz. **P** and **F** together determine the oscillator frequency: $f_0 = F + 2^{(P-69)/12} \cdot 440\text{Hz}$
- **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

Differentiator

Filter



The differentiator gives the slope of the input signal in change units per millisecond. The effect is like a high pass filter where the amplification is proportional to frequency. Unity gain at 159 Hz.

- **In:** Audio signal input for the signal to be differentiated

- **Out:** Audio output for the differentiated signal

Integrator

Filter



Integrator with Reset input. The output value changes with a slope given by the input in change units per millisecond. The effect is like a low pass filter where amplification is proportional to $1/\text{frequency}$. Unity gain at 159 Hz.

- **In:** Audio signal input for the signal to be integrated
- **Rst:** 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

12 Delay

Static Delay

Delay



Polyphonic delay line for audio signals. The input signal appears at the output with a delay corresponding to the set time. The delay time is controlled by events at the **Dly** input and is always rounded to an integer number of samples.

The upper limit for the delay time can be adjusted in the module's properties dialog window (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.

- **Dly:** Event input for controlling the delay time. Value in milliseconds
- **In:** Audio input for the signal to be delayed.
- **Out:** Audio output for the delayed signal

Modulation Delay

Delay



Polyphonic delay line for audio signals. The input signal appears at the output with a delay corresponding to the value 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 upper limit for the delay time can be adjusted in the module's properties dialog window (default: 1 second) and affects the memory usage. 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 172 kB of RAM for each second of buffer length and for each voice. For one minute you need 10 MB.

- **Dly:** Audio input for controlling the delay time. Value in milliseconds
- **In:** Audio input for the signal to be delayed.
- **Out:** Audio output for the delayed signal

Diffuser Delay

Delay



The diffuser is an all-pass filter containing a delay line with feedback. Its function is to smear (decorrelate) the input signal without emphasizing any frequency components. Its typical use is as a building block for reverb type effects, where you would connect several of these modules in series and give them different delay times of a few milliseconds.

The delay time is controlled by events at the **Dly** input and is always rounded to an integer number of samples. 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:** Event 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

Grain Delay

Delay



 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.

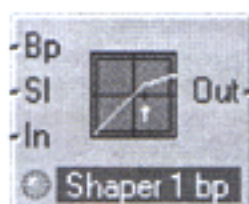
In the properties dialog window the playback quality and also the upper limit of the delay time can be set. The available maximum delay time can deviate from the set amount by up to 50 %.

- **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.

13 Shaper

Shaper 1 BP

Shaper



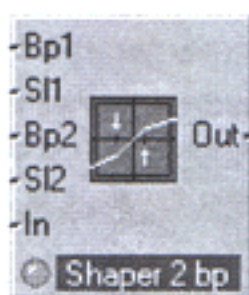
Signal shaper with piecewise linear input/output curve and one breakpoint.

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

Shaper



Signal shaper with piecewise linear input/output curve and two breakpoint.

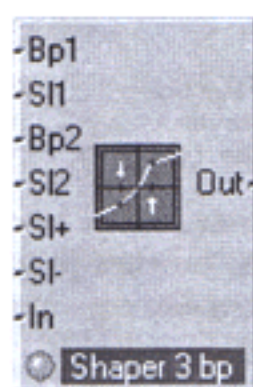
The slope of the curve above the upper and below the lower breakpoint can be adjusted. Signal values between the breakpoints are passed unchanged.

- **Bp1:** Audio input for controlling the level of the upper breakpoint

- **SI1:** Audio input for controlling the slope of the upper part of the input/output curve (1 = no change in signal).
- **Bp2:** Audio input for controlling the level of the lower breakpoint
- **SI2:** Audio input for controlling the slope of the lower part of the input/output curve (1 = no change in signal).
- **In:** Audio input for the signal to be shaped
- **Out:** Audio output for the shaped signal

Shaper 3 BP

Shaper



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
- **SI1:** 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
- **SI2:** Audio input for controlling the slope of the lower part of the input/output curve (1 = no change in signal).
- **SI+:** Audio input for controlling the slope of the input/output curve between zero and the upper breakpoint (1 = no change in signal).
- **SI-:** 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

Shaper



Signal shaper with parabolic (2^{nd} order polynomial) input/output curve.

The linear and square parts of the signal can be adjusted ($\text{Out} = \text{Lin In} + \text{Par In}^2$).

- **Lin:** Audio input for controlling the level of the linear, undistorted part
- **Par:** Audio input for controlling the level of the square, distorted part
- **In:** Audio input for the signal to be shaped
- **Out:** Audio output for the shaped signal

Shaper Cubic

Shaper



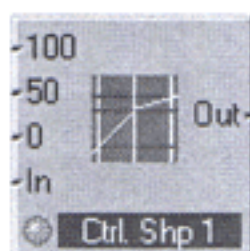
Signal shaper with cubic parabolic (3^{rd} order polynomial) input/output curve.

The linear, square and cubic parts of the signal can be adjusted ($\text{Out} = \text{Lin In} + \text{Par In}^2 + \text{Cub In}^3$).

- **Lin:** Audio input for controlling the level of the linear, undistorted part
- **Par:** Audio input for controlling the level of the square, distorted part
- **Cub:** Audio input for controlling the level of the cubic, distorted part
- **In:** Audio input for the signal to be shaped
- **Out:** Audio output for the shaped signal

Ctrl. Shaper 1 BP

Shaper

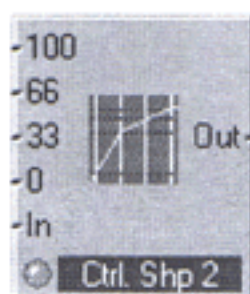


Event signal shaper with piecewise linear input/output curve and one fixed breakpoint. The output it produced by linear interpolation between the given points.

- **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

Shaper

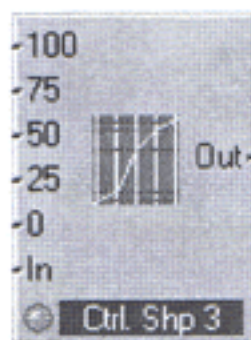


Event signal shaper with piecewise linear input/output curve and two fixed breakpoints. The output it produced by linear interpolation between the given points.

- **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

Shaper

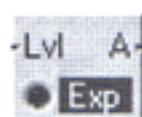


Event signal shaper with piecewise linear input/output curve and three fixed breakpoints. The output it produced by linear interpolation between the given points.

- **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

Event Expon. (A)

Shaper

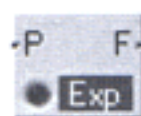


Event exponential function with scaling as for level control inputs (e.g. on a **Mixer** module). With this converter a linear amplitude control input can be controlled like a level control input with a logarithmic event value in dB.

- **Lvl:** Event input for the logarithmic amplitude control signal. 0 [dB] at **A** = 1, 20 [dB] at **A** = 10
- **A:** Event output for the linear amplitude control signal

Event Expon. (F)

Shaper



Event exponential function with scaling as for pitch control inputs (e.g. on an oscillator module). With this converter a linear frequency control input can be controlled like a pitch control input with a logarithmic event value.

- **P:** Event input for the logarithmic pitch control signal. Value in semitones, 69 = A3 (440 Hz).
- **F:** Event output for the linear frequency control signal

Event Log (A)

Shaper



Logarithmic converter for level control. A linear amplitude control signal is converted to a logarithmic level control signal (in dB).

This converter can also be used to control logarithmic time control inputs in dB_{time} (e.g. in envelopes) using linear time values in ms (according to the table on page 122).

- **A:** Event input for the amplitude control signal
- **Lvl:** Event output for the logarithmic level control signal.
0 [dB] at **A** = 1, 20 [dB] at **A** = 10

Event Log (F)

Shaper



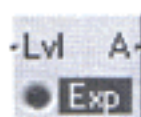
Logarithmic converter for level control. A linear amplitude control signal is converted to a logarithmic level control signal (in dB).

- **F:** Event input for the linear frequency control signal

- **P:** Event output for the logarithmic pitch control signal. Value in semitones, 69 = A3 (440 Hz).

Audio Expon. (A)

Shaper

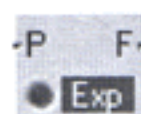


Audio exponential function with scaling as for logarithmic amplitude control inputs. With this converter a linear control input can be controlled with a logarithmic audio signal.

- **Lvl:** Audio input for the logarithmic amplitude control signal. 0 [dB] at **A** = 1, 20 [dB] at **A** = 10
- **A:** Audio output for the linear amplitude control signal

Audio Expon. (F)

Shaper



Audio exponential function with scaling as for logarithmic frequency control inputs. With this converter a linear control input can be controlled with a logarithmic audio signal.

- **P:** Audio input for the logarithmic frequency control signal. Value in semitones, 69 = A3 (440 Hz).
- **F:** Audio output for the linear frequency control signal. Value in Hz

14 Modifier

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

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

Rectifier

Audio Modifier



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:** Audio input for signal to be rectified. Negative values become positive with equal magnitude.
- **Sign:** Audio output for the sign of the input signal (-1 or +1).
- **Out:** Audio output for the rectified 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 and 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 output 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

Slew Limiter

Audio Modifier

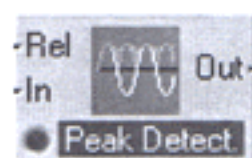


Slew rate limiter with separately adjustable maximum rate for rising and falling signals. The output signal tracks the input signal, but for fast movement and jumps at the input, the output follows with a limited rate of change (ramp) until its value reaches that of the input signal.

- **Up:** Audio input for controlling the maximum slew rate for rising signals. Value in units of 1/sec
- **Dn:** Audio input for controlling the maximum slew rate for falling signals. Value in units of 1/sec
- **In:** Audio input for signal to be slew rate limited
- **Out:** Audio output for slew rate limited signal

Peak Detector

Audio Modifier



Detector for the peak amplitude. The input signal is rectified and smoothed with an adjustable release time. The attack time is zero.

The result of the Peak Detector's action is that the output value follows the amplitude envelope of the input – use this module as an envelope follower.

- **Rel:** Logarithmic event input for controlling the release time. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (i.e. in $\text{dB}_{1\text{ms}}$).
- **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

Quantizer

Audio Modifier



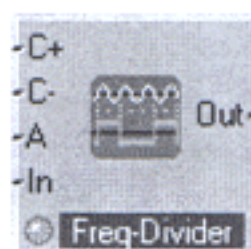
Quantizer for audio signals, with adjustable step size.

The input signal is rounded to the nearest quantization step before being output.

- **St:** Audio input for controlling the size of the quantization step. When set to zero, the signal is not quantized.
- **In:** Audio input for the signal to be quantized
- **Out:** Audio output for the quantized signal
- **Err:** Audio output for the quantization error that is generated by rounding: $\text{Err} = \text{Out} - \text{In}$

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 ($f_{out} = 2 f_{in} / (C+ + C-)$). 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

Relay 1

Audio Modifier



On/off switch for audio signals, with control input. The output signal is either zero or equal to the input signal.

- **Ctrl:** Audio input for control of switching. A value greater than zero switches the input to the output.
- **In:** Audio input for the signal to be switched
- **Out:** Audio output for the switched signal

Relay 2

Audio Modifier



Change-over switch for two audio inputs, with control input. The output signal is equal to the signal of either the first or the second input.

- **Ctrl:** Audio input for control of switching. A value greater than zero switches input 1 to the output, otherwise input 2 is output.
- **In1:** Audio input for the first signal to be switched
- **In2:** Audio input for the second signal to be switched
- **Out:** Audio output for the switched signal

15 Event Processing

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.

Separator

Event Processing



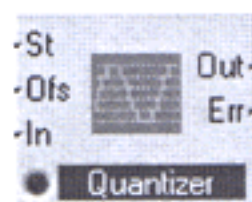
All received events are compared to the threshold value and are sent to either one or the other output.

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.

Quantizer

Event Processing



Quantizer for events, with adjustable step size and grid offset.

The input signal is rounded to the nearest quantization step before being output. The distance between the steps of the quantization grid is set with **St**, and the position of the grid is adjusted with **Ofs**, where **Ofs** gives the value of one of the steps.

- **St**: Audio input for controlling the size of the quantization step. When set to zero, the signal is not quantized.

- **Ofs:** Audio input for controlling the position of the quantization steps. One step is at the value **Ofs**, the next is at **Ofs + St**, then **Ofs + 2 St** etc., but also at **Ofs - St**, **Ofs - 2 St**...
- **In:** Input for the events to be quantized
- **Out:** Output for the quantized events
- **Err:** Output for the quantization error that is generated by rounding the events:

$$\text{Err} = \text{Out} - \text{In}$$

Randomizer

Event Processing



Event randomizer with adjustable spread.

The arriving events are modified with a random positive or negative offset in the given range ($\text{Out} = \text{In} + X$, where $-\text{Rng} \leq X \leq \text{Rng}$).

- **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

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.

Timer

Event Processing



The elapsed time between the two last received events is measured and output as an event. Also, the frequency whose period of oscillation is equal to the measured duration is calculated and output.

- **In:** Polyphonic input for the events whose distance in time is to be measured.
- **F:** Polyphonic event output for the frequency of input events in Hz.
- **T:** Polyphonic event output for the time between events in milliseconds.

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 ... 2.99 : division by 2, 3 ... 3.99 : division by 3, etc.).
- **W:** Control input for the pulse width of the output signal. Range of values: 0 ... 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

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.

Hold

Event Processing



Hold envelope.

When the module is triggered by a positive event at the **T** 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.

- **In:** 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.

Delay

Event Processing



Delay line for events. An event arriving at the input appears at the output after the set time.

In this version of REAKTOR only one event can be in the delay at any one time. If another event arrives before the first one has been output, the first event is lost.

- **Dly:** Event input for controlling the delay time. Value in milliseconds
- **In:** Event input for the signal to be delayed.
- **Out:** Event output for the delayed signal

Router 1

Event Processing

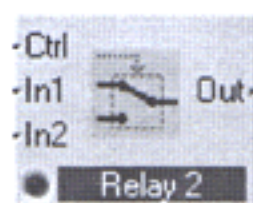


On/off switch for event signals, with control input. The last event passed through is held at the output even while off.

- **Ctrl:** Audio 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 2

Event Processing



Change-over switch for two event inputs, with control input. Depending on the value at the control input, the events from either the first or the second input are passed to the output.

- **Ctrl:** Audio input for control of switching. While the value is greater than zero the events from input 1 are passed to the output, otherwise those from input 2.
- **In1:** Event input for the first signal to be switched
- **In2:** Event input for the second signal to be switched
- **Out:** Event output for the switched signal. After switching inputs, the last value from the old input is held at the output until a new event arrives at the new input.

16 Auxiliary

Tapedeck 1-Ch

Auxiliary



1-channel Tapedeck module for recording and playback of audio signals.

WAV files can be imported for playback with the entry **WAV Import...** in the context menu of the Tapedeck module. A file that is already loaded can be imported again by selecting **WAV Reload**, e.g. if it has been changed using a sample editor.

A recording can be and exported using **WAV Export...** in the context menu. If the file has already been exported, it can be overwritten with **WAV Overwrite**, e.g. if you have made a new recording in the Tapedeck.

If **Show Control** 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.

The maximum recording time is set with **Size (sec)** in the module's properties dialog window (value 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.

If interruptions to the sound due to operating system disturbance or processor overload should occur during recording, this will not affect the recorded sound. It is even possible to set the system sample rate so high that real time processing is no longer possible, and ignore the resulting noises - the recording will be pure.

When importing a WAV 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 **Size (sec)** in the module's properties dialog window.

- **R:** 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.
- **P:** 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.
- **Out:** Audio output for the playback signal or the signal being recorded.

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 WAV 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



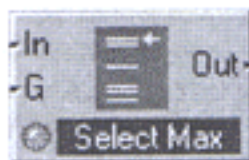
Event Voice Combiner.

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



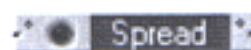
Event Voice Combiner with minimum value selection.

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

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.

Event Merge

Auxiliary

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.

When several events with the same value occur in a row, only the first is passed through, the others are filtered out. Like this, superfluous events can be eliminated.

A to E

Auxiliary

Audio to event converter.

The audio signal is sampled using the Control Rate specified in the **Settings** menu and the values are sent out as a stream of events.

A to E (Trig)

Auxiliary

Audio to event converter with trigger input.

When the trigger input signal leaves zero to positive values (rising edge) the audio signal is sampled and output as an event.

- **T:** Audio input for triggering the conversion. Triggers on a rising edge
- **In:** Audio input for signal to be sampled and output as events
- **Out:** Event output for the sampled signal

A to E (Perm)

Auxiliary

Audio to event converter with permanent conversion at regular intervals and adjustable sampling frequency.

The audio signal is sampled with the given frequency and output as a stream of events. When running this module at a high frequency (above 1000 Hz) the CPU-Load caused by event processing can rise significantly. Typically $F = 200$ Hz is sufficient.

- **F:** Audio input for controlling the sample frequency. Value in Hz
- **In:** Audio input for signal to be sampled and output as events
- **Out:** Event output for the sampled signal

A to Gate

Auxiliary



Audio to gate event converter with gate amplitude input.

When the trigger signal leaves zero to positive values (rising edge), the gate signal is switched on with an amplitude of the current value of the amplitude input. When the trigger input returns to zero or negative values (falling edge) the gate signal is switched off.

- **T:** Audio input for triggering the gate signal
- **A:** Audio input for controlling the amplitude of the gate events
- **Out:** Event output for the gate event signal

Audio Smoother

Auxiliary



Smoother for monophonic event signals with audio output. Typically, a smoother is connected after a fader or button to get smooth transitions.

The jumps in the value of the input event signal are smoothed to ramps. The **Transition Time** (in milliseconds) can be adjusted in the properties. Exactly after this time has elapsed, the output reaches the same value as the input, assuming that no further jumps occurred at the input. The larger the **Transition Time**, the stronger the smoothing effect.

- **In:** Mono event input for the signal to be smoothed.
- **Out:** Mono audio output for the smoothed signal.

Event Smoother

Auxiliary



Smoother for monophonic event signals. Typically, a smoother is connected after a fader or button to get smooth transitions.

The jumps in the value of the input event signal are smoothed to ramps. The **Transition Time** (in milliseconds) can be adjusted in the properties. Exactly after this time has elapsed, the output reaches the same value as the input, assuming that no further jumps occurred at the input. The larger the **Transition Time**, the stronger the smoothing effect.

During the transition, events are output with the rate selected as the **Control Rate** in the **Settings** menu. The higher the rate, the higher the resolution of the smoothing transition.

- **In:** Mono event input for the signal to be smoothed.
- **Out:** Mono event output for the smoothed signal.

Scope

Auxiliary

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 with **Pixel in X** and **Pixel in Y**.

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.

17 Conversion Tables for Control Values

17.1 Control Value / Time

Conversion of logarithmic time control value in $\text{dB}_{1\text{ms}}$, e.g. for envelopes (A, D, D1, D2, R, H). The control value is calculated in dB with the reference value 1 ms. Therefore, the modules **Log (A)** and **Expon. (A)** can also be used for conversion.

Control Value	Time
-40	10 μs
-30	31,62 μs
-20	100 μs
-10	316,2 μs
0	1 ms
10	3,162 ms
20	10 ms
30	31,62 ms
40	100 ms
50	316,2 ms
60	1 s
70	3,162 s
80	10 s
90	31,62 s
100	100 s
110	316,2 s
120	1000 s

Time	Control Value
100 μs	-20
200 μs	-14
500 μs	-6
1 ms	0
2 ms	6
5 ms	14
10 ms	20
20 ms	26
50 ms	34
100 ms	40
200 ms	46
500 ms	54
1 s	60
2 s	66
5 s	74
10 s	80
20 s	86
50 s	94

Conversion Formula

- $\text{Time} = 10^{\text{Value} / 20} \text{ ms}$
- $\text{Value} = 20 \log(\text{Time} / 1 \text{ ms})$

17.2 Pitch / Frequency

Conversion of logarithmic pitch control value, e.g. for oscillators.

Pitch (Semitone)	Frequency
-100	0,02535 Hz
-90	0,04517 Hz
-80	0,08048 Hz
-70	0,1434 Hz
-60	0,2555 Hz
-50	0,4552 Hz
-40	0,8111 Hz
-30	1,4453 Hz
-20	2,5752 Hz
-10	4,5885 Hz
0	8,1758 Hz
10	14,568 Hz
20	25,957 Hz
30	46,249 Hz
40	82,407 Hz
50	146,83 Hz
60	261,63 Hz
69	440 Hz
70	466,16 Hz
80	830,61 Hz
90	1480,0 Hz
100	2637,0 Hz
110	4698,6 Hz
120	8372,0 Hz
130	14917 Hz

Frequency	Pitch (Semitone)
0,01 Hz	-116,10
0,02 Hz	-104,10
0,05 Hz	-88,24
0,1 Hz	-76,24
0,2 Hz	-64,24
0,5 Hz	-48,38
1 Hz	-36,38
2 Hz	-24,38
5 Hz	-8,51
10 Hz	3,49
20 Hz	15,49
50 Hz	31,35
100 Hz	43,35
200 Hz	55,35
500 Hz	71,21
1000 Hz	83,21
2000 Hz	95,21
5000 Hz	111,08
10000 Hz	123,08
20000 Hz	135,08

Conversion Formula

- Frequency = $2^{(P-69)/12} \cdot 440$ Hz
- Pitch = $69 + 12 \log(\text{Frequency} / 440 \text{ Hz}) / \log(2)$

The MIDI note range (C-2 to G8) corresponds to the pitch range 0 to 127