



**NATIVE INSTRUMENTS**

# **REAKTOR 5**

**MODUL- UND CORE REFERENZ**

Der Inhalt dieses Dokuments kann sich unangekündigt ändern und stellt keine Verpflichtung seitens der NATIVE INSTRUMENTS GmbH dar. Die in diesem Dokument beschriebene Software wird unter einer Lizenzvereinbarung zur Verfügung gestellt und darf nicht kopiert werden. Ohne ausdrückliche schriftliche Genehmigung der NATIVE INSTRUMENTS GmbH, im Folgenden als NATIVE INSTRUMENTS bezeichnet, darf kein Teil dieses Handbuchs in irgendeiner Form kopiert, übertragen oder anderweitig reproduziert werden. Alle Produkt- und Firmennamen sind Warenzeichen ihrer jeweiligen Eigentümer.

Desweiteren bedeutet die Tatsache, dass Sie diesen Text lesen, dass Sie der Besitzer einer legalen Version sind und nicht einer illegalen Raubkopie. Nur aufgrund Ihrer Loyalität und Ehrlichkeit kann die NATIVE INSTRUMENTS auch in Zukunft innovative Audio-Software entwickeln. Wir bedanken uns im Namen der gesamten Belegschaft.

Der Autor dieses Handbuchs: **NATIVE INSTRUMENTS** und **Len Sasso**

Besonderer Dank gebührt dem Beta-Test-Team, das uns nicht nur eine unschätzbare Hilfe beim Aufspüren von Fehlern war, sondern mit seinen Vorschlägen ein besseres Produkt entstehen lassen hat.



**NATIVE INSTRUMENTS**

© NATIVE INSTRUMENTS GmbH, 2007. Alle Rechte vorbehalten.

#### **Deutschland**

NATIVE INSTRUMENTS GmbH

Schlesische Str. 28

10997 Berlin

Germany

[info@native-instruments.de](mailto:info@native-instruments.de)

[www.native-instruments.de](http://www.native-instruments.de)

#### **USA**

NATIVE INSTRUMENTS North America, Inc.

5631 Hollywood Boulevard

Los Angeles, CA 90028

USA

[sales@native-instruments.com](mailto:sales@native-instruments.com)

[www.native-instruments.com](http://www.native-instruments.com)

# Inhalt

<b>Modul-Referenz.....</b>	<b>19</b>
<b>Panel .....</b>	<b>21</b>
Fader.....	21
Knob .....	24
Button .....	24
List.....	25
Switch .....	26
Lamp.....	29
Level Lamp .....	30
RGB Lamp .....	30
Meter.....	31
Level Meter .....	32
Picture.....	33
Multi Picture.....	33
Text.....	34
Multi Text.....	34
XY.....	35
Scope .....	37
Multi Display und Poly Display .....	38
Mouse Area .....	41
Stacked Macro.....	44
<b>MIDI In .....</b>	<b>45</b>
Note Pitch.....	45
Pitchbend.....	45
Gate .....	46
Single Trig. Gate.....	46
Sel. Note Gate .....	46
On Velocity.....	47
Off Velocity .....	47
Controller .....	47
Ch. Aftertouch .....	48
Poly Aftertouch .....	48
Sel. Poly AT.....	48
Program Change.....	49
Start/Stop .....	49
1/96 Clock.....	49
Sync Clock .....	50
Song Pos.....	50
Channel Message .....	51

<b>MIDI Out.....</b>	<b>52</b>
Note Pitch/Gate .....	52
Pitchbend.....	52
Controller .....	53
Ch. Aftertouch .....	53
Poly Aftertouch .....	53
Sel. Poly AT.....	54
Program Change.....	54
Start/Stop .....	54
1/96 Clock .....	55
Song Pos .....	55
Channel Message .....	56
<b>Math .....</b>	<b>57</b>
Constant .....	57
Add .....	57
Subtract.....	58
Invert, -X.....	58
Multiply.....	58
a * b+ c .....	59
Reciprocal 1/x.....	59
Divide x/y .....	59
Modulo x % y.....	60
Rectifier .....	60
Rect./Sign .....	60
Compare .....	61
Compare/Equal .....	61
Quantize .....	62
Expon. (A).....	62
Expon. (F).....	63
Log (A) .....	63
Log (F).....	63
Power x y .....	64
Square Root .....	64
1 / Square Root .....	64
Sine.....	65
Sin/Cos .....	65
Arcsin .....	66
Arccos .....	66
Arctan.....	66

<b>Signal Path .....</b>	<b>67</b>
Selector/Scanner .....	67
Relay 1,2 .....	68
Crossfade .....	68
Distributor/Panner .....	69
Stereo Pan .....	69
Amp/Mixer .....	70
Stereo Amp/Mixer .....	71
<b>Oscillator .....</b>	<b>72</b>
Sawtooth .....	72
Saw FM .....	72
Saw Sync .....	73
Saw Pulse .....	74
Bi-Saw .....	74
Triangle .....	75
Tri FM .....	75
Tri Sync .....	76
Tri/Par Symm .....	76
Parabol .....	77
Par FM .....	77
Par Sync .....	78
Par PWM .....	79
Sine .....	79
Sine FM .....	80
Sine Sync .....	80
Multi-Sine .....	81
Pulse .....	82
Pulse FM .....	83
Pulse Sync .....	83
Pulse 1-ramp .....	84
Pulse 2-ramp .....	85
Bi-Pulse .....	86
Impulse .....	86
Impulse FM .....	87
Impulse Sync .....	87
Multi-Step .....	88
4-Step .....	88
5-Step .....	89
6-Step .....	89
8-Step .....	89

Multi-Ramp .....	89
4-Ramp .....	89
5-Ramp .....	90
6-Ramp .....	90
8-Ramp .....	90
Ramp.....	90
Clock Oscillator .....	91
Noise.....	92
Random .....	92
Geiger.....	93
<b>Sampler .....</b>	<b>94</b>
Sampler .....	96
Sampler FM.....	97
Sampler Loop .....	98
Grain Resynth .....	101
Grain Pitch Former .....	105
Grain Cloud .....	109
Beat Loop .....	111
Sample Lookup .....	114
<b>Sequencer.....</b>	<b>115</b>
Sequencer.....	115
6-Step .....	116
8-Step .....	116
12-Step .....	116
16-Step .....	116
Multiplex16 .....	117
<b>LFO, Envelope.....</b>	<b>119</b>
LFO .....	119
Slow Random .....	120
H-Env .....	121
HR-Env .....	121
D-Env .....	122
DR-Env .....	122
DSR-Env .....	123
DBDR-Env.....	123
DBDSR-Env.....	124
AD-Env .....	125
AR-Env .....	126
ADR-Env .....	126
ADSR-Env .....	127

ADBDR-Env .....	128
ADBDSR-Env .....	129
AHDSR - Env .....	130
AHDBDR - Env.....	131
4-Ramp .....	132
5-Ramp .....	133
6-Ramp .....	134
<b>Filter .....</b>	<b>136</b>
HP/LP 1-Pole.....	136
HP/LP 1-Pole FM .....	137
Allpass 1-Pole.....	137
Multi 2-Pole .....	138
Multi 2-Pole FM .....	138
Multi/Notch 2-Pole .....	139
Multi/Notch 2-Pole FM.....	140
Multi/LP 4-Pole.....	141
Multi/LP 4-Pole FM .....	142
Multi/HP 4-Pole .....	143
Multi/HP 4-Pole FM .....	144
Pro-52 Filter.....	145
Ladder Filter.....	145
Ladder Filter FM .....	146
Peak EQ.....	147
Peak EQ FM .....	147
High Shelf EQ.....	148
High Shelf EQ FM .....	149
Low Shelf EQ.....	149
Low Shelf EQ FM .....	150
Differentiator .....	150
Integrator .....	151
<b>Delay .....</b>	<b>152</b>
Single Delay .....	152
Multi-Tap Delay .....	153
Diffuser Delay.....	154
Grain Delay.....	155
Grain Cloud Delay .....	156
Unit Delay .....	158
<b>Audio Modifier.....</b>	<b>159</b>
Saturator.....	159
Saturator 2.....	159

Clipper.....	160
Mod. Clipper.....	160
Mirror 1 Level.....	161
Mirror 2 Levels.....	161
Chopper.....	162
Shaper 1 BP.....	162
Shaper 2 BP.....	163
Shaper 3 BP.....	163
Shaper Parabolic.....	164
Shaper Cubic.....	165
Slew Limiter.....	165
Peak Detector.....	166
Sample & Hold.....	166
Frequency Divider.....	166
Audio Table.....	167
<b>Event Processing.....</b>	<b>169</b>
Accumulator.....	169
Counter.....	170
Randomizer.....	170
Frequency Divider.....	171
Ctrl. Shaper 1 BP.....	171
Ctrl. Shaper 2 BP.....	172
Ctrl. Shaper 3 BP.....	172
Logic AND.....	173
Logic OR.....	173
Logic EXOR.....	173
Logic NOT.....	174
Order.....	174
Iteration.....	175
Separator.....	175
Value.....	176
Merge.....	176
Step Filter.....	176
Router M->1.....	177
Router 1,2.....	177
Router 1->M.....	178
Timer.....	178
Hold.....	179
Event Table.....	179



<b>Auxiliary .....</b>	<b>182</b>
Tapedeck 1-Ch.....	182
Tapedeck 2-Ch.....	185
Audio Voice Combiner .....	185
Event V.C. All .....	186
Event V.C. Max.....	186
Event V.C. Min .....	186
A to E .....	187
A to E (Trig).....	187
A to E (Perm).....	187
A to Gate.....	188
To Voice .....	188
From Voice .....	189
Voice Shift .....	189
Audio Smoother .....	190
Event Smoother .....	190
Master Tune/Level .....	191
Tempo Info.....	191
Voice Info.....	192
Tuning Info .....	192
System Info .....	193
Note Range Info.....	193
MIDI Channel Info .....	194
Snapshot.....	194
Set Random .....	196
Unison Spread .....	196
Snap Value .....	197
Snap Value Array.....	197
<b>Terminals .....</b>	<b>200</b>
In Port .....	200
Out Port.....	200
Send.....	200
Receive .....	200
IC Send .....	202
IC Receive.....	202
OSC Send .....	203
OSC Receive.....	203
<b>Anhang .....</b>	<b>204</b>
Transponieren ankommender MIDI-Noten.....	204

<b>Erste Schritte in REAKTOR Core .....</b>	<b>205</b>
Was ist REAKTOR Core.....	205
Wie Sie Core Cells verwenden .....	206
Der Einsatz von Core Cells in der Praxis.....	209
Grundlegende Bearbeitung von Core Cells .....	211
<b>Der Einstieg in REAKTOR Core.....</b>	<b>217</b>
Die Core-Cell-Typen Event und Audio .....	217
Wie Sie Ihre erste Core Cell erstellen .....	219
Audio- und Kontroll-Signale .....	232
Wie Sie Ihre ersten REAKTOR-Core-Macros bauen .....	239
Wie Sie Audio als Kontroll-Signal verwenden .....	247
Event-Signale.....	249
Logic-Signale.....	254
<b>Grundlagen von REAKTOR Core: Das Core-Signal-Modell .....</b>	<b>256</b>
Werte.....	256
Events.....	256
Gleichzeitige Events.....	259
Verarbeitungsreihenfolge .....	261
Event-Core-Cells im Rückblick .....	263
<b>Strukturen mit internem Zustand.....</b>	<b>269</b>
Clock-Signale.....	269
Object Bus Connections .....	270
Initialisierung.....	273
Wie Sie einen Event-Akkumulierer bauen.....	276
Event Merging.....	278
Event-Akkumulierer mit Reset und Initialisierung.....	279
<b>Audio-Verarbeitung im Kern .....</b>	<b>290</b>
Audio-Signale .....	290
Sampling Rate Clock Bus .....	292
Verbindungs-Feedback .....	293
Feedback um Macros herum.....	297
Denormale Werte .....	301
Andere böse Zahlen.....	305
Wie Sie ein 1-Pol-Tiefpassfilter bauen .....	306
<b>Bedingte Verarbeitung.....</b>	<b>310</b>
Event-Routing .....	310
Wie Sie einen Signal-Beschneider bauen.....	312
Wie Sie einen einfachen Sägezahn-Oszillator aufbauenr .....	313

<b>Weitere Signal-Typen .....</b>	<b>315</b>
Fließkomma-Signale .....	315
Integer-Signale.....	317
Wie Sie einen Event-Zähler bauen .....	321
Wie Sie einen Flanken-Zähler bauen .....	322
<b>Arrays.....</b>	<b>326</b>
Einführung in das Thema “Arrays” .....	326
Wie Sie einen Audio-Signal-Wahlschalter bauen .....	329
Wie Sie ein Delay aufbauen .....	336
Tabellen .....	342
<b>Wie Sie optimale Strukturen aufbauen .....</b>	<b>348</b>
Latches und Modulations-Macros .....	348
Routing und Merging .....	349
Numerische Operationen .....	350
Konvertierungen zwischen Float und Integer .....	351
<b>Anhang A. REAKTOR Cores Bedienoberfläche .....</b>	<b>352</b>
A.1. Core cells .....	352
A.2. Core-Module und -Macros .....	352
A.3. Core ports .....	353
A.4. Core-Strukturen bearbeiten .....	353
<b>Anhang B. Konzepte von REAKTOR Core .....</b>	<b>355</b>
B.1. Signale und Events .....	355
B.2. Initialisierung.....	355
B.3. Verbindungs-Typ OBC .....	355
B.4. Routing.....	356
B.5. Latching.....	356
B.6. Clocking.....	356
<b>Anhang C. Core-Macro-Ports .....</b>	<b>357</b>
C.1. In .....	357
C.2. Out.....	357
C.3. Latch (Eingang).....	357
C.4. Latch (Ausgang).....	357
C.5. Bool C (Eingang) .....	357
C.6. Bool C (Ausgang) .....	358
<b>Anhang D. Core-Cell-Ports .....</b>	<b>359</b>
D.1. In (Audio-Modus).....	359
D.3. In (Event-Modus).....	359

<b>Anhang E. Built-in buses.....</b>	<b>360</b>
E.1. SR.C.....	360
E.2. SR.R.....	360
<b>Anhang F. Built-in modules.....</b>	<b>361</b>
F.1. Const.....	361
F.2. Math > + .....	361
F.3. Math > - .....	361
F.4. Math > * .....	361
F.5. Math > / .....	362
F.6. Math >  x  .....	362
F.7. Math > -x.....	362
F.8. Math > DN Cancel .....	362
F.9. Math > ~log .....	362
F.10. Math > ~exp.....	363
F.11. Bit > Bit AND .....	363
F.13. Bit > Bit XOR.....	363
F.14. Bit > Bit NOT .....	364
F.15. Bit > Bit <<.....	364
F.17. Flow > Router.....	364
F.18. Flow > Compare.....	365
F.19. Flow > Compare Sign.....	365
F.20. Flow > ES Ctl .....	366
F.21. Flow > ~BoolCtl .....	366
F.22. Flow > Merge .....	366
F.23. Flow > EvtMerge.....	366
F.24. Memory > Read .....	367
F.25. Memory > Write.....	367
F.26. Memory > R/W Order .....	367
F.27. Memory > Array .....	368
F.28. Memory > Size [ ] .....	368
F.30. Memory > Table.....	369
F.31. Macro .....	369
<b>Anhang G. Expert macros.....</b>	<b>370</b>
G.1. Clipping > Clip Max / IClip Max .....	370
G.2. Clipping > Clip Min / IClip Min .....	370
G.3. Clipping > Clip MinMax / IClipMinMax .....	370
G.4. Math > 1 div x .....	370
G.5. Math > 1 wrap.....	370
G.6. Math > Imod .....	371

G.7. Math > Max / IMax .....	371
G.8. Math > Min / IMin .....	371
G.9. Math > round.....	371
G.10. Math > sign +- .....	371
G.11. Math > sqrt (>0).....	372
G.12. Math > sqrt .....	372
G.13. Math > x(>0)^y .....	372
G.14. Math > x^2 / x^3 / x^4 .....	372
G.15. Math > Chain Add / Chain Mult .....	372
G.16. Math > Trig-Hyp > 2 pi wrap.....	372
G.17. Math > Trig-Hyp > arcsin / arccos / arctan .....	373
G.18. Math > Trig-Hyp > sin / cos / tan .....	373
G.19. Math > Trig-Hyp > sin -pi..pi / cos -pi..pi / tan -pi..pi.....	373
G.20. Math > Trig-Hyp > tan -pi4..pi4 .....	373
G.21. Math > Trig-Hyp > sinh / cosh / tanh.....	373
G.22. Memory > Latch / ILatch.....	373
G.23. Memory > z^-1 / z^-1 ndc.....	374
G.24. Memory > Read [] .....	374
G.25. Memory > Write [] .....	374
G.27. Modulation > x * a / Integer > lx * a .....	375
G.28. Modulation > x - a / Integer > lx - a.....	375
G.29. Modulation > a - x / Integer > la - x.....	375
G.30. Modulation > x / a .....	375
G.31. Modulation > a / x .....	376
G.32. Modulation > xa + y.....	376

## **Anhang H. Standard macros.....377**

H.1. Audio Mix-Amp > Amount .....	377
H.2. Audio Mix-Amp > Amp Mod .....	377
H.3. Audio Mix-Amp > Audio Mix .....	377
H.4. Audio Mix-Amp > Audio Relay .....	377
H.5. Audio Mix-Amp > Chain (amount).....	378
H.6. Audio Mix-Amp > Chain (dB) .....	378
H.7. Audio Mix-Amp > Gain (dB) .....	378
H.8. Audio Mix-Amp > Invert.....	379
H.9. Audio Mix-Amp > Mixer 2 ... 4 .....	379
H.10. Audio Mix-Amp > Pan .....	379
H.11. Audio Mix-Amp > Ring-Amp Mod .....	379
H.13. Audio Mix-Amp > Stereo Mixer 2 ... 4 .....	380
H.14. Audio Mix-Amp > VCA.....	380
H.15. Audio Mix-Amp > XFade (lin) .....	381

H.16. Audio Mix-Amp > XFade (par) .....	381
H.17. Audio Shaper > 1+2+3 Shaper .....	381
H.18. Audio Shaper > 3-1-2 Shaper .....	382
H.19. Audio Shaper > Broken Par Sat.....	382
H.20. Audio Shaper > Hyperbol Sat .....	382
H.21. Audio Shaper > Parabol Sat.....	383
H.22. Audio Shaper > Sine Shaper 4 / 8.....	383
H.23. Control > Ctl Amount.....	383
H.24. Control > Ctl Amp Mod .....	383
H.25. Control > Ctl Bi2Uni .....	384
H.26. Control > Ctl Chain.....	384
H.27. Control > Ctl Invert.....	384
H.28. Control > Ctl Mix .....	384
H.29. Control > Ctl Mixer 2 .....	385
H.30. Control > Ctl Pan .....	385
H.31. Control > Ctl Relay .....	385
H.32. Control > Ctl XFade.....	385
H.33. Control > Par Ctl Shaper.....	386
H.34. Convert > dB2AF .....	386
H.35. Convert > dP2FF .....	386
H.36. Convert > logT2sec .....	387
H.37. Convert > ms2Hz .....	387
H.39. Convert > P2F .....	387
H.40. Convert > sec2Hz .....	387
H.41. Delay > 2 / 4 Tap Delay 4p.....	388
H.42. Delay > Delay 1p / 2p / 4p .....	388
H.43. Delay > Diff Delay 1p / 2p / 4p.....	388
H.44. Envelope > ADSR .....	389
H.45. Envelope > Env Follower .....	390
H.46. Envelope > Peak Detector .....	390
H.47. EQ > 6dB LP/HP EQ .....	390
H.48. EQ > 6dB LowShelf EQ .....	390
H.49. EQ > 6dB HighShelf EQ.....	391
H.50. EQ > Peak EQ .....	391
H.51. EQ > Static Filter > 1-pole static HP.....	391
H.52. EQ > Static Filter > 1-pole static HS .....	391
H.53. EQ > Static Filter > 1-pole static LP.....	392
H.54. EQ > Static Filter > 1-pole static LS.....	392
H.55. EQ > Static Filter > 2-pole static AP .....	392
H.56. EQ > Static Filter > 2-pole static BP .....	392
H.57. EQ > Static Filter > 2-pole static BP1 .....	392

H.58. EQ > Static Filter > 2-pole static HP .....	393
H.59. EQ > Static Filter > 2-pole static HS .....	393
H.60. EQ > Static Filter > 2-pole static LP.....	393
H.61. EQ > Static Filter > 2-pole static LS.....	393
H.63. EQ > Static Filter > 2-pole static Pk.....	394
H.64. EQ > Static Filter > Integrator .....	394
H.65. Event Processing > Accumulator .....	394
H.67. Event Processing > Clk Gen .....	395
H.68. Event Processing > Clk Rate.....	395
H.69. Event Processing > Counter.....	395
H.71. Event Processing > Dup Flt / IDup Flt .....	396
H.72. Event Processing > Impulse.....	396
H.73. Event Processing > Random .....	396
H.74. Event Processing > Separator / ISeparator .....	397
H.75. Event Processing > Thld Crossing.....	397
H.76. Event Processing > Value / IValue.....	397
H.77. LFO > MultiWave LFO .....	397
H.78. LFO > Par LFO .....	398
H.79. LFO > Random LFO.....	398
H.80. LFO > Rect LFO.....	398
H.81. LFO > Saw(down) LFO .....	398
H.82. LFO > Saw(up) LFO.....	399
H.84. LFO > Tri LFO .....	399
H.85. Logic > AND .....	399
H.86. Logic > Flip Flop.....	400
H.87. Logic > Gate2L .....	400
H.88. Logic > GT / IGT .....	400
H.89. Logic > EQ.....	400
H.90. Logic > GE.....	400
H.91. Logic > L2Clock.....	401
H.92. Logic > L2Gate.....	401
H.93. Logic > NOT .....	401
H.94. Logic > OR.....	401
H.95. Logic > XOR .....	401
H.96. Logic > Schmitt Trigger .....	402
H.97. Oscillators > 4-Wave Mst.....	402
H.98. Oscillators > 4-Wave Slv .....	402
H.99. Oscillators > Binary Noise .....	402
H.100. Oscillators > Digital Noise .....	403
H.101. Oscillators > FM Op .....	403
H.102. Oscillators > Formant Osc .....	403

H.103. Oscillators > MultiWave Osc.....	403
H.104. Oscillators > Par Osc .....	404
H.105. Oscillators > Quad Osc.....	404
H.106. Oscillators > Sin Osc .....	404
H.107. Oscillators > Sub Osc 4 .....	404
H.108. VCF > 2 Pole SV .....	405
H.109. VCF > 2 Pole SV C.....	405
H.110. VCF > 2 Pole SV (x3) S .....	405
H.111. VCF > 2 Pole SV T (S) .....	406
H.112. VCF > Diode Ladder.....	406
H.113. VCF > D/T Ladder.....	406
H.114. VCF > Ladder x3 .....	407
<b>Anhang I. Core cell library.....</b>	<b>408</b>
I.1. Audio Shaper > 3-1-2 Shaper.....	408
I.2. Audio Shaper > Broken Par Sat.....	408
I.3. Audio Shaper > Hyperbol Sat.....	408
I.4. Audio Shaper > Parabol Sat.....	409
I.5. Audio Shaper > Sine Shaper 4/8 .....	409
I.6. Control > ADSR .....	409
I.7. Control > Env Follower.....	410
I.8. Control > Flip Flop .....	410
I.9. Control > MultiWave LFO.....	410
I.10. Control > Par Ctl Shaper.....	411
I.11. Control > Schmitt Trigger.....	411
I.12. Control > Sine LFO .....	411
I.13. Delay > 2/4 Tap Delay 4p .....	412
I.14. Delay > Delay 4p .....	412
I.15. Delay > Diff Delay 4p.....	412
I.16. EQ > 6dB LP/HP EQ .....	413
I.17. EQ > HighShelf EQ.....	413
I.18. EQ > LowShelf EQ .....	413
I.19. EQ > Peak EQ .....	413
I.20. EQ > Static Filter > 1-pole static HP.....	414
I.21. EQ > Static Filter > 1-pole static HS.....	414
I.22. EQ > Static Filter > 1-pole static LP .....	414
I.23. EQ > Static Filter > 1-pole static LS.....	414
I.24. EQ > Static Filter > 2-pole static AP.....	415
I.25. EQ > Static Filter > 2-pole static BP.....	415
I.26. EQ > Static Filter > 2-pole static BP1.....	415
I.27. EQ > Static Filter > 2-pole static HP.....	415



I.28. EQ > Static Filter > 2-pole static HS .....	416
I.29. EQ > Static Filter > 2-pole static LP .....	416
I.30. EQ > Static Filter > 2-pole static LS .....	416
I.31. EQ > Static Filter > 2-pole static N.....	417
I.32. EQ > Static Filter > 2-pole static Pk .....	417
I.33. Oscillator > 4-Wave Mst.....	417
I.34. Oscillator > 4-Wave Slv .....	418
I.35. Oscillator > Digital Noise.....	418
I.36. Oscillator > FM Op .....	418
I.37. Oscillator > Formant Osc .....	419
I.38. Oscillator > Impulse .....	419
I.39. Oscillator > MultiWave Osc .....	419
I.40. Oscillator > Quad Osc .....	420
I.41. Oscillator > Sub Osc .....	420
I.42. VCF > 2 Pole SV C.....	420
I.43. VCF > 2 Pole SV T.....	421
I.44. VCF > 2 Pole SV x3 S.....	421
I.45. VCF > Diode Ladder .....	422
I.46. VCF > D/T Ladder .....	422
I.47. VCF > Ladder x3.....	423
<b>Index .....</b>	<b>424</b>



# Modul-Referenz

Module sind die elementarsten Komponenten eines REAKTOR-Ensembles. Dieser Teil des Handbuchs richtet sich vor allem an REAKTOR-Anwender, die Ihre Ensembles von Grund auf selbst konstruieren möchten. Wenn Sie dies nicht möchten oder meinen, dass Ihre Erfahrung für die Konstruktion eigener Ensembles nicht ausreichend ist, bietet REAKTOR auch andere Möglichkeiten, sein Potential auszuschöpfen:

- Falls Sie überhaupt nicht an der Konstruktion eigener Ensembles interessiert sind, arbeiten Sie auf Ensemble-Ebene.
- Falls Sie Ihre Lieblings-Instrumente in einem Ensemble zusammenstellen und diese gemeinsam verwenden möchten, arbeiten Sie auf Instrumenten-Ebene.
- Falls Sie Ihre eigenen Instrumente aus vorgefertigten Bausteinen zusammenbauen möchten, arbeiten Sie auf Macro-Ebene.
- Falls Sie die Kontrolle über jeden einzelnen Parameter eines Ensembles behalten möchten, arbeiten Sie auf Modul-Ebene.

Dieses Referenzhandbuch führt alle REAKTOR-Module auf und bietet eine einführende Beschreibung für jedes Modul. Die Beschreibung enthält Properties-Einstellungen, soweit diese für das Modul verfügbar sind, sowie eine Liste aller Ein- und Ausgangsports des Moduls.

Alle REAKTOR-Objekte (Module, Macros, Instrumente und Ensembles) besitzen ein Label-Feld in ihren Properties. Dieses Label erscheint auch in den Objekt-Icons innerhalb des Struktur-Fensters. In der Standardeinstellung beschreibt das Label auch die Funktion des Moduls. Ändern Sie den Modulnamen deshalb mit Umsicht, besonders wenn Sie möchten, dass andere REAKTOR-Anwender die Strukturen Ihrer Kreationen auch verstehen sollen.

## Hybrid-Module

Eine Reihe von Modulen in REAKTOR sind Hybrid-Module. Nach dem Einfügen eines solchen Moduls erscheint es zunächst als Event-verarbeitendes Modul, was mit einem roten Label angezeigt wird. Ein grüner Punkt auf einem Port zeigt an, dass man sowohl ein Audiokabel als auch ein Eventkabel mit dem Port verbinden kann. Sobald Sie ein Audiokabel an einen der Eingangs-Ports anschliessen, wird das Modul in ein Audio-verarbeitendes Modul umgewandelt, was durch schwarze Punkte und Labels auf den Ports angezeigt wird. Wenn Sie ein Eventkabel anschliessen, verwandelt sich der grüne Punkt auf einem Port in einen roten Punkt.

Ein Hybrid-Modul weist die folgenden Eigenheiten auf:

- Wenn Sie Audio- und Eventkabel an den Eingängen mischen, oder wenn Sie ausschliesslich Audiokabel verwenden, arbeitet das Modul mit Audiorate.
- Wenn Sie ausschliesslich Eventkabel an den Eingängen anschliessen, arbeitet das Modul mit Eventrate
- Wenn der Modulausgang mit einem Eventeingang eines anderen Moduls verbunden wird, wird das Modul automatisch ein Event-verarbeitendes Modul, und es ist nicht mehr möglich, Audiokabel an den Moduleingängen anzuschliessen.
- Wenn der Modulausgang mit einem Audioeingang eines anderen Moduls verbunden wird, kann das Modul entweder mit Audiorate oder mit Eventrate arbeiten, abhängig von den Kabeln, die an den Moduleingängen angeschlossen werden.
- Wenn das Modul wegen seiner Eingangsverbindungen schon mit Audiorate arbeitet, können Sie den Ausgang nicht mehr an den Eventeingang eines anderen Moduls anschliessen.

### **Dynamische Port-Verwaltung**

Wo es Sinn macht, verfügen Module über eine dynamische Verwaltung der Ein- und/oder Ausgangsports. Sie können Eingänge im Modul hinzufügen, indem Sie zusätzliche Kabel auf den Portbereich des Moduls ziehen, während Sie die Ctrl-Taste gedrückt halten. Sie können sehen, wo der Port hinzugefügt wird, wenn Sie den Mauszeiger über das Zielmodul halten. Die vertikale Position des Mauszeigers bestimmt, wo der Port hinzugefügt wird, so dass es möglich ist, neue Ports zwischen schon vorhandenen Ports zu erzeugen.

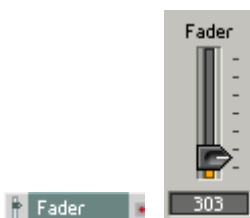
# Panel

Panel-Module liefern die Bedienelement zur Steuerung der Reaktor-Instrumente. Sie können in den zwei Panels A und B unabhängig voneinander angeordnet und sichtbar gemacht werden. Einige Panel-Module dienen nur zur Anzeige von Werten. Dazu gehören Lamps, Meters und Scopes, um Reaktor-Prozesse zu visualisieren, sowie Grafik- und Text-Module zur Verschönerung der Oberfläche. Die übrigen Module erzeugen oder routen Daten zur Siganlverarbeitung in Reaktor. Zu diesen Modulen gehören Fader, Knobs, Buttons, Switches, Menüs und ein XY-Kontrollfeld (welches sowohl Anzeige als auch Bedienelement sein kann).

---

## Fader

## Panel



Mit dem Schieberegler im Panel stellt man den Wert ein, den das Modul in der Structure als Event und Audio ausgibt. Das Signal ist monophon– bei Verbindung mit einem polyphonen Eingang erhalten alle Stimmen den gleichen Wert.

### Properties - Function-Seite

Wertebereich (Range): Das Ausgangssignal liegt entsprechend der Position des Knopfes im Bereich zwischen **Min** und **Max**. Mit dem **Step**-Parameter kann eine Schrittweite eingestellt werden, um die angezeigten und ausgegebenen Werte zu quantisieren.

Alternativ ist es auch möglich, das Num Steps-Feld zur Einstellung der Step-Auflösung des Faders zu verwenden. Die Werte beider Felder, Stepsize und Num Steps, sind abhängig voneinander. Wenn man also den Werte in einem der Felder verändert, verändert sich auch der andere. Während man den Wertebereich pro Step im Stepsize-Feld eingibt, repräsentiert das Num Steps-Feld die Anzahl von Schritten über den gesamten Wertebereich des Faders. Die höchstmögliche Auflösung eines Faders beträgt 127.000 Steps, welche man erhält, wenn man 0 im **Stepsize**-Feld bzw. 127.000 im **Num Step**-Feld einträgt.

---

Sie haben die Möglichkeit, im **Num Steps** eine Step-Auflösung zu definieren, welche über der MIDI-Standard-Auflösung von 128 liegt. Denken Sie daran, daß Reaktor zwar intern eine höhere Auflösung berechnen kann, mit externer Hard- oder Software aber Daten lediglich in der MIDI-Auflösung austauschen kann. Unter bestimmten Umständen kann es darum dazu kommen, daß Funktionalität oder Klang eines Ensembles innerhalb verschiedener Produktionsumgebungen differieren. Im Regelfall stellt dies kein Problem dar, da die MIDI-Auflösung zur Parameterkontrolle völlig ausreichend ist. In manchen Situationen allerdings (z.B. wenn Sie einen Filter mit hoher Resonanz verwenden und die Grenzfrequenz bewegen) wünschen Sie sich möglicherweise eine höhere Auflösung, welche Ihnen Reaktor dann auch ermöglicht.

---

Mit **Mouse Res** kann festgelegt werden, welche Distanz in Pixel der Mauszeiger zurückzulegen hat, um vom kleinsten Wert des Faders zum größten zu gelangen. Wenn Sie zum Beispiel einen Wert unter **Mouse Res** eingeben, der doppelt so hoch ist wie der im Pixel in **Y-Feld** auf der **Appearance**-Seite innerhalb der Properties, müssen Sie die Maus doppelt so weit ziehen wie der Fader lang ist, um vom kleinsten auf den größten Wert zu wechseln.

Wenn **Num Steps** größer als **Mouse Res** ist, kann nicht mehr jeder Schritt mit der Maus erreicht werden. Wenn andererseits **Mouse Res** größer ist als **Num Steps**, kann die Anzahl der Pixel pro Schritt auf einen höheren Wert als 1 gestellt werden.

Der Default-Wert wird immer dann verwendet, wenn eine Initialisierung des Reglers stattfindet. Der Wert wird in den folgenden Situationen benutzt:

- Betätigen des **Default**-Buttons im Snapshot-Fenster setzt alle Bedienelemente im Ensemble/Instrument zurück.
- Wenn "default" als eines des Morph-Ziele im Snapshot-Fenster erscheint, können Sie zwischen einen Snapshot und den Default-Einstellungen aller Bedienelemente morphen.
- Wenn Sie ein Bedienelement einem Instrument hinzufügen, welches schon eine Snapshotliste enthält, wird der Default-Wert des neuen Elements in allen Snapshots verwendet bis Sie die entsprechenden Snapshots mit einem neuen Wert überschreiben oder **Snap Isolate** aktivieren.

Wenn Snap Isolate aktiviert ist, reagiert der Fader nicht auf Snapshotaufrufe. Die ID-Nummer wird für das Snapshot-Management von REAKTOR verwendet. Die ID-Nummern werden immer automatisch eingetragen, wenn Sie ein Bedienelement einfügen.

Wenn Sie diese ID ändern (um beispielsweise Snapshots von einem Instrument

mit ähnlichen Bedienelementen zu importieren), werden schon existierende Snapshots des Panel-Element nicht mehr erkennen und es ignorieren. Verändern Sie die ID-Nummer also nur, wenn Sie genau wissen, was Sie tun.

### Properties - Info-Seite

Ein Infotext zur Erläuterung der Funktion des Faders kann unter **Info** eingegeben werden. Dieser Text erscheint als Schnellhilfe (Hint), so lange der Mauszeiger auf dem Fader im Panel ruht, sofern Show-Hints angeschaltet ist.

### Properties - Appearance-Seite

Das Label, welches Sie hier eingeben können, wird im Panel nur dann über dem Fader gezeigt, wenn in der **Visible**-Sektion **Label** aktiviert ist. Ebenso wird der eingestellte Wert nur dann als Zahl unter dem Fader gezeigt, wenn **Value** aktiviert ist. Es ist auch möglich, das Bitmap des Faders selbst zu verstecken, so dass nur noch das **Value**-Feld sichtbar ist. In diesem Fall können Sie den Fader aber trotzdem noch benutzen, indem Sie im Panel auf das Value-Feld klicken und die Maus nach oben oder unten ziehen. Fader können vertikal oder horizontal im Panel dargestellt werden. Kreuzen Sie das entsprechende Feld in der **Form**-Sektion an, um das gewünschte Aussehen zu erzielen.

Die Länge des Faders kann im Feld **Pixel in Y** in der **Size**-Sektion frei eingestellt werden. Sie können auch die Breite des Faders mit den Optionen **Big**, **Medium** und **Small** definieren.

### Properties - Connection-Seite

Die Connection-Seite der Properties für Bedienelemente wird im Abschnitt *Connection-Properties von Bedienelementen* in Kapitel 18.5. erläutert.



Wie Fader, aber als Drehregler im Panel dargestellt.



Mit einem Druck-Knopf im Panel wählt man den Wert aus, den das Modul in der Structure als Event und Audio ausgibt. Das Signal ist monophon– bei Verbindung mit einem polyphonen Eingang erhalten alle Stimmen den gleichen Wert.

### Properties - Function-Seite

- **Range:** Der Wert, der beim Drücken des Buttons gesendet wird, und der, der beim Loslassen gesendet wird, kann in den Feldern On Value und Off Value definiert werden.
- **Mode:** Es stehen die Betriebsarten **Trigger**, **Gate** und **Toggle** zur Auswahl. Im Trigger-Modus wird nur beim Hineindrücken ein Event gesendet. Im Gate-Modus wird beim Loslassen zusätzlich ein Event mit dem Wert Null gesendet. Im Toggle-Modus wird bei jedem Hineindrücken zwischen den beiden Zuständen gewechselt.
- **Default = On:** Setzt den Default-Wert, der bei verschiedenen Aktionen in REAKTOR verwendet wird, auf On.
- Die Aktivierung des **Snap Isolate**-Schalters verhindert, dass der Button auf Snapshot-Aufrufe reagiert.
- Die Aktivierung des **Random Isolate**-Schalters verhindert, dass der Button auf die Zufallsfunktion für Snapshots reagiert.

### Properties - Info-Seite

Ein Infotext zur Erläuterung der Funktion des Buttons kann unter **Info** eingegeben werden. Dieser Text erscheint als Schnellhilfe (Hint), so lange der Mauszeiger auf dem Button im Panel ruht, sofern Show-Hints angeschaltet ist.



## Properties - Appearance-Seite

Das Label wird im Panel nur dann über dem Button gezeigt, wenn im Properties-Dialog **Label** aktiviert ist. Ebenso wird der eingestellte Wert nur dann als Zahl unter dem Button gezeigt, wenn in den Properties **Value** aktiviert ist.

Die Größe des Buttons kann in drei Stufen eingestellt werden: **Small**, **Medium** und **Big**.

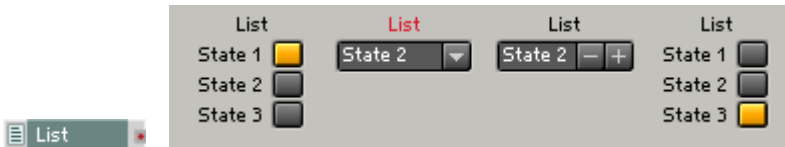
## Properties - Connection-Seite

Die Connection-Seite der Properties für Bedienelemente wird im Abschnitt *Connection-Properties von Bedienelementen* in Kapitel 18.5. erläutert.

---

## List

## Panel



Das List-Modul dient zur Konstruktion von Panel-Listen, Dropdown-Menüs, Button-Menüs und Textanzeige mit Scrolling-Funktion.

## Properties - Function-Seite

Die **Function-Seite** der Properties enthält eine Liste, in welcher Sie einzelne Einträge anfügen (**Append**), einfügen (**Insert**) und löschen (**Delete**) können. Es ist auch möglich, die Zahl der Einträge mit einer numerischen Eingabe festzulegen. Die Einträge in der Liste werden im Panel mit dem ausgewählten Anzeige-Style dargestellt. Für jeden Eintrag können Sie einen Wert eingeben, der am Modulausgang gesendet wird, wenn der Eintrag ausgewählt wird.

Die Function-Seite enthält auch einen Value Generator. Mit diesem kleinen Tool können Sie Werte für mehrere Listen-Einträge gleichzeitig generieren. Beispiel: Sie möchten zwischen den Einträgen eine Schrittweite von 4 anstatt von 1 haben. Dazu geben Sie im Value Generator „4“ unter Stepsize ein und betätigen den Apply-Button. Nun werden die Werte in der Value-Spalte der Liste entsprechend den Einstellungen im Value Generator angepasst.

Die **Mouse Resolution**-Einstellung ist nur dann von Bedeutung, wenn für das Bedienelement der Spin-Style auf der **Appearance-Seite** ausgewählt ist, wo man auf den Eintrag im Bedienelement klicken und die Maus nach oben oder unten ziehen kann, um den Eintrag zu ändern.

## Properties - Appearance-Seite

Die Panel-Darstellung des Bedienelements für dieses Modul ändert sich abhängig vom auf der **Appearance-Seite** in den Properties ausgewählten Style. Die folgenden Styles sind verfügbar:

- **Button:** Jeder Eingangsport des Moduls erzeugt einen Button. Alle Buttons werden vertikal als Buttonleiste im Instrumenten-Panel aneinandergehängt. Der gerade ausgewählte Eingang erscheint in der Indicator-Farbe des Instruments.
- **Menu:** Jeder Eingangsport des Moduls erzeugt einen neuen Eintrag in einer Dropdown-Liste.
- **Text Panel:** Jeder Eingangsport des Moduls erzeugt einen neuen Eintrag in einer Liste, welche mehrere Einträge gleichzeitig darstellt. Wenn Sie mehr Einträge erzeugen als in das Panel-Display passen, dessen Größe Sie mit den Feldern Size X und Size Y auf der Appearance-Seite der Properties definieren, erhält das Panel-Display Scrollbalken.
- **Spin:** Jeder Eingangsport des Moduls erzeugt einen neuen Eintrag in einer Liste. Sie können mit den + und - Buttons rechts vom Panel-Display durch die Listeneinträge navigieren.

Die **Size X**- und **Size Y**-Felder regeln die Display-Grösse des Bedienelements im Panel.

## Ports

- **Out:** Eventausgang für den Wert, der mit dem ausgewählten Eintrag verbunden ist.

---

## Switch

## Panel



Der Switch dient zum Umschalten von Signalwegen. Er enthält keine eigene Signalverarbeitung, sondern stellt nur eine Verbindung zwischen anderen

Modulen her. Der Ausgang wird durch Drücken einer Taste oder durch Auswahl eines Listeneintrags mit dem dadurch ausgewählten Eingang verbunden. Alle übrigen Eingangsports sind abgeschaltet.

Module, deren Ausgänge nicht verbunden sind, werden automatisch deaktiviert, um unnötige Belastung des Prozessors zu vermeiden. Die Statuslampe bleibt bei deaktivierten Modulen dunkel.

Das Modul ist hybrid, kann also Audio und Eventsignale je nach vorgenommener Verkabelung gleichermaßen verarbeiten, und verfügt über eine dynamische Eingangsverwaltung.

### Properties - Function page

**Enable Switch Off:** Wenn diese Option aktiviert ist, kann der Switch einen Zustand einnehmen, bei dem kein Eingang ausgewählt ist. Wenn Button-Style für das Bedienelement auf der Appearance-Seite ausgewählt ist, ist es möglich, ein zweites Mal auf den gerade selektierten Schalter zu klicken, um alle Eingänge auszuschalten. Bei Verwendung aller anderen Styles erhalten Sie den zusätzlichen Eintrag „Off“.

Das Modul besitzt eine dynamische Eingangsport-Verwaltung. Die Anzahl der Eingänge kann mit **Min Num Port Groups** auf der Function-Seite der Properties definiert werden.

Die **Mouse Resolution**-Einstellung ist nur dann von Bedeutung, wenn für das Bedienelement der Spin-Style auf der Appearance-Seite ausgewählt ist, wo man auf den Eintrag im Bedienelement klicken und die Maus nach oben oder unten ziehen kann, um den Eintrag zu ändern.

### Properties - Appearance-Seite

Das Label wird im Panel nur dann über dem Switch gezeigt, wenn in den Properties **Label** aktiviert ist.

Das Bedienelement kann in drei verschiedenen Größen dargestellt werden: **Small**, **Medium** und **Big**.

Falls Sie einen Switch mit nur zwei Eingängen verwenden, wird im Button-Style nur ein Button im Panel angezeigt (der obere), wenn die Option **1 Toggle Button** aktiviert ist: Wenn der Button angeschaltet ist, ist Eingang 1 ausgewählt, wenn er ausgeschaltet ist, Eingang 2.

Die Panel-Darstellung des Bedienelements für dieses Modul ändert sich abhängig vom auf der **Appearance**-Seite in den Properties ausgewählten Style. Die folgenden Styles sind verfügbar:

- **Button:** Jeder Eingangsport des Moduls erzeugt einen Button. Alle Buttons werden vertikal als Buttonleiste im Instrumenten-Panel aneinandergehängt. Der gerade ausgewählte Eingang erscheint in der Indicator-Farbe des Instruments.
- **Menu:** Jeder Eingangsport des Moduls erzeugt einen neuen Eintrag in einer Dropdown-Liste.
- **Text Panel:** Jeder Eingangsport des Moduls erzeugt einen neuen Eintrag in einer Liste, welche mehrere Einträge gleichzeitig darstellt. Wenn Sie mehr Einträge erzeugen als in das Panel-Display passen, dessen Größe Sie mit den Feldern **Size X** und **Size Y** auf der Appearance-Seite der Properties definieren, erhält das Panel-Display Scrollbalken.
- **Spin:** Jeder Eingangsport des Moduls erzeugt einen neuen Eintrag in einer Liste. Sie können mit den + und - Buttons rechts vom Panel-Display durch die Listeneinträge navigieren.

## Ports

- **Out:** Eventausgang für den mit dem ausgewählten Eintrag assoziierten Wert.



Die Lampe im Panel leuchtet, solange das (mit 25 Hz abgetastete) Eingangssignal in dem Bereich ist, der im Properties-Dialog mit **Min** und **Max** eingestellt ist, d.h. die Lampe ist an, wenn der Wert größer als **Min** und kleiner oder gleich **Max** ist.

Wenn der **Continuous-Modus** in den Properties aktiviert ist, wird die Lampen-Farbe zwischen den **Min-** und **Max-Werten** ein- und ausgeblendet.

Die Farbe der Lampe (rot, grün, blau, gelb oder Indicator-Farbe) lässt sich in den Properties auswählen. Wenn Sie die Option **Indicator Color** wählen, verwendet die Lampe die Indicator-Farbe, die in den Instrument-Properties definiert wurde.

Es ist auch möglich, eigene Farben für die An- und Aus-Position der Lampe mit Hilfe der Buttons **Set On Color** und **Set Off Color** zu definieren.

Wenn Sie die Option **Has Frame** nicht aktivieren, können Sie Lampen im Panel pixelgenau lückenlos nebeneinander anordnen.

Das Label wird im Panel nur dann über der Lampe gezeigt, wenn in den Properties **Label** aktiviert ist.



Die Lampe im Panel leuchtet auf, sobald der Pegel am Eingang innerhalb des Bereichs ist, der in den Properties mit **Min** und **Max** (in dB) eingestellt ist, d.h. die Lampe ist an, wenn der Pegel größer als **Min** und kleiner oder gleich **Max** ist.

Wenn der Continuous-Modus in den Properties aktiviert ist, wird die Lampen-Farbe zwischen den Min- und Max-Werten ein- und ausgeblendet.

Die Farbe der Lampe (rot, grün, blau, gelb oder Indicator-Farbe) lässt sich in den Properties auswählen. Wenn Sie die Option **Indicator Color** wählen, verwendet die Lampe die Indicator-Farbe, die in den Instrument-Properties definiert wurde.

Es ist auch möglich, eigene Farben für die An- und Aus-Position der Lampe mit Hilfe der Buttons **Set On Color** und **Set Off Color** zu definieren.

Wenn Sie die Option **Has Frame** nicht aktivieren, können Sie Lampen im Panel pixelgenau lückenlos nebeneinander anordnen.

Das Label wird im Panel nur dann über der Lampe gezeigt, wenn in den Properties **Label** aktiviert ist.

---

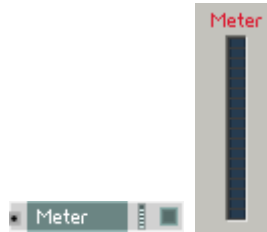
## RGB Lamp

## Panel



Dieses Modul erzeugt eine in der Größe einstellbare Farbanzeige, welche aus den drei Farben, welche an den entsprechenden Moduleingängen anliegen, gemischt wird. Die horizontale und vertikale Größe kann in Pixel auf der Appearance-Seite der Properties eingestellt werden.

- **R:** Audioeingang für die Intensität der roten Farbkomponente.  
Range: 0 - 1.
- **G:** Audioeingang für die Intensität der grünen Farbkomponente.  
Range: 0 - 1.
- **B:** Audioeingang für die Intensität der blauen Farbkomponente.  
Range: 0 - 1.



Das anliegende Signal wird (mit 25 Hz) abgetastet und auf einer linearen Skala dargestellt. Der angezeigte Bereich wird in den Properties mit **Min** und **Max** eingestellt.

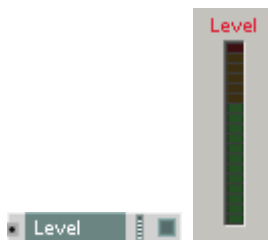
Die Farbe des Meters (rot, grün, blau, gelb oder Indicator-Farbe) lässt sich in den Properties auswählen. Wenn Sie die Option **Indicator Color** wählen, verwendet die Lampe die Indicator-Farbe, die in den Instrument-Properties definiert wurde.

Es ist auch möglich, eigene Farben für den oberen und unteren Teil des Meters mit Hilfe der Buttons **Set On Color** und **Set Off Color** zu definieren.

**Number Of Segments** legt fest, aus wie vielen einzelnen Elementen das Meter zusammengesetzt ist.

Unter **Size X (Segment)** und **Size Y (Segment)** können Sie die Größe eines Elements des Meters definieren. Die Gesamthöhe des Meters ergibt sich aus **Size Y (Segment)** multipliziert mit **Number of Segments**.

Das Label wird im Panel nur dann über dem Meter gezeigt, wenn in den Properties **Label** aktiviert ist.



Die Amplitude des anliegenden Audiosignals wird auf einer logarithmischen Skala dargestellt. Der angezeigte Bereich wird in den Properties mit **Min** und **Max** in dB eingestellt.

Die Farbe des Meters (rot, grün, blau, gelb oder Indicator-Farbe) lässt sich in den Properties auswählen. Wenn Sie die Option **Indicator Color** wählen, verwendet die Lampe die Indicator-Farbe, die in den Instrument-Properties definiert wurde.

Es ist auch möglich, eigene Farben für den oberen und unteren Teil des Meters mit Hilfe der Buttons **Set On Color** und **Set Off Color** zu definieren.

**Number Of Segments** legt fest, aus wie vielen einzelnen Elementen das Meter zusammengesetzt ist.

Unter **Size X (Segment)** und **Size Y (Segment)** können Sie die Größe eines Elements des Meters definieren. Die Gesamthöhe des Meters ergibt sich aus **Size Y (Segment)** multipliziert mit **Number of Segments**.

Das Label wird im Panel nur dann über dem Level Meter gezeigt, wenn in den Properties **Label** aktiviert ist.



### Picture

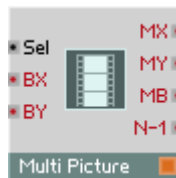
Erlaubt den Import eines Dekorations-Bitmaps für das Panel im TGA-Format (Dateierweiterung \*.tga). Das Bitmap-Modul hat keine Ein- und Ausgänge. Die Anzeigegröße wird auf die Originalgröße der geladenen Bitmap gesetzt. Es ist möglich, den sichtbaren Rahmen in Pixel in den Properties zu skalieren. Das Bild selbst kann nicht innerhalb von Reaktor skaliert werden; dafür benötigen Sie eine zusätzliche Bildbearbeitungs-Software.

Aktivieren Sie die Option **Save bitmap with ensemble** in den Properties, um die Bilddaten als Teil eines Reaktor-Ensembles zu speichern.

---

## Multi Picture

## Panel



Das Multi Picture-Modul ist ein 2-dimensionales Bedienelement (ähnlich wie das XY-Modul), welches die Mausposition und den Status der Maustaste an den Modulausgängen ausgibt. Das Modul unterstützt die Animation eines Filmstreifens, wenn dieser in einem einzelnen Bild vorliegt.

Die Animation kann im Picture Properties-Fenster vorgenommen werden, indem die Anzahl der Frames (Animations) und deren Richtung (horizontal oder vertikal) angegeben werden.

24-Bit BMP und 32-Bit Targa (unkomprimiert) formatierte Bildformate können in REAKTOR an verschiedenen Stellen eingesetzt werden: als Hintergrund für Instrumenten- und Macropanel, für die Structure-Icons für Instrumente und Macros, und in den Picture- und Multi-Picture-Modulen. Der Vorteil des Targa-Formats besteht in der Unterstützung eines Alpha-Kanals, welcher als Maske für die sichtbaren Bereiche des Bildes verwendet werden kann (wobei unmaskierte Bereiche dann transparent erscheinen). Dies ist beispielsweise nützlich, um runde Drehregler auf einem rechteckigen Hintergrund zu erzeugen.

Sie können Bilder mit Hilfe des Select Picture-Dropdown-Menüs in den Properties des betreffenden Objekts laden. Beim Laden eines Bildes wird automatisch das Picture Properties-Fenster geöffnet, wo Sie alle relevanten Bildeinstellungen vornehmen können. Alle Bilder werden automatisch im Arbeitsspeicher geteilt und stehen allen anderen entsprechenden Objekten zur Verfügung.

- **Sel:** Audioeingang für die Auswahl des Bildausschnitts mit einer Nummer. Range 0 - Nummer des letzten Frames.
- **MX:** Eventausgang für die horizontale Mausposition (X), wenn die Maus sich innerhalb des sichtbaren Bildausschnitts im Panel befindet.
- **MY:** Eventausgang für die vertikale Mausposition (Y), wenn die Maus sich innerhalb des sichtbaren Bildausschnitts im Panel befindet
- **MB:** Eventausgang für den Status der Maustaste (0 = nicht gedrückt, 1 = gedrückt)
- **N-1:** Eventausgang für die Anzahl der in den Picture Properties definierten Bildausschnitte (Num animations) -1.

---

## Text

## Panel


 A rectangular button with a light gray background and the word "Text" in a dark gray sans-serif font.

Das Text-Modul verarbeitet kein Signal, es dient nur dazu, erläuternde Texte in die Structure einzufügen. So kann man hier z. B. Autor und Erstellungsdatum eines Instruments vermerken und die Funktionsweise erläutern.

---

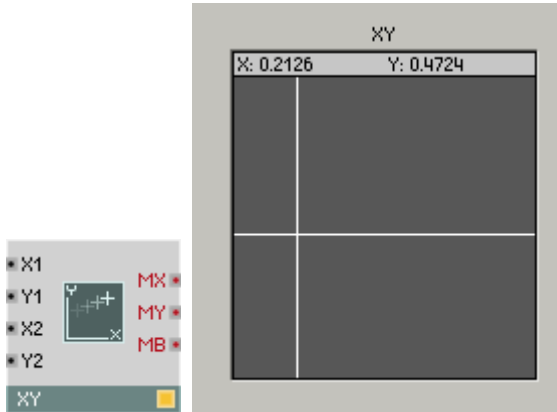
## Multi Text

## Panel


 A rectangular button with a light gray background. It contains the text "Multi Text" in a dark gray sans-serif font, followed by a small square icon with a white document symbol on a dark background.

Das Multi Text-Modul ermöglicht eine änderbare Textanzeige im Bedienpanel. Eine beliebige Anzahl von Texten kann in den Modul-Properties hinzugefügt werden. Dort können die Texte auch editiert oder gelöscht werden.

**In:** Audioeingang für die Nummer des anzuzeigenden Textes.



Das XY-Kontrollfeld hat zwei Funktionen: Es zeigt Audioeingangssignale an und fungiert als 2-dimensionales Kontrollelement für Mausoperationen.

### Properties - Function-Seite

Die Option **Always Active** schaltet die Fähigkeit des Moduls ein, einen Signal-Weg, welcher an einen der Eingangsports des Moduls angeschlossen ist, zu aktivieren.

Im **Incremental Mouse Mode** reagiert das Bedienelement ähnlich wie ein Drehregler. In diesem Modus können Sie irgendwo innerhalb des Panel-Displays des Moduls klicken und die Maus bewegen, ohne den Wert direkt auf die Position des Mauszeigers zu setzen. Stattdessen folgt der Wert dem Mauszeiger mit einem festen Versatz.

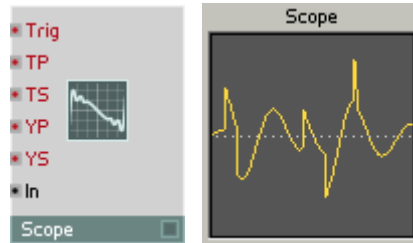
### Properties - Appearance-Seite

In den Modul-Properties können Sie den Anzeigetyp für die Visualisierung des Audiosignals einstellen. Die Eingänge **X1** und **Y1** steuern die Position des sichtbaren Objekts (**Pixel** oder **Cross**). Wenn als Objekt **Bar** oder **Rectangle** ausgewählt ist, dann steuern **X1** und **Y1** die eine Ecke und **X2** und **Y2** die gegenüberliegende Ecke des sichtbaren Objekts. Um sich schnell verändernde Audiodaten anzuzeigen, wählen Sie den **Scope**-Modus. Alle anderen Modi greifen den Eingang nur in einer niedrigeren Abtastrate ab.

Sie können auch die Größe des Fadenkreuzes einstellen, welches an der Mausposition erscheint.

Alles, was in das Anzeigefeld gezeichnet wird, verblasst mehr oder weniger schnell, abhängig von der Einstellung für **Fade Time** in den Properties (der Maximalwert beträgt 99).

- **X1**: Audioeingang für die **X1** Koordinate des sichtbaren Objekts.
- **Y1**: Audioeingang für die **Y1** Koordinate des sichtbaren Objekts.
- **X2**: Audioeingang für die **X2** Koordinate des sichtbaren Objekts.
- **Y2**: Audioeingang für die **Y2** Koordinate des sichtbaren Objekts.
- **MX**: Eventausgang für die X-Position des Mauszeigers, wenn die Maustaste innerhalb der Anzeige gedrückt ist.
- **MY**: Eventausgang für die Y-Position des Mauszeigers, wenn die Maustaste innerhalb der Anzeige gedrückt ist.
- **MB**: Status der linken Maustaste (1=gedrückt, 0=nicht gedrückt).



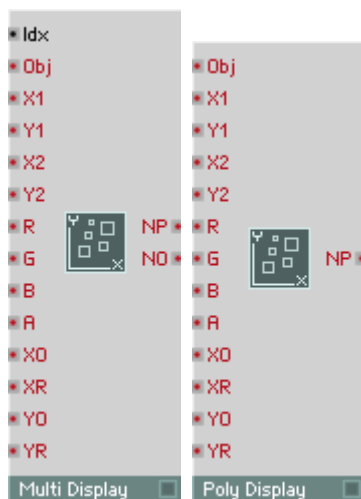
Oszilloskop zur Darstellung sich schnell ändernder Signale, besonders Audio. Jedes Mal wenn am Trigger-Eingang (**Trg**) ein Event empfangen wird, beginnt das Modul das Audiosignal am Eingang (**In**) aufzunehmen und stellt es als Kurve im Panel dar. Wenn keine Trigger-Events kommen, wird das gespeicherte Signal weiterhin angezeigt und kann in verschiedenen Vergrößerungen und Positionen dargestellt werden, indem man die Werte der entsprechenden Steuereingänge verändert.

Die Größe der Darstellung kann auf der Appearance-Seite der Properties mit **Pixel in X** und **Pixel in Y** eingestellt werden.

Auf der Function-Seite können Sie den Buffer, der für die Scope-Darstellung verwendet wird, in ms einzustellen.

Üblicherweise verbindet man eine **A to E Trig**-Modul mit dem **Trg**-Eingang, um das Oszilloskop mit einem Audiosignal zu synchronisieren.

- **Trg**: Mono Event-Eingang für das Trigger-Signal, das die Darstellung auslöst.
- **TP**: Mono Event-Eingang für die Zeit-Verschiebung (Time Position) der dargestellten Kurve in Millisekunden. Die Kurve fängt am linken Rand der Anzeige um **TP** ms nach dem Trigger-Event an.
- **TS**: Mono Event- Eingang für die Zeit-Skala (Time Scale) der dargestellten Kurve. Vom linken bis zum rechten Rand der Darstellung werden **TP** ms des Signals gezeigt.
- **YP**: Mono Event- Eingang für eine Amplituden-Verschiebung (Y Position) der Darstellung. **YP** = -1 entspricht der Unterkante der Anzeige, +1 der Oberkante.
- **YS**: Mono Event- Eingang für die Amplituden-Skalierung (Y Scale) der dargestellten Kurve. Die Differenz zwischen dem Signal an der Oberkante und an der Unterkante der Anzeige beträgt 2 **YS**. Wenn **YP** = 0, werden Werte zwischen +**YS** und -**YS** angezeigt.
- **In**: Mono Audio-Eingang für das darzustellende Signal.



Die Module Multi Display und Poly Display ermöglichen es, eine Vielzahl von grafische Objekten (Kreuze, Balken, Bilder, Animationen, etc.) anzuzeigen und zu verändern. Sie können für jedes grafische Objekt eine Reihe von Parametern (z. B. Art, Position, Größe und Farbe) individuell festlegen.

Der Hauptunterschied zwischen den beiden Modulen Multi Display und Poly Display ist, dass Sie die Anzahl der grafischen Objekte im Fall des Moduls Multi Display im Feld Number of Objects im Dialog Properties festlegen, während die Anzahl der grafischen Objekte im Fall des Moduls Poly Display von der Stimmenanzahl des Instruments abhängt.

Der Vorteil des Moduls Multi Display ist, dass es eine beliebige Anzahl von grafischen Objekten anzeigen kann, unabhängig von der Anzahl polyphoner Stimmen. Jedes Objekt kann dann unabhängig von den anderen Objekten über den Eingang Idx adressiert werden. Das Modul Poly Display ist dagegen einfacher zu programmieren, weil es keine Index-Verarbeitung erfordert und Sie aufgrund der parallelen polyphonen Verarbeitung alle Objekte gleichzeitig ansprechen können. Allerdings begrenzt die Stimmenanzahl des Instruments immer die Anzahl der Objekte. Der Dialog Properties enthält eine Reihe von Optionen, mit denen Sie die Darstellung anpassen können, darunter Wahlmöglichkeiten für Hintergrundfarbe und -bild.

Wenn Sie die Module Multi Display und Poly Display in Verbindung mit den Modulen Mouse Area und Snap Value Array verwenden, können Sie hoch entwickelte Elemente für die Bedienoberfläche konstruieren (z. B. Sequencer).

Sämtliche Eingänge des Moduls Multi Display nehmen monophone Event-Signale entgegen. Der Eingang Idx akzeptiert auch monophone Audio-Signale.

- **Idx:** Index des zu adressierenden grafischen Objekts. Der Index ist 1-basiert, d. h., die ID des ersten Elements ist 1 (nicht 0). Nicht ganzzahlige Werte werden auf die nächste Ganzzahl (Integer) gerundet. Wie der Eingang Idx außerhalb des Bereichs zwischen 1 und N liegende Werte behandelt, hängt von der Einstellung der Option Index Behaviour im Dialog Properties ab. Die Reihenfolge, in der die grafischen Objekte geschichtet werden, hängt von ihren über den Eingang Idx empfangenen Werten ab. Objekte mit niedrigeren Idx-Werten erscheinen auf Objekten mit höheren Idx-Werten. Stellen Sie den Idx-Wert immer ein, bevor Sie Events an andere Ports übermitteln lassen.
- **Obj:** Auswahlmöglichkeit für grafische Objekte oder Bild-Frames.
  - 4: Kreuz
  - 3: Linie von X1, Y1 nach X1, Y1 des nächsten Objekts derselben Art
  - 2: Linie von X1, Y1 nach X2, Y2
  - 1: Balken
  - 0: Rechteck
  - 1 ...NP: Bild-Index, der ein einzelnes Bild (1) oder eine Serie von Bildern in einer Animation (1, 2, 3, ...,NP) adressiert, wobei NP die Gesamtanzahl der Bilder in der Animation ist.

Nicht ganzzahlige Werte werden auf die nächste Ganzzahl (Integer) gerundet.

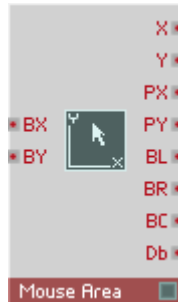
Wenn Sie die Option Ignore Index Obj (im Dialog Properties) einschalten, werden alle grafischen Objekte auf denselben Typ eingestellt.

- **X1, Y1:** Koordinaten für die erste Ecke der Fläche des grafischen Objekts; wenn Sie die Option Center to X1, Y1 im Dialog Properties einschalten, bestimmen diese Koordinaten die Mitte des grafischen Objekts.

- **X2, Y2:** Koordinaten für die erste Ecke der Fläche des grafischen Objekts.
- **R, G, B:** Diese Eingänge bestimmen die Anteile der Komponenten Rot, Grün und Blau an der Objekt-Farbe (Bereich: 0 bis 1). Wenn Sie die Option Ignore Index RGB (im Dialog Properties) einschalten, haben alle grafischen Objekte dieselbe Farbe.
- **A:** Objekt-Transparenz (0 = vollständig transparent, 1 = vollständig deckend).
- **X0, Y0:** Die niedrigsten sichtbaren X- und Y-Werte (für den Bildlauf). Eingaben an diesen Ports überschreiben die Einträge in den Feldern X Origin und Y Origin im Dialog Properties.
- **XR, YR:** Horizontaler/vertikaler Darstellungsbereich (für die Zoom-Funktion). Eingaben an diesen Ports überschreiben die Einträge in den Feldern X Range und Y Range im Dialog Properties.
- **NP:** Ausgang, an dem die Anzahl der Einzelbilder in der Animation anliegt.
- **NO:** Ausgang, an dem die Anzahl der grafischen Objekte anliegt.

Die Eingänge und Ausgänge des Moduls Poly Display entsprechen denen des Moduls Multi Display, abgesehen davon, dass dem Modul Poly Display der Eingang Idx und der Ausgang NO fehlt. Die Eingänge des Moduls Poly Display nehmen polyphone Event-Signale entgegen, eine Ausnahme bilden die Eingänge X0, XR, Y0 und YR, die monophone Event-Signale akzeptieren.





Das Modul Mouse Area bemerkt und übermittelt Maus-Aktionen, darunter Tastenklicks, Ziehen und Änderungen der Position. Das Modul Mouse Area kann eine eigene Umrandung und Füllfarbe besitzen und deshalb selbst als Panel-Bedienelement zum Einsatz kommen. Dennoch liegt es meistens als unsichtbares (transparentes) Overlay über anderen Modulen, beispielsweise Multi Display und Poly Display. Mit derartigen Kombinationen lassen sich sehr eigenständige Bedienelemente erschaffen.

Die Ausgänge Mouse Area X und Y arbeiten entweder im absoluten oder im inkrementellen Modus (wie im Dialog Properties eingestellt). Wenn Sie die Option Incremental Mode wählen, werden die X- und Y-Positionen durch mehrfaches Ziehen der Maus (mit gedrückter Taste) inkrementell angepasst, genau wie es auch im Fall der eingebauten Drehregler und Fader in REAKTOR geschieht. Konkret bedeutet dies, dass die Ausgänge X und Y bei jedem neu angesetzten Ziehen der Maus Werte übertragen, die eine relative Positionsänderung gegenüber dem Endpunkt des letzten Zieh-Vorgangs enthalten (und nicht die absolute Position des Mauszeigers). Die Eingänge BX und BY sind nur im inkrementellen Modus wirksam und dienen dazu, die inkrementelle "Basis" für aufeinander folgende Zieh-Aktionen (die das jeweils letzte Ziehen der Maus überschreiben) festzulegen. Typischerweise sind diese Eingänge an Module des Typs Snap Value angeschlossen, um sicherzustellen, dass die inkrementelle Basis der letzten in einem Snapshot gespeicherten Mausbewegung entsprechend eingestellt wird.

Im Dialog Properties können Sie mit der Option Outline Style das Aussehen der Umrandung der Maus-sensitiven Fläche (Mouse Area) einstellen. Wenn Sie hier die Option Rectangle wählen, wird ein 1 Pixel breiter Rand um die Fläche gezogen. Wenn Sie Bar wählen, wird die gesamte Maus-sensitive Fläche farbig ausgefüllt.

Mit der Option Active State stellen Sie die Aktion ein, die eine Änderung des Zustands der Umrandung von inaktiv in aktiv bewirkt. Hier können Sie zwischen Selection (der aktive Zustand tritt ein, wenn die Mouse Area ausgewählt wird), und den Schalt-Optionen Left/Right/Center wählen (der aktive Zustand trifft ein, sobald die relevante Maustaste gedrückt wird). Die Wahlmöglichkeit Outline Color im Dialog Properties erlaubt es, Farb- und Transparenz-Einstellungen für den aktiven und den inaktiven Zustand getrennt festzulegen. Ebenfalls im Dialog Properties können Sie Werte für den X- und X-Offset (Versatz) einstellen und damit die Position der Mouse Area im Instrumenten-Panel gegen REAKTORs 4x4-Raster versetzen.

- **BX:** Dieser Eingang setzt den Basiswert für inkrementelle Änderungen am Ausgang X. Die erlaubten Werte definieren Sie im Wertefeld Range X im Dialog Properties.
- **BY:** Dieser Eingang setzt den Basiswert für inkrementelle Änderungen am Ausgang Y. Die erlaubten Werte definieren Sie im Wertefeld Range Y im Dialog Properties.
- Beachten Sie, dass die Eingänge BX und BY nur dann die Ausgänge X und Y beeinflussen, wenn Sie im Dialog Properties den Modus Incremental Mode gewählt haben.
- **X:** Dieser Ausgang stellt die horizontale Maus-Position bereit, und zwar auf den im Wertefeld Range X (im Dialog Properties) eingestellten Bereich skaliert und begrenzt. Der Ausgang X sendet nur Maus-Positionen, die innerhalb der Mouse Area (die Sie in den Feldern Size X und Size Y im Dialog Properties definiert haben) liegen.
- **Y:** Dieser Ausgang stellt die vertikale Maus-Position bereit, und zwar auf den im Wertefeld Range Y (im Dialog Properties) eingestellten Bereich skaliert und begrenzt. Der Ausgang X sendet nur Maus-Positionen, die innerhalb der Mouse Area (die Sie in den Feldern Size X und Size Y im Dialog Properties definiert haben) liegen.
- **PX: Horizontale Maus-Position in Pixel relativ zur X-Ursprungslinie** (linker Rand der Mouse Area). Wenn Sie die Maus links der X-Ursprungslinie bewegen, werden negative Werte ausgegeben, wenn Sie die Maus rechts der Linie bewegen, werden positive Werte ausgegeben. Im Gegensatz zum Ausgang X, der auf Maus-Bewegungen innerhalb der Mouse Area beschränkt ist, meldet PX Maus-Positionen auf der gesamten Fläche zwischen dem linken und dem rechten Rand des Bildschirms.

- **PY: Horizontale Maus-Position in Pixel relativ zur Y-Ursprungslinie** (linker Rand der Mouse Area). Wenn Sie die Maus oberhalb der Y-Ursprungslinie bewegen, werden negative Werte ausgegeben, wenn Sie die Maus unterhalb der Linie bewegen, werden positive Werte ausgegeben. Im Gegensatz zum Ausgang Y der auf Maus-Bewegungen innerhalb der Mouse Area beschränkt ist, meldet PY Maus-Positionen auf der gesamten Fläche zwischen dem linken und dem rechten Rand des Bildschirms.
- **BL:** Zustand der linken (bzw. einzigen) Maustaste: 1, falls gedrückt, sonst 0
- **BR:** Zustand der rechten Maustaste: 1, falls gedrückt, sonst 0
- **BC:** Zustand der mittleren Maustaste: 1, falls gedrückt, sonst 0
- **Db:** Gibt ein einzelnes Event mit dem Wert 1 aus, wenn ein Doppelklick ausgeführt wird. Der Wert Db bleibt nach einem Doppelklick auf dem Wert 1 stehen: Sie müssen also Events am Ausgang Db abfragen, keine statischen Werte.



Stacked Macro ermöglicht es, dass sich mehrere grafische Objekte denselben Bereich des Instrumenten-Panels teilen. Welche Objekte zu einem Zeitpunkt innerhalb des gemeinsamen Bereichs angezeigt werden, können Sie mit dem Modul Panel Index steuern.

Nachdem Sie ein *Stacked Macro* in Ihrer Struktur eingesetzt haben, platzieren Sie es an einer geeigneten Stelle im Panel. Stellen Sie die gewünschte Größe im Dialog **Properties** ein. Setzen Sie dann zwei oder mehr Macros („normale“ Macros, keine weiteren *Stacked Macros*) in das Innere des *Stacked Macro*, zusammen mit einem Modul des Typs *Panel Index*. Nur eins der „normalen“ Macros (und mit ihm beliebige Bedienelemente, die zu ihm gehören) ist zu einem gegebenen Zeitpunkt sichtbar. Welches Macro sichtbar ist, hängt vom Eingangswert am Modul *Panel Index* ab. (Um die Index-Nummer eines Macros zu ermitteln, führen Sie unter Windows XP einen Rechts-Klick, unter Mac OS X einen Ctrl-Klick die Panel-Ansicht dieses Macros aus – die Index-Nummer wird dann im Kontextmenü angezeigt).

# MIDI In

MIDI In-Module dienen dazu, MIDI-Daten von externen Geräten in Reaktor verfügbar zu machen. Es gibt getrennte Module für verschiedene MIDI-Daten-Typen, darunter Noten, Velocity, Pitchbend, Mono und Poly Aftertouch, Controller, Program Change und MIDI Clock. Reaktor erzeugt auch Gate-Daten aus der Noten-Information, welche als verschiedene Gate-Module zur Verfügung stehen. Einige MIDI In-Module können auch für interne oder OSC-Connections eingesetzt werden.

---

## Note Pitch

## MIDI In



Polyphone Event-Quelle für die Tonhöhe von MIDI-Note-On-Events.

Ein Note-On-Event setzt den Ausgang auf einen Wert, der der Tastennummer (Note Number) entspricht. Der Bereich des Ausgangssignals wird im Properties-Dialog mit **Min** und **Max** eingestellt. Die Auflösung beträgt 128 Stufen. Ein Note-Off-Event hat hier keine Wirkung.

Wenn man bei dem standardmäßigen Ausgangsbereich von **Min** = 0 bis **Max** = 127 den **P**-Eingang eines Oscillators (zur logarithmischen Tonhöhen-Steuerung = Pitch) ansteuert, spielt man mit der üblichen wohltemperierte Stimmung. Eine Einheit entspricht dann einem Halbtonschritt. Das mittlere C liegt bei 60. Wenn man **Min** und **Max** anders einstellt, kann man die Tonhöhe entsprechend verschieben und skalieren, z. B. um in Vierteltönen zu spielen.

---

## Pitchbend

## MIDI In



Monophone Event-Quelle für MIDI-Pitchbend (Pitch Bender). Die Auflösung beträgt 16384 Stufen. Steht der Bitch-Bender in der Mittelstellung, ist der Ausgangswert immer 0. Der Bereich des Ausgangswerts kann für Pitchbend nach oben und unten unabhängig voneinander eingestellt werden. Für Pitchbend nach unten wird der Bereich mit **Min** und nach oben mit **Max** eingestellt.

Wenn man den **P**-Eingang eines Oscillators (zur logarithmischen Tonhöhen-Steuerung = Pitch) ansteuert (z. B. nach Addition mit einem Note-Pitch-Wert) kann man bei einem Ausgangsbereich von **Min** = -1 und **Max** = 1 die Tonhöhe um einen Halbton nach oben oder unten verändern. Ein Bereich von -12 bis +12 bedeutet Pitchbend innerhalb von  $\pm 12$  Halbtonschritten, d. h.  $\pm 1$  Oktave.



Polyphone Event-Quelle für MIDI Note-On- und Note-Off-Events.

Ein Note-On-Event setzt den Ausgang auf den Wert, der der Anschlagstärke (Velocity) entspricht. Der Bereich des Ausgangssignals wird im Properties-Dialog mit **Min** und **Max** eingestellt. Die Auflösung beträgt 128 Stufen. Ein Note-Off-Event setzt den Ausgang auf Null. Wenn man die Anschlagdynamik ausschalten will, muss man **Min** und **Max** auf den gleichen Wert einstellen.

---

## Single Trig. Gate



Monophone Event-Quelle für MIDI-Note-On- und Note-Off-Events mit Einmal-Trigger-Charakteristik.

Ein Note-On-Event setzt den Ausgang auf den Wert, der der Anschlagstärke (Velocity) entspricht. Der Bereich des Ausgangssignals wird im Properties-Dialog mit **Min** und **Max** eingestellt. Die Auflösung beträgt 128 Stufen. Ein Note-Off-Event setzt den Ausgang auf Null. Wenn man die Anschlagdynamik ausschalten will, muss man **Min** und **Max** auf den gleichen Wert einstellen.

Nur die erste Note erzeugt ein Event und startet damit angeschlossene Envelopes von vorne (Trigger). Noten, die angeschlagen werden, so lange noch andere gehalten werden (Legato-Spiel), erzeugen im Gegensatz zu **Gate** kein Event und triggert so z. B. Envelopes nicht erneut.

---

## Sel. Note Gate



Monophone Event-Quelle für ausgewählte MIDI-Note-On- und -Note-Off-Events.

Ein Note-On-Event mit der in den Properties eingestellten Tastennummer (Note Number) setzt den Ausgang auf den Wert, der der Anschlagstärke (Velocity) entspricht. Der Bereich des Ausgangssignals wird im Properties-Dialog mit **Min** und **Max** eingestellt. Die Auflösung beträgt 128 Stufen. Ein Note-Off-Event mit der eingestellten Tastennummer (Note Number) setzt den Ausgang auf Null. Wenn man die Anschlagdynamik ausschalten will, muss man **Min** und **Max** auf den gleichen Wert einstellen.

---

## On Velocity

MIDI In



Polyphone Event-Quelle für MIDI-Note-On-Velocity-Events. Ein Note-On-Event setzt den Ausgang auf den Wert, der der Anschlagstärke (Velocity) entspricht. Der Bereich des Ausgangssignals wird im Properties-Dialog mit **Min** und **Max** eingestellt. Die Auflösung beträgt 128 Stufen.

---

## Off Velocity

MIDI In



Polyphone Event-Quelle für MIDI-Note-Off-Velocity-Events. Ein Note-Off-Event setzt den Ausgang auf einen Wert, der der „Loslassgeschwindigkeit“ der Taste entspricht. Der Bereich des Ausgangssignals wird im Properties-Dialog mit **Min** und **Max** eingestellt. Die Auflösung beträgt 128 Stufen.

Anmerkung: Nur sehr wenige Keyboards erzeugen Note-Off-Velocity-Events.

---

## Controller

MIDI In



Monophone Event-Quelle für MIDI-Controller-Events (Control Change). Ein Event setzt den Ausgang auf einen Wert, der der Position des Steuerelements (Controller) entspricht. Die Nummer des Controllers (1–128) wird im Properties-Dialog mit **Note No** und der Bereich des Ausgangssignals mit **Min** und **Max** eingestellt. Die Auflösung beträgt 128 Stufen.

Die aktuellen Werte aller Controller (und Fader) eines Instruments kann in einem Snapshot abgespeichert und als solcher wieder geladen werden. Die Aktivierung des **Snap-Isolate**-Schalters in den Properties des Controllers verhindert, dass sich der Controller-Wert bei Snapshot-Recall ändert. Das Ausgangssignal des Controller-Moduls ist als Event- oder Audiosignal verwendbar.

Die Nummern einiger häufig auftretender MIDI-Controller sind:

- 1 Modulations-Rad (Mod Wheel)
- 7 Lautstärke (Volume)
- 10 Panorama (Pan)
- 64 Sustain
- 65 Portamento
- 66 Hold (Sostenuto)

Konsultieren Sie die MIDI-Implementation-Chart Ihres MIDI-Instruments, um herauszufinden, welche MIDI-Controller es übertragen kann.

---

## Ch. Aftertouch

MIDI In



Monophone Event-Quelle für monophone MIDI-Aftertouch-Events (Channel's After Touch). Ein Event setzt den Ausgang auf den Wert, der dem Druck auf die Taste entspricht. Der Bereich des Ausgangssignals wird im Properties-Dialog mit **Min** und **Max** eingestellt. Die Auflösung beträgt 128 Stufen.

---

## Poly Aftertouch

MIDI In



Polyphone Event-Quelle für so genannte polyphone MIDI-Aftertouch-Events (Key Aftertouch). Ein Event setzt den Ausgang auf einen Wert, der dem Druck auf die Taste entspricht. Dabei wird der Wert jeweils nur für die Stimme des REAKTOR-Instruments ausgegeben, die die zur Taste zugehörige Note spielt. Der Bereich des Ausgangssignals wird im Properties-Dialog mit **Min** und **Max** eingestellt. Die Auflösung beträgt 128 Stufen.

Anmerkung: Nur sehr wenige Keyboards erzeugen polyphonen Aftertouch.

---

## Sel. Poly AT

MIDI In



Monophone Event-Quelle für ausgewählte polyphone Aftertouch-Events (Key Aftertouch).

Ein Event mit der eingestellten Tastennummer (Note Number) setzt den Ausgang auf einen Wert, der dem Druck auf die Taste entspricht. Die Nummer der Taste (1–128) wird im Properties-Dialog mit **Controller No.**, der Bereich des Ausgangssignals mit **Min** und **Max** eingestellt. Die Auflösung beträgt 128 Stufen. Die aktuelle Position aller Controller (und Fader) eines Instruments kann in einem Snapshot abgespeichert und als solcher wieder geladen werden. Die Aktivierung des **Snap-Isolate**-Schalters in den Properties des Controllers verhindert, dass sich der Controller-Wert bei Snapshot-Recall ändert.

Anmerkung: Nur sehr wenige Keyboards erzeugen polyphonen Aftertouch.



---

## Program Change

MIDI In



Monophone Event-Quelle für MIDI-Program-Change-Events (Programmwechsel). Ein Event setzt den Ausgang auf einen Wert, der der Programmnummer entspricht. Der Bereich des Ausgangssignals wird im Properties-Dialog mit **Min** und **Max** eingestellt. Die Auflösung beträgt 128 Stufen.

Bei Verwendung dieses Moduls schaltet man normalerweise in den Instrument Properties **Prog. Change Enable** aus, damit die MIDI-Program-Change-Events nicht außerdem Snapshots aufrufen.

---

## Start/Stop

MIDI In



Quelle für Start- und Stop-Events zur Synchronisation mit externen MIDI-Geräten oder der internen zentralen Clock.

Das **Start/Stop**-Modul liefert am Ausgang ein monophones Gate-Signal, dessen Wert in den Properties mit **Output Value** eingestellt wird. Bei Drücken des Start-Schalters bzw. Empfang eines MIDI-Start-Events, springt das Signal auf den eingestellten Wert. Bei Drücken des Stop-Schalters bzw. Empfang eines MIDI-Stop-Events springt das Signal auf Null zurück.

Das Modul verbindet man typischerweise mit dem Reset-Eingang von Sequenzern und Event-Dividern, die zur MIDI Clock synchronisiert sind, um einen gleichzeitigen Start zu erzwingen.

---

## 1/96 Clock

MIDI In



Quelle für ein Taktsignal, das der externen MIDI-Clock oder der zentralen internen Clock entspricht.

Pro 96-ten Note liefert der Ausgang ein Event, dessen Wert in den Properties mit **Output Value** eingestellt wird.

Im Unterschied zur **Sync Clock**-Quelle müssen die Events der **1/96 Clock** über einen **Event Freq. Divider** (siehe Seite 414) aufbereitet werden. Der Vorteil liegt darin, dass Sie dabei mit beliebigen Teilerfaktoren experimentieren können.



Quelle für ein Taktsignal, das von der externen MIDI-Clock oder der zentralen internen Clock abgeleitet wird.

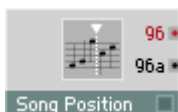
Der Ausgang liefert ein Gate-Signal, dessen Wert in den Properties mit **Output Value** eingestellt wird. Bei jedem Taktschlag springt das Gate auf den eingestellten Wert und nach einer bestimmten Zeit wieder auf Null zurück. Das Modul wird durch Start-Stop-Events aktiviert und deaktiviert.

Die Geschwindigkeit des Taktsignals (Viertel-, Achtel-, Sechzehntelnoten etc.) wird in den Properties unter **Rate** eingestellt.

Die Dauer, für die das Gate jeweils „an“ ist (eine Achtel-, Sechzehntelnote etc.) wird in den Properties unter **Duration** eingestellt.

---

## Song Pos



Modul zur Übermittlung von Song Position Informationen. Es wird in 96tel Noten vom Beginn eines Songs gezählt. Normalerweise verbindet man das Modul mit Eingang **A** des **Modulo** Moduls und eine Konstante mit dem Wert **6** mit Eingang **B**, um ein 16tel Notenraster am Div Ausgang zu bekommen.

- **96**: Eventausgang für die Anzahl von 96tel Noten (das sind 24 pro Viertelnote) seit Song-Start. Benutzen Sie das **Modulo** Modul, um andere Noteneinheiten zu bekommen.
- **96a**: Audioausgang für die sample-genaue Position in 96tel Noten.



Monophone Quelle, die MIDI-Channel-Nachrichten von einem externen MIDI-Gerät (Keyboard, Sequencer, etc.) oder von anderen Instrumenten innerhalb des Ensembles empfängt.

Die Ausgabe-Reihenfolge der Ports (siehe unten) ist genau festgelegt, von oben nach unten. Das stellt sicher, dass Typ (z. B. Änderung einer Reglerposition) und Quelle (z. B. Controller 7 auf Kanal 1) der MIDI-Channel-Nachricht immer vor dem eigentlichen Wert übermittelt werden.

- **St:** Ausgangs-Port, der den Typ der empfangenen Nachricht weitermeldet.
  - 0 = Note Off
  - 1 = Note On
  - 2 = Poly Aftertouch
  - 3 = Control Change
  - 4 = Program Change
  - 5 = Channel Aftertouch
  - 6 = Pitchbend
- **Ch:** Nummer des MIDI-Kanals (1-16).
- **Nr:** Nummer einer Note, eines Control-Change-Befehls oder eines Program-Change –Befehls (0-127).
- **Val:** Velocity-Wert einer Note, Aftertouch-Druck oder Wert von Control Change und Pitchbend. Externe MIDI-Nachrichten werden auf 7 Bit quantisiert (14 Bit für Pitchbend). Interne MIDI-Nachrichten werden im 32-Bit-Fließkomma-Format übermittelt. Alle Werte werden in den Feldern Min und Max im Dialog Properties getroffenen Einstellungen entsprechend skaliert, sodass sich in der Default-Einstellung ein Wertebereich von 0 bis 1 ergibt. Eine andere häufig verwendete Einstellung ist 0 bis 127, was in manchen Fällen zu einer besseren Auswertbarkeit der Werte führt (z. B. bei Program-Change-Nachrichten).

# MIDI Out

Reaktor kann MIDI-Daten sowohl verarbeiten als auch selbst erzeugen. MIDI Out-Module senden diese an externe MIDI-Geräte. Für jedes MIDI In-Module gibt es auch ein entsprechendes MIDI Out-Modul. Einige MIDI Out-Module können auch für interne oder OSC-Connections eingesetzt werden.

---

## Note Pitch/Gate

## MIDI Out



Wandelt ein monophones oder polyphones Event-Signal in MIDI-Noten-Events um. Jedes Event am Gate-Eingang **G** erzeugt eine MIDI-Nachricht an dem MIDI-Port, den REAKTOR für die Ausgabe benutzt. Der Wert des Events bestimmt die Anschlagstärke (Velocity). Ein Event mit Wert 1 erzeugt eine Velocity von 127. Ein Event mit Wert Null erzeugt ein Note Off-Event, schaltet also den Ton aus.

Die Tonhöhe (Tastenummer) der Note On und Note Off Events ist der aktuelle Wert am Pitch-Eingang **P** in dem Bereich, der mit **Min** und **Max** eingestellt wurde. Ein Event mit Wert gleich **Min** setzt den MIDI Note Pitch auf 0, ein Event mit Wert gleich **Max** setzt den MIDI Note Pitch auf 127.

---

## Pitchbend

## MIDI Out



Wandelt ein monophones Event-Signal in MIDI-Pitchbend-Events um. Jedes Event erzeugt eine MIDI-Nachricht an dem MIDI-Port, den REAKTOR für die Ausgabe benutzt. Der Bereich des Eingangssignal wird mit **Min** und **Max** bestimmt. Die Auflösung am Ausgang beträgt 16384 Schritte. Ein Event mit Wert gleich **Min** setzt den MIDI Pitchbend-Wert auf -8192, ein Event mit Wert gleich **Max** setzt den MIDI Pitchbend-Wert auf +8191.



Wandelt ein monophones Event-Signal in MIDI-Controller-Events um. Jedes Event erzeugt eine MIDI-Nachricht an dem MIDI-Port, den REAKTOR für die Ausgabe benutzt. Die Nummer des MIDI Controllers wird in den Properties unter **Controller No** eingestellt. Der Bereich des Eingangssignal wird mit **Min** und **Max** bestimmt. Die Auflösung am Ausgang beträgt 128 Schritte. Ein Event mit Wert gleich **Min** setzt den MIDI Controller auf 0, ein Event mit Wert gleich **Max** setzt den MIDI Controller auf 127.

---

## Ch. Aftertouch

## MIDI Out



Wandelt ein monophones Event-Signal in MIDI-Channel-Aftertouch-Events um. Jedes Event erzeugt eine MIDI-Nachricht an dem MIDI-Port, den REAKTOR für die Ausgabe benutzt. Der Bereich des Eingangssignal wird mit **Min** und **Max** bestimmt. Die Auflösung am Ausgang beträgt 128 Schritte. Ein Event mit Wert gleich **Min** setzt MIDI-Channel-Aftertouch auf 0, ein Event mit Wert gleich **Max** setzt Aftertouch auf 127.

---

## Poly Aftertouch

## MIDI Out



Die Aftertouch-Events (AT) werden zu MIDI-Poly-Aftertouch-Events umgewandelt. Der gerade am Pitch-Eingang (P) anliegende Wert wird in die Noten-Nummer umgewandelt. Werte zwischen Min und Max ergeben Noten-Nummern zwischen 0 und 127. Aftertouch-Werte zwischen 0 und 1 werden in Werte zwischen 0 und 127 umgewandelt.

- P: Pitch-Eingang für die Tonhöhe, die in eine Noten-Nummer gewandelt wird. Werte zwischen Min und Max ergeben Noten-Nummern zwischen 0 und 127.
- AT: Eingang für das Aftertouch-Signal. Werte zwischen 0 und 1 werden in MIDI-Aftertouch-Werte zwischen 0 und 127 umgewandelt werden.



Wandelt ein monophones Event-Signal in MIDI-Poly-Aftertouch-Events um. Jedes Event erzeugt eine MIDI-Nachricht an dem MIDI-Port, den REAKTOR für die Ausgabe benutzt. Die Nummer der Taste für die der Druck eingestellt wird, wird in den Properties unter **Note No.** Eingestellt. Der Bereich des Eingangssignal wird mit **Min** und **Max** bestimmt.

Die Auflösung am Ausgang beträgt 128 Schritte. Ein Event mit Wert gleich **Min** setzt den MIDI Aftertouch-Wert auf 0, ein Event mit Wert gleich **Max** setzt den MIDI Aftertouch-Wert auf 127.

Anmerkung: Nur sehr wenige MIDI-Geräte können polyphonen Aftertouch verarbeiten.

---

## Program Change

## MIDI Out



Wandelt ein monophones Event-Signal in MIDI-Program-Change-Events (Programmwechsel) um. Jedes Event erzeugt eine MIDI-Nachricht an dem MIDI-Port, den REAKTOR für die Ausgabe benutzt. Der Bereich des Eingangssignal wird mit **Min** und **Max** bestimmt. Die Auflösung am Ausgang beträgt 128 Schritte. Ein Event mit Wert gleich **Min** setzt MIDI-Program-Change auf Nummer 0, ein Event mit Wert gleich **Max** setzt das Programm auf 127.

---

## Start/Stop

## MIDI Out



Eingangsereignisse erzeugen Start/Continue/Stop-Ereignisse am MIDI-Ausgang.

- **G:** Ein positives Event sendet einen Start- oder Continue-Befehl, ein negatives oder Null-Event sendet einen Stop-Befehl.
- **Rst:** Nachdem ein positives Event an diesem Rst-Eingang empfangen wurde, sendet das nächste positive Event am G(ate)-Eingang einen Startbefehl (bei 0).



Eingangsevents erzeugen 1/96-Clock-Events am MIDI-Ausgang.

- **In:** Ein Event mit positivem Wert erzeugt ein MIDI-Clock-Event.

---

## Song Pos



Der am Pos-Eingang anliegende Wert wird mit einem Event am Trig-Eingang als Song-Position gesendet.

- **Trig:** Jedes Event mit einem positiven Wert erzeugt ein MIDI-Song-Position-Event.
- **Pos:** Der Wert an diesem Eingang wird (als ein Vielfaches von 1/96-Noten) mit einem Trigger-Event genommen und als MIDI-Song-Position gesendet.



Monophone Quelle, die MIDI-Channel-Nachrichten an externe MIDI-Geräte (Synthesizer, Sequencer, etc.) oder intern an andere Instrumente innerhalb des Ensembles sendet. Channel-Nachrichten werden immer dann übertragen, wenn ein Event am Port St eintrifft. Daher müssen der gewünschte Wert und das Ziel der Nachricht durch geeignete Nachrichten an den anderen Eingängen korrekt definiert sein, bevor Events am Eingang St ankommen. Alle Eingänge nehmen nur monophone Signale entgegen.

- **St:** Event-Eingang, der den Typ der Channel-Nachricht bestimmt, die gesendet wird. Eine Channel-Nachricht wird immer dann gesendet, wenn ein Event an diesem Eingang ankommt.
  - 0 = Note Off
  - 1 = Note On
  - 2 = Poly Aftertouch
  - 3 = Control Change
  - 4 = Program Change
  - 5 = Channel Aftertouch
  - 6 = Pitchbend
- **Ch:** Audio-Eingang für die Nummer des MIDI-Kanals (1-16). Nicht ganzzahlige Werte, die an diesem Eingang ankommen, werden auf die nächste Ganzzahl (Integer) aufgerundet; so würde z. B. der Wert 0.5 auf 1 aufgerundet werden.
- **Nr:** Audio-Eingang für die Nummer einer Note, eines Control-Change-Befehls oder eines Program-Change –Befehls (0-127).
- **Val:** Audio-Eingang für den Velocity-Wert einer Note, Aftertouch-Druck oder Wert von Control Change und Pitchbend.



# Math

In REAKTOR gibt es Module für gebräuchliche mathematische Operationen wie Addition, Subtraktion, Multiplikation und Division sowie weniger vertraute Funktionen wie Arcus Tangens und Kehrwert der Wurzel. Es gibt auch Module zur exponentialen und logarithmischen Umrechnung, um unterschiedlich skalierte Moduleingänge anzusteuern. Beispielsweise verfügen einige Module in REAKTOR über lineare Frequenzeingänge (gemessen in Hertz), während andere exponentielle Frequenzeingänge besitzen (gemessen in Halbtönen und auf MIDI-Noten abgestimmt). Deshalb gibt es Module, welche zwischen diesen zwei Formaten konvertieren können.

Alle Module in dieser Sektion sind Hybrid-Module. Dies bedeutet, dass sie sowohl als Audio- als auch als Eventmodule verwendet werden können, abhängig von der Verkabelung ihrer Eingänge. Viele dieser Module (Add und Mult zum Beispiel) besitzen auch eine dynamische Verwaltung der Moduleingänge, was durch drei kleine Punkte in der unteren linken Hälfte des Moduls angezeigt wird. Wenn ein Wire bei gedrückter Ctrl-Taste an eine leere Stelle im Eingangsport-Bereich (die linke Modulseite) eines solchen Moduls gezogen wird, wird automatisch ein neuer Eingangsport erzeugt.

---

## Constant

## Math



Monophone Quelle für eine Konstante. Der Wert wird im Properties-Dialog eingestellt. Das Modul erzeugt nur einmal ein Event, und zwar wenn es bei der Aktivierung initialisiert wird.

---

## Add

## Math



Addierer für zwei oder mehr Audio- oder Eventsignale. Das Ausgangssignal ist die Summe der Eingangssignale (**Out** = **In1** + **In2** + **In3** etc.).

Das Modul kann als einfacher Multi-Kanal-Mixer verwendet werden, wo alle Kanäle fest auf 0 dB stehen.

---

## Subtract

Math



Subtrahierer für zwei Audio- oder Eventsignale. Das Ausgangssignal ist die Subtraktion des zweiten Eingangssignals vom ersten (**Out = In1 - In2**).

---

## Invert, -X

Math



Inverter für ein Audio- oder Eventsignal. Das Ausgangssignal ist das invertierte Eingangssignal (**Out = -In**).

Einen Klang einfach zu invertieren, ist normalerweise nicht hörbar, außer wenn es mit dem uninvertierten Signal kombiniert wird. Aber Invertierung von Steuersignalen hat meist eine sehr deutliche Wirkung.

---

## Multiply

Math



Multiplizierer für zwei oder mehr Audio- oder Eventsignale. Das Ausgangssignal ist das Produkt der beiden Eingangssignale (**Out = In1 \* In2 \* In3 etc**).

Die typische Anwendung ist als signalgesteuerter Verstärker (entspricht VCA in analogen Synthesizern), wenn an einem Eingang ein Audiosignal und an dem anderen ein Verstärkungsfaktor anliegt.

Wenn eine Null an einem der Eingänge anliegt, wird am Ausgang immer ein Null-Signal ausgegeben.

Eine Funktion als Quadrierer läßt sich realisieren, indem zwei Eingängen dasselbe Signal zugeführt wird (**Out = In · In = In<sup>2</sup>**).

Wenn den Eingängen verschiedene Signale zugeführt werden, ist das Ausgangssignal deren Ringmodulation.

---

$$a * b + c$$

**Math**



Kombinierter Multiplizierer-Addierer:  
Der Ausgangswert ist das Ergebnis von  $(A * B) + C$ .

---

## Reciprocal 1/x

**Math**



Das Ausgangssignal ist der Kehrwert des Eingangssignals.  
Vorsicht ist geraten bei kleinen Eingangswerten (nahe Null), da dann der Ausgangswert sehr groß wird. Bei genau Null am Eingang ist das Ausgangssignal auch Null.

Die Anwendung auf Klangsignale ist normalerweise nicht sinnvoll. Das Modul ist eher für Steuersignale (die nicht in die Nähe von Null kommen) brauchbar.

---

## Divide x/y

**Math**



Das Ausgangssignal ist der Quotient aus dem x-Signal am oberen Eingang und dem y-Signal am unteren Eingang.

Vorsicht ist geraten bei kleinen Werten (nahe Null) am unteren Eingang, da dann der Ausgangswert sehr groß wird. Bei genau Null am Eingang ist das Ausgangssignal auch Null.

Die Anwendung auf Klangsignale ist normalerweise nicht sinnvoll.

Wann immer möglich, benutzen Sie einen Multiplizierer anstatt eines Dividierers bei Audiosignalen, denn der Prozessor wird so deutlich entlastet. Zum Beispiel, statt durch eine Konstante oder ein Event-Signal zu teilen, invertieren Sie das Event-Signal (1/x) und multiplizieren Sie mit dem Ergebnis das Audiosignal.

---

## Modulo x % y

Math



Modulo und Div. berechnen die ganzzahlige Division der zwei Eingangswerte sowie den Rest der Division.

- **A:** Hybrideingang für das Signal **A**, welches durch Signal **B** geteilt wird.
- **B:** Hybrideingang für das Signal **B**, welches zum Teilen von **A** benutzt wird. **B** ist normalerweise ein ganzzahliger Wert, muss es aber nicht sein.
- **Div:** Hybridausgang der ganzzahligen Division von **A** und **B**. Dieses ist der größte ganzzahlige Wert, der kleiner oder gleich **A/B** ist.
- **Mod:** Hybridausgang für das Modulo von **A** und **B**. Dies ist der Rest der ganzzahligen Division von **A** und **B**. Range: [ 0 ... **B** ].

---

## Rectifier

Math



Vollwellen-Gleichrichter. Das Eingangssignal wird gleichgerichtet, d.h. negative Werte werden invertiert und werden positiv.

- **In:** Hybrideingang für das gleichzurichtende Signal. Negative Werte werden positiv bei gleichbleibender Magnitude.
- **Out:** Hybridausgang für das gleichgerichtete Signal.

---

## Rect./Sign

Math

Vollwellen-Gleichrichter mit zusätzlichem Vorzeichen-Ausgang. Das Eingangssignal wird gleichgerichtet, d.h. negative Werte werden invertiert und werden positiv. Das Vorzeichen des Eingangssignals ist am Sign-Ausgang verfügbar.

- **In:** Hybrideingang für das gleichzurichtende Signal. Negative Werte werden positiv bei gleichbleibender Magnitude.

- **Sign:** Hybridausgang für das Vorzeichen des Eingangssignals (-1 oder +1).
- **Out:** Hybridausgang für das gleichgerichtete Signal.

---

## Compare

Math



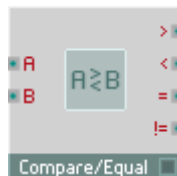
Vergleichende Logikfunktion. Vergleicht die zwei Eingangswerte miteinander und setzt den Ausgang auf den Wert, der dem logischen Ergebnis dieses Vergleichs entspricht.

- **A:** Eingang für den ersten der zwei Werte, die verglichen werden sollen.
- **B:** Eingang für den zweiten der zwei Werte, die verglichen werden sollen.
- **>:** Ausgang für das Ergebnis des Vergleichs „**A** größer als **B**“ (0=FALSCH, 1=WAHR)
- **<=:** Ausgang für das Ergebnis des Vergleichs „**A** kleiner als **B**“. (0 = FALSCH, 1 =WAHR)

---

## Compare/Equal

Math



Vergleichende Logikfunktion. Vergleicht die zwei Eingangswerte miteinander und setzt den Ausgang auf den Wert, der dem logischen Ergebnis dieses Vergleichs entspricht.

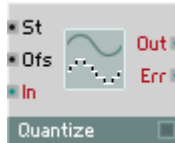
- **A:** Eingang für den ersten der zwei Werte, die verglichen werden sollen.
- **B:** Eingang für den zweiten der zwei Werte, die verglichen werden sollen.
- **>:** Ausgang für das Ergebnis des Vergleichs „**A** größer als **B**“ (0=FALSCH, 1=WAHR)

- **<:** Ausgang für das Ergebnis des Vergleichs „**A** kleiner als **B**“.  
(0 = FALSCH, 1 =WAHR)
- **=:** Ausgang für das Ergebnis des Vergleichs „**A** gleich **B**“.  
(0 = FALSCH, 1 = WAHR)
- **!:=:** Ausgang für das Ergebnis des Vergleichs „**A** ungleich **B**“.  
(0 = FALSCH, 1 =WAHR)

---

## Quantize

Math



Quantisierer mit einstellbarer Stufenweite. Das Eingangssignal wird auf die jeweils nächstliegende Quantisierungsstufe gerundet ausgegeben.

- **St:** Steuereingang für die Größe der Quantisierungsstufen.  
Bei **St** = 0 bleibt das Signal unquantisiert.
- **In:** Hybrideingang für das zu quantisierende Signal.
- **Out:** Hybridausgang für das quantisierte Signal.
- **Err:** Hybridausgang für den Quantisierungsfehler, der bei der Rundung auftritt: **Err** = **Out** – **In**.

---

## Expon. (A)

Math



Exponential-Funktion zur Konvertierung logarithmischer Pegel-Werte in dB in lineare Amplituden-Werte.

- **Lvl:** Hybrideingang für logarithmische Pegel-Werte, die in lineare Amplituden-Werte umgewandelt werden sollen.  
Typ. range: [50 ... 10].
- **A:** Hybridausgang für das lineare Amplituden-Steuersignal.



Exponential-Funktion zur Konvertierung logarithmischer Pitch-Werte in Halbtönen in lineare Frequenz-Werte in Hz.

- **P:** Hybrideingang für logarithmische Pitch-Werte in Halbtönen, die in lineare Frequenz-Werte in Hz umgewandelt werden sollen. Typ. range: [0 ... 127].
- **F:** Hybridausgang für das lineare Frequenz-Steuersignal.



Logarithmier-Funktion zur Konvertierung linearer Amplituden-Werte in logarithmische Pegel-Werte in dB. Dieses Modul kann auch zur Ansteuerung logarithmischer Time-Eingänge von Envelopes etc. verwendet werden.

- **A:** Hybrideingang für lineare Amplituden-Werte, die in logarithmische Pegel-Werte in dB umgewandelt werden sollen. Typ. range: [0 ... 1000].
- **Lvl:** Hybridausgang für den Pegel in dB. Typ. range: [-60 ... 0].



Logarithmier-Funktion zur Konvertierung linearer Frequenz-Werte in Hz in logarithmische Pitch-Werte in Halbtönen.

- **F:** Hybrideingang für lineare Frequenz-Werte in Hz, die in logarithmische Pitch-Werte in Halbtönen umgewandelt werden sollen. Typ. range: [0 ... 5000].
- **P:** Hybridausgang für den Pitch in Halbtönen. Typ. range: [0 ... 100].

---

## Power x y

Math



Das Power-Modul berechnet die Potenz X hoch Y (normalerweise notiert als  $X^Y$  oder  $X^{\wedge}Y$ ).

- X: Hybrid-Eingang für die Basis.
- Y: Hybrid-Eingang für den Exponenten.
- $X^Y$ : Hybrid-Eingang für das Ergebnis der Berechnung.

---

## Square Root

Math



Berechnet die Quadratwurzel des Eingangswertes.

- **In**: Hybrideingang für das Argument der Wurzelfunktion. Bei negativen Eingangswerten ist der Ausgang gleich 0.
- **Out**: Hybridausgang für die Quadratwurzel des Eingangswertes.

---

## 1 / Square Root

Math



Das Modul berechnet den Kehrwert der Wurzel für den am Eingang anliegenden Wert. Der Einsatz dieses Moduls ist effizienter als die Verwendung der beiden Module Square Root und Reciprocal  $1/x$ . Bei negativen Eingangswerten wird 0 ausgegeben.

In: Hybrideingang für das Argument.

Out: Hybridausgang für das Ergebnis.





Das Sine-Modul berechnet die trigonometrische Sinus-Funktion. Sowohl Eingang als auch Ausgang haben einen Wertebereich von -1 bis 1. Um einen Eingangswert, der in Grad vorliegt, zu berechnen, teilen Sie diesen zunächst durch 360. Um einen Eingangswert, der im Bogenmass (Radian) vorliegt, zu berechnen, teilen Sie diesen zunächst durch  $2\pi$  (ungefähr 6,283). Der Ausgangsbereich liegt zwischen 1 (Eingangswert 0,25) und -1 (Eingangswert 0,75).

- In: Hybrideingang für das Argument.
- Out: Hybridausgang für das Ergebnis.



Für jedes Eingangsevent werden der Sinus und der Cosinus berechnet. Ein Eingangswert von 1.0 entspricht einer ganzen Periode der Sinus- und Cosinusfunktion (also 360 Grad).

- **In**: Hybrideingang für das Argument (Winkel) der Sinusfunktion. Ein Wert von 1.0 entspricht einer Periode der Sinusfunktion (360 Grad). Typ. Range: [ -1 ... 1 ].
- **Sin**: Hybridausgang für den Sinus des Eingangswertes.
- **Cos**: Hybridausgang für den Cosinus des Eingangswertes.



Das ArcSin-Modul berechnet die umgekehrte Sinus-Funktion (Arcussinus von  $x$  ist die Nummer zwischen 0 und 1, dessen Sinus  $x$  ist). Da der Ausgangswert für die Sinus-Funktion zwischen -1 und 1 liegt, ist die Arcussinus-Funktion auch nur für Eingangswerte in diesem Bereich gültig. Für Argumente, die kleiner als -1 sind, gibt das Modul -0,25 aus, und für solche, die grösser als 1 sind, gibt es 0,25 aus.

- In: Hybrideingang für das Argument.
- Out: Hybridausgang für den Arcussinus des Eingangswerts.



Das ArcCos-Modul berechnet die umgekehrte Cosinus-Funktion (Arcuscosinus von  $x$  ist die Nummer zwischen 0 und 1, dessen Cosinus  $x$  ist). Da der Ausgangswert für die Cosinus-Funktion zwischen -1 und 1 liegt, ist die Arcuscosinus-Funktion auch nur für Eingangswerte in diesem Bereich gültig. Für Argumente, die kleiner als -1 sind, gibt das Modul 0,5 aus, und für solche, die grösser als 1 sind, gibt es 0 aus.

- In: Hybrideingang für das Argument.
- Out: Hybridausgang für den Arcuscosinus des Eingangswerts.



Das ArcTan-Modul berechnet die umgekehrte Tangens-Funktion (der Tangens ist der Sinus geteilt durch Cosinus). Der Ausgang des ArcTan-Moduls liegt im Bereich zwischen -0,25 und 0,25, welches -90 bis 90 Grad entspricht.

- In: Hybrideingang für das Argument.
- Out: Hybridausgang für den Arcustangens des Eingangswerts.

# Signal Path

Die Module in dieser Kategorie erlauben es, Event- und Audiodaten in den Strukturen flexibel zu verkabeln. Es stehen Mixer, Router für Ein- und Ausgänge, steuerbare Schaltungen (Relays), Crossfader und Panner zur Verfügung.

---

## Selector/Scanner

## Signal Path



Selector/Scanner. Die Eingänge werden abhängig von dem am **Pos**-Eingang anliegenden Wert abgetastet. Wenn **Pos** eine Integerzahl ist, erhalten Sie immer nur ein Eingangssignal, andererseits eine Mischung zweier Eingänge. Das Ausgangssignal setzt sich aus den zwei überblendeten Eingängen zusammen, deren Indexe dem Wert am Pos-Eingang am nächsten liegen.

Wenn der **Wrap**-Modus in den Properties aktiviert ist, arbeitet **Pos** als Schleife, so daß Max +1 als 0, Max +2 als 1 usw. interpretiert wird.

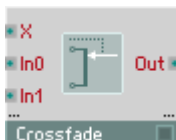
Der Selector/Scanner kann interessante Effekte erzeugen, wenn die Eingänge mit Signalen aus dem Multitap-Delays gespeist werden und die Abtastposition vom Ramp Oscillator gesteuert wird. Das Modul besitzt eine dynamische Eingangsport-Verwaltung. Die Anzahl der Eingänge kann mit Min Num Port Groups auf der Function-Seite der Properties definiert werden.

- **Pos**: Audiosteuerungseingang für die Auswahl des Eingangs/der Eingänge, der/die gescannt werden soll(en). **Pos** = 0 selektiert **In0**, **Pos** = 1 selektiert **In1**, **Pos** = 0.5 erzeugt einen Mix aus **In0** und **In1**. Typ. Wertebereich: [ 0 ... Max ]
- **In0...Max**: Eingänge für die zu scannenden Audiosignale.
- **Out**: Audioausgang für das gescannte Signal.



Relay. Der obere Eingang ist mit dem Ausgang verbunden, wenn  $Ctl > 0$  ist. Andernfalls wird der untere Eingang mit dem Ausgang verbunden. Wenn der untere Eingang nicht existiert, wird ein Null-Signal am Ausgang erzeugt. Das Modul kann einen oder zwei Eingangsports haben. Die Anzahl der Eingänge kann mit Min Num Port Groups auf der Function-Seite der Properties definiert werden.

- **Ctl**: Steuereingang zur Auswahl eines Signaleingangs.
- **In**: Signaleingang (bis zu zwei).
- **Out**: Ausgang für das ausgewählte Signal.



Crossfading-Modul für zwei Signale. Das Ausgangssignal ist eine gewichtete Mischung (lineare Interpolation) der beiden Eingangssignale **In0** und **In1**. Das Verhältnis der Anteile beider Eingangssignale am Ausgangssignal ist über den **X**-Port einstellbar.

Das Modul besitzt eine dynamische Eingangsport-Verwaltung. Die Anzahl der Eingangspaare (**In0** und **In1**) und die der Ausgangsports kann mit Min Num Port Groups auf der Function-Seite der Properties definiert werden.

- **X**: Hybrider Steuereingang für das Mischverhältnis.  
Wert zwischen 0 (**Out** = **In1**) und 1 (**Out** = **In2**).
- **In1, In2**: Hybrideingänge für die beiden zu mischenden Signale
- **Out**: Ausgang für das gemischte Signal (**Out** =  $(1 - X) \text{ In1} + X \text{ In2}$ ).



Distributor/Panner. Das Eingangssignal wird mittels des Pos-Eingangs mit einem oder mehreren Ausgängen verbunden bzw. es wird zwischen ihnen gepannt. Ist Pos eine ganze Zahl, wird das Signal nur eines Eingangs genommen, andernfalls erhalten Sie ein Panning zwischen zwei Eingängen.

Wenn der **Wrap**-Modus in den Properties aktiviert ist, arbeitet **Pos** als Schleife, so daß Max +1 als 0, Max +2 als 1 usw. interpretiert wird.

Das Modul besitzt eine dynamische Ausgangsport-Verwaltung. Die Anzahl der Ausgangsports kann mit Min Num Port Groups auf der Function-Seite der Properties definiert werden.

- Pos: Hybrideingang zur Auswahl des Ausgangs/der Ausgänge, welche(r) mit dem Eingang verbunden werden soll(en).
- In: Hybrider Signaleingang.
- Out: Ausgang oder mehrere Ausgänge für das Eingangssignal.

---

## Stereo Pan

## Signal Path



Panoramaregler. Durch Veränderung der Signalpegel beider Ausgänge wird das Eingangssignal im Stereofeld plaziert. Die Summe der Werte an den Ausgängen **L** und **R** ist immer genau zwei mal so groß wie der Eingangswert, d. h. der Eingang wird in einem variablen Verhältnis auf die beiden Ausgänge aufgeteilt.

Das Modul besitzt eine dynamische Port-Verwaltung. Die Anzahl der Eingänge und die der Ausgangspaare (L und R) kann mit Min Num Port Groups auf der Function-Seite der Properties definiert werden.

- **Pan**: Steuereingang zur Festlegung der Stereo-Position (-1 = links, 0 = Mitte, 1 = rechts).
- **In**: Hybrideingang für das im Stereobild zu positionierende Signal.
- **L**: Hybridausgang für das linke Signal.
- **R**: Hybridausgang für das rechte Signal.



Amp/Mixer für eine einstellbare Anzahl an Audiosignalen. Die Eingangssignale werden jeweils um den am zugehörigen Lvl-Port anliegenden Wert (in dB) verstärkt oder abgeschwächt und am Ausgang aufsummiert. Das Modul besitzt eine dynamische Eingangsport-Verwaltung. Die Anzahl der Eingangspaare (Lvl und In) kann mit Min Num Port Groups auf der Function-Seite der Properties definiert werden.

- **Lvl:** Logarithmischer Steuereingang für die Verstärkung (Gain) in dB.
- **In:** Signaleingang.
- **Out:** Ausgang für das gemixte Signal ( $\text{Out} = \text{In1} \cdot 10^{\text{Lvl1}/20} + \text{In2} \cdot 10^{\text{Lvl2}/20}$  etc.).



Panoramaregler mit Verstärker für eine einstellbare Anzahl an Audio-Signalen. Die Eingangssignale werden um die an den Lvl-Ports anliegenden Werte (in dB) verstärkt oder abgeschwächt. Durch Veränderung der Signalpegel beider Ausgänge wird das Eingangssignal im Stereofeld plziert.

Das Modul besitzt eine dynamische Eingangsport-Verwaltung. Die Anzahl der Eingangsgruppen (Lvl, Pan und In) kann mit Min Num Port Groups auf der Function-Seite der Properties definiert werden.

- **Lvl:** Logarithmischer Steuereingang für die Verstärkung (Gain) in dB. Wenn der Wert kleiner als 0 ist, ist das Ausgangssignal kleiner als das Eingangssignal.
- **Pan:** Steuereingang zur Festlegung der Stereo-Position (-1 = links, 0 = Mitte, 1 = rechts).
- **In:** Audioeingang für das zu verstärkende und im Stereofeld zu positionierende Audio-Signal.
- **L:** Ausgang für das verstärkte linke Audio-Signal.
- **R:** Ausgang für das verstärkte rechte Audio-Signal.

# Oscillator

In REAKTOR wird der Begriff Oscillator für eine grosse Anzahl an Signalgeneratoren verwendet. Tatsächlich wird jedes Klangerzeugungsverfahren, welches keine Samples verwendet, Oscillator genannt. Dies schliesst die gewöhnlichen Wellenformgeneratoren (Sine, Pulse, Sawtooth etc.) ein wie auch Impulse, Step, Noise und Audio-Table.

Alle Oscillatoren in REAKTOR können mit jeder Frequenz laufen, von 0 Hz (Stillstand) über das gesamte Audiospektrum bis zur Grenze, die von der Abtastrate gesetzt wird. Damit sind alle Oscillatoren gleichermaßen für den Einsatz als LFO wie auch zur Klangwellenformgenerierung geeignet. Wenn man einen Oscillator als LFO nimmt, um einen Eingang eines anderen Moduls zu modulieren, der nur Events annimmt (z. B. P im Gegensatz zu F), muss man einen A to E (perm) Konverter dazwischenschalten.

---

## Sawtooth

## Oscillator



Oscillator für Sägezahn-Wellenform mit logarithmischem Tonhöhen-Steuereingang (Pitch) und linearer Amplitudenmodulation.

- **P:** Logarithmischer Event-Steuereingang für die Tonhöhe (Oscillator-Frequenz) in Halbtonschritten (69 = 440 Hz).
- **A:** Audio-Steuereingang für die Amplitude. Das Ausgangssignal bewegt sich zwischen **+A** und **-A**.
- **Out:** Audio-Ausgang für die Sägezahn-Wellenform.

---

## Saw FM

## Oscillator



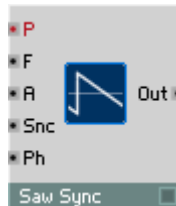
Oscillator für Sägezahn-Wellenform mit logarithmischem Tonhöhen-Steuereingang (Pitch), linearer Frequenzmodulation (FM) und linearer Amplitudenmodulation.



- **P**: Logarithmischer Event-Steuereingang für die Tonhöhe (Oscillator-Frequenz) in Halbtonschritten ( $69 = 440 \text{ Hz}$ ).
- **F**: Audio-Steuereingang für die lineare Frequenzmodulation in Hz. Die Oscillatorfrequenz bestimmt sich aus **P** und **F**.
- **A**: Audio-Steuereingang für die Amplitude. Das Ausgangssignal bewegt sich zwischen **+A** und **-A**.
- **Out**: Audio-Ausgang für die Sägezahn-Wellenform.

## Saw Sync

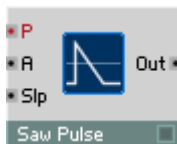
## Oscillator



Oscillator für Sägezahn-Wellenform mit Synchronisationseingang, logarithmischem Tonhöhen-Steuereingang (Pitch), linearer Frequenzmodulation (FM) und linearer Amplitudenmodulation.

Immer wenn das Synchronisationssignal vom Nullpunkt in positive Richtung geht (steigende Flanke), wird die Phase des Oscillators auf Null zurückgesetzt, um die Wellenform von Anfang zu starten.

- **P**: Logarithmischer Event-Steuereingang für die Tonhöhe (Oscillator-Frequenz) in Halbtonschritten ( $69 = 440 \text{ Hz}$ ).
- **F**: Audio-Steuereingang für die lineare Frequenzmodulation in Hz. Die Oscillatorfrequenz bestimmt sich aus **P** und **F**.
- **A**: Audio-Steuereingang für die Amplitude. Das Ausgangssignal bewegt sich zwischen **+A** und **-A**.
- **Snc**: Audio-Steuereingang für das Synchronisationssignal. Positiver Nulldurchgang bewirkt einen Neustart des Oscillators bei Null.
- **Ph**: Eingang zur Bestimmung der Phase (Position in der Wellenform), auf die der Oscillator bei Synchronisation zurückgesetzt wird.
- **Out**: Audio-Ausgang für die Sägezahn-Wellenform.



Oscillator für variable Sägezahn/Nadelpuls-Wellenform mit logarithmischem Tonhöhen-Steuereingang (Pitch) und linearer Amplitudenmodulation. Die Steilheit der Rampe läßt sich verändern und damit die Wellenform zwischen einem normalen Sägezahn und einer kurzen Spitze einstellen.

- **P:** Logarithmischer Event-Steuereingang für die Tonhöhe (Oscillator-Frequenz) in Halbtonschritten ( $69 = 440$  Hz).
- **A:** Audio-Steuereingang für die Amplitude. Das Ausgangssignal bewegt sich zwischen **+A** und **-A**.
- **Slp:** Audio-Steuereingang für die Steigung (Slope) der Wellenform. Der Wert 0 ergibt einen normalen Sägezahn, ein größerer Wert verkürzt die Rampe bis zu einer kurzen Spitze (Nadel).
- **Out:** Audio-Ausgang für die Sägezahn/Nadelpuls-Wellenform



Oscillator für bipolare Sägezahn-Wellenform mit Nullphase, mit logarithmischem Tonhöhen-Steuereingang (Pitch), Pulsbreitenmodulation (PWM) und linearer Amplitudenmodulation.

- **P:** Logarithmischer Event-Steuereingang für die Tonhöhe (Oscillator-Frequenz) in Halbtonschritten ( $69 = 440$  Hz).
- **A:** Audio-Steuereingang für die Amplitude. Das Ausgangssignal bewegt sich zwischen **+A** und **-A**.
- **W:** Audio-Steuereingang für die Pulsbreitenmodulation (PWM). Wertebereich: 0 bis ca. 6. Bei **W** = 0 erhält man eine normale Sägezahn-Wellenform (Saw Wave), bei größerem **W** kürzere Pulse und eine längere Nullphase.
- **Out:** Audio-Ausgang für die bipolare Sägezahn-Wellenform.

---

## Triangle

## Oscillator



Oscillator für symmetrische Dreieck-Wellenform mit logarithmischem Tonhöhen-Steuereingang (Pitch) und linearer Amplitudenmodulation.

- **P:** Logarithmischer Event-Steuereingang für die Tonhöhe (Oscillator-Frequenz) in Halbtonschritten ( $69 = 440$  Hz).
- **A:** Audio-Steuereingang für die Amplitude. Das Ausgangssignal bewegt sich zwischen **+A** und **-A**.
- **Out:** Audio-Ausgang für die Dreieck-Wellenform.

---

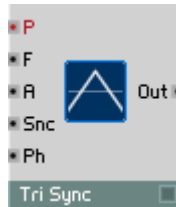
## Tri FM

## Oscillator



Oscillator für symmetrische Dreieck-Wellenform mit logarithmischem Tonhöhen-Steuereingang (Pitch), linearer Frequenzmodulation (FM) und linearer Amplitudenmodulation.

- **P:** Logarithmischer Event-Steuereingang für die Tonhöhe (Oscillator-Frequenz) in Halbtonschritten ( $69 = 440$  Hz).
- **F:** Audio-Steuereingang für die lineare Frequenzmodulation in Hz. Die Oscillatorfrequenz bestimmt sich aus **P** und **F**.
- **A:** Audio-Steuereingang für die Amplitude. Das Ausgangssignal bewegt sich zwischen **+A** und **-A**.
- **Out:** Audio-Ausgang für die Dreieck-Wellenform.



Oscillator für symmetrische Dreieck-Wellenform mit Synchronisationseingang, logarithmischem Tonhöhen-Steuereingang (Pitch), linearer Frequenzmodulation (FM) und linearer Amplitudenmodulation.

Immer wenn das Synchronisationssignal vom Nullpunkt in positive Richtung geht (steigende Flanke), wird die Phase des Oscillators auf 0 zurückgesetzt.

- **P**: Logarithmischer Event-Steuereingang für die Tonhöhe (Oscillator-Frequenz) in Halbtonschritten ( $69 = 440$  Hz).
- **F**: Audio-Steuereingang für die lineare Frequenzmodulation in Hz. Die Oscillatorfrequenz bestimmt sich aus **P** und **F**.
- **A**: Audio-Steuereingang für die Amplitude. Das Ausgangssignal bewegt sich zwischen **+A** und **-A**.
- **Snc**: Audio-Steuereingang für das Synchronisationssignal. Positiver Nulldurchgang bewirkt einen Neustart des Oscillators bei Null.
- **Ph**: Eingang zur Bestimmung der Phase (Position in der Wellenform), auf die der Oscillator bei Synchronisation zurückgesetzt wird.
- **Out**: Audio-Ausgang für die Dreieck-Wellenform.



Oscillator für variable Dreieck- und Parabel-Wellenformen, mit logarithmischem Tonhöhen-Steuereingang (Pitch) und linearer Amplitudenmodulation. Die mit (W) einstellbare Symmetrie erlaubt obertonarme symmetrische Wellenformen bis hin zu obertonreichen Sägezahn-Wellenformen bei hoher Asymmetrie.

- **P**: Logarithmischer Event-Steuereingang für die Tonhöhe (Oscillator-Frequenz) in Halbtonschritten ( $69 = 440$  Hz).

- **A:** Audio-Steuereingang für die Amplitude. Das Ausgangssignal bewegt sich zwischen **+A** und **-A**.
- **W:** Eingang für die Steuerung der Wellensymmetrie: -1=Fallender Sägezahn, 0=Dreieck, +1=Aufsteigender Sägezahn. Typ. Range: [0...1]
- **Par:** Audio-Ausgang für die Parabel-Wellenform.
- **Tri:** Audio-Ausgang für die Dreieck -Wellenform.

## Parabol

## Oscillator



Oscillator für Parabel-Wellenform mit logarithmischem Tonhöhen-Steuereingang (Pitch) und linearer Amplitudenmodulation. Die Wellenform besteht aus zwei Hälften, die jeweils aus einer Parabelfunktion resultieren. Der Oscillator klingt wie ein Sinus mit einer leichten Beimischung von ungeradzahligten Obertönen und ist somit oft ein ausreichender Ersatz für einen Sinusgenerator, verursacht aber weniger CPU-Last als dieser.

- **P:** Logarithmischer Event-Steuereingang für die Tonhöhe (Oscillator-Frequenz) in Halbtonschritten (69 = 440 Hz).
- **A:** Audio-Steuereingang für die Amplitude. Das Ausgangssignal bewegt sich zwischen **+A** und **-A**.
- **Out:** Audio-Ausgang für die Parabel-Wellenform

## Par FM

## Oscillator

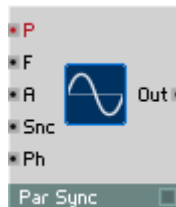


Oscillator für Parabel-Wellenform mit logarithmischem Tonhöhen-Steuereingang (Pitch), linearer Frequenzmodulation (FM) und linearer Amplitudenmodulation. Die Wellenform besteht aus zwei Hälften, die jeweils aus einer Parabelfunktion resultieren. Der Oscillator klingt wie ein Sinus mit einer leichten Beimischung von ungeradzahligten Obertönen und kann als Ersatz für einen Sinusgenerator herangezogen werden, wobei dieser weniger Rechenaufwand erfordert.

- **P**: Logarithmischer Event-Steuereingang für die Tonhöhe (Oscillator-Frequenz) in Halbtonschritten ( $69 = 440 \text{ Hz}$ ).
- **F**: Audio-Steuereingang für die lineare Frequenzmodulation in Hz. Die Oscillatorfrequenz bestimmt sich aus **P** und **F**.
- **A**: Audio-Steuereingang für die Amplitude. Das Ausgangssignal bewegt sich zwischen **+A** und **-A**.
- **Out**: Audio-Ausgang für die Parabel-Wellenform

## Par Sync

## Oscillator



Oscillator für Parabel-Wellenform mit Synchronisationseingang, logarithmischem Tonhöhen-Steuereingang (Pitch), linearer Frequenzmodulation (FM) und linearer Amplitudenmodulation. Die Wellenform besteht aus zwei Hälften, die jeweils aus einer Parabelfunktion resultieren. Der Oscillator klingt wie ein Sinus mit einer leichten Beimischung von ungeradzahligem Obertönen und kann als Ersatz für einen Sinusgenerator herangezogen werden, wobei er aber weniger Rechenaufwand als dieser erfordert.

Immer wenn das Synchronisationssignal vom Nullpunkt in positive Richtung geht (steigende Flanke) wird die Phase des Oscillators auf 0 zurückgesetzt.

- **P**: Logarithmischer Event-Steuereingang für die Tonhöhe (Oscillator-Frequenz) in Halbtonschritten ( $69 = 440 \text{ Hz}$ ).
- **F**: Audio-Steuereingang für die lineare Frequenzmodulation in Hz. Die Oscillatorfrequenz bestimmt sich aus **P** und **F** zusammen.
- **A**: Audio-Steuereingang für die Amplitude. Das Ausgangssignal bewegt sich zwischen **+A** und **-A**.
- **Snc**: Audio-Steuereingang für das Synchronisationssignal. Positiver Nulldurchgang bewirkt einen Neustart des Oscillators bei Null.
- **Ph**: Eingang zur Bestimmung der Phase (Position in der Wellenform), auf die der Oscillator bei Synchronisation zurückgesetzt wird.
- **Out**: Audio-Ausgang für die Parabel-Wellenform



Oscillator für variable Parabol-Wellenform mit logarithmischem Tonhöhen-Steuereingang (Pitch) und linearer Amplitudenmodulation. Das Verhältnis zwischen der Länge des unteren und oberen Teils der Kurve lässt sich verändern und damit die Wellenform zwischen einer normalen symmetrischen Parabol-Welle und einer einfachen Parabel einstellen.

- **P:** Logarithmischer Event-Steuereingang für die Tonhöhe (Oscillator-Frequenz) in Halbtonschritten (69 = 440 Hz).
- **A:** Audio-Steuereingang für die Amplitude. Das Ausgangssignal bewegt sich zwischen **+A** und **-A**.
- **W:** Event-Steuereingang für die Symmetrie der Wellenform. Der Wert -1 ergibt nach unten offene Parabeln, 0 eine normale symmetrische Parabol-Wellenform und +1 nach oben offene Parabeln.
- **Out:** Audio-Ausgang für die variable Parabol-Wellenform.



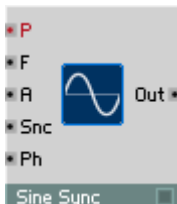
Oscillator für Sinus-Wellenform mit logarithmischem Tonhöhen-Steuereingang (Pitch) und linearer Amplitudenmodulation. Wenn der Sinus nicht ganz rein sein muss, eignet sich der Parabol-Oscillator als Ersatz mit weniger Rechenaufwand.

- **P:** Logarithmischer Event-Steuereingang für die Tonhöhe (Oscillator-Frequenz) in Halbtonschritten (69 = 440 Hz).
- **A:** Audio-Steuereingang für die Amplitude. Das Ausgangssignal bewegt sich zwischen **+A** und **-A**.
- **Out:** Audio-Ausgang für die Sinus-Wellenform.



Oscillator für Sinus-Wellenform mit logarithmischem Tonhöhen-Steuereingang (Pitch), linearer Frequenzmodulation (FM) und linearer Amplitudenmodulation. Wenn der Sinus nicht ganz rein sein muss, eignet sich der Parabol-Oscillator als Ersatz mit weniger Rechenaufwand.

- **P**: Logarithmischer Event-Steuereingang für die Tonhöhe (Oscillator-Frequenz) in Halbtonschritten ( $69 = 440$  Hz).
- **F**: Audio-Steuereingang für die lineare Frequenzmodulation in Hz. Die Oscillatorfrequenz bestimmt sich aus **P** und **F**.
- **A**: Audio-Steuereingang für die Amplitude. Das Ausgangssignal bewegt sich zwischen **+A** und **-A**.
- **Out**: Audio-Ausgang für die Sinus-Wellenform.



Oscillator für Sinus-Wellenform mit Synchronisationseingang, logarithmischem Tonhöhen-Steuereingang (Pitch), linearer Frequenzmodulation (FM) und linearer Amplitudenmodulation.

Immer wenn das Synchronisationssignal vom Nullpunkt in positive Richtung geht (steigende Flanke) wird die Phase des Oscillators auf eine Position zurückgesetzt, die mit dem Phasen-Eingang bestimmt wird.

Wenn der Sinus nicht ganz rein sein muss, eignet sich der Parabol-Oscillator als Ersatz mit weniger Rechenaufwand.

- **P**: Logarithmischer Event-Steuereingang für die Tonhöhe (Oscillator-Frequenz) in Halbtonschritten ( $69 = 440$  Hz).
- **F**: Audio-Steuereingang für die lineare Frequenzmodulation in Hz. Die Oscillatorfrequenz bestimmt sich aus **P** und **F**.



- **A:** Audio-Steuereingang für die Amplitude. Das Ausgangssignal bewegt sich zwischen **+A** und **-A**.
- **Snc:** Audio-Steuereingang für das Synchronisationssignal. Positiver Nulldurchgang bewirkt einen Neustart des Oscillators bei Null.
- **Ph:** Eingang zur Bestimmung der Phase (Position in der Wellenform), auf die der Oscillator bei Synchronisation zurückgesetzt wird. **Ph** = 0: Phase = 0° (Mitte des ansteigenden Teils), **Ph** = 0.5: Phase = 180° (Mitte des abfallenden Teils), **Ph** = 1: Phase = 360° (wie 0°).
- **Out:** Audio-Ausgang für die Sinus-Wellenform

## Multi-Sine

## Oscillator



Oscillator für Additive Synthese. Eine Wellenform wird durch das Überlagern einzelner Sinuswellen (die Obertöne) erzeugt. Für jede Komponente lassen sich die Amplitude (**A**) und das Frequenzvielfache (**F**) getrennt einstellen.

Die Sinuswellenkomponenten haben eine feste Phasenlage zueinander. Das bedeutet, daß sich die Wellenform exakt mit der Frequenz wiederholt, die mit dem Wert am **P**-Eingang eingestellt ist.

Die Frequenzvielfachen sind normalerweise ganze Zahlen (die Obertonnummer), sonst ergeben sich keine reinen Sinuswellen.

- **P:** Logarithmischer Event-Steuereingang für die Tonhöhe (Oscillator-Frequenz) in Halbtonschritten (69 = 440 Hz).
- **F1:** Audio-Steuereingang für den Frequenzfaktor (Obertonnummer) der ersten Sinuswelle, als Vielfaches der Grundfrequenz. Typ. Range: [1...20]
- **A1:** Linearer Audio-Steuereingang für die Amplitude der ersten Sinuswelle. Kann auch für Tremolo und Ringmodulation benutzt werden. Typ. Range: [ 0 ... 1 ].

- **F2:** Wie **F1**, aber für die zweite Sinuswelle.
- **A2:** Wie **A1**, aber für die zweite Sinuswelle.
- **F3:** Wie **F1**, aber für die dritte Sinuswelle.
- **A3:** Wie **A1**, aber für die dritte Sinuswelle.
- **F4:** Wie **F1**, aber für die vierte Sinuswelle.
- **A4:** Wie **A1**, aber für die vierte Sinuswelle.
- **S1:** Ausgang für die erste Sinuswelle.
- **S2:** Ausgang für die zweite Sinuswelle.
- **S3:** Ausgang für die dritte Sinuswelle.
- **S4:** Ausgang für die vierte Sinuswelle.
- **Out:** Audio-Ausgang für das Signal, das durch Addieren der Sinuswellen erzeugt wird.

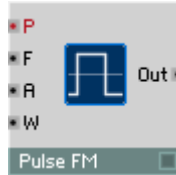
## Pulse

## Oscillator



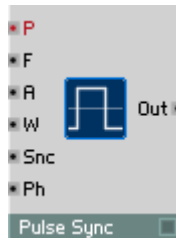
Oscillator für Rechteck-Wellenform mit logarithmischem Tonhöhen-Steuereingang (Pitch), Pulsbreitenmodulation (PWM) und linearer Amplitudenmodulation.

- **P:** Logarithmischer Event-Steuereingang für die Tonhöhe (Oscillator-Frequenz) in Halbtonschritten (69 = 440 Hz).
- **A:** Audio-Steuereingang für die Amplitude. Das Ausgangssignal bewegt sich zwischen **+A** und **-A**.
- **W:** Audio-Steuereingang für die Pulsbreitenmodulation (PWM). Wertebereich -1 bis 1. Symmetrische Wellenform (Square Wave) 50:50 bei **W** = 0, Pulsverhältnis 33:66 bei -0.33, 66:33 bei 0.33, 75:25 bei 0.5, 90:10 bei 0.8.  $Lo : Hi = (1 + W) : (1 - W)$ .
- **Out:** Audio-Ausgang für die Rechteck-Wellenform.



Oscillator für Rechteck-Wellenform mit logarithmischem Tonhöhen-Steuereingang (Pitch), linearer Frequenzmodulation (FM), Pulsbreitenmodulation (PWM) und linearer Amplitudenmodulation.

- **P**: Logarithmischer Event-Steuereingang für die Tonhöhe (Oscillator-Frequenz) in Halbtonschritten (69 = 440 Hz).
- **F**: Audio-Steuereingang für die lineare Frequenzmodulation in Hz. Die Oscillatorfrequenz bestimmt sich aus **P** und **F**.
- **A**: Audio-Steuereingang für die Amplitude. Das Ausgangssignal bewegt sich zwischen **+A** und **-A**.
- **W**: Audio-Steuereingang für die Pulsbreitenmodulation (PWM). Wertebereich -1 bis 1. Symmetrische Wellenform (Square Wave) 50:50 bei **W** = 0, Pulsverhältnis 33:66 bei -0.33, 66:33 bei 0.33, 75:25 bei 0.5, 90:10 bei 0.8.  $Lo : Hi = (1 + W) : (1 - W)$ .
- **Out**: Audio-Ausgang für die Rechteck-Wellenform.



Oscillator für Rechteck-Wellenform mit Synchronisationseingang, logarithmischem Tonhöhen-Steuereingang (Pitch), linearer Frequenzmodulation (FM), Pulsbreitenmodulation (PWM) und linearer Amplitudenmodulation.

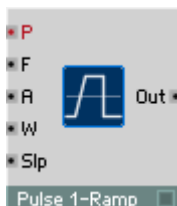
Immer wenn das Synchronisationssignal vom Nullpunkt in positive Richtung geht (steigende Flanke), wird die Phase des Oscillators auf 0 zurückgesetzt.

- **P**: Logarithmischer Event-Steuereingang für die Tonhöhe (Oscillator-Frequenz) in Halbtonschritten (69 = 440 Hz).

- **F:** Audio-Steuereingang für die lineare Frequenzmodulation in Hz. Die Oscillatorfrequenz bestimmt sich aus **P** und **F**.
- **A:** Audio-Steuereingang für die Amplitude. Das Ausgangssignal bewegt sich zwischen **+A** und **-A**.
- **W:** Audio-Steuereingang für die Pulsbreitenmodulation (PWM). Wertebereich -1 bis 1. Symmetrische Wellenform (Square Wave) 50:50 bei **W** = 0, Pulsverhältnis 33:66 bei -0.33, 66:33 bei 0.33, 75:25 bei 0.5, 90:10 bei 0.8.  $Lo : Hi = (1 + W) : (1 - W)$ .
- **Snc:** Audio-Steuereingang für das Synchronisationssignal. Positiver Nulldurchgang bewirkt einen Neustart des Oscillators bei Null.
- **Ph:** Eingang zur Bestimmung der Phase (Position in der Wellenform), auf die der Oscillator bei Synchronisation zurückgesetzt wird.
- **Out:** Audio-Ausgang für die Rechteck-Wellenform.

## Pulse 1-ramp

## Oscillator



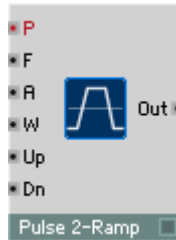
Oscillator für rechtwinklige Trapez-Wellenform mit logarithmischem Tonhöhen-Steuereingang (Pitch), linearer Frequenzmodulation (FM), Pulsbreitenmodulation (PWM) und linearer Amplitudenmodulation. Die Wellenform ist eine Mischung aus Rechteck und Sägezahn: Die steigende Flanke einer Rechteckwellenform kann abgeflacht und ihre Steilheit eingestellt werden. Die fallende Flanke ist senkrecht.

- **P:** Logarithmischer Event-Steuereingang für die Tonhöhe (Oscillator-Frequenz) in Halbtonschritten ( $69 = 440$  Hz).
- **F:** Audio-Steuereingang für die lineare Frequenzmodulation in Hz. Die Oscillatorfrequenz bestimmt sich aus **P** und **F**.
- **A:** Audio-Steuereingang für die Amplitude. Das Ausgangssignal bewegt sich zwischen **+A** und **-A**.
- **W:** Audio-Steuereingang für die Pulsbreitenmodulation (PWM). Wertebereich -1 bis 1.

- **Slp**: Audio-Steuereingang für die Steilheit der ansteigenden Rampe. Wenn **Slp** = 0 bzw. der Eingang nicht verbunden ist, steigt die Wellenform nicht an und am Ausgang liegt 0 an, d. h. es ist nichts zu hören. Typischer Wertebereich: 1 ... 20.
- **Out**: Audio-Ausgang für die Trapez-Wellenform.

## Pulse 2-ramp

## Oscillator



Oscillator für Trapez-Wellenform mit logarithmischem Tonhöhen-Steuereingang (Pitch), linearer Frequenzmodulation (FM), Pulsbreitenmodulation (PWM) und linearer Amplitudenmodulation. Die Wellenform ist eine Mischung aus Rechteck, Sägezahn und Dreieck: Beide Flanken einer Rechteckwellenform können abgeflacht werden, die Steilheit ist einstellbar.

- **P**: Logarithmischer Event-Steuereingang für die Tonhöhe (Oscillator-Frequenz) in Halbtonschritten (69 = 440 Hz).
- **F**: Audio-Steuereingang für die lineare Frequenzmodulation in Hz. Die Oscillatorfrequenz bestimmt sich aus **P** und **F**.
- **A**: Audio-Steuereingang für die Amplitude. Das Ausgangssignal bewegt sich zwischen **+A** und **-A**.
- **W**: Audio-Steuereingang für die Pulsbreitenmodulation (PWM). Wertebereich -1 bis 1.
- **Up**: Audio-Steuereingang für die Steilheit der ansteigenden Rampe.
- **Dn**: Audio-Steuereingang für die Steilheit der fallenden Rampe.
- **Out**: Audio-Ausgang für die Trapez-Wellenform.



Oscillator für bipolare Rechteck-Wellenform mit Nullphase. Mit logarithmischem Tonhöhen-Steuereingang (Pitch), Pulsbreitenmodulation (PWM) und linearer Amplitudenmodulation.

- **P**: Logarithmischer Event-Steuereingang für die Tonhöhe (Oscillator-Frequenz) in Halbtonschritten ( $69 = 440$  Hz).
- **A**: Audio-Steuereingang für die Amplitude. Das Ausgangssignal bewegt sich zwischen **+A** und **-A**.
- **W**: Audio-Steuereingang für die Pulsbreitenmodulation (PWM). Wertebereich  $-0$  bis  $1$ . Bei **W** =  $0$  erhält man eine normale symmetrische Rechteck-Wellenform (Square Wave) 50:50, bei größerem Wert an **W** kürzere Pulse und eine längere Nullphase.
- **Out**: Audio-Ausgang für die bipolare Rechteck-Wellenform.



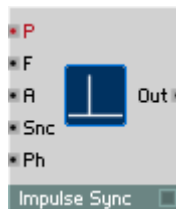
Oscillator für Impuls-Wellenform mit logarithmischem Tonhöhen-Steuereingang (Pitch), und linearer Amplitudenmodulation.

- **P**: Logarithmischer Event-Steuereingang für die Tonhöhe (Oscillator-Frequenz) in Halbtonschritten ( $69 = 440$  Hz).
- **A**: Audio-Steuereingang für die Amplitude. Das Ausgangssignal bewegt sich zwischen **+A** und **-A**.
- **Out**: Audio-Ausgang für die Impuls-Wellenform.



Oscillator für Impuls-Wellenform mit logarithmischem Tonhöhen-Steuereingang (Pitch), linearer Frequenzmodulation (FM) und linearer Amplitudenmodulation.

- **P:** Logarithmischer Event-Steuereingang für die Tonhöhe (Oscillator-Frequenz) in Halbtonschritten (69 = 440 Hz).
- **F:** Audio-Steuereingang für die lineare Frequenzmodulation in Hz. Die Oscillatorfrequenz bestimmt sich aus **P** und **F**.
- **A:** Audio-Steuereingang für die Amplitude.
- **Out:** Audio-Ausgang für die Impuls-Wellenform.



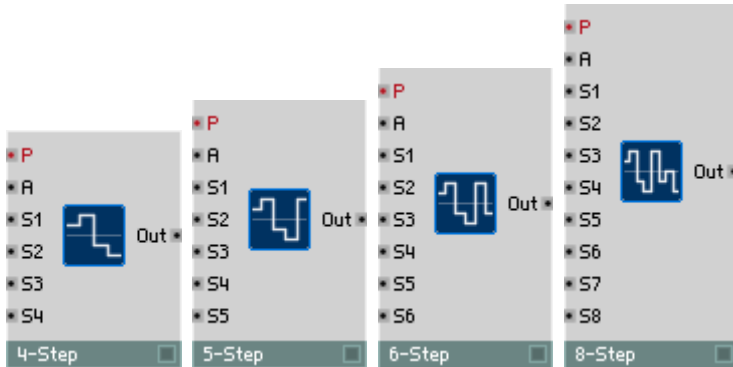
Oscillator für Impuls-Wellenform mit Synchronisationseingang, logarithmischem Tonhöhen-Steuereingang (Pitch), linearer Frequenzmodulation (FM), und linearer Amplitudenmodulation. Immer wenn das Synchronisationssignal vom Nullpunkt in positive Richtung geht (steigende Flanke) wird die Phase des Oscillators auf 0 zurückgesetzt.

- **P:** Logarithmischer Event-Steuereingang für die Tonhöhe (Oscillator-Frequenz) in Halbtonschritten (69 = 440 Hz).
- **F:** Audio-Steuereingang für die lineare Frequenzmodulation in Hz. Die Oscillatorfrequenz bestimmt sich aus **P** und **F** zusammen.
- **A:** Audio-Steuereingang für die Amplitude. Das Ausgangssignal bewegt sich zwischen **+A** und **-A**.
- **Snc:** Audio-Steuereingang für das Synchronisationssignal. Ein positiver Nulldurchgang bewirkt einen Neustart des Oscillators bei Null.

- **Ph:** Eingang zur Bestimmung der Phase (Position in der Wellenform), auf die der Oscillator bei Synchronisation zurückgesetzt wird.
- **Out:** Audio-Ausgang für die Impuls-Wellenform.

## Multi-Step

## Oscillator



## 4-Step

## Oscillator

Oscillator für 4-Schritt-Wellenform mit logarithmischem Tonhöhen-Steuereingang (Pitch) und linearer Amplitudenmodulation. Die Höhe jedes Schrittes läßt sich unabhängig von den anderen einstellen.

- **P:** Logarithmischer Event-Steuereingang für die Tonhöhe (Oscillator-Frequenz) in Halbtonschritten ( $69 = 440 \text{ Hz}$ ).
- **A:** Audio-Steuereingang für die Amplitude. Das Ausgangssignal bewegt sich zwischen **+A** und **-A**.
- **S1:** Audio-Steuereingang für die Höhe des ersten Schritts.
- **S2:** Audio-Steuereingang für die Höhe des zweiten Schritts.
- **S3:** Audio-Steuereingang für die Höhe des dritten Schritts.
- **S4:** Audio-Steuereingang für die Höhe des vierten Schritts.
- **Out:** Audio-Ausgang für die 4-Schritt-Wellenform.



---

## 5-Step

## Oscillator

Wie 4-Step, aber mit 5 Schritten.

---

## 6-Step

## Oscillator

Wie 4-Step, aber mit 6 Schritten.

---

## 8-Step

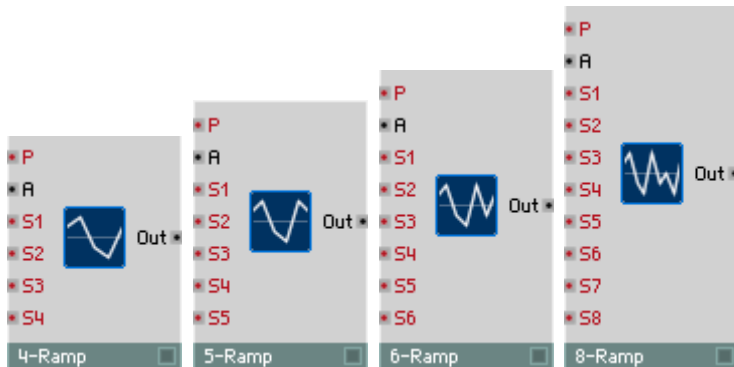
## Oscillator

Wie 4-Step, aber mit 8 Schritten.

---

## Multi-Ramp

## Oscillator



---

## 4-Ramp

## Oscillator

Oscillator für 4-Rampen-Wellenform mit logarithmischem Tonhöhen-Steuereingang (Pitch) und linearer Amplitudenmodulation. Die Höhe jedes der Punkte, die durch Rampen verbunden sind (Breakpoints), läßt sich unabhängig von den anderen einstellen.

- **P:** Logarithmischer Event-Steuereingang für die Tonhöhe (Oscillator-Frequenz) in Halbtonschritten (69 = 440 Hz).
- **A:** Audio-Steuereingang für die Amplitude. Das Ausgangssignal bewegt sich zwischen **+A** und **-A**.
- **S1:** Event-Steuereingang für die Höhe des ersten Breakpoint.
- **S2:** Event-Steuereingang für die Höhe des zweiten Breakpoint.

- **S3**: Event-Steuereingang für die Höhe des dritten Breakpoint.
- **S4**: Event-Steuereingang für die Höhe des vierten Breakpoint.
- **Out**: Audio-Ausgang für die 4-Rampen-Wellenform.

---

## 5-Ramp

## Oscillator

Wie 4-Ramp, aber mit 5 Rampen.

---

## 6-Ramp

## Oscillator

Wie 4-Ramp, aber mit 6 Rampen.

---

## 8-Ramp

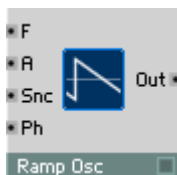
## Oscillator

Wie 4-Ramp, aber mit 8 Rampen.

---

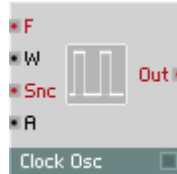
## Ramp

## Oscillator



Oscillator zur Erzeugung von Rampen-Wellenformen, welche typischerweise als Steuersignal benutzt werden, um zum Beispiel ein Audio Table-Modul als Wellenform-Oscillator zu verwenden. Das Signal steigt von 0 auf A an und wird dann schlagartig zurückgesetzt.

- **F**: Audioeingang zur Steuerung der Oscillatorfrequenz in Hz. Um die Tonhöhe in Halbtönen zu steuern, verwenden Sie ein Event Expon. (F)-Modul.
- **A**: Eingang zur Amplitudensteuerung des Rampensignals. Typ. Wert: 1.
- **Snc**: Audio-Steuereingang für das Synchronisationssignal. Ein positiver Nulldurchgang bewirkt einen Reset des Oscillators auf **Ph**.
- **Ph**: Audioeingang für die Synchronisationsphase. Wenn der Oscillator synchronisiert wird, springt sein Ausgang auf diesen Wert mal **A** zurück. Typ. Bereich: 0...1.
- **Out**: Audioausgang für das Rampensignal. Wertebereich 0...A.



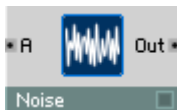
Freilaufende Taktsignal-Quelle. Ein interner Oscillator (monophon) erzeugt regelmäßige Clock An/Aus-Pulse in der Form von Events, z. B. um damit einen Sequencer anzutreiben. Der Wert der An-Events kann in den Modul-Properties eingestellt werden.

- **F:** Event-Eingang zur Steuerung der Clock-Takt-Frequenz in Hz. Zur Angabe des Tempos in BPM (Beats per Minute = Taktschläge pro Minute) berechnet man den Wert in Hz wie folgt: *Clocks-pro-Taktschlag* x BPM/60. Für 16<sup>tel</sup> Noten-Clocks bei 4/4-Takt (das sind vier 16<sup>tel</sup> pro Taktschlag) bekommt man **F** = 4/60 x BPM oder 0.0667 x BPM.
- **W:** Event-Eingang zur Steuerung der Pulsbreite, d. h. dem Verhältnis der Dauer der An-Phase zur Aus-Phase. Der Wertebereich ist -1 bis 1. Symmetrische Signalform (Gleiche Dauer von An und Aus) bei **W** = 0; Aus:An = ( 1 + **W** ) / ( 1 - **W** ).
- **Snc:** Event-Eingang zur Synchronisierung des Taktsignals. Ein positiver Event synchronisiert die Taktsignal-Quelle.
- **A:** Amplitudeneingang. Es muss ein Wert größer 0 angeschlossen sein, damit nicht lediglich Null-Events produziert werden. Typ. Value: 1.
- **Out:** Event-Ausgang für das Taktsignal. Der Wert wechselt zwischen dem An-Wert und Null.

---

## Noise

## Oscillator



Rauschgenerator, erzeugt weißes Rauschen, d.h. ein Zufallssignal, in dem alle möglichen Frequenzanteile gleichermaßen enthalten sind. Das Signal besteht aus nur zwei Werten,  $A/2$  und  $-A/2$ , die aber zufällig aufeinander folgen.

- **A:** Audio-Steuereingang für die Amplitude des Ausgangssignals.
- **Out:** Audio-Ausgang für das Rauschsignal.

---

## Random

## Oscillator



Zufallswertgenerator. Erzeugt eine Schrittwellenform, bei der die Höhe jenes Schritts ein Zufallswert aus dem gegebenen Bereich ist. Die Funktionsweise ist vergleichbar mit einem gleichverteilten, weißen Rauschgenerator und nachgeschaltetem Abtast-Glied (Sample & Hold), das mit einer gleichmäßigen Frequenz getaktet ist.

- **P:** Logarithmischer Event-Steuereingang für die Schrittrate (Sample-&-Hold-Frequenz) in Halbtonschritten ( $69 = 440$  Hz).
- **A:** Audio-Steuereingang für die Amplitude. Das Ausgangssignal ist ein Zufallswert zwischen  $+A$  und  $-A$ .
- **Out:** Audio-Ausgang für das Zufallswert-Signal.



Erzeugt Events in zufälligen Abständen, ähnlich wie ein „Geigerzähler“ für das Aufspüren radioaktiver Teilchen. Die durchschnittliche Rate der Events kann am **P**-Eingang eingestellt werden. **Rnd** steuert die „Zufälligkeit“.

- **P**: Logarithmischer Event-Steuereingang für die Durchschnittsrate (oder Dichte) der zufällig erzeugten Events.
- **Rnd**: Event-Steuereingang für die „Zufälligkeit“ oder Regelmäßigkeit der zeitlichen Verteilung der Events: 0 = völlig zufällig, 1 = völlig periodisch.
- **Clk**: Audioausgang für die zu zufälligen Zeiten erzeugten Clicks.
- **Out**: Ausgang für die zu zufälligen Zeiten erzeugten Events. Diese können beispielsweise zum Auslösen einer Hüllkurve (**G**-Eingang) benutzt werden.

# Sampler

Wenn ein Modul von sich aus ein Audiosignal erzeugt und es sich nicht um einen Oscillator handelt, haben Sie es mit einem Sampler zu tun. REAKTOR enthält einfache Sample-Player sowie ausgeklügelte Sample-Prozessoren für granulare Synthese mit Pitch- und Time Shifting und Beat-Slicing. Bei manchen Modulen gibt es sogar die Möglichkeit, ein individuelles Sample mittels einer Nummer über den Sel-Eingang auszuwählen.

## Properties - Function-Seite

- **Wellenform-Button:** Ein Klick auf den Button mit dem Wellenform-Symbol öffnet den Sample Map Editor.
- **Embed Samples In Ensemble** aktiviert die Option zum Speichern von Samples mit dem Ensemble.
- **No Stereo** unterbindet die Stereo-Wiedergabe von Samples. Bei Aktivierung sinkt die Rechenlast.
- Über das Quality-Dropdown-Menü wird die Qualität der Sample-Wiedergabe in drei Stufen (**Poor**, **Good** und **Excellent**) gesteuert. Höhere Qualität wird dabei mit einem Ansteigen der Rechenlast erkaufte. Der Begriff "Qualität" steht hier für das Fehlen von Störgeräuschen. Selbstverständlich können gerade diese Störgeräusche die Qualität eines Samples im musikalischen Sinne erhöhen.
- **Oscil. Mode** versetzt das Modul bei Aktivierung in den Oscillator-Modus.

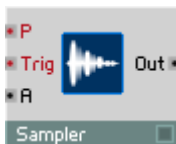
Sampler im Oscillator-Modus gehen davon aus, daß die verwendeten Samples *Wellenformen* oder *Wellensätze* enthalten. Eine Wellenform ist die Aufzeichnung einer Periode aus einer periodischen Schwingung, also im allgemeinen ein sehr kurzes Sample. Durch wiederholte Wiedergabe des Samples kann die periodische Schwingung wiederhergestellt werden. Das Sampler-Modul wird so zu einem digitalen Oscillator. Sampler-Module im Oscillator-Modus interpretieren die am **P**-Eingang anliegenden Werte genau wie REAKTOR-Oscillatoren, nämlich als MIDI-Notennummern und produzieren eine periodische Schwingung mit der entsprechenden Grundtonhöhe.

Die Module **Sampler** und **Sampler FM** interpretieren im Oscillator-Modus das gesamte Sample als eine Periode. Um einen Ton der Grundtonhöhe 440 Hertz zu erzeugen, wird das gesamte Sample in einer Sekunde 440 mal durchlaufen, unabhängig von der Dauer des Samples.

**Sampler Loop** interpretiert im Oscillator-Modus die Loop-Länge als Periode. Die Loop-Länge wird aus dem Soundfile gelesen, kann aber auch über den **LL**-Eingang des Moduls verändert werden. In einem Sample können also mehrere – tatsächlich beliebig viele – Wellenformen untergebracht werden; es handelt sich dann um ein WaveSet, also um eine Aneinanderreihung von Wellenformen. Angenommen, ein WaveSet umfaßt 100 Abtastwerte; dann passen 100 solcher Wellenformen in ein 10000 Abtastwerte großes Sample. Die erste Wellenform geht von Abtastwert 0 bis Abtastwert 99, die zweite von 100 bis 199, etc.. Wenn die Loop-Länge die Periode einer Wellenform bestimmt, bestimmt logischerweise die Position des Loops die Auswahl der Wellenform aus dem WaveSet. **Sampler Loop** verfügt über den **LS**-Eingang zur Steuerung des Loop-Startpunkts. Die an diesem Eingang anliegenden Werte werden im Oscillator-Modus auf Wellenform-Grenzen quantisiert, so daß ein knackfreier Übergang zwischen Wellenformen gewährleistet ist. Die Position im WaveSet kann so einfach über Hüllkurven, Anschlagsdynamik etc. gesteuert werden. Offensichtlich muss man solche WaveSet-Samples konstruieren oder von synthetischen Klangquellen aufnehmen. Die Bibliothek von REAKTOR enthält eine Reihe interessanter WaveSets. **Sampler Loop** integriert WaveSet-Synthese in die Palette der Syntheseformen von REAKTOR, so daß WaveSet-Synthese nun erstmalig in Verbindung mit FM zur Verfügung steht.

### Properties - Appearance page

- **Picture:** Aktivieren Sie diese Option, um eine Wellenformdarstellung für das Sampler-Modul im Panel zu sehen.
- **Size X/Size Y:** Erlaubt die Einstellung der Grösse der Wellenformanzeige für das Sampler-Modul in Pixel.
- **Scroll Bar:** Aktivieren Sie diese Option, um eine Scrollbar unter der Wellenformanzeige zu Navigationszwecken zu sehen. Die Scrollbar erlaubt das Bewegen in der Wellenform (indem man die Scrollbar in der Mitte zieht) und das Zoomen in die Wellenform hinein und aus ihr heraus (indem man die Scrollbar an einer der Seiten nach links bzw. rechts zieht).



Player für die polyphone und transponierte Wiedergabe eines Samples oder einer Sample-Map. Die Sample-Verwaltung erfolgt über den Sample Map Editor (Siehe *Sample Map Editor* in Kapitel 21.3).

Bei aktiviertem Loop wird das gesamte Sample ständig wiederholt und durch positive Events am Trigger-Eingang neu gestartet. Bei de-aktiviertem Loop läuft das Sample auf ein positives Trigger-Event hin einmal von Anfang bis Ende, oder, bei Auswahl der Laufrichtung **Backward**, von Ende bis Anfang.

Bei aktiviertem **Waveform-Mode** wird die Wiedergabe-Rate an die Länge des aktuellen Samples angepaßt, um beim Betrieb als Wellenform-Oscillator die richtige Tonhöhe zu erhalten.

- **P**: Logarithmischer Steuereingang für die Wiedergaberate (Pitch) und für die Auswahl eines Samples aus der geladenen Sample-Map. Wenn **P** gleich dem **Root-Key** des aktuellen Samples ist, wird das Sample in Original-Tonhöhe wiedergegeben.
- **Trig**: Ein positives Event an diesem Eingang startet (trigger) einen erneuten Auslesevorgang am Sampleanfang.
- **A**: Steuereingang für die Ausgangs-Amplitude.
- **Out**: Ausgang des Sample-Players.





Player für die polyphone und transponierte Wiedergabe eines Samples oder einer Sample-Map. Der **F**-Eingang und der Startpunkt-Steuereingang (**St**) erlauben die Manipulation der Sample-Wiedergabe.

Die Sample-Verwaltung erfolgt über den Sample Map Editor (Siehe *Sample Map Editor* auf Seite 253.).

Bei aktiviertem Loop wird das gesamte Sample ständig wiederholt; durch positive Events am Trigger-Eingang wird auf die Position des über den **St**-Eingang zu bestimmenden Startpunkts zurückgesprungen. Bei deaktiviertem Loop läuft das Sample auf ein positives Trigger-Event hin einmal vom Startpunkt bis zum Ende, oder, bei Auswahl der Laufrichtung **Backward**, vom Ende bis zum Anfang.

Bei aktiviertem **Waveform-Mode** wird die Wiedergabe-Rate an die Länge des aktuellen Samples angepaßt, um beim Betrieb als Wellenform-Oscillator die richtige Tonhöhe zu erhalten.

- **P**: Logarithmischer Steuereingang für die Wiedergaberate (Pitch) und für die Auswahl eines Samples aus der geladenen Sample-Map. Wenn **P** gleich dem **Root-Key** des aktuellen Samples ist, wird das Sample in Original-Tonhöhe wiedergegeben.
- **F**: Linearer Steuereingang für die Modulation der Wiedergaberate. Die Wirkung ist eine Frequenz-Modulation wie bei den Oscillator-Modulen in REAKTOR. Bei großen negativen Werten wird das Sample rückwärts wiedergegeben.
- **St**: Steuereingang für den Startpunkt beim nächsten Triggern; in Millisekunden vom Sampleanfang.
- **Trig**: Ein positives Event an diesem Eingang startet (trigger) einen erneuten Auslesevorgang am Sampleanfang.
- **A**: Steuereingang für die Ausgangs-Amplitude.
- **Out**: Ausgang des Sample-Players.
- **Lng**: Polyphoner Event-Ausgang für die Länge des aktuellen Samples in Millisekunden.



Universal-Player für die polyphone und transponierte Wiedergabe von Mono- oder Stereo- Samples, Sample-Maps und WaveSets.

Die Sample-Verwaltung erfolgt über den Sample Map Editor (Siehe *Sample Map Editor* auf Seite 253.).

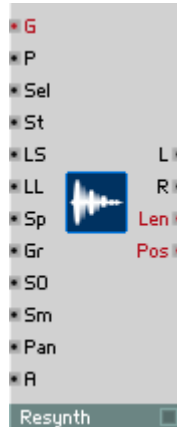
Bei einem positiven **Gate**-Event beginnt die Sample-Wiedergabe beim Startpunkt (einstellbar über den **St**-Eingang). Bei de-aktiviertem Loop läuft das Sample einmal vom Startpunkt bis zum Ende, oder, bei Auswahl der Laufrichtung **Backward**, vom Ende bis zum Anfang. Bei aktiviertem Loop wird das Sample innerhalb des Loop-Bereichs endlos wiederholt, sobald die Wiedergabe in diesen Bereich eingetreten ist. Der Loop-Bereich ist durch Daten aus dem Soundfile vorgegeben, aus dem das Sample geladen wurde. Falls das Soundfile keine Loop-Daten enthält, gilt der Sample-Beginn als Loop-Beginn und die Sample-Länge als Loop-Länge. Die Loop-Daten aus dem Soundfile sind Defaults (Grundeinstellungen), die immer dann gelten, wenn der Loop-Start-Eingang (**LS**), beziehungsweise der Loop-Length-Eingang (**LL**) nicht mit anderen Modulen verbunden sind.

Falls die **Loop in Release**-Option de-aktiviert ist, wird die Loop-Wiedergabe bei einem **Gate**-Event kleiner oder gleich Null beendet; das Sample läuft je nach Laufrichtung auf den Sample-Beginn oder das Sample-Ende zu und verstummt dann. Bei aktivierter **Loop in Release**-Option, oder bei ausgeschalteter Loop-Wiedergabe haben **Gate**-Events kleiner oder gleich Null keine Auswirkung.

Bei aktivierter **No Stereo**-Option (erreichbar über den Properties-Dialog) werden Stereo-Samples wie Mono-Samples behandelt, das heißt auf dem **L**- und auf dem **R**-Ausgang wird ein identisches Signal ausgegeben. **Sampler Loop** verwendet dann nur den linken Kanal von Stereo-Samples. Diese Option steht zur Verfügung, weil Mono-Wiedergabe geringere Anforderungen an die Rechenkapazität stellt.

- **G:** Ein positives Event an diesem Eingang startet den Auslesevorgang bei der Position, die am **St**-Eingang eingestellt ist. Bei negativen Werten wird die Loop-Wiedergabe abgebrochen, außer wenn die Option **Loop in Release** aktiv ist.
- **P:** Logarithmischer Steuereingang für die Wiedergaberate (Pitch). Das Sample wird in Original-Tonhöhe wiedergegeben, wenn **P** gleich dem **Root-Key** des aktuellen Samples ist. **P** bestimmt bei nicht verbundenem **Sel**-Eingang außerdem die Auswahl von Samples aus der Sample-Map. Im **Oscillator** Modus arbeitet **Sampler Loop** als digitaler Oscillator; **P** bestimmt wie bei den Oscillator-Modulen in REAKTOR die Grundtonhöhe der erzeugten Schwingung.
- **Sel:** Steuereingang für die Auswahl (Select) eines Samples aus der Sample-Map. Wird dieser Eingang nicht verbunden, werden die am **P**-Eingang anliegenden Werte verwendet.
- **F:** Linearer Steuereingang für die Modulation der Wiedergaberate. Die Wirkung ist eine Frequenz-Modulation wie bei den Oscillator-Modulen in REAKTOR. Bei großen negativen Werten wird das Sample rückwärts wiedergegeben.
- **St:** Steuereingang für den Startpunkt beim nächsten Triggern; in Millisekunden vom Sampleanfang.
- **LS:** Steuereingang für den Loop-Startpunkt, in Millisekunden vom Sampleanfang. Wird dieser Eingang nicht verbunden, gelten die Loop-Daten aus dem Soundfile. Sind im Soundfile keine Loop-Daten gespeichert, gilt als Default: **LS** = 0. Veränderungen an diesem Eingang werden beim Re-Triggern von Samples und beim Erreichen einer Loop-Begrenzung wirksam.

- **LL**: Steuereingang für die Loop-Länge in Millisekunden. Wird dieser Eingang nicht verbunden, gelten die Loop-Daten aus dem Soundfile. Sind im Soundfile keine Loop-Daten gespeichert, gilt als Default: **LL** = Länge des Samples. Veränderungen an diesem Eingang werden beim Re-Triggern von Samples und beim Erreichen einer Loop-Begrenzung wirksam.
- **A**: Steuereingang für die Ausgangs-Amplitude.
- **L**: Audio-Ausgang für den linken Kanal des Sample-Players.
- **R**: Audio-Ausgang für den rechten Kanal des Sample-Players. Bei Verarbeitung von Mono-Samples oder bei Aktivierung der **No Stereo**-Option liegt an den Ausgängen **L** und **R** das selbe Signal an.
- **Lng**: Polyphoner Event-Ausgang für die Länge des aktuellen Samples in Millisekunden.



Echtzeit-Re-Synthesizer für die polyphone, transponierte Wiedergabe von Mono- oder Stereo- Samples und Sample-Maps. **Resynth** erlaubt die unabhängige Echtzeit-Kontrolle von Tonhöhe und Abspielgeschwindigkeit und ermöglicht umfassende Manipulationen von Samples.

“Normale” Sampler wie die bekannten Hardware-Sampler oder die Module **Sampler**, **Sampler FM** und **Sampler Loop** in REAKTOR unterhalten pro Stimme einen *Zeiger*, der auf die aktuelle Position im Sample verweist. Am Ausgang des Samplers liegt immer die Amplitude des Samples an der Zeigerposition an. Der Zeiger bewegt sich mehr oder weniger schnell durch das Sample und erzeugt so an den Ausgängen eine zeitveränderliche Amplitude – also eine Schwingung.

Die Geschwindigkeit der Zeigerbewegung bestimmt die Geschwindigkeit der Sample-Wiedergabe (beispielsweise das Tempo eines Beat-Loops). Gleichzeitig bestimmt sie die empfundene Tonhöhe: Je langsamer das im Sample aufgezeichnete Signal abgetastet wird, desto größer werden die Perioden der am Ausgang entstehenden Schwingungen – die Tonhöhe fällt. Kommt der Zeiger zum Stillstand, verändert sich die Ausgangsamplitude gar nicht mehr. Es entsteht kein hörbares Signal.

**Resynth** verwendet, genau wie konventionelle Sampler, einen Zeiger auf die aktuelle Sample-Position. Das Signal an den Ausgängen ist jedoch nicht einfach die Sample-Amplitude an der Zeigerposition. Tatsächlich wird das Ausgangssignal von einem Modul-internen Synthesizer erzeugt, der das Signal an der Zeigerposition *re-synthetisiert*. Die Tonhöhe des von diesem Synthesizer erzeugten Signals ist von der Geschwindigkeit des Zeigers unabhängig. Auch wenn sich der Zeiger im Stillstand befindet, erzeugt der Synthesizer weiter Klang.

Während die Tonhöhe des ausgegebenen Signals wie üblich über den **P**-Eingang bestimmt wird, legt der **Sp**-Eingang die Zeigergeschwindigkeit fest.

Selbstverständlich entspricht das verlangsamte oder “eingefrorene” Signal nicht in allen Fällen der Vorstellung, sofern eine solche überhaupt besteht: Wie klingt der Aufprall des Hammers auf dem Nagel in unendlicher Verlangsamung? Der Re-Synthese-Algorithmus von **Resynth** ist so ausgelegt, dass ein weiterer Bereich von Klängen im Sinn der musikalischen Erwartung sinnvoll und subtil oder drastisch verarbeitet werden kann. Der Algorithmus ist über den **Gr** (Granularity) und den **Sm**-Parameter (Smoothness) zu beeinflussen, so daß manuelle Anpassungen an das Sample möglich sind und drastische Klangverfremdungen provoziert werden können. Über den Properties-Dialog kann – für alle Samples einer Map gesondert – die Option **Signal-Informed Granulation** aktiviert werden. Wenn diese Option eingeschaltet ist, berücksichtigt der Re-Synthese-Algorithmus von **Resynth** Informationen über das Sample, die aus einer beim erstmaligen Laden von Samples stattfindenden Analyse des Samples stammen. Der Algorithmus reagiert so auf die Eigenschaften des Materials. Wenn diese Option de-aktiviert ist, klingen die Resultate im allgemeinen ziemlich “elektronisch”.

Die Sample-Verwaltung erfolgt über den Sample Map Editor (Siehe *Sample Map Editor* auf Seite 253.).

Bei einem positiven **Gate**-Event beginnt die Sample-Wiedergabe beim Startpunkt (einstellbar über den **St**-Eingang). Bei de-aktiviertem Loop läuft das Sample einmal vom Startpunkt bis zum Ende, oder, bei Auswahl der Laufrichtung **Backward**, vom Startpunkt bis zum Anfang. Bei aktiviertem Loop wird das Sample innerhalb des Loop-Bereichs endlos wiederholt, sobald die Wiedergabe in diesen Bereich eingetreten ist. Der Loop-Bereich ist durch Daten aus dem Soundfile vorgegeben, aus dem das Sample geladen wurde. Falls das Soundfile keine Loop-Daten enthält, gilt der Sample-Beginn als Loop-Beginn und die Sample-Länge als Loop-Länge. Die Loop-Daten aus dem Soundfile sind Defaults (Grundeinstellungen), die immer dann gelten, wenn der Loop-Start-Eingang (**LS**), beziehungsweise der Loop-Length-Eingang (**LL**) nicht mit anderen Modulen verbunden sind.

Falls die **Loop in Release**-Option de-aktiviert ist, wird die Loop-Wiedergabe bei einem **Gate**-Event kleiner oder gleich Null beendet; das Sample läuft je nach Laufrichtung auf den Sample-Beginn oder das Sample-Ende zu und verstummt dann. Bei aktivierter **Loop in Release**-Option, oder bei ausgeschalteter Loop-Wiedergabe haben **Gate**-Events kleiner oder gleich Null keine Auswirkung.

Bei aktivierter **No Stereo**-Option (erreichbar über den Properties-Dialog) werden Stereo-Samples wie Mono-Samples behandelt, das heißt auf dem

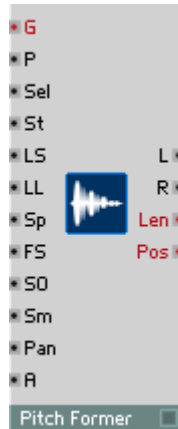
**L-** und auf dem **R-**Ausgang wird ein identisches Signal ausgegeben. **Resynth** verwendet dann nur den linken Kanal von Stereo-Samples. Diese Option steht zur Verfügung, weil Mono-Wiedergabe geringere Anforderungen an die Rechenkapazität stellt.

Alle Eingänge von **Resynth** außer **Gate** sind als Audioeingänge ausgelegt. Die anliegenden Werte werden beim (Re-) Triggern von Samples (also bei positiven **Gate**-Events) übernommen. Während der Sample-Wiedergabe werden Veränderungen der Werte im **Granularity**-Intervall (am **Gr**-Eingang einzustellen) wirksam.

- **G**: Ein positives Event an diesem Eingang startet den Auslesevorgang bei der Position, die am **St**-Eingang eingestellt ist. Bei negativen Werten wird die Loop-Wiedergabe abgebrochen, außer wenn die Option **Loop in Release** aktiv ist.
- **P**: Logarithmischer Audio-Steuereingang für die Wiedergaberate (Pitch). Das Sample wird in Original-Tonhöhe wiedergegeben, wenn **P** gleich dem **Root-Key** des aktuellen Samples ist. **P** bestimmt bei nicht verbundenem **Sel**-Eingang außerdem die Auswahl von Samples aus der Sample-Map.
- **Sel**: Audio-Steuereingang für die Auswahl (Select) eines Samples aus der Sample-Map. Wird dieser Eingang nicht verbunden, werden die am **P**-Eingang anliegenden Werte verwendet.
- **St**: Audio-Steuereingang für den Startpunkt beim nächsten Triggern; in Millisekunden vom Sampleanfang.
- **LS**: Audio-Steuereingang für den Loop-Startpunkt, in Millisekunden vom Sampleanfang. Wird dieser Eingang nicht verbunden, gelten die Loop-Daten aus dem Soundfile. Sind im Soundfile keine Loop-Daten gespeichert, gilt als Default: **LS** = 0.
- **LL**: Audio-Steuereingang für die Loop-Länge in Millisekunden. Wird dieser Eingang nicht verbunden, gelten die Loop-Daten aus dem Soundfile. Sind im Soundfile keine Loop-Daten gespeichert, gilt als Default: **LL** = Länge des Samples. Bei **LL** = 0 kommt die Bewegung im Sample ab Erreichen des Loop-Startpunkts zum Stillstand; der Klang an dieser Stelle wird eingefroren.
- **Sp**: Audio-Steuereingang für die tonhöhen-unabhängige Auslesegeschwindigkeit (Speed). Anliegende Werte werden als Faktor interpretiert: Bei **Sp** = 1 erfolgt die Wiedergabe in Originalgeschwindigkeit, **Sp** = 2 entspricht doppelter Geschwindigkeit, **Sp** = 0 bedeutet Stillstand. Wenn dieser Eingang nicht verbunden ist, gilt als Default die transponierte Originalgeschwindigkeit, so daß – wie bei konventionellen Samplern – die Geschwindigkeit mit fallender Tonhöhe abnimmt.

- **Gr:** Audio-Steuereingang für die Granularität des Re-Synthese-Prozesses in Millisekunden. Dieser Parameter bestimmt die Größe der für die Re-Synthese benutzten Klangpartikel. Bei aktivierter **Signal-Informed Granulation**-Option werden anliegende Werte als Vorgabe verwendet; die tatsächlich verwendeten Werte werden an das Material angepaßt.
- **SO:** Audio-Steuereingang für die Modulation der Sample-Position (Sample Offset) in Millisekunden. Dieser Eingang wird verwendet, um die Position im Sample tonhöhen-unabhängig, beispielsweise über einen Rauschgenerator, zu modulieren.
- **Sm:** Audio-Steuereingang für die *Smoothness* (Gleichmäßigkeit) des Re-Syntheseprozesses. Beeinflußt wird die Formgebung der Klangpartikel. Kleine Werte führen im allgemeinen zu einem rauheren Klangergebnis.
- **Pan:** Audio-Steuereingang für die Position im Stereofeld (-1 = Links, 0 = Mitte, 1 = Rechts).
- **A:** Audio-Steuereingang für die Ausgangs-Amplitude.
- **L:** Audio-Ausgang für den linken Kanal des Re-Synthesizers
- **R:** Audio-Ausgang für den rechten Kanal des Re-Synthesizers. Bei Verarbeitung von Mono-Samples oder bei Aktivierung der **No Stereo**-Option liegt and den Ausgängen **L** und **R** das selbe Signal an.
- **Lng:** Polyphoner Event-Ausgang für die Länge des aktuellen Samples in Millisekunden.
- **Pos:** Polyphoner Event-Ausgang für die aktuelle Position im Sample in Millisekunden.





Echtzeit-Re-Synthesizer für die polyphone Wiedergabe von Mono- oder Stereo- Samples, Sample-Maps oder WaveSets. **Pitch Former** ist ein WaveSet-Synthesizer, in den nicht nur WaveSets, sondern beliebige Samples geladen werden können. **Pitch Former** eliminiert den Tonhöhenverlauf eines Samples und prägt dem Sample eine beliebige, neue Tonhöhe auf. Neben unabhängiger Echtzeit-Kontrolle über die Abspielgeschwindigkeit erlaubt **Pitch Former** eine spektrale Transposition, also eine Tonhöhen-unabhängige Transposition der Klangfarbe.

“Normale” Sampler wie die bekannten Hardware-Sampler oder die Module **Sampler**, **Sampler FM** und **Sampler Loop** in REAKTOR unterhalten pro Stimme einen *Zeiger*, der auf die aktuelle Position im Sample verweist. Am Ausgang des Samplers liegt immer die Amplitude des Samples an der Zeigerposition an. Der Zeiger bewegt sich mehr oder weniger schnell durch das Sample und erzeugt so an den Ausgängen eine zeitveränderliche Amplitude – also eine Schwingung.

Die Geschwindigkeit der Zeigerbewegung bestimmt die Geschwindigkeit der Sample-Wiedergabe (beispielsweise das Tempo eines Beat-Loops). Gleichzeitig bestimmt sie die empfundene Tonhöhe: Je langsamer das im Sample aufgezeichnete Signal abgetastet wird, desto größer werden die Perioden der am Ausgang entstehenden Schwingungen – die Tonhöhe fällt. Kommt der Zeiger zum Stillstand, verändert sich die Ausgangsamplitude gar nicht mehr. Es entsteht kein hörbares Signal.

**Pitch Former** verwendet, genau wie konventionelle Sampler, einen Zeiger auf die aktuelle Sample-Position. Das Signal an den Ausgängen ist jedoch nicht

einfach die Sample-Amplitude an der Zeigerposition. Tatsächlich wird das Ausgangssignal von einem Modul-internen Synthesizer erzeugt, der das Signal an der Zeigerposition *re-synthetisiert*. Die Tonhöhe des von diesem Synthesizer erzeugten Signals ist von der Geschwindigkeit des Zeigers unabhängig. Auch wenn sich der Zeiger im Stillstand befindet, erzeugt der Synthesizer weiter Klang. Während die Tonhöhe des ausgegebenen Signals wie üblich über den **P**-Eingang bestimmt wird, legt der **Sp**-Eingang die Zeigergeschwindigkeit fest.

Während konventionelle Sampler und das Modul **Resynth** REAKTOR durch *Transposition* eines Samples eine *relative* Tonhöhen-Veränderung erreichen, zwingt **Pitch Former** einem Sample eine beliebige, *absolut bestimmte* Tonhöhe auf. **Pitch Former** kann so wie ein REAKTOR-Oscillator verwendet werden. Je nach Art des verarbeiteten Klangmaterials und der verwendeten Einstellungen klingt das Resultat mehr oder weniger “elektronisch” verfremdet. **Pitch Former** erzeugt auch dann Signale mit einer bestimmten Tonhöhe, wenn das verarbeitete Sample keine eindeutige Tonhöhe hat (zum Beispiel Aufnahmen von Becken oder Gong). **Pitch Former** verursacht umso weniger Verfremdungen, je eindeutiger die Grundtonhöhe des verarbeiteten Materials zu bestimmen ist und je geringer die Originaltonhöhe von der erzeugten Tonhöhe abweicht.

Bei konventionellen Samplern und dem Modul **Resynth** in REAKTOR geht die Transposition der Grundtonhöhe mit der Transposition *aller* spektralen Anteile einher. Dies wird oft als Einschränkung empfunden, weil von der Transposition auch diejenigen spektralen Anteile betroffen sind, die sich der Hörerwartung gemäß nicht ändern sollten. Der “Mickey-Mouse-Effekt” beim Verstimmen von Sprachaufnahmen etwa ist auf die Transposition der Formanten zurückzuführen, deren Lage bei einem “realen” menschlichen Sprecher weitgehend unabhängig von der Grundtonhöhe ist. **Pitch Former** entkoppelt in gewissen Grenzen Tonhöhe und Formantlage. Über den Formant-Shift-Eingang (**FS**) läßt sich die Formantlage unabhängig von der Tonhöhe verschieben. Da das Ohr zur Identifikation der Grundtonhöhe alle spektralen Anteile heranzieht, kann es durch Manipulationen dieses Parameters – vor allem bei sehr niedrigen Grundtonhöhen – zu interessanten Vertauschungen von Grundtonhöhe und Klangfarbe kommen. Ähnliche klangliche Effekte werden von Synthesizer-Kennern oft durch *Oscillator-Synchronisation* erzielt.

Die Sample-Verwaltung erfolgt über den Sample Map Editor (Siehe *Sample Map Editor* auf Seite 253.).

Bei einem positiven **Gate**-Event beginnt die Sample-Wiedergabe beim Startpunkt (einstellbar über den **St**-Eingang). Bei deaktiviertem Loop läuft das Sample einmal vom Startpunkt bis zum Ende, oder, bei Auswahl der Laufrichtung **Backward**, vom Startpunkt bis zum Anfang. Bei aktiviertem Loop wird das Sample innerhalb des Loop-Bereichs endlos wiederholt, sobald

die Wiedergabe in diesen Bereich eingetreten ist. Der Loop-Bereich ist durch Daten aus dem Soundfile vorgegeben, aus dem das Sample geladen wurde. Falls das Soundfile keine Loop-Daten enthält, gilt der Sample-Beginn als Loop-Beginn und die Sample-Länge als Loop-Länge. Die Loop-Daten aus dem Soundfile sind *Defaults* (Grundeinstellungen), die immer dann gelten, wenn der Loop-Start-Eingang (**LS**), beziehungsweise der Loop-Length-Eingang (**LL**) nicht mit anderen Modulen verbunden sind.

Falls die **Loop in Release**-Option deaktiviert ist, wird die Loop-Wiedergabe bei einem **Gate**-Event kleiner oder gleich Null beendet; das Sample läuft je nach Laufrichtung auf den Sample-Beginn oder das Sample-Ende zu und verstummt dann. Bei aktivierter **Loop in Release**-Option, oder bei ausgeschalteter Loop-Wiedergabe haben **Gate**-Events kleiner oder gleich Null keine Auswirkung.

Bei aktivierter **No Stereo**-Option (erreichbar über den Properties-Dialog) werden Stereo-Samples wie Mono-Samples behandelt, das heißt auf dem **L**- und auf dem **R**-Ausgang wird ein identisches Signal ausgegeben. **Pitch Former** verwendet dann nur den linken Kanal von Stereo-Samples. Diese Option steht zur Verfügung, weil eine Mono-Wiedergabe geringere Anforderungen an die Rechenkapazität stellt.

Alle Eingänge von **Pitch Former** außer **Gate** sind als Audioeingänge ausgelegt. Die anliegenden Werte werden beim (Re-) Triggern von Samples (also bei positiven **Gate**-Events) übernommen. Während der Sample-Wiedergabe werden Veränderungen der Werte im Intervall der Grundton-Periode (einzustellen über den **P**-Eingang) wirksam.

- **G**: Ein positives Event an diesem Eingang startet den Auslesevorgang bei der Position, die am **St**-Eingang eingestellt ist. Bei negativen Werten wird die Loop-Wiedergabe abgebrochen, außer wenn die Option **Loop in Release** aktiv ist.
- **P**: Logarithmischer Audio-Steuereingang für die Wiedergaberate (Pitch). **P** bestimmt bei nicht verbundenem **Sel**-Eingang außerdem die Auswahl von Samples aus der Sample-Map.
- **Sel**: Audio-Steuereingang für die Auswahl (Select) eines Samples aus der Sample-Map. Wird dieser Eingang nicht verbunden, werden die am **P**-Eingang anliegenden Werte verwendet.
- **St**: Audio-Steuereingang für den Startpunkt beim nächsten Triggern; in Millisekunden vom Sampleanfang.
- **LS**: Audio-Steuereingang für den Loop-Startpunkt, in Millisekunden vom Sampleanfang. Wird dieser Eingang nicht verbunden, gelten die Loop-Daten aus dem Soundfile. Sind im Soundfile keine Loop-Daten gespeichert, gilt als Default: **LS** = 0.

- **LL:** Audio-Steuereingang für die Loop-Länge in Millisekunden. Wird dieser Eingang nicht verbunden, gelten die Loop-Daten aus dem Soundfile. Sind im Soundfile keine Loop-Daten gespeichert, gilt als Default: **LL** = Länge des Samples. Bei **LL** = 0 kommt die Bewegung im Sample ab Erreichen des Loop-Startpunkts zum Stillstand; der Klang an dieser Stelle wird eingefroren.
- **Sp:** Audio-Steuereingang für die Tonhöhen-unabhängige Auslesegeschwindigkeit (Speed). Anliegende Werte werden als Faktor interpretiert: Bei **Sp** = 1 erfolgt die Wiedergabe in Originalgeschwindigkeit, **Sp** = 2 entspricht doppelter Geschwindigkeit, **Sp** = 0 bedeutet Stillstand. Wenn dieser Eingang nicht verbunden ist, gilt als Default die transponierte Originalgeschwindigkeit, so daß – wie bei konventionellen Samplern – die Geschwindigkeit mit dem Fallen der Tonhöhe abnimmt.
- **FS:** Audio-Steuereingang für die Tonhöhen-unabhängige Transposition der Formantlage in Halbtönen.
- **SO:** Audio-Steuereingang für die Modulation der Sample-Position (Sample Offset) in Millisekunden. Dieser Eingang wird verwendet, um die Position im Sample Tonhöhen-unabhängig, beispielsweise über einen Rauschgenerator, zu modulieren.
- **Sm:** Audio-Steuereingang für die *Smoothness* (Gleichmäßigkeit) des Re-Syntheseprozesses. Beeinflußt wird die Formgebung der Klangpartikel. Kleine Werte führen im allgemeinen zu einem rauheren Klangergebnis.
- **Pan:** Audio-Steuereingang für die Position im Stereofeld (-1 = Links, 0 = Mitte, 1 = Rechts).
- **A:** Audio-Steuereingang für die Ausgangs-Amplitude.
- **L:** Audio-Ausgang für den linken Kanal des Re-Synthesizers.
- **R:** Audio-Ausgang für den rechten Kanal des Re-Synthesizers. Bei Verarbeitung von Mono-Samples oder bei Aktivierung der **No Stereo**-Option liegt and den Ausgängen **L** und **R** das selbe Signal an.
- **Lng:** Polyphoner Event-Ausgang für die Länge des aktuellen Samples in Millisekunden.
- **Pos:** Polyphoner Event-Ausgang für die aktuelle Position im Sample in Millisekunden.



Stereo-Multisample-Granular-Synthesizer mit unabhängiger Kontrolle über Tonhöhe **P**, Portamento **PS** (Pitch-Slide), Sampleselektion **Sel**, Sampleposition **Pos** und Länge **Len** jedes Grains. Die Hüllkurve für jedes Grains kann mit Attack **Att** und Decay **Dec** gesteuert werden.

Für jedes Grain kann die Deltazeit **dt** für die Dauer bis zum Start des nächsten Grains eingestellt werden. Die maximale Anzahl gleichzeitig wiedergegebener Grains kann in den Properties bestimmt werden.

Es gibt eine Reihe von Jitter-Eingängen, welche einen Wertebereich für den zugehörigen Eingang festlegen.

Die Sample-Verwaltung erfolgt über den Sample Map Editor (Siehe *Sample Map Editor* auf Seite 253.).

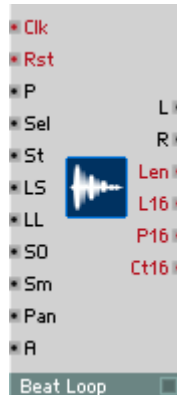
- **Trig:** Eventeingang für das Gatesignal. (G) > 0 startet unverzüglich das nächste Grain.
- **P:** Audioeingang zur logarithmischen Tonhöhenkontrolle (in Halbtönen). Die Tonhöhe ist unabhängig von der Abspielgeschwindigkeit. Das Sample wird in der Originaltonhöhe wiedergegeben, wenn P=Root Key ist. Typ. Bereich: [0...127], Default: 60.

- **D/F:** Audioeingang zur Steuerung der Abspielrichtung, wenn **P** verbunden ist. Andernfalls dient er zur Frequenzsteuerung. Das Sample wird in der Originaltonhöhe wiedergegeben, wenn  $F=1$ , und rückwärts abgespielt, wenn  $F=-1$  ist. Typ. Bereich: [-4...4], Default: 1.
- **PJ:** Audioeingang für zufällige Tonhöhenabweichungen (Pitch Jitter) in Halbtönen. Typ. Bereich: [0...3].
- **PS:** Audioeingang zur logarithmischen Steuerung des Tonhöhenversatzes (Pitch Shift) des aktuellen Grains in Halbtönen. Typ. Bereich: [-3...3].
- **Sel:** Audioeingang zur Multisample-Auswahl (in Halbtönen). Wenn der Eingang nicht verbunden ist, werden die Werte am (**P**) Eingang benutzt. Typ. Bereich: [0...127].
- **Pos:** Audioeingang zur Einstellung der Position in der Audiodatei in ms. Bereich [0...<Samplelänge in ms>].
- **PsJ:** Audioeingang zur Steuerung der zufälligen Positionsabweichung (Position Jitter) in ms. Typ. Bereich: [0...<Samplelänge in ms>]
- **Len:** Audioeingang zur Einstellung der Grain-Länge in ms. Typ. Bereich: [10...100]. Default: 20 ms.
- **LnJ:** Audioeingang für die Steuerung der zufälligen Längenabweichung (Length Jitter) in ms. Typ. Bereich: [10...100]. Default: 0 ms.
- **Att:** Audioeingang zur Einstellung der Attackzeit. Bereich: [0...1]. Default: 0.2.
- **Dec:** Audioeingang zur Einstellung der Decayzeit. Bereich: [0...1]. Default: 0.2.
- **Dist:** Audioeingang zur Einstellung der Deltazeit für die Dauer bis zum Start des nächsten Grains in ms. Typ. Bereich: [5...100]. Default: 20.
- **DisJ:** Audioeingang für die Steuerung der zufälligen Deltazeitabweichung (Delta time Jitter). Typ. Bereich: [10...100]. Default: 20 ms.
- **Pan:** Audioeingang zur Einstellung der Position im Stereofeld. Bereich: [-1(Links)...1(Rchts)].
- **PnJ:** Audioeingang zur Steuerung der zufälligen Stereopositionsabweichung (Pan Jitter). Bereich: [0...1].
- **A:** Audioeingang zur Amplitudensteuerung. Typ. Bereich: [0...1]. Default: 1.
- **L:** Polyphoner Ausgang für den linken Stereokanal. Typ. Bereich: [-1...1].

- **R:** Polyphoner Ausgang für den rechten Stereokanal.  
Typ. Bereich: [-1...1].
- **Lng:** Eventausgang für die Samplelänge in ms.  
Typ. Bereich: [0...<Samplelänge in ms>].
- **GTr:** Eventausgang für das Auslösen eines Grains. Gibt 1 aus, wenn ein Grain startet, 0, wenn es stoppt.

## Beat Loop

## Sampler



Echtzeit-Re-Synthesizer für die synchronisierte Wiedergabe von Beatloop-Samples. Die Transposition der Beatloops kann über den **P**-Eingang unabhängig von der Wiedergabegeschwindigkeit eingestellt werden. **Beat Loop** synchronisiert sich zu einer über den **C**-Eingang verbundenen 96-tel-Clock-Quelle, normalerweise nimmt man dazu das **1/96 Clock**-Modul. So kann man leicht alle **Beat Loops** in REAKTOR – unabhängig vom internen Tempo der Samples – zueinander synchronisieren. **Beat Loop** ermöglicht außerdem die einfache Kopplung von REAKTOR-internen Sequencer-Modulen und von MIDI-Clock an rhythmisches Sample-Material.

Die Sample-Verwaltung erfolgt über den Sample Map Editor (Siehe *Sample Map Editor* auf Seite 253.).

**Beat Loop** erwartet exakt geschnittene Samples, die 2, 4, 8, 16, oder 32 etc. Schläge enthalten. Das Tempo der verwendeten Beat Loops sollte zwischen 87 und 174 BPM liegen. Erfüllen die verwendeten Samples diese Bedingungen, kann **Beat Loop** die Samples auch bei veränderter Wiedergabegeschwindigkeit in guter Qualität wiedergeben. Durch Aktivierung der über den Properties-Dialog erreichbaren, Sample-bezogenen **Pitched Sound**-Option können eventuelle

Tonhöhen-Verfälschungen von Bassläufen u. ä. vermieden werden; dabei sind aber Einbußen hinsichtlich der rhythmischen Präzision hinzunehmen.

Über den Loop-Start-Eingang (**LS**) und den Loop-Length-Eingang (**LL**) wird der Loop-Bereich des Samples eingestellt. Sind **LS** und **LL** nicht beschaltet, wird das gesamte Sample im Loop wiedergegeben. Durch positive **Rst**-Events wird die Sample-Wiedergabe beim Startpunkt (einstellbar über den **St**-Eingang) fortgesetzt.

Bei aktivierter **No Stereo**-Option (erreichbar über den Properties-Dialog) werden Stereo-Samples wie Mono-Samples behandelt, das heißt auf dem **L**- und auf dem **R**-Ausgang wird ein identisches Signal ausgegeben. **Beat Loop** verwendet dann nur den linken Kanal von Stereo-Samples. Diese Option steht zur Verfügung, weil Mono-Wiedergabe geringere Anforderungen an die Rechenkapazität stellt.

Alle Eingänge von **Beat Loop** außer **C** und **Rst** sind als Audioeingänge ausgelegt. Während der Sample-Wiedergabe werden Veränderungen der Werte im Intervall von Sechzehntel-Notenwerten wirksam.

- **C**: Ein positives Event an diesem Eingang schaltet das **Beat Loop**-Modul um einen 96-tel Notenwert weiter.
- **Rst**: Ein positives Event an diesem Eingang setzt die Wiedergabe auf die am **St**-Eingang eingestellte Position zurück.
- **P**: Logarithmischer Audio-Steuereingang für die Wiedergaberate (Pitch). **P** bestimmt bei nicht verbundenem **Sel**-Eingang außerdem die Auswahl von Samples aus der Sample-Map.
- **Sel**: Audio-Steuereingang für die Auswahl (Select) eines Samples aus der Sample-Map. Wird dieser Eingang nicht verbunden, werden die am **P**-Eingang anliegenden Werte verwendet.
- **St**: Audio-Steuereingang für den Startpunkt beim nächsten Triggern; in Sechzehntel-Notenwerten vom Sample-Beginn aus gerechnet.
- **LS**: Audio-Steuereingang für den Loop-Startpunkt, in Sechzehntel-Notenwerten. Wird dieser Eingang nicht verbunden, gilt als Default: **LS** = 0.
- **LL**: Audio-Steuereingang für die Loop-Länge in Sechzehntel-Notenwerten. Wird dieser Eingang nicht verbunden, gilt als Default: **LL** = Länge des Samples.
- **SO**: Audio-Steuereingang für die Modulation der Sample-Position (Sample Offset) in Sechzehntel-Notenwerten. Dieser Eingang wird verwendet, um Sprünge im Sample zu erreichen, die beispielsweise über ein Sequencer-Modul gesteuert werden können.



- **Sm:** Audio-Steuereingang für die *Smoothness* (Gleichmäßigkeit) des Re-Syntheseprozesses. Beeinflußt wird die Formgebung der Klangpartikel. Sehr kleine Werte führen im allgemeinen zu „Knacksen“ in Sechzehntel-Intervallen.
- **Pan:** Audio-Steuereingang für die Position im Stereofeld (-1 = Links, 0 = Mitte, 1 = Rechts).
- **A:** Audio-Steuereingang für die Ausgangs-Amplitude.
- **L:** Audio-Ausgang für den linken Kanal des Re-Synthesizers.
- **R:** Audio-Ausgang für den rechten Kanal des Re-Synthesizers. Bei Verarbeitung von Mono-Samples oder bei Aktivierung der **No Stereo**-Option liegt an den Ausgängen **L** und **R** das selbe Signal an.
- **Lng:** Polyphoner Event-Ausgang für die Länge des aktuellen Samples in Millisekunden.
- **L16:** Polyphoner Event-Ausgang für die Länge des aktuellen Samples in Sechzehntel-Notenwerten.
- **P16:** Polyphoner Event-Ausgang für die aktuelle Position im Sample in Sechzehntel-Notenwerten.
- **Ct16:** Polyphoner Event-Ausgang für eine Zähler (Count) der Sechzehntel-Notenwerte, die seit dem Start / Reset vergangen sind.



Dieses Modul stellt ein Sample als Funktionswert-Tabelle (look-up table) zur Verfügung. Über den **Pos**-Eingang wird eine Position im Sample in Millisekunden bestimmt. An den Ausgängen liegt der Wert des Samples an dieser Stelle an. Soundfiles werden über den **Load Audio...**-Eintrag des Kontext-Menüs geladen.

Die Sample-Verwaltung erfolgt über den Sample Map Editor (Siehe *Sample Map Editor* auf Seite 253.).

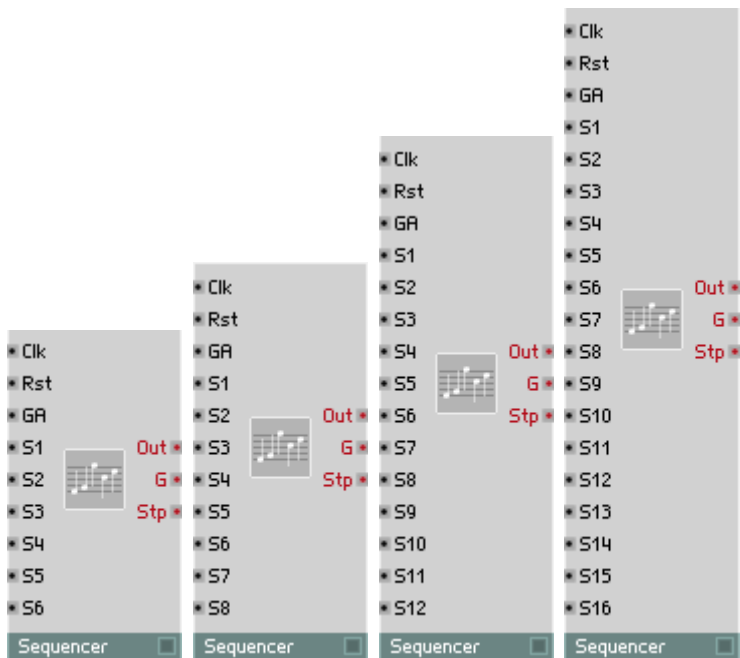
- **Pos**: Audio-Eingang für die Position im Sample in Millisekunden.
- **A**: Audio-Eingang für die Amplituden-Modulation.
- **L**: Audio-Ausgang für den linken Kanal des Samples.
- **R**: Audio-Ausgang für den rechten Kanal des Samples. Bei Verarbeitung von Mono-Samples oder bei Aktivierung der **No Stereo**-Option liegt an den Ausgängen **L** und **R** das selbe Signal an.
- **Lng**: Polyphoner Event-Ausgang für die Länge des aktuellen Samples in Millisekunden.

# Sequencer

REAKTOR verfügt über Step-Sequencer mit Gate-Ausgang in vier verschiedenen Grössen sowie ein Modul, welches mittels eines Positionswertes einen Eingang für die Signalverarbeitung auswählt.

## Sequencer

## Sequencer



---

## 6-Step

## Sequencer

Sequencer mit 6 Schritten. Der Ausgangswert bei jedem Schritt (z. B. für die Tonhöhe) läßt sich unabhängig von den anderen einstellen. Zusätzlich wird bei jedem Schritt am Ausgang ein Gate-Signal mit dem aktuellen Eingangswert als Amplitude ausgegeben.

- **C:** Audio-Steuereingang für das Takt-Signal (Clock). Ein positiver Nulldurchgang schaltet zum nächsten Schritt. Typischerweise schließt man hier einen Pulse Oscillator oder das MIDI Sync Modul an.
- **Rst:** Audio-Steuereingang für das Initialisierungs-Signal (Reset). Positiver Nulldurchgang setzt den Sequencer zurück zum ersten Schritt. Typischerweise schließt man hier einen Button oder MIDI Start Modul an.
- **GA:** Audio-Steuereingang für die Amplitude des Gate-Ausgangssignals. Wenn der Wert 0 beträgt oder der Eingang nicht verbunden ist, erscheint am Gate-Ausgang kein Signal.
- **S1:** Audio-Steuereingang für den Wert des ersten Schritts.
- **S2:** Audio-Steuereingang für den Wert des zweiten Schritts.
- **S3:** Audio-Steuereingang für den Wert des dritten Schritts.
- **S4:** Audio-Steuereingang für den Wert des vierten Schritts.
- **S5:** Audio-Steuereingang für den Wert des fünften Schritts.
- **S6:** Audio-Steuereingang für den Wert des sechsten Schritts.
- **Out:** Eventsignal-Ausgang für die Sequencer-Schritte.
- **G:** Eventsignal-Ausgang für das Gate-Signal.
- **St:** Eventsignal-Ausgang für den Schritt, zu dem geschaltet wird.

---

## 8-Step

## Sequencer

Wie **6-Step** Sequencer, aber mit 8 Schritten.

---

## 12-Step

## Sequencer

Wie **6-Step** Sequencer, aber mit 12 Schritten.

---

## 16-Step

## Sequencer

Wie **6-Step** Sequencer, aber mit 16 Schritten.



Gesteuerter Schalter für die Auswahl von Werten, z.B. zur Nutzung als Sequencer. Ein Event am **Pos**-Eingang adressiert mit seinem Wert einen der 16 Eingänge. Der am entsprechenden Eingang aktuell anliegende Wert wird an den Ausgang weitergegeben.

Wenn der **Pos**-Eingang von einem Signal gespeist wird, welches schrittweise ansteigt, sendet der Ausgang **Out** eine Sequenz der jeweiligen Werte an den Inputs **0...15**. Die Werte am **Pos**-Eingang sind in den Zahlenbereich  $[0 \dots (\text{Len} - 1)]$  „gefaltet“, so daß man Sequenzen mit variabler Länge (**Len**) erzeugen kann.

Der **Pos**-Eingang kann auch von einem Bedienelement, einem **Beat Loop**-Modul, einer Zufalls-Eventquelle oder der Masterclock (**Song Pos**-Modul) gesteuert werden. **Multiplex 16** ist das Modul, welches vorher als **Demux** bzw. Select 16 bekannt war.

- **Pos**: Eventsteuereingang für die Auswahl. Jedes Event an diesem Eingang erzeugt ein Event an den Ausgängen. Um **Multiplex 16** als Sequencer zu benutzen, wird hier als Steuerwert die Zahl von 16tel Noten eingegeben, welche seit Start/Reset vergangen sind.

- **Len**: Eingang zur Festlegung der Sequenzlänge. Range: [ 1 ... 16 ].
- **0-15**: Audioeingang zur Steuerung des Wertes des entsprechenden Schritts.
- **Stp**: Aktuelle Sequenzerschnitt-Nummer. Die Zahl an diesem Ausgang wird als „**Pos** modulo **Len**“ berechnet. Range: [ 0...**Len** ].
- **Bar**: Aktuelle Taktzahl. Die Zahl an diesem Ausgang wird als „Integer(**Pos** / **Len**)“ berechnet.
- **Out**: Der Ausgangswert des aktuellen Sequenzerschnitt.

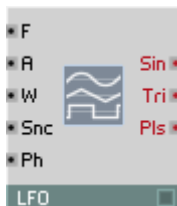
# LFO, Envelope

Diese Kategorie einen vollständig modulierbaren LFO mit verschiedenen Wellenformen, einen Zufallsgenerator und Hüllkurvengeneratoren jeglicher Art, einschliesslich Ramp-Envelopes mit mehreren Time-/Level-Parametern in drei Grössen.

Alle Hüllkurven (Envelopes) können optional ihren Verlauf in einer Grafik im Panel darstellen. Dies kann mit der Option **Visible** auf der **Appearance-Seite** in den Modul-Properties aktiviert werden. Die Größe der Darstellung kann in den Properties mit **Size X** und **Size Y** eingestellt werden. Die Zeitachse der Kurve ist nicht maßstabsgetreu.

## LFO

## LFO, Envelope



Low Frequency Oscillator (Niederfrequenz-Oscillator) mit Rechteck-, Dreieck- und Sinus-Wellen-Ausgängen. Er wird typischerweise eingesetzt als Quelle für Modulationssignale (für Vibrato, Tremolo usw.). Das Ausgangssignal ist ein Strom von Events mit der Rate die im **Settings** Menü als **Control Rate** eingestellt ist. Da der LFO mit Control Rate arbeitet, ist er viel effizienter und belastet den Prozessor weitaus weniger als ein Audio-Oscillator, der dieselbe Aufgabe übernehmen könnte.

- **F**: Event-Eingang zur Steuerung der Oscillator-Frequenz in Hz. Zur Angabe des Tempos in BPM (Beats per Minute = Taktschläge pro Minute) berechnet man den Wert in Hz wie folgt: *Schwingungen-pro-Taktschlag* x BPM/60. Für 3 Schwingungen pro Takt im 4/4-Takt (das sind  $\frac{3}{4}$  Schwingungen pro Taktschlag) bekommt man z. B.  
 $F = (\frac{3}{4})/60 \times \text{BPM}$  oder  $0.0125 \times \text{BPM}$ .
- **A**: Steuereingang für die Amplitude. Das Ausgangssignal bewegt sich zwischen **+A** und **-A**.
- **W**: Event-Eingang zur Steuerung der Pulsbreite, d. h. dem Verhältnis der Dauer der An-Phase zur Aus-Phase der Rechteck-Welle. Die anderen Wellenformen werden ebenso entsprechend verschoben. Der Wertebereich ist -1 bis 1. Symmetrische Signalform bei **W** = 0.

- **Snc**: Event-Eingang zur Synchronisierung der LFO-Wellenform. Ein positives Event synchronisiert den Oscillator, indem die Phase auf den Wert gesetzt wird, der am **Ph**-Eingang anliegt.
- **Ph**: Eingang zur Bestimmung der Phase (Position in der Wellenform) auf die der Oscillator zum Synchronisationszeitpunkt zurückgesetzt wird. **Ph** = 0: Phase = 0° (Mitte der ansteigenden Flanke), **Ph** = 0.5: Phase = 180° (Mitte der fallenden Flanke), **Ph** = 1: Phase = 360° (das selbe wie 0°).
- **Par**: Event-Ausgang für das Signal mit Sinus-Wellenform.
- **Tri**: Event-Ausgang für das Signal mit Dreieck-Wellenform.
- **Pls**: Event-Ausgang für das Signal mit Rechteck-Wellenform.

## Slow Random

## LFO, Envelope



Low Frequency Oscillator (Niederfrequenz-Oscillator), der eine schrittweise Zufallswellenform ausgibt.

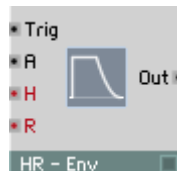
- **F**: Steuereingang für die Schritt-Frequenz in Hz.
- **A**: Steuereingang für die Amplitude. Das Ausgangssignal bewegt sich zwischen **+A** und **-A**.
- **Out**: Event-Ausgang für das Zufallswert-Signal.





Hüllkurve mit Hold-Charakteristik. Wenn die Hüllkurve ausgelöst wird, springt der Ausgangswert auf den aktuellen Wert des Amplitudeneingangs und bleibt dort, bis die Haltezeit abgelaufen ist. Danach springt der Ausgang auf 0 zurück. Die Hüllkurve kann jederzeit, auch während des Haltens, neu ausgelöst werden.

- **T:** Audio-Steuereingang für die Auslösung (Trigger), wenn der Wert von 0 in positive Richtung geht.
- **A:** Audio-Steuereingang für den Ausgangswert während der Haltezeit.
- **H:** Logarithmischer Event-Steuereingang für die Haltedauer (Hold Time). 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (also in  $\text{dB}_{\text{1ms}}$ ).
- **Out:** Audio-Ausgang für das Hüllkurvensignal.



Hüllkurven-Generator mit Hold-Release-Charakteristik. Wenn die Hüllkurve ausgelöst wird, springt der Ausgangswert auf den aktuellen Wert des Amplitudeneingangs und bleibt dort, bis die Haltezeit abgelaufen ist. Danach klingt der Ausgang gemäß der Release-Zeit exponentiell auf Null ab.

- **T:** Audio-Steuereingang für die Auslösung (Trigger), wenn der Wert von 0 in positive Richtung geht
- **A:** Audio-Steuereingang für den Ausgangswert während der Halte- und Release-Zeit.
- **H:** Logarithmischer Event-Steuereingang für die Haltedauer (hold time). 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (also in  $\text{dB}_{\text{1ms}}$ ).
- **R:** Logarithmischer Event-Steuereingang für die Release-Zeit. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (also in  $\text{dB}_{\text{1ms}}$ ).
- **Out:** Audio-Ausgang für das Hüllkurvensignal.



Hüllkurven-Generator mit Decay-Charakteristik. Wenn die Hüllkurve ausgelöst wird, springt der Ausgangswert auf den aktuellen Wert des Amplitudeneingangs und klingt dann gemäß der Decay-Zeit exponentiell auf Null ab.

- **T:** Audio-Steuereingang für die Auslösung (Trigger), wenn der Wert von 0 in positive Richtung geht
- **A:** Audio-Steuereingang für den Ausgangswert.
- **D:** Logarithmischer Event-Steuereingang für die Decay-Zeit. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (also in  $\text{dB}_{1\text{ms}}$ ).
- **Out:** Audio-Ausgang für das Hüllkurvensignal.



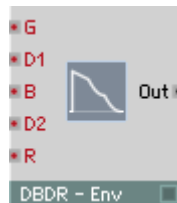
Hüllkurven-Generator mit Decay-Release-Charakteristik. Wenn die Hüllkurve mit einem Gate-Event ausgelöst wird, springt der Ausgangswert auf die Amplitude des Gate-Events und klingt dann gemäß der Decay-Zeit exponentiell auf 0 ab. Ein Gate-Event mit Amplitude 0 (bei Note-Off) setzt die Abklingzeit auf die Release-Zeit.

- **G:** Event-Steuereingang für das Gate-Signal, das die Hüllkurve auslöst (Trigger). Die Amplitude des Gate-Signals bestimmt den anfänglichen Ausgangswert.
- **D:** Logarithmischer Event-Steuereingang für die Decay-Zeit. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (also in  $\text{dB}_{1\text{ms}}$ ).
- **R:** Logarithmischer Event-Steuereingang für die Release-Zeit. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (also in  $\text{dB}_{1\text{ms}}$ ).
- **Out:** Audio-Ausgang für das Hüllkurvensignal.



Hüllkurven-Generator mit Decay-Sustain-Release-Charakteristik. Wenn die Hüllkurve mit einem Gate-Event ausgelöst wird, springt der Ausgangswert auf die Amplitude des Gate-Events, klingt dann gemäß der Decay-Zeit exponentiell auf den Sustain-Pegel (multipliziert mit der Amplitude) ab und wird dort gehalten. Nach Empfang eines Gate-Events mit Amplitude 0 (bei Note-Off) sinkt der Ausgangswert gemäß der Release-Zeit exponentiell auf 0 ab.

- **G:** Event-Steuereingang für das Gate-Signal, das die Hüllkurve auslöst (Trigger). Die Amplitude des Gate-Signals bestimmt den anfänglichen Ausgangswert.
- **D:** Logarithmischer Event-Steuereingang für die Decay-Zeit. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (also in  $\text{dB}_{1\text{ms}}$ ).
- **S:** Event-Steuereingang für den Sustain-Pegel. Typischer Wertebereich ist 0 (Abklingen auf 0) bis 1 (auf Anfangspegel halten), aber Sustain kann auch größer als 1 oder sogar kleiner als 0 sein.
- **R:** Logarithmischer Event-Steuereingang für die Release-Zeit. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (also in  $\text{dB}_{1\text{ms}}$ ).
- **Out:** Audio-Ausgang für das Hüllkurvensignal.



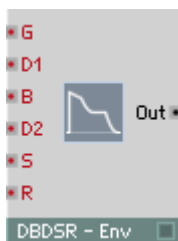
Hüllkurven-Generator mit Decay-Breakpoint-Release-Charakteristik. Wenn die Hüllkurve mit einem Gate-Event ausgelöst wird, springt der Ausgangswert auf die Amplitude des Gate-Events und klingt dann mit der Decay-1-Zeit exponentiell ab, bis er den Breakpoint-Pegel (multipliziert mit der Amplitude) erreicht hat, worauf er mit der Decay-2-Zeit weiter abklingt. Nach Empfang

eines Gate-Events mit Amplitude 0 (bei Note-Off) sinkt der Ausgangswert gemäß der Release-Zeit exponentiell auf 0 ab.

- **G:** Event-Steuereingang für das Gate-Signal, das die Hüllkurve auslöst (Trigger). Die Amplitude des Gate-Signals bestimmt den anfänglichen Ausgangswert.
- **D1:** Logarithmischer Event-Steuereingang für die erste Decay-Zeit. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (also in  $\text{dB}_{1\text{ms}}$ ).
- **B:** Event-Steuereingang für den Breakpoint-Pegel. Wertebereich 0 (nie auf Decay-2 schalten) bis 1 (sofort auf Decay-2 schalten).
- **D2:** Logarithmischer Event-Steuereingang für die 2. Decay-Zeit. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (also in  $\text{dB}_{1\text{ms}}$ ).
- **R:** Logarithmischer Event-Steuereingang für die Release-Zeit. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (also in  $\text{dB}_{1\text{ms}}$ ).
- **Out:** Audio-Ausgang für das Hüllkurvensignal.

## DBDSR-Env

## LFO, Envelope



Hüllkurven-Generator mit Decay-Breakpoint-Decay-Sustain-Release-Charakteristik. Wenn die Hüllkurve mit einem Gate-Event ausgelöst wird, springt der Ausgangswert auf die Amplitude des Gate-Events und klingt dann mit der Decay-1-Zeit exponentiell ab, bis er den Breakpoint-Pegel (multipliziert mit der Amplitude) erreicht hat, worauf er mit der Decay-2-Zeit weiter auf den Sustain-Pegel (multipliziert mit der Amplitude) abklingt. Dort wird er gehalten. Nach Empfang eines Gate-Events mit Amplitude 0 (bei Note-Off) sinkt der Ausgangswert gemäß der Release-Zeit exponentiell auf 0 ab.

- **G:** Event-Steuereingang für das Gate-Signal, das die Hüllkurve auslöst (Trigger). Die Amplitude des Gate-Signals bestimmt den anfänglichen Ausgangswert.
- **D1:** Logarithmischer Event-Steuereingang für die erste Decay-Zeit. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (also in  $\text{dB}_{1\text{ms}}$ ).

- **B:** Event-Steuereingang für den Breakpoint-Pegel. Wertebereich 0 (nie auf Decay-2 schalten) bis 1 (sofort auf Decay-2 schalten).
- **D2:** Logarithmischer Event-Steuereingang für die 2. Decay-Zeit. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (also in  $\text{dB}_{1\text{ms}}$ ).
- **S:** Event-Steuereingang für den Sustain-Pegel. Typischer Wertebereich ist 0 (Abklingen auf 0) bis 1 (auf Anfangspegel halten), aber Sustain kann auch größer als 1 oder sogar kleiner als 0 sein.
- **R:** Logarithmischer Event-Steuereingang für die Release-Zeit. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (also in  $\text{dB}_{1\text{ms}}$ ).
- **Out:** Audio-Ausgang für das Hüllkurvensignal.

## AD-Env

## LFO, Envelope



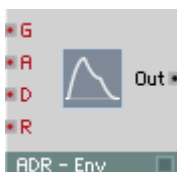
Hüllkurven-Generator mit Attack-Decay-Charakteristik. Wenn die Hüllkurve mit einer positiven Flanke am Trigger-Eingang ausgelöst wird, steigt der Ausgangswert während der Attack-Zeit linear auf die Amplitude des A-Eingangs. Der Ausgang klingt dann gemäß der Decay-Zeit exponentiell auf 0 ab.

- **T:** Audio-Steuereingang für die Auslösung (Trigger), wenn der Wert von 0 in positive Richtung geht
- **A:** Audio-Steuereingang für den Spitzenwert.
- **A:** Logarithmischer Event-Steuereingang für die Attack-Zeit. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (also in  $\text{dB}_{1\text{ms}}$ ).
- **D:** Logarithmischer Event-Steuereingang für die Decay-Zeit. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (also in  $\text{dB}_{1\text{ms}}$ ).
- **Out:** Audio-Ausgang für das Hüllkurvensignal.



Hüllkurven-Generator mit Attack-Release-Charakteristik. Wenn die Hüllkurve mit einem Gate-Event ausgelöst wird, steigt der Ausgangswert während der Attack-Zeit linear auf die Amplitude des Gate-Events. Der Ausgang wird dann auf dem Maximalwert gehalten, bis er nach einem Gate-Event mit Amplitude 0 (Note-Off) gemäß der Release-Zeit exponentiell auf 0 absinkt.

- **G:** Event-Steuereingang für das Gate-Signal, das die Hüllkurve auslöst (Trigger). Die Amplitude des Gate-Signals bestimmt den maximalen Ausgangswert.
- **A:** Logarithmischer Event-Steuereingang für die Attack-Zeit. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (also in  $\text{dB}_{1\text{ms}}$ ).
- **R:** Logarithmischer Event-Steuereingang für die Release-Zeit. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (also in  $\text{dB}_{1\text{ms}}$ ).
- **Out:** Audio-Ausgang für das Hüllkurvensignal.



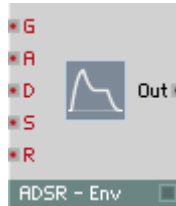
Hüllkurven-Generator mit Attack-Decay-Release-Charakteristik. Wenn die Hüllkurve mit einem Gate-Event ausgelöst wird, steigt der Ausgangswert während der Attack-Zeit linear auf die Amplitude des Gate-Events an und klingt dann mit der Decay-Zeit linear auf 0 ab. Nach einem Gate-Event mit Amplitude 0 (Note-Off) sinkt der Pegel in der Release-Zeit exponentiell auf 0.

- **G:** Event-Steuereingang für das Gate-Signal, das die Hüllkurve auslöst (Trigger). Die Amplitude des Gate-Signals bestimmt den maximalen Ausgangswert.
- **A:** Logarithmischer Event-Steuereingang für die Attack-Zeit. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (also in  $\text{dB}_{1\text{ms}}$ ).

- **D:** Logarithmischer Event-Steuereingang für die Decay-Zeit. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (also in  $\text{dB}_{1\text{ms}}$ ).
- **R:** Logarithmischer Event-Steuereingang für die Release-Zeit. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (also in  $\text{dB}_{1\text{ms}}$ ).
- **Out:** Audio-Ausgang für das Hüllkurvensignal.

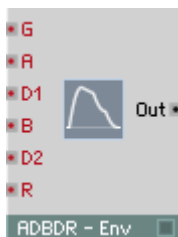
## ADSR-Env

## LFO, Envelope



Hüllkurven-Generator mit Attack-Hold-Decay-Sustain-Release-Charakteristik. Wenn die Hüllkurve mit einem Gate-Event ausgelöst wird, steigt der Ausgangswert während der Attack-Zeit linear auf die Amplitude des Gate-Events und klingt dann mit der Decay-Zeit exponentiell auf den Sustain-Pegel (multipliziert mit der Amplitude) ab. Der Ausgang wird auf dem Sustain-Pegel gehalten, bis er nach einem Gate-Event mit Amplitude 0 (Note-Off) gemäß der Release-Zeit exponentiell auf Null absinkt.

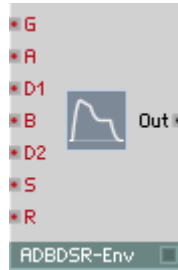
- **G:** Event-Steuereingang für das Gate-Signal, das die Hüllkurve auslöst (Trigger). Die Amplitude des Gate-Signals bestimmt den maximalen Ausgangswert.
- **A:** Logarithmischer Event-Steuereingang für die Attack-Zeit. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (also in  $\text{dB}_{1\text{ms}}$ ).
- **D:** Logarithmischer Event-Steuereingang für die Decay-Zeit. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (also in  $\text{dB}_{1\text{ms}}$ ).
- **S:** Event-Steuereingang für den Sustain-Pegel. Typischer Wertebereich ist 0 (Abklingen auf 0) bis 1 (am Ende der Attack-Phase halten), aber Sustain kann auch größer als 1 oder sogar kleiner als 0 sein.
- **R:** Logarithmischer Event-Steuereingang für die Release-Zeit. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (also in  $\text{dB}_{1\text{ms}}$ ).
- **Out:** Audio-Ausgang für das Hüllkurvensignal.



Hüllkurven-Generator mit Attack-Hold-Decay-Sustain-Release-Charakteristik. Wenn die Hüllkurve mit einem Gate-Event ausgelöst wird, steigt der Ausgangswert während der Attack-Zeit linear auf die Amplitude des Gate-Events und klingt dann mit der Decay-1-Zeit exponentiell ab, bis er den Breakpoint-Pegel (multipliziert mit der Amplitude) erreicht hat, worauf dieser gemäß der Decay-2-Zeit weiter auf 0 absinkt. Nach Empfang eines Gate-Events mit Amplitude 0 (bei Note-Off) sinkt der Ausgangswert mit der Release-Zeit exponentiell auf 0 ab.

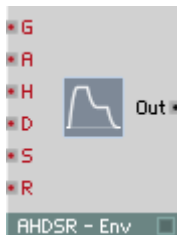
- **G:** Event-Steuereingang für das Gate-Signal, das die Hüllkurve auslöst (Trigger). Die Amplitude des Gate-Signals bestimmt den maximalen Ausgangswert.
- **A:** Logarithmischer Event-Steuereingang für die Attack-Zeit. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (also in  $\text{dB}_{1\text{ms}}$ ).
- **D1:** Logarithmischer Event-Steuereingang für die 1. Decay-Zeit. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (also in  $\text{dB}_{1\text{ms}}$ ).
- **B:** Event-Steuereingang für den Breakpoint-Pegel. Wertebereich 0 (nie auf Decay-2 schalten) bis 1 (sofort auf Decay-2 schalten).
- **D2:** Logarithmischer Event-Steuereingang für die 2. Decay-Zeit. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (also in  $\text{dB}_{1\text{ms}}$ ).
- **R:** Logarithmischer Event-Steuereingang für die Release-Zeit. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (also in  $\text{dB}_{1\text{ms}}$ ).
- **Out:** Audio-Ausgang für das Hüllkurvensignal.





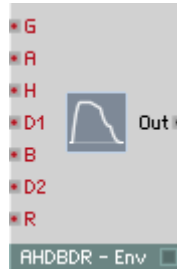
Hüllkurven-Generator mit Attack-Hold-Decay-Sustain-Release-Charakteristik. Wenn die Hüllkurve mit einem Gate-Event ausgelöst wird, steigt der Ausgangswert während der Attack-Zeit linear auf die Amplitude des Gate-Events und klingt dann mit der ersten Decay-Zeit linear auf den Breakpoint-Pegel ab. Von dort geht der Verlauf mit der zweiten Decay-Zeit exponentiell bis zum Sustain-Pegel weiter. (Die Pegel sind mit der Amplitude skaliert). Der Ausgang wird auf dem Sustain-Pegel gehalten, bis er nach einem Gate-Event mit Wert 0 (Note-Off) gemäß der Release-Zeit exponentiell auf Null absinkt.

- **G:** Event-Steuereingang für das Gate-Signal, das die Hüllkurve auslöst (Trigger). Die Amplitude des Gate-Signals bestimmt den maximalen Ausgangswert.
- **A:** Logarithmischer Event-Steuereingang für die Attack-Zeit. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (also in  $\text{dB}_{1\text{ms}}$ ).
- **D1:** Logarithmischer Event-Steuereingang für die erste Decay-Zeit. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (also in  $\text{dB}_{1\text{ms}}$ ).
- **B:** Event-Steuereingang für den Breakpoint-Pegel. Typischer Wertebereich ist 0 bis 1.
- **D2:** Logarithmischer Event-Steuereingang für die zweite Decay-Zeit. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (also in  $\text{dB}_{1\text{ms}}$ ).
- **S:** Event-Steuereingang für den Sustain-Pegel. Typischer Wertebereich ist 0 bis 1.
- **R:** Logarithmischer Event-Steuereingang für die Release-Zeit. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (also in  $\text{dB}_{1\text{ms}}$ ).
- **Out:** Audio-Ausgang für das Hüllkurvensignal.



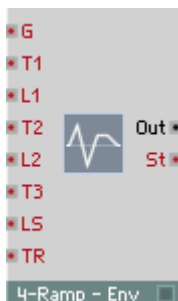
Hüllkurven-Generator mit Attack-Hold-Decay-Sustain-Release-Charakteristik. Wenn die Hüllkurve mit einem Gate-Event ausgelöst wird, steigt der Ausgangswert während der Attack-Zeit linear auf die Amplitude des Gate-Events und bleibt dort, bis die Haltezeit abgelaufen ist, klingt dann mit der Decay-Zeit exponentiell auf den Sustain-Pegel (multipliziert mit der Amplitude) ab. Der Ausgang wird auf dem Sustain-Pegel gehalten, bis er nach einem Gate-Event mit Amplitude 0 (Note-Off) gemäß der Release-Zeit exponentiell auf Null absinkt.

- **G:** Event-Steuereingang für das Gate-Signal, das die Hüllkurve auslöst (Trigger). Die Amplitude des Gate-Signals bestimmt den maximalen Ausgangswert.
- **A:** Logarithmischer Event-Steuereingang für die Attack-Zeit. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (also in  $\text{dB}_{1\text{ms}}$ ).
- **H:** Logarithmischer Event-Steuereingang für die Haltedauer (hold time). 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (also in  $\text{dB}_{1\text{ms}}$ ).
- **D:** Logarithmischer Event-Steuereingang für die Decay-Zeit. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (also in  $\text{dB}_{1\text{ms}}$ ).
- **S:** Event-Steuereingang für den Sustain-Pegel. Typischer Wertebereich ist 0 (Abklingen auf 0) bis 1 (am Ende der Attack-Phase halten), aber Sustain kann auch größer als 1 oder sogar kleiner als 0 sein.
- **R:** Logarithmischer Event-Steuereingang für die Release-Zeit. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (also in  $\text{dB}_{1\text{ms}}$ ).
- **Out:** Audio-Ausgang für das Hüllkurvensignal.



Hüllkurven-Generator mit Attack-Hold-Decay-Sustain-Release-Charakteristik. Wenn die Hüllkurve mit einem Gate-Event ausgelöst wird, steigt der Ausgangswert während der Attack-Zeit linear auf die Amplitude des Gate-Events und bleibt dort, bis die Haltezeit abgelaufen ist, klingt dann mit der Decay-1-Zeit exponentiell ab, bis er den Breakpoint-Pegel (multipliziert mit der Amplitude) erreicht hat, worauf dieser gemäß der Decay-2-Zeit weiter auf 0 absinkt. Nach Empfang eines Gate-Events mit Amplitude 0 (bei Note-Off) sinkt der Ausgangswert mit der Release-Zeit exponentiell auf 0 ab.

- **G:** Event-Steuereingang für das Gate-Signal, das die Hüllkurve auslöst (Trigger). Die Amplitude des Gate-Signals bestimmt den maximalen Ausgangswert.
- **A:** Logarithmischer Event-Steuereingang für die Attack-Zeit. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (also in  $\text{dB}_{1\text{ms}}$ ).
- **H:** Logarithmischer Event-Steuereingang für die Haltedauer (hold time). 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (also in  $\text{dB}_{1\text{ms}}$ ).
- **D1:** Logarithmischer Event-Steuereingang für die 1. Decay-Zeit. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (also in  $\text{dB}_{1\text{ms}}$ ).
- **B:** Event-Steuereingang für den Breakpoint-Pegel. Wertebereich 0 (nie auf Decay-2 schalten) bis 1 (sofort auf Decay-2 schalten).
- **D2:** Logarithmischer Event-Steuereingang für die 2. Decay-Zeit. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (also in  $\text{dB}_{1\text{ms}}$ ).
- **R:** Logarithmischer Event-Steuereingang für die Release-Zeit. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (also in  $\text{dB}_{1\text{ms}}$ ).
- **Out:** Audio-Ausgang für das Hüllkurvensignal.



Hüllkurven-Generator mit vier Phasen und linearen Übergängen.

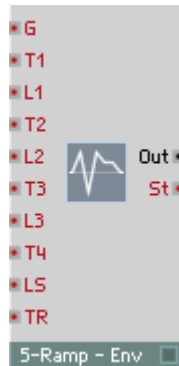
Wenn die Hüllkurve mit einem Gate-Event ausgelöst wird, steigt der Ausgangswert während der ersten Zeit linear auf den ersten Pegel, dann innerhalb der zweiten Zeit auf den zweiten Pegel usw. Der dritte Pegel ist der Sustain-Pegel, auf dem der Ausgang gehalten wird, bis ein Gate-Event mit Wert Null (Note-Off) empfangen wird, wonach der Ausgang mit dem letzten Zeitwert auf Null läuft.

- **G:** Event-Steuereingang für das Gate-Signal, das die Hüllkurve auslöst (Trigger). Die Amplitude des Gate-Signals bestimmt zusammen mit allen Level-Werten, welche Pegel wirklich erreicht werden.
- **T1:** Logarithmischer Event-Steuereingang für die Zeit der ersten Phase. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (also in  $\text{dB}_{1\text{ms}}$ ).
- **L1:** Steuereingang für den Pegel, der am Ende der ersten Phase erreicht wird.
- **T2:** Logarithmischer Event-Steuereingang für die Zeit der zweiten Phase. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (also in  $\text{dB}_{1\text{ms}}$ ).
- **L2:** Steuereingang für den Pegel, der am Ende der zweiten Phase erreicht wird.
- **T3:** Logarithmischer Event-Steuereingang für die Zeit der dritten Phase. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (also in  $\text{dB}_{1\text{ms}}$ ).
- **L3:** Steuereingang für den Pegel, der am Ende der dritten Phase erreicht wird. Dies ist der Sustain-Pegel.
- **TR:** Logarithmischer Event-Steuereingang für die Zeit der letzten Phase, der Release-Phase. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (also in  $\text{dB}_{1\text{ms}}$ ).

- **St:** Event-Ausgang für die Nummer der aktuellen Phase, in der sich die Hüllkurve befindet (1, 2 ...). Nach dem Ende der Release-Phase und vor einem neuen Gate-On-Event ist der Wert 0. Mit entsprechender Verschaltung kann dieser Wert benutzt werden, um weitere Envelope-Module in Reihe zu schalten, oder so, daß das Modul sich selber erneut anstößt.
- **Out:** Audio-Ausgang für das Hüllkurvensignal.

## 5-Ramp

## LFO, Envelope



Hüllkurven-Generator mit fünf Phasen und linearen Übergängen.

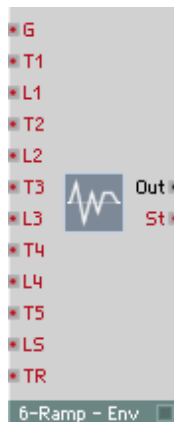
Wenn die Hüllkurve mit einem Gate-Event ausgelöst wird, steigt der Ausgangswert während der ersten Zeit linear auf den ersten Pegel, dann innerhalb der zweiten Zeit auf den zweiten Pegel usw. Der vierte Pegel ist der Sustain-Pegel, auf dem der Ausgang gehalten wird, bis ein Gate-Event mit Wert Null (Note-Off) kommt, wonach der Ausgang mit dem letzten Zeitwert auf Null läuft.

- **G:** Event-Steuereingang für das Gate-Signal, das die Hüllkurve auslöst (Trigger). Die Amplitude des Gate-Signals bestimmt zusammen mit allen Level-Werten, welche Pegel wirklich erreicht werden.
- **T1:** Logarithmischer Event-Steuereingang für die Zeit der ersten Phase. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (also in  $\text{dB}_{\text{1ms}}$ ).
- **L1:** Steuereingang für den Pegel, der am Ende der ersten Phase erreicht wird.
- **T2:** Logarithmischer Event-Steuereingang für die Zeit der zweiten Phase. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (also in  $\text{dB}_{\text{1ms}}$ ).

- **L2:** Steuereingang für den Pegel, der am Ende der zweiten Phase erreicht wird.
- **T3:** Logarithmischer Event-Steuereingang für die Zeit der dritten Phase. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (also in  $\text{dB}_{1\text{ms}}$ ).
- **L3:** Steuereingang für den Pegel, der am Ende der dritten Phase erreicht wird.
- **T4:** Logarithmischer Event-Steuereingang für die Zeit der vierten Phase. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (also in  $\text{dB}_{1\text{ms}}$ ).
- **LS:** Steuereingang für den Pegel, der am Ende der vierten Phase erreicht wird. Dies ist der Sustain-Pegel.
- **TR:** Logarithmischer Event-Steuereingang für die Zeit der letzten Phase, der Release-Phase. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (also in  $\text{dB}_{1\text{ms}}$ ).
- **St:** Event-Ausgang für die Nummer der aktuellen Phase, in der sich die Hüllkurve befindet (1, 2 ...). Nach dem Ende der Release-Phase und vor einem neuen Gate-On-Event ist der Wert 0. Mit entsprechender Verschaltung kann dieser Wert benutzt werden, um weitere Envelope-Module in Reihe zu schalten, oder so, daß das Modul sich selber erneut anstößt.
- **Out:** Audio-Ausgang für das Hüllkurvensignal.

## 6-Ramp

## LFO, Envelope



Hüllkurven-Generator mit sechs Phasen und linearen Übergängen. Wenn die Hüllkurve mit einem Gate-Event ausgelöst wird, steigt der Ausgangswert

während der ersten Zeit linear auf den ersten Pegel, dann innerhalb der zweiten Zeit auf den zweiten Pegel usw. Der fünfte Pegel ist der Sustain-Pegel, auf dem der Ausgang gehalten wird, bis ein Gate-Event mit Wert Null (Note-Off) kommt, wonach der Ausgang mit dem letzten Zeitwert auf Null läuft.

- **G:** Event-Steuereingang für das Gate-Signal, das die Hüllkurve auslöst (Trigger). Die Amplitude des Gate-Signals bestimmt zusammen mit allen Level-Werten, welche Pegel wirklich erreicht werden.
- **T1:** Logarithmischer Event-Steuereingang für die Zeit der ersten Phase. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (also in  $\text{dB}_{1\text{ms}}$ ).
- **L1:** Steuereingang für den Pegel, der am Ende der ersten Phase erreicht wird.
- **T2:** Logarithmischer Event-Steuereingang für die Zeit der zweiten Phase. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (also in  $\text{dB}_{1\text{ms}}$ ).
- **L2:** Steuereingang für den Pegel, der am Ende der zweiten Phase erreicht wird.
- **T3:** Logarithmischer Event-Steuereingang für die Zeit der dritten Phase. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (also in  $\text{dB}_{1\text{ms}}$ ).
- **L3:** Steuereingang für den Pegel, der am Ende der dritten Phase erreicht wird.
- **T4:** Logarithmischer Event-Steuereingang für die Zeit der vierten Phase. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (also in  $\text{dB}_{1\text{ms}}$ ).
- **L4:** Steuereingang für den Pegel, der am Ende der vierten Phase erreicht wird.
- **T5:** Logarithmischer Event-Steuereingang für die Zeit der fünften Phase. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (also in  $\text{dB}_{1\text{ms}}$ ).
- **LS:** Steuereingang für den Pegel, der am Ende der fünften Phase erreicht wird. Dies ist der Sustain-Pegel.
- **TR:** Logarithmischer Event-Steuereingang für die Zeit der letzten Phase, der Release-Phase. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (also in  $\text{dB}_{1\text{ms}}$ ).
- **St:** Event-Ausgang für die Nummer der aktuellen Phase, in der sich die Hüllkurve befindet (1, 2 ...). Nach dem Ende der Release-Phase und vor einem neuen Gate-On-Event ist der Wert 0. Mit entsprechender Verschaltung kann dieser Wert benutzt werden, um weitere Envelope-Module in Reihe zu schalten, oder so, daß das Modul sich selber erneut anstößt.
- **Out:** Audio-Ausgang für das Hüllkurvensignal.

# Filter

Es gibt eine grosse Anzahl an verschiedenen Filtermodulen in REAKTOR. Sie finden hier gleich 22 unterschiedliche Typen. Es werden sowohl die Standards mit Tiefpass, Hochpass und Bandpass abgedeckt, als auch Emulationen klassischer Synthesizer-Filter wie Sequential und Moog angeboten. Es existiert auch ein Allpass-Filter zur Hallerzeugung und Signal-Streuung und die Filtertypen Integrator und Differentiator.

Alle Filter in REAKTOR können mit jeder Frequenz laufen, von 0 Hz (konstantes Signal) über das gesamte Audiospektrum bis zur Grenze, die von der Abtastrate gesetzt wird. Damit sind sie alle gleichermaßen für die Verarbeitung von Audiosignalen wie für die Glättung von Steuersignalen (z. B. Portamento) geeignet. Wenn man mit einem Filter das Eingangssignal für einen Port bearbeitet, der nur Events annimmt (z. B. P im Gegensatz zu F), muss man einen A to E (perm) Konverter dazwischenschalten.

Alle Filter und Equalizer können optional ihren Frequenzgang in einer Grafik im Panel darstellen. Dies kann mit der Option **Visible** auf der **Appearance-Seite** in den Modul-Properties aktiviert werden. Die Größe der Darstellung kann in den Properties mit **Size X** und **Size Y** eingestellt werden. Die Frequenzachse der Kurve ist logarithmisch skaliert und reicht von 10 Hz bis 20 kHz.

---

## HP/LP 1-Pole

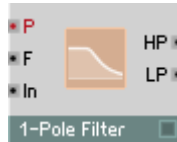
## Filter



1-Pol-Filter mit Hoch- und Tiefpaßausgang (Flankensteilheit 6 dB/Oktave) und logarithmischer Einstellung der Grenzfrequenz.

- **P**: Logarithmischer Event-Steuereingang für die Grenzfrequenz (Cutoff Frequency) in Halbtonschritten (69 = 440 Hz).
- **In**: Audio-Eingang für das zu filternde Signal.
- **HP**: Audio-Ausgang für das Hochpaßfilter-Signal (High Pass Filter).
- **LP**: Audio-Ausgang für das Tiefpaßfilter-Signal (Low Pass Filter).





1-Pol-Filter mit Hoch- und Tiefpaßausgang (Flankensteilheit 6 dB/Oktave), logarithmischer und linearer Einstellung der Grenzfrequenz.

- **P**: Logarithmischer Event-Steuereingang für die Grenzfrequenz (Cutoff Frequency) in Halbtonschritten ( $69 = 440$  Hz).
- **F**: Audio-Steuereingang für die lineare Einstellung der Grenzfrequenz in Hz. Die Grenzfrequenz bestimmt sich aus **P** und **F**.
- **In**: Audio-Eingang für das zu filternde Signal.
- **HP**: Audio-Ausgang für das Hochpaßfilter-Signal (High Pass Filter).
- **LP**: Audio-Ausgang für das Tiefpaßfilter-Signal (Low Pass Filter).

---

**Allpass 1-Pole****Filter**

Allpass-Filter erster Ordnung. Dieser Filtertyp hat eine geringe Flankensteilheit, aber die Phasenverschiebung zwischen Ein- und Ausgang vergrößert sich von 0 Grad bei tiefen Frequenzen bis -180 Grad bei hohen Frequenzen. An der vom P-Eingang gesteuerten Eckfrequenz wird die Phase um -90 Grad versetzt.

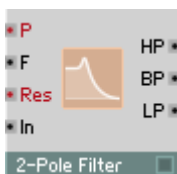
- **P**: Logarithmischer Steuereingang für die Eckfrequenz, bei der die Phasenverschiebung -90 Grad beträgt. Skalierung: 1 Halbton pro Einheit,  $43 = G1 = 98$  Hz,  $84 = C5 = 1047$  Hz.
- **In**: Eingang für das Signal, das mit dem Allpass gefiltert (phasenverschoben) werden soll.
- **Out**: Ausgang für das mit dem Allpass gefilterten (phasenverschobenen) Signal.



2-Pol-Filter mit Hoch- und Tiefpaßausgang (Flankensteilheit 12 dB/Oktave), Bandpaßausgang (oben und unten je 6 dB/Oktave), variabler Filterresonanz und logarithmischer Einstellung der Grenzfrequenz. Bei sehr hoher Resonanz kann das Filter auch als Oscillator verwendet werden (Selbstoszillation).

Die Verstärkung im Durchlaßbereich ist immer 1 (0 dB), während die Verstärkung bei der Grenzfrequenz mit der Resonanz steigt. Achtung: sehr große Amplituden bei **Res** nahe 1.

- **P**: Logarithmischer Event-Steuereingang für die Grenzfrequenz (cutoff frequency) in Halbtonschritten (69 = 440 Hz).
- **Res**: Event-Steuereingang für die Resonanz/Dämpfung des Filters. Wertebereich 0 (maximale Dämpfung, keine Resonanz) bis 1 (keine Dämpfung, maximale Resonanz, Selbstoszillation). Bei hoher Resonanz beträgt der Q-Faktor = Bandmitte [Hz] / Bandbreite [Hz]  $\cong 1 / (2 - \text{Res})$ .
- **In**: Audio-Eingang für das zu filternde Signal.
- **HP**: Audio-Ausgang für das Hochpaßfilter-Signal (High Pass Filter).
- **BP**: Audio-Ausgang für das Bandpaßfilter-Signal (Band Pass Filter).
- **LP**: Audio-Ausgang für das Tiefpaßfilter-Signal (Low Pass Filter).



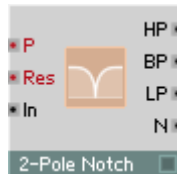
2-Pol-Filter mit Hoch- und Tiefpaßausgang (Flankensteilheit 12 dB/Oktave), Bandpaßausgang (oben und unten je 6 dB/Oktave), variabler Filterresonanz, logarithmischer und linearer Einstellung der Grenzfrequenz. Bei sehr hoher Resonanz kann das Filter auch als Oscillator verwendet werden (Selbstoszillation). Die Verstärkung im Durchlaßbereich ist immer 1 (0 dB),

während die Verstärkung bei der Grenzfrequenz mit der Resonanz steigt. Achtung: sehr große Amplituden bei **Res** nahe 1.

- **P**: Logarithmischer Event-Steuereingang für die Grenzfrequenz (cutoff frequency) in Halbtonschritten (69 = 440 Hz).
- **F**: Audio-Steuereingang für die lineare Einstellung der Grenzfrequenz in Hz. Die Grenzfrequenz bestimmt sich aus **P** und **F**.
- **Res**: Event-Steuereingang für die Resonanz/Dämpfung des Filters. Wertebereich 0 (maximale Dämpfung, keine Resonanz) bis 1 (keine Dämpfung, maximale Resonanz, Selbstoszillation). Bei hoher Resonanz beträgt der Q-Faktor = Bandmitte [Hz]/Bandbreite [Hz]  $\cong 1/(2 - 2 \text{ Res})$ .
- **In**: Audio-Eingang für das zu filternde Signal.
- **HP**: Audio-Ausgang für das Hochpaßfilter-Signal (High Pass Filter).
- **BP**: Audio-Ausgang für das Bandpaßfilter-Signal (Band Pass Filter).
- **LP**: Audio-Ausgang für das Tiefpaßfilter-Signal (Low Pass Filter).

## Multi/Notch 2-Pole

## Filter



2-Pol-Filter mit Bandsperre-Ausgang, Hoch- und Tiefpaßausgang (Flankensteilheit 12 dB/Oktave), Bandpaß-Ausgang (oben und unten je 6 dB/Oktave), variabler Filterresonanz und logarithmischer Einstellung der Grenzfrequenz.

Die Verstärkung bei der Grenzfrequenz ist immer 1 (0 dB), während die Verstärkung im Durchlaßbereich mit steigender Resonanz sinkt. Am Bandsperre-Ausgang ist die eingestellte Frequenz vollständig unterdrückt.

- **P**: Logarithmischer Event-Steuereingang für die Grenzfrequenz (Cutoff Frequency) in Halbtonschritten (69 = 440 Hz).
- **Res**: Event-Steuereingang für die Resonanz/Dämpfung des Filters. Wertebereich 0 (maximale Dämpfung, keine Resonanz) bis 1 (keine Dämpfung, maximale Resonanz, Selbstoszillation). Bei hoher Resonanz beträgt der Q-Faktor = Bandmitte [Hz]/Bandbreite [Hz]  $\cong 1/(2 - 2 \text{ Res})$ .

- **In:** Audiosignal-Eingang für das zu filternde Signal.
- **HP:** Audio-Ausgang für das Hochpaßfilter-Signal (High Pass Filter).
- **BP:** Audio-Ausgang für das Bandpaßfilter-Signal (Band Pass Filter).
- **LP:** Audio-Ausgang für das Tiefpaßfilter-Signal (Low Pass Filter).
- **N:** Audio-Ausgang für das Bandsperre-Signal (Band-Reject-/Notch-Filter).

## Multi/Notch 2-Pole FM

## Filter

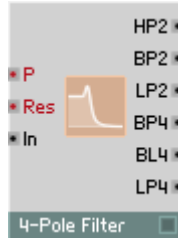


2-Pol-Filter mit Bandsperre-Ausgang, Hoch- und Tiefpaßausgang (Flankensteilheit 12 dB/Oktave), Bandpaß-Ausgang (oben und unten je 6 dB/Oktave), variabler Filterresonanz, logarithmischer und linearer Einstellung der Grenzfrequenz.

Die Verstärkung bei der Grenzfrequenz ist immer 1 (0 dB), während die Verstärkung im Durchlaßbereich mit steigender Resonanz sinkt.

Am Bandsperre-Ausgang ist die eingestellte Frequenz vollständig unterdrückt.

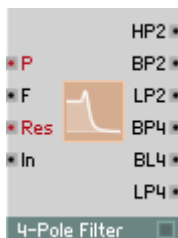
- **P:** Logarithmischer Event-Steuereingang für die Grenzfrequenz (Cutoff Frequency) in Halbtonschritten (69 = 440 Hz).
- **F:** Audio-Steuereingang für die lineare Einstellung der Grenzfrequenz in Hz. Die Frequenz bestimmt sich aus **P** und **F**.
- **Res:** Event-Steuereingang für die Resonanz/Dämpfung des Filters. Wertebereich 0 (maximale Dämpfung, keine Resonanz) bis 1 (keine Dämpfung, maximale Resonanz, Selbstoszillation). Bei hoher Resonanz beträgt der Q-Faktor = Bandmitte [Hz]/Bandbreite [Hz]  $\cong 1/(2 - 2 \text{ Res})$ .
- **In:** Audiosignal-Eingang für das zu filternde Signal.
- **HP:** Audio-Ausgang für das Hochpaßfilter-Signal (High Pass Filter).
- **BP:** Audio-Ausgang für das Bandpaßfilter-Signal (Band Pass Filter).
- **LP:** Audio-Ausgang für das Tiefpaßfilter-Signal (Low Pass Filter).
- **N:** Audio-Ausgang für das Bandsperre-Signal (Band-Reject-/Notch-Filter).



4-Pol-Filter mit Tiefpaß- (24 dB/Oktave), Bandpaß- (12/12 dB/Oktave) und Bandpaß-Tiefpaß-Ausgängen (6/18 dB/Oktave), 2-Pol Hochpaß-, Tiefpaß- (12 dB/Oktave) und Bandpaßausgängen (6/6 dB/Oktave), variabler Filterresonanz und logarithmischer Einstellung der Grenzfrequenz.

Die Verstärkung im Durchlaßbereich ist immer 1 (0 dB), während die Verstärkung bei der Grenzfrequenz mit der Resonanz steigt. Achtung: sehr große Amplituden bei **Res** nahe 1.

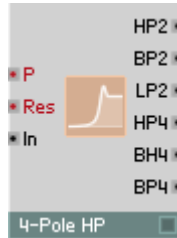
- **P**: Logarithmischer Event-Steuereingang für die Grenzfrequenz (Cutoff Frequency) in Halbtonschritten (69 = 440 Hz).
- **Res**: Event-Steuereingang für die Resonanz/Dämpfung des Filters. Wertebereich 0 (maximale Dämpfung, keine Resonanz) bis 1 (keine Dämpfung, maximale Resonanz, Selbstoszillation!).
- **In**: Audio-Eingang für das zu filternde Signal.
- **HP2**: Audio-Ausgang für das 2-Pol-Hochpaßfilter-Signal.
- **BP2**: Audio-Ausgang für das 2-Pol-Bandpaßfilter-Signal.
- **LP2**: Audio-Ausgang für das 2-Pol-Tiefpaßfilter-Signal.
- **BP4**: Audio-Ausgang für das 4-Pol-Bandpaßfilter-Signal.
- **BL4**: Audio-Ausgang für das Bandpaß-Tiefpaß-Signal.
- **LP4**: Audio-Ausgang für das 4-Pol-Tiefpaßfilter-Signal.



4-Pol-Filter mit Hochpaß- (24 dB/Oktave), Bandpaß- (12/12 dB/Oktave) und Bandpaß-Hochpaß-Ausgängen (18/6 dB/Oktave), 2-Pol Hochpaß-, Tiefpaß- (12 dB/Oktave) und Bandpaßausgängen (6/6 dB/Oktave), variabler Filterresonanz, logarithmischer und linearer Einstellung der Grenzfrequenz.

Die Verstärkung im Durchlaßbereich ist immer 1 (0 dB), während die Verstärkung bei der Grenzfrequenz mit der Resonanz steigt. Achtung: sehr große Amplituden bei **Res** nahe 1.

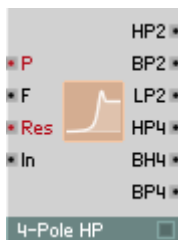
- **P**: Logarithmischer Event-Steuereingang für die Grenzfrequenz (Cutoff Frequency) in Halbtonschritten ( $69 = 440$  Hz).
- **F**: Audio-Steuereingang für die lineare Einstellung der Grenzfrequenz in Hz. Die Grenzfrequenz bestimmt sich aus **P** und **F**.
- **Res**: Event-Steuereingang für die Resonanz/Dämpfung des Filters. Wertebereich 0 (maximale Dämpfung, keine Resonanz) bis 1 (keine Dämpfung, maximale Resonanz, Selbstoszillation).
- **In**: Audio-Eingang für das zu filternde Signal.
- **HP2**: Audio-Ausgang für das 2-Pol-Hochpaßfilter-Signal.
- **BP2**: Audio-Ausgang für das 2-Pol-Bandpaßfilter-Signal.
- **LP2**: Audio-Ausgang für das 2-Pol-Tiefpaßfilter-Signal.
- **HP4**: Audio-Ausgang für das 4-Pol-Hochpaßfilter-Signal.
- **BH4**: Audio-Ausgang für das Bandpaß-Hochpaß-Signal.
- **BP4**: Audio-Ausgang für das 4-Pol-Bandpaßfilter-Signal.



4-Pol-Filter mit Tiefpaß- (24 dB/Oktave), Bandpaß- (12/12 dB/Oktave) und Bandpaß-Tiefpaß-Ausgängen (6/18 dB/Oktave), 2-Pol Hochpaß-, Tiefpaß- (12 dB/Oktave) und Bandpaßausgängen (6/6 dB/Oktave), variabler Filterresonanz und logarithmischer Einstellung der Grenzfrequenz.

Die Verstärkung im Durchlaßbereich ist immer 1 (0 dB), während die Verstärkung bei der Grenzfrequenz mit der Resonanz steigt. Achtung: sehr große Amplituden bei **Res** nahe 1.

- **P**: Logarithmischer Event-Steuereingang für die Grenzfrequenz (Cutoff Frequency) in Halbtonschritten (69 = 440 Hz).
- **Res**: Event-Steuereingang für die Resonanz/Dämpfung des Filters. Wertebereich 0 (maximale Dämpfung, keine Resonanz) bis 1 (keine Dämpfung, maximale Resonanz, Selbstoszillation!).
- **In**: Audio-Eingang für das zu filternde Signal.
- **HP2**: Audio-Ausgang für das 2-Pol-Hochpaßfilter-Signal.
- **BP2**: Audio-Ausgang für das 2-Pol-Bandpaßfilter-Signal.
- **LP2**: Audio-Ausgang für das 2-Pol-Tiefpaßfilter-Signal.
- **BP4**: Audio-Ausgang für das 4-Pol-Bandpaßfilter-Signal.
- **BL4**: Audio-Ausgang für das Bandpaß-Tiefpaß-Signal.
- **LP4**: Audio-Ausgang für das 4-Pol-Tiefpaßfilter-Signal.

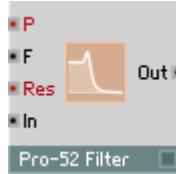


4-Pol-Filter mit Hochpaß- (24 dB/Oktave), Bandpaß- (12/12 dB/Oktave) und Bandpaß-Hochpaß-Ausgängen (18/6 dB/Oktave), 2-Pol Hochpaß-, Tiefpaß- (12 dB/Oktave) und Bandpaßausgängen (6/6 dB/Oktave), variabler Filterresonanz, logarithmischer und linearer Einstellung der Grenzfrequenz.

Die Verstärkung im Durchlaßbereich ist immer 1 (0 dB), während die Verstärkung bei der Grenzfrequenz mit der Resonanz steigt. Achtung: sehr große Amplituden bei **Res** nahe 1.

- **P**: Logarithmischer Event-Steuereingang für die Grenzfrequenz (Cutoff Frequency) in Halbtonschritten (69 = 440 Hz).
- **F**: Audio-Steuereingang für die lineare Einstellung der Grenzfrequenz in Hz. Die Grenzfrequenz bestimmt sich aus **P** und **F**.
- **Res**: Event-Steuereingang für die Resonanz/Dämpfung des Filters. Wertebereich 0 (maximale Dämpfung, keine Resonanz) bis 1 (keine Dämpfung, maximale Resonanz, Selbstoszillation).
- **In**: Audio-Eingang für das zu filternde Signal.
- **HP2**: Audio-Ausgang für das 2-Pol-Hochpaßfilter-Signal.
- **BP2**: Audio-Ausgang für das 2-Pol-Bandpaßfilter-Signal.
- **LP2**: Audio-Ausgang für das 2-Pol-Tiefpaßfilter-Signal.
- **HP4**: Audio-Ausgang für das 4-Pol-Hochpaßfilter-Signal.
- **BH4**: Audio-Ausgang für das Bandpaß-Hochpaß-Signal.
- **BP4**: Audio-Ausgang für das 4-Pol-Bandpaßfilter-Signal.





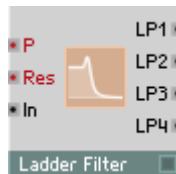
Filter des Pro-52 Analog-Synthesizers. Es handelt sich um einen 4-Pole Tiefpaß-Filter (24 dB/Oktave) mit variabler Filterresonanz, logarithmischer und linearer Einstellung der Grenzfrequenz.

Das Filter beginnt mit der Selbstoszillation, wenn **Res** sich 1 annähert. Die Amplitude der Selbstoszillation ist ungefähr 1.

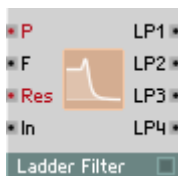
- **P**: Logarithmischer Event-Steuereingang für die Grenzfrequenz (Cutoff Frequency) in Halbtonschritten (69 = 440 Hz).
- **F**: Audio-Steuereingang für die lineare Einstellung der Grenzfrequenz in Hz. Die Grenzfrequenz bestimmt sich aus **P** und **F**.
- **Res**: Event-Steuereingang für die Resonanz/Dämpfung des Filters. Wertebereich 0 (maximale Dämpfung, keine Resonanz) bis 1 (keine Dämpfung, maximale Resonanz, Selbstoszillation).
- **In**: Audio-Eingang für das zu filternde Signal.
- **Out**: Audio-Ausgang für das 4-Pol-Tiefpaßfilter-Signal.

## Ladder Filter

## Filter



Genauso wie **Ladder Filter FM**, aber ohne den Steuereingang für die Frequenzmodulation. Siehe nächste Modulbeschreibung.

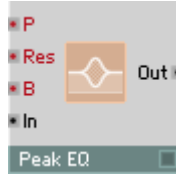


Filter, das auf dem klassischen, patentierten Ladder-Schaltkreis von Bob Moog basiert. Es handelt sich um einen 4-Pole Filter mit verschiedenen Tiefpaß-Ausgängen: 24 dB/Oktave, 18 dB/Oktave, 12 dB/Oktave, 6 dB/Oktave. Das Filter besitzt außerdem eine variable Filterresonanz, logarithmische und lineare Grenzfrequenz.

Optional kann die Sättigungs-Charakteristik des analogen Schaltkreises simuliert werden, wenn Sie **Distortion** in den Properties aktivieren. Wenn **Distortion** aktiv ist, geht das Filter in die Selbstoszillation über, wenn **Res** 1 oder höher ist. Die Amplitude der Selbstoszillation ist ungefähr 1, wenn **Res** 1 ist, kann aber auch sehr viel höher sein, wenn **Res** mit noch größeren Werten angesteuert wird.

Das Filter besitzt auch Optionen zur Einstellung der Qualität der Simulation: **Standard**, **High** und **Excellent**. Dieser Effekt ist besonders dann wahrnehmbar, wenn **Distortion** aktiv ist. Natürlich erzeugt eine höhere Qualitätseinstellung auch eine höhere CPU-Last.

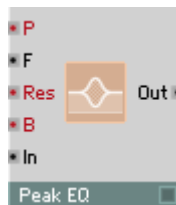
- **P**: Logarithmischer Event-Steuereingang für die Grenzfrequenz (Cutoff Frequency) in Halbtonschritten (69 = 440 Hz).
- **F**: Audio-Steuereingang für die lineare Einstellung der Grenzfrequenz in Hz. Die Grenzfrequenz bestimmt sich aus **P** und **F**.
- **Res**: Event-Steuereingang für die Resonanz/Dämpfung des Filters. Wertebereich 0 (maximale Dämpfung, keine Resonanz) bis 1 (keine Dämpfung, maximale Resonanz, Selbstoszillation). Werte über 1 sind möglich im Distortion-Modus.
- **In**: Audio-Eingang für das zu filternde Signal.
- **LP1**: Audio-Ausgang für das 6 dB/Oktave-Tiefpaßfilter-Signal.
- **LP2**: Audio-Ausgang für das 12 dB/Oktave-Tiefpaßfilter-Signal.
- **LP3**: Audio-Ausgang für das 18 dB/Oktave-Tiefpaßfilter-Signal.
- **LP4**: Audio-Ausgang für das 24 dB/Oktave-Tiefpaßfilter-Signal.



Parametrischer Equalizer mit einstellbarer Anhebung/Absenkung, Bandbreite und logarithmisch regelbarer Frequenz.

Mit dem Peak-EQ kann eine bestimmte Frequenz und ihre mehr oder weniger große Umgebung gezielt verstärkt oder gedämpft werden. Weiter entfernte Frequenzen werden nicht verändert.

- **P**: Logarithmischer Event-Steuereingang für die Frequenz in Halbtonschritten ( $69 = 440$  Hz).
- **Res**: Event-Steuereingang für die Resonanz (Q-Faktor) des Filters. Wertebereich 0 (minimale Resonanz, maximale Bandbreite) bis 1 (maximale Resonanz, minimale Bandbreite). Bei hoher Resonanz beträgt der Q-Faktor  $= \text{Bandmitte [Hz]} / \text{Bandbreite [Hz]} \cong 1 / (2 - 2 \text{ Res})$ .
- **B**: Event-Steuereingang für Anhebung/Absenkung (Boost/Cut) in dB. Bei **B** = 0 bleibt das Signal unverändert.
- **In**: Audio-Eingang für das Equalizer-Eingangssignal.
- **Out**: Audio-Ausgang für das Equalizer-Ausgangssignal.



Parametrischer Equalizer mit einstellbarer Anhebung/Absenkung, Bandbreite und logarithmisch und linear regelbarer Frequenz.

Mit dem Peak-EQ kann eine bestimmte Frequenz und ihre mehr oder weniger große Umgebung gezielt verstärkt oder gedämpft werden. Weiter entfernte Frequenzen werden nicht verändert.

- **P**: Logarithmischer Event-Steuereingang für die Frequenz in Halbtonschritten ( $69 = 440 \text{ Hz}$ ).
- **F**: Audio-Steuereingang für die lineare Einstellung der Frequenz in Hz. Die Frequenz bestimmt sich aus **P** und **F**.
- **Res**: Event-Steuereingang für die Resonanz (Q-Faktor) des Filters. Wertebereich 0 (minimale Resonanz, maximale Bandbreite) bis 1 (maximale Resonanz, minimale Bandbreite). Bei hoher Resonanz beträgt der Q-Faktor = Bandmitte [Hz] / Bandbreite [Hz]  $\cong 1 / (2 - 2 \text{ Res})$ .
- **B**: Event-Steuereingang für Anhebung/Absenkung (Boost/Cut) in dB. Bei **B** = 0 bleibt das Signal unverändert.
- **In**: Audio-Eingang für das Equalizer-Eingangssignal.
- **Out**: Audio-Ausgang für das Equalizer-Ausgangssignal.

## High Shelf EQ

## Filter



Parametrischer Equalizer mit High-Shelving-Charakteristik, einstellbarer Anhebung/Absenkung und logarithmisch regelbarer Frequenz.

Der Pegel von Frequenzen oberhalb der Eckfrequenz wird um den eingestellten Betrag angehoben oder abgesenkt, tiefe Frequenzen werden nicht verändert.

- **P**: Logarithmischer Event-Steuereingang für die Eckfrequenz in Halbtonschritten ( $69 = 440 \text{ Hz}$ ).
- **B**: Event-Steuereingang für Anhebung/Absenkung (Boost/Cut) in dB. Bei **B** = 0 bleibt das Signal unverändert.
- **In**: Audio-Eingang für das Equalizer-Eingangssignal.
- **Out**: Audio-Ausgang für das Equalizer-Ausgangssignal.



Parametrischer Equalizer mit High-Shelving-Charakteristik, einstellbarer Anhebung/Absenkung und logarithmisch und linear regelbarer Frequenz.

Der Pegel von Frequenzen oberhalb der Eckfrequenz wird um den eingestellten Betrag angehoben oder abgesenkt, tiefe Frequenzen werden nicht verändert.

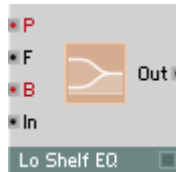
- **P**: Logarithmischer Event-Steuereingang für die Eckfrequenz in Halbtonschritten (69 = 440 Hz).
- **F**: Audio-Steuereingang für die lineare Einstellung der Eckfrequenz in Hz. Die Frequenz bestimmt sich aus **P** und **F**.
- **B**: Event-Steuereingang für Anhebung/Absenkung (Boost/Cut) in dB. Bei **B** = 0 bleibt das Signal unverändert.
- **In**: Audio-Eingang für das Equalizer-Eingangssignal.
- **Out**: Audio-Ausgang für das Equalizer-Ausgangssignal.



Parametrischer Equalizer mit Low-Shelving-Charakteristik, einstellbarer Anhebung/Absenkung und logarithmisch regelbarer Frequenz.

Der Pegel von Frequenzen unterhalb der Eckfrequenz wird um den eingestellten Betrag angehoben oder abgesenkt, hohe Frequenzen werden nicht verändert.

- **P**: Logarithmischer Event-Steuereingang für die Eckfrequenz in Halbtonschritten (69 = 440 Hz).
- **B**: Event-Steuereingang für Anhebung/Absenkung (Boost/Cut) in dB. Bei **B** = 0 bleibt das Signal unverändert.
- **In**: Audio-Eingang für das Equalizer-Eingangssignal.
- **Out**: Audio-Ausgang für das Equalizer-Ausgangssignal.



Parametrischer Equalizer mit Low-Shelving-Charakteristik, einstellbarer Anhebung/Absenkung und logarithmisch und linear regelbarer Frequenz.

Der Pegel von Frequenzen unterhalb der Eckfrequenz wird um den eingestellten Betrag angehoben oder abgesenkt, hohe Frequenzen werden nicht verändert.

- **P**: Logarithmischer Event-Steuereingang für die Eckfrequenz in Halbtonschritten ( $69 = 440$  Hz).
- **F**: Audio-Steuereingang für die lineare Einstellung der Eckfrequenz in Hz. Die Frequenz bestimmt sich aus **P** und **F**.
- **B**: Event-Steuereingang für Anhebung/Absenkung (Boost/Cut) in dB. Bei **B** = 0 bleibt das Signal unverändert.
- **In**: Audio-Eingang für das Equalizer-Eingangssignal.
- **Out**: Audio-Ausgang für das Equalizer-Ausgangssignal.



Der Differentiator liefert am Ausgang die Steigung des Eingangssignals in “Units pro Millisekunde”. Er wirkt damit wie ein Hochpass-Filter, dessen Verstärkung proportional mit der Frequenz ansteigt. Bei 159 Hz ist die Verstärkung gleich 1.

- **In**: Eingang für das zu differenzierende Signal.
- **Out**: Ausgang für das differenzierte Signal.



Der Integrator liefert am Ausgang ein Signal, dessen Steigung in “Units pro Millisekunde” durch die Amplitude des Eingangssignals gesteuert wird. Er wirkt damit wie ein Tiefpass-Filter, dessen Verstärkung proportional mit der Frequenz absinkt. Bei 159 Hz ist die Verstärkung gleich 1. Ein Event am Set-Eingang setzt den Ausgang sprunghaft auf den Wert des Events.

- **Set:** Ein Event an diesem Eingang setzt den Ausgang des Integrators auf den Wert des Events.
- **In:** Eingang für das zu integrierende Signal. Es steuert die Steigung des Ausgangs in „Einheiten pro Millisekunde“.
- **Out:** Ausgang für das integrierte Signal.

# Delay

REAKTOR verfügt über sechs Delay-Typen zur Realisierung von Verzögerungseffekten. Es existieren Two-Tap-Delays, zwei granulare Delays, ein Diffuser (für Halleffekte) und das Unit Delay, welches für Audio-Feedbacks und Physical Modeling verwendet werden kann.

## Single Delay

## Delay



Verzögerungsglied für Audiosignale. Das Eingangssignal erscheint am Ausgang mit einer Verzögerung entsprechend der eingestellten Zeit. Die Verzögerung wird am **Dly**-Eingang gesteuert.

Die Obergrenze für die Verzögerungszeit läßt sich im Properties-Dialog des Moduls im Feld Max Delay Buffer einstellen (normal 1 Sekunde) und beeinflusst den Speicherverbrauch. Wie groß dieser Wert eingestellt werden kann, hängt vom verfügbaren RAM des Rechners ab. Bei einer Abtastrate von 44,1 kHz braucht man 172 kB RAM pro Sekunde Pufferlänge und pro Stimme, für eine Minute werden 10 MB benötigt. Wenn das Modul als Event-Delay verwendet wird (der In-Port ist rot und zeigt an, dass sich das Modul im Event-Verarbeitungs-Modus befindet) können Sie an gleicher Stelle die Zahl der im Delay-Buffer zwischengespeicherten Events unter Max Count Of Buffered Events einstellen.

Dieses Modul ersetzt verschiedene Module früherer REAKTOR-Versionen:

- Es verhält sich wie ein REAKTOR 3 Static Delay, wenn Sie ein Audiosignal an den In-Eingang und ein Eventsignal an den Dly-Eingang anschliessen. Wenn die Verzögerungszeit nicht einer ganzen Zahl von Samples entspricht, wird das Ausgangssignal durch Interpolation bestimmt. Dadurch kann es zu geringfügiger Veränderung des Klangs kommen. Die Interpolationsmethode kann im Properties-Dialog ausgewählt werden. Lineare Interpolation kann den Gehalt hoher Frequenzen im Klang ein wenig reduzieren.
- Es verhält sich wie ein REAKTOR 3 Modulation Delay, wenn Sie ein Audiosignal an den In-Eingang und ein Audiosignal an den Dly-Eingang anschliessen. Die Verzögerung kann durch ein Audio-Signal am **Dly**-Eingang kontinuierlich moduliert werden. Wenn die Verzögerungszeit



nicht einer ganzen Zahl von Samples entspricht, wird das Ausgangssignal durch Interpolation bestimmt. Dadurch kann es zu geringfügiger Veränderung des Klangs kommen. Die Interpolationsmethode kann im Properties-Dialog ausgewählt werden.

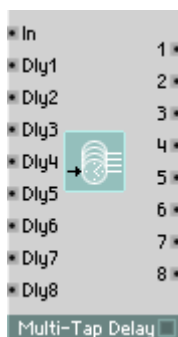
- Es verhält sich wie ein REAKTOR 3 Event Delay, wenn Sie ein Eventsignal an den In-Eingang anschliessen.

## Ports

- **Dly:** Hybrideingang für die Verzögerungszeit (Delay Time) in Millisekunden (ms).
- **In:** Hybrideingang für das zu verzögernde Audiosignal.
- **Out:** Ausgang für das verzögerte Audiosignal.

## Multi-Tap Delay

## Delay



Multi-Tap-Delay-Kette für Audiosignale. Wenn die eingestellte Verzögerungszeit einer Sampleanzahl entspricht, die nicht ganzzahlig ist, wird interpoliert. Die Interpolations-Methode kann in den Properties gewählt werden.

Der Ausgang wird üblicherweise an einen Mixer oder Scanner angeschlossen.

- **In:** Audioeingang für das zu verzögernde Signal.
- **Dly1...8:** Audioeingänge zur Steuerung der Verzögerungszeit in Millisekunden. Typ. Wertebereich: [0...1000].
- **1...8:** Audioausgänge für das verzögerte Eingangssignal. **1** wird mit der Zeit **Dly1**, **2** mit **Dly2** usw.verzögert.



Der Diffuser ist ein Allpaßfilter, der ein Verzögerungsglied (Delay-Line) mit Rückkopplung (Feedback) enthält. Die Wirkung ist eine Zerstreuung des Eingangssignals, ohne dabei bestimmte Frequenzen zu betonen. Das typische Anwendungsgebiet ist der Bau von Hall-Effekten (Reverb), wobei man meist mehrere dieser Module hintereinander schaltet und ihnen unterschiedliche Delay-Zeiten im Millisekundenbereich gibt.

Die Verzögerung wird am **Dly**-Eingang gesteuert. Die Verzögerung kann durch ein Audio-Signal am **Dly**-Eingang kontinuierlich moduliert werden. Wenn die Verzögerungszeit nicht einer ganzen Zahl von Samples entspricht, wird das Ausgangssignal durch Interpolation bestimmt. Dadurch kann es zu geringfügiger Veränderung des Klangs kommen. Die Interpolationsmethode kann im Properties-Dialog ausgewählt werden. Der Grad an interner Rückkopplung wird mit dem **Dffs**-Eingang eingestellt.

Wenn man als Delay-Zeit Null einstellt, funktioniert der Diffuser als 1-Pol-Allpaßfilter. So kann man z. B. einen Phaser bauen, indem man mehrere Diffuser in Reihe schaltet und den **Dffs**-Parameter moduliert.

Die Obergrenze für die Verzögerungszeit läßt sich im Properties-Dialog des Moduls einstellen (normal 200 ms) und beeinflußt den Speicherverbrauch.

- **Dly**: Hybrideingang für die Verzögerungszeit (Delay Time) in Millisekunden.
- **Dffs**: Event-Steuereingang für den Diffusions-Koeffizienten. Wertebereich: -1...1. **Dffs** = 0 : das Modul ist ein pures Delay, **Dffs** = 1 : Ausgang = Eingang, **Dffs** = -1 : Ausgang = -Eingang. Die brauchbarsten Werte für **Dffs** sind in der Nähe von 0.5 zu finden.
- **In**: Eingang für das zu zerstreute Audiosignal.
- **Out**: Ausgang für das zerstreute Audiosignal.



Verzögerungsglied und Pitch-Shifter für Audiosignale. Das Eingangssignal erscheint an den Ausgängen mit einer Verzögerung entsprechend der am **Dly**-Eingang eingestellten Zeit, transponiert um den am **P**-Eingang anliegenden Wert in Halbtönen.

Das Eingangssignal wird in Klangpartikel “zerhackt”, deren Größe über den **Granularity**-Eingang (**Gr**) gesteuert wird. Über den **Smoothness**-Eingang (**Sm**) kann die “Rauheit” des Klangergebnisses gesteuert werden. Die Position von Klangpartikeln im Stereofeld wird über den **Pan**-Eingang festgelegt.

Die Delayzeit von **Grain Delay** kann ohne Auswirkung auf die Tonhöhe variiert werden. Interessante Effekte sind in Verbindung mit Zufalls- und Rauschgeneratoren möglich.

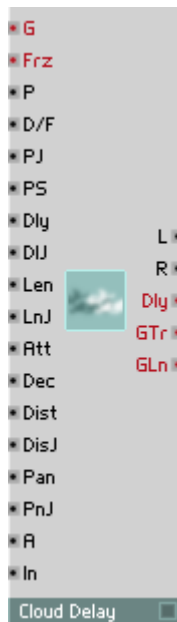
Im Properties-Dialog kann neben der Wiedergabe-Qualität auch die Obergrenze für die Verzögerungszeit eingestellt werden. Die tatsächlich zur Verfügung stehende maximale Delayzeit kann von dem eingestellten Wert um bis zu 50 % abweichen.

- **P**: Logarithmischer Audio-Steuereingang für die Transposition in Halbtönen (Pitch).
- **Dly**: Audio-Steuereingang für die Verzögerungszeit (Delay Time) in Millisekunden.
- **Gr**: Audio-Steuereingang für die Granularität des Re-Synthese-Prozesses in Millisekunden. Dieser Parameter bestimmt die Größe der für die Re-Synthese benutzten Klangpartikel.
- **Sm**: Audio-Steuereingang für die *Smoothness* (Gleichmäßigkeit) des Re-Syntheseprozesses. Beeinflusst wird die Formgebung der Klangpartikel. Kleine Werte führen im allgemeinen zu einem rauheren Klangergebnis.
- **Pan**: Audio-Steuereingang für die Position im Stereofeld (-1 = Links, 0 = Mitte, 1 = Rechts).

- **A:** Audio-Steuereingang für die Ausgangs-Amplitude.
- **In:** Eingang für das zu verzögernde Audiosignal.
- **L:** Audio-Ausgang für den linken Kanal des Delays.
- **R:** Audio-Ausgang für den rechten Kanal des Delays.
- **Dly:** Polyphoner Event-Ausgang, an dem die aktuelle Verzögerungszeit anliegt. Dieser Ausgang produziert immer dann ein Event, wenn ein neues Klangpartikel erzeugt wird.

## Grain Cloud Delay

## Delay



Das Grain Cloud Delay-Modul ist dem Grain Cloud-Modul aus der Sampler-Sektion sehr ähnlich. Der Unterschied besteht darin, dass das Grain Cloud-Modul mit im Ram abgelegten Samples arbeitet, während das entsprechende Delay-Modul den konstant wechselnden Audiobuffer bearbeitet, der in den Eingangsport des Moduls (In-Label) gespeist wird.

- **Trig:** Eventeingang, um das nächste Grain auszulösen. Werte > 0 starten unverzüglich das nächste Grain. Wert = -1 unterdrückt das nächste Grain (siehe Dist-Eingang).
- **Frz:** Eventeingang zum Einfrieren des Audiobuffers. Werte > 0 frieren den Buffer ein.

- **P**: Audioeingang zur logarithmischen Tonhöhenkontrolle (in Halbtönen). Die Tonhöhe ist unabhängig von der Abspielgeschwindigkeit. Typ. Bereich: [-20...20].
- **D/F**: Audioeingang zur Steuerung der Abspielrichtung, wenn **P** verbunden ist. Andernfalls dient er zur Frequenzsteuerung. Der Audiobuffer wird in der Originaltonhöhe wiedergegeben, wenn  $F=1$ , und rückwärts abgespielt, wenn  $F=-1$  ist. Typ. Bereich: [-4...4], Default: 1.
- **PJ**: Audioeingang für zufällige Tonhöhenabweichungen (Pitch Jitter) in Halbtönen. Typ. Bereich: [0...3].
- **PS**: Audioeingang zur logarithmischen Steuerung des Tonhöhenversatzes (Pitch Shift) des aktuellen Grains in Halbtönen. Typ. Bereich: [-3...3].
- **Dly**: Audioeingang zur Steuerung der Verzögerungszeit in ms. Erlaubter Bereich: [0...Bufferlänge].
- **Dlj**: Audioeingang für zufällige Abweichungen der Verzögerungszeit. Erlaubter Bereich: [0...Bufferlänge].
- **Len**: Audioeingang zur Einstellung der Grain-Länge in ms. Typ. Bereich: [10...100]. Default: 30 ms.
- **LnJ**: Audioeingang für die Steuerung der zufälligen Längenabweichung (Length Jitter) in ms. Typ. Bereich: [10...100]. Default: 0 ms.
- **Att**: Audioeingang zur Einstellung der Attackzeit. Bereich: [0...1]. Default: 0.2.
- **Dec**: Audioeingang zur Einstellung der Decayzeit. Bereich: [0...1]. Default: 0.2.
- **Dist**: Audioeingang zur Einstellung der Deltazeit für die Dauer bis zum Start des nächsten Grains in ms. Typ. Bereich: [5...100]. Default: 20.
- **DisJ**: Audioeingang für die Steuerung der zufälligen Deltazeitabweichung (Delta time Jitter). Typ. Bereich: [10...100]. Default: 20 ms.
- **Pan**: Audioeingang zur Einstellung der Position im Stereofeld. Bereich: [-1(Links)...1(Rchts)].
- **PnJ**: Audioeingang zur Steuerung der zufälligen Stereopositionsabweichung (Pan Jitter). Bereich: [0...1].
- **A**: Audioeingang zur Amplitudensteuerung. Typ. Bereich: [0...1]. Default: 1.
- **In**: Audioeingang für das zu verzögernde Audiosignal.
- **L**: Polyphoner Ausgang für den linken Stereokanal. Typ. Bereich: [-1...1].

- **R:** Polyphoner Ausgang für den rechten Stereokanal.  
Typ. Bereich: [-1...1].
- **Dly:** Polyphoner Eventausgang für die Verzögerungszeit bei jedem Grain-Start.
- **GTr:** Eventausgang für das Auslösen eines Grains. Gibt 1 aus, wenn ein Grain startet, 0, wenn es stoppt.

## Unit Delay

## Delay



Verzögert ein Audiosignal um die Zeit eines Audiosamples ( $1/\text{Samplerate}$ ). Jede Struktur, die irgendeine Art von Feedback verwendet, hat immer irgendwo in der Feedbackschleife ein Unit-Delay integriert. Wenn nicht explizit ein Unit-Delay-Modul verwendet wurde, fügt REAKTOR eigenständig eine solche Verzögerung irgendwo in der Schleife ein, deren Position durch einen vertikalen blauen Strich an einem Modul-Port sichtbar gemacht wird.

- In: Eingang für das Signal, dass um ein Audiosample verzögert werden soll.
- Out: Ausgang für das zu verzögernde Signal.

# Audio Modifier

Diese Audioverarbeitungsmodule erzeugen verschiedene Arten von Verzerrung (Shaping, Clipping und Mirroring zum Beispiel). Zusätzlich gibt es die Module Slew Limiter, Peak Detector, Sample and Hold, und das Frequency Divider Modul.

---

## Saturator

## Audio Modifier



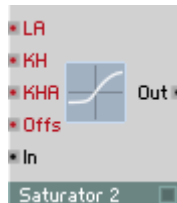
Verzerrer mit gerundeter Kennlinie für weiches Übersteuern in die Sättigung (Saturation). Der Ausgangswert wird begrenzt auf  $\pm 2$  (erreicht bei Eingangswerten größer  $\pm 4$ ). Sehr kleine Eingangswerte werden nicht verändert.

- **In:** Audio-Eingang für das zu begrenzende Signal.
- **Out:** Audio-Ausgang für das begrenzte Signal.

---

## Saturator 2

## Audio Modifier



Saturator 2 ist ein asymmetrischer, parabolischer Saturator vierter Ordnung, welcher den Zugriff auf die Verzerrungskurve ermöglicht.

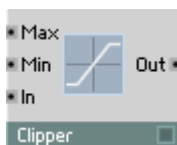
- **LA:** Level-Asymmetrie. Bei  $LA = 0$  sind die Sättigungs-Level für positive und negative Signale identisch. Für  $LA > 0$  wird der positive Level reduziert. Bei  $LA = 1$  ist es 0. Für  $LA < 0$  wird der negative Level reduziert. Bei  $LA = -1$  ist es 0.
- **KH:** Knee Hardness. Bei  $KH = 0$  steigt die Sättigung so sanft wie möglich an. Der vollständige Wertebereich zwischen 0 und dem Sättigungs-Level wird für die gerundete Kurve verwendet. Mit steigenden Werten für KH wird der Kurvenbereich reduziert auf  $(1 - KH)$  des Sättigungs-Levels. Bei  $KH = 1$  wird das Signal am Sättigungs-Level hart abgeschnitten (clipped).

- **KHA:** Knee Hardness-Asymmetrie. Bei  $KHA = 0$  ist die Knee Hardness für positive und negative Signale identisch. Für  $KHA > 0$  wird die Knee Hardness für positive Signale reduziert. Bei  $KHA = 1$  ist sie 0. Für  $KHA < 0$  wird die Knee Hardness für negative Signale reduziert. Bei  $KHA = -1$  ist sie 0.
- **Offs:** Dieser Eingang fügt dem Eingangssignal einen Offset hinzu und verschiebt es relativ zur Sättigungskurve. Für ein Null-Signal wird der Offset am Ausgang vollständig kompensiert.
- **In:** Eingang für das zu verzerrende Signal.
- **Out:** Ausgang für das verzerrte Signal.

---

## Clipper

## Audio Modifier



Verzerrer mit harter Begrenzungskennlinie und einstellbarer oberer und unterer Begrenzung. Wenn das Eingangssignal den Maximalwert übersteigt, wird es auf diesen Wert begrenzt (Clipping). Wenn es den Minimalwert unterschreitet, wird es auf jenen Wert begrenzt. Signalwerte dazwischen werden unverändert ausgegeben.

- **Max:** Audiosignal-Steuereingang für den Maximalwert des Signals.
- **Min:** Audiosignal-Steuereingang für den Minimalwert des Signals.
- **In:** Audio-Eingang für das zu begrenzende Signal.
- **Out:** Audio-Ausgang für das begrenzte Signal.

---

## Mod. Clipper

## Audio Modifier



Verzerrer mit harter Begrenzungskennlinie und modulierbarer Begrenzung. Wenn der Betrag des Eingangssignals den Maximalwert übersteigt, wird es auf diesen Wert begrenzt. Signalwerte mit kleinerem Betrag werden unverändert ausgegeben.



- **M:** Audiosignal-Steuereingang für den maximalen Betrag des Signals.
- **In:** Audio-Eingang für das zu begrenzende Signal.
- **Out:** Audio-Ausgang für das begrenzte Signal.

---

## Mirror 1 Level

## Audio Modifier



Signalwert-Spiegeler mit einstellbarem Spiegelwert. Signalwerte, die über dem Spiegelwert liegen, werden an diesem „reflektiert“, so daß sie nun kleiner sind. Kleinere Signalwerte werden unverändert ausgegeben.

- **Max:** Audiosignal-Steuereingang für den Spiegelwert.
- **In:** Audio-Eingang für das zu modifizierende Signal.
- **Out:** Audio-Ausgang für das modifizierte Signal.

---

## Mirror 2 Levels

## Audio Modifier



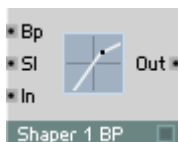
Signalwert-Doppelspiegeler mit einstellbaren Spiegelwerten. Signalwerte, die über dem oberen Spiegelwert liegen, werden an diesem nach unten „reflektiert“. Signalwerte, die unter dem unteren Spiegelwert liegen, werden nach oben gespiegelt. Signalwerte in der Mitte werden unverändert ausgegeben.

- **Max:** Audiosignal-Steuereingang für den oberen Spiegelwert.
- **Min:** Audiosignal-Steuereingang für den unteren Spiegelwert.
- **In:** Audio-Eingang für das zu modifizierende Signal.
- **Out:** Audio-Ausgang für das modifizierte Signal.



Chopper-Modulator, der die Verstärkung des Eingangssignals zwischen einem variablen Wert und 1 hin und her schaltet. Wenn das Modulationssignal positiv ist, wird der Eingangswert mit dem Wert **X** multipliziert, bei negativem Modulationssignal wird der Eingangswert unverändert ausgegeben.

- **M**: Audio-Eingang für das Modulationssignal (nur das Vorzeichen ist relevant).
- **X**: Audiosignal-Steuereingang für den Verstärkungsfaktor bei positivem **M**.
- **In**: Audio-Eingang für das zu modulierende Signal.
- **Out**: Audio-Ausgang für das modulierte Signal.



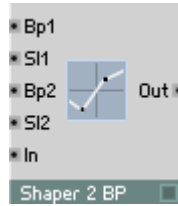
Signalformer mit stückweise linearer Kennlinie mit einem Breakpoint. Die Steigung der Kennlinie über dem Breakpoint läßt sich einstellen. Eingangssignalswerte unter dem Breakpoint werden unverändert ausgegeben.

- **Bp**: Audiosignal-Steuereingang für den Breakpoint-Wert.
- **Sl**: Audiosignal-Steuereingang für die Steigung des oberen Teils der Kennlinie (1 = keine Signaländerung).
- **In**: Audio-Eingang für das zu formende Signal.
- **Out**: Audio-Ausgang für das geformte Signal.

---

## Shaper 2 BP

## Audio Modifier



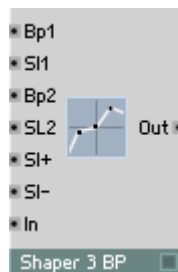
Signalformer mit stückweise linearer Kennlinie mit zwei Breakpoints. Die Steigungen der Kennlinien über dem oberen und unter dem unteren Breakpoint lassen sich einstellen. Eingangssignalwerte zwischen den Breakpoints werden unverändert ausgegeben.

- **Bp1:** Audiosignal-Steuereingang für den oberen Breakpoint-Wert.
- **SI1:** Audiosignal-Steuereingang für die Steigung des oberen Teils der Kennlinie (1 = keine Signal-Änderung).
- **Bp2:** Audiosignal-Steuereingang für den unteren Breakpoint-Wert.
- **SI2:** Audiosignal-Steuereingang für die Steigung des unteren Teils der Kennlinie (1 = keine Signal-Änderung).
- **In:** Audio-Eingang für das zu formende Signal.
- **Out:** Audio-Ausgang für das geformte Signal.

---

## Shaper 3 BP

## Audio Modifier



Signalformer mit stückweise linearer Kennlinie mit drei Breakpoints. Die Steigungen der Kennlinien über dem oberen Breakpoint, unter dem unteren Breakpoint und zwischen den Breakpoints und dem Nullpunkt lassen sich einstellen.

- **Bp1:** Audiosignal-Steuereingang für den oberen Breakpoint-Wert.
- **SI1:** Audiosignal-Steuereingang für die Steigung des oberen Teils der Kennlinie.

- **Bp2:** Audiosignal-Steuereingang für den unteren Breakpoint-Wert.
- **SI2:** Audiosignal-Steuereingang für die Steigung des unteren Teils der Kennlinie.
- **SI+:** Audiosignal-Steuereingang für die Steigung der Kennlinie zwischen dem Nullpunkt und dem oberen Breakpoint (1 = keine Signal-Änderung).
- **SI-:** Audiosignal-Steuereingang für die Steigung der Kennlinie zwischen dem unteren Breakpoint und dem Nullpunkt (1 = keine Signal-Änderung).
- **In:** Audio-Eingang für das zu formende Signal.
- **Out:** Audio-Ausgang für das geformte Signal.

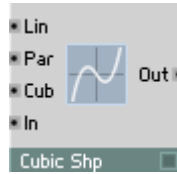
## Shaper Parabolic

## Audio Modifier



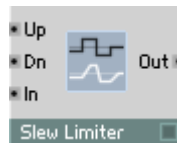
Signalformer mit parabolischer Kennlinie (Polynom 2. Ordnung). Der lineare und der quadratische Signalanteil lassen sich unabhängig voneinander einstellen. (**Out** = **Lin In** + **Par In**<sup>2</sup>)

- **Lin:** Audiosignal-Steuereingang für den linearen unverzerrten Anteil.
- **Par:** Audiosignal-Steuereingang für den quadratischen verzerrten Anteil.
- **In:** Audio-Eingang für das zu formende Signal.
- **Out:** Audio-Ausgang für das geformte Signal.



Signalformer mit kubisch-parabolischer Kennlinie (Polynom 3. Ordnung). Der lineare, quadratische und kubische Signalanteil lassen sich unabhängig voneinander einstellen. ( $\text{Out} = \text{Lin In} + \text{Par In}^2 + \text{Cub In}^3$ )

- **Lin:** Audiosignal-Steuereingang für den linearen unverzerrten Anteil.
- **Par:** Audiosignal-Steuereingang für den quadratischen verzerrten Anteil.
- **Cub:** Audiosignal-Steuereingang für den kubischen verzerrten Anteil.
- **In:** Audio-Eingang für das zu formende Signal.
- **Out:** Audio-Ausgang für das geformte Signal.



Begrenzer mit getrennt einstellbarer maximaler Rate (Slew) für positive und negative Richtungen. Das Ausgangssignal folgt dem Eingangssignal, bei schnellen Signaländerungen und Sprüngen folgt der Ausgang jedoch nur mit einer begrenzten Steigung (Rampe), bis er das Eingangssignal wieder erreicht hat.

- **Up:** Audiosignal-Steuereingang für die maximale Rate (in 1/sec) bei ansteigenden Signalen.
- **Dn:** Audiosignal-Steuereingang für die maximale Rate (in 1/sec) bei abfallenden Signalen.
- **In:** Audio-Eingang für das zu begrenzende Signal.
- **Out:** Audio-Ausgang für das begrenzte Signal.

---

## Peak Detector

## Audio Modifier



Spitzenwert-Gleichrichter mit Glättung.

Die Anstiegszeit ist gleich 0, die Rücklaufzeit regelbar.

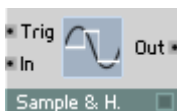
Das Wirkung des Peak Detectors ist, daß der Ausgangswert der Amplitudenhüllkurve des Eingangssignals folgt – verwenden Sie dieses Modul als Hüllkurvenfolger (Envelope Follower).

- **In:** Audio-Eingang für das gleichzurichtende Signal.
- **Rel:** Steuereingang für die Rücklaufzeit.
- **Out:** Audio-Ausgang für das gleichgerichtete Signal.

---

## Sample & Hold

## Audio Modifier



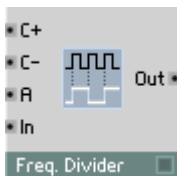
Abtast- und Halteglied mit Takteingang. Wenn das Taktsignal (Clock) über 0 steigt, wird der aktuelle Eingangswert an den Ausgang gelegt und bis zum nächsten Taktimpuls gehalten. So ergibt sich am Ausgang eine Schrittwellenform.

- **C:** Audio-Eingang für das Taktsignal (Clock). Der Eingang wird bei einer steigenden Flanke hier abgetastet.
- **In:** Audio-Eingang für das abzutastende Signal.
- **Out:** Audio-Ausgang für das abgetastete Signal.

---

## Frequency Divider

## Audio Modifier



Frequenzteiler (Rechteck-SubOscillator) mit unabhängig einstellbarer Zeit für die beiden Halbwellen. Durch Zählen der Nulldurchgänge wird eine

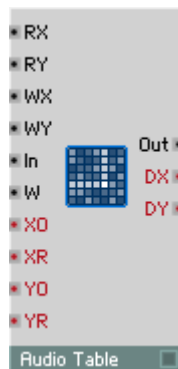
Rechteck-wellenform erzeugt, deren Frequenz ein Bruchteil der Frequenz des Eingangssignals ist. Das Frequenzverhältnis läßt sich einstellen ( $f_{\text{out}} = 2 f_{\text{in}} / (\mathbf{C+} + \mathbf{C-})$ ).

Eine asymmetrische Wellenform ergibt sich, wenn **C+** und **C-** unterschiedlich gewählt werden.

- **C+**: Steuereingang für die Anzahl der Nulldurchgänge des Eingangssignals während der Hochphase des Ausgangssignals.
- **C-**: Steuereingang für die Anzahl der Nulldurchgänge des Eingangssignals während der Tiefphase des Ausgangssignals.
- **A**: Steuereingang für die Ausgangsamplitude.
- **In**: Audio-Eingang für das zu teilende Signal.
- **Out**: Audio-Ausgang für die Rechteckwelle niedriger Frequenz.

## Audio Table

## Audio Modifier



Enthält eine Tabelle von Werten. Die Tabelle kann als Audiosignal ausgelesen, Audio kann in der Tabelle gespeichert werden, und der Inhalt der Tabelle kann angezeigt und graphisch editiert werden. Die Tabelle kann 1-dimensional (eine Reihe von Werten, die mit X adressiert werden) oder 2-dimensional (eine Matrix von Reihen und Spalten, welche mit X und Y adressiert werden) verwendet werden.

Der ausgegebene Wert wird durch die Leseposition bestimmt, welche sich mit den Eingängen **RX** und **RY** bestimmt. Ein Signal, das am Eingang **In** anliegt, wird in einzelnen Zellen der Tabelle entsprechend der Schreibposition, welche durch die Eingänge **WX** und **WY** vorgegeben werden, gespeichert, X ist die horizontale Position von links nach rechts, Y die vertikale Position von oben nach unten. Die Zählung beginnt immer mit 0 für das erste Element.

Die Panelanzeige für das Modul kann alle Daten oder diejenigen eines festgelegten Bereichs anzeigen. Viele Optionen in den Properties erlauben die Festlegung des Verhaltens durch den Benutzer.

Eine vollständigen Beschreibung der Properties, Menüs und Keyboard-Shortcuts lesen Sie im Abschnitt *Die Table-Module* ab Seite 260.

- **RX**: Audioeingang für die X-Position der Tabellenzelle, von welcher Daten gelesen werden.
- **RY**: Audioeingang für die Y-Position der Tabellenzelle, von welcher Daten gelesen werden. Dieser wird im 2D-Modus verwendet oder um die Reihenummer zu adressieren, wenn mehr als eine Reihe existiert.
- **WX**: Audioeingang für die X-Position der Tabellenzelle, in welche Daten geschrieben werden.
- **WY**: Audioeingang, um den Schreibvorgang auszulösen.
- **W**: Audioeingang zum Auslösen eines Schreibvorgangs an der Position.
- **In**: Audioeingang für das Signal, welches in die Tabelle geschrieben wird. Wenn der Wert **W** größer als 0 ist, wird der an **In** anliegende Wert in die Tabelle unter der durch **WX** und **WY** festgelegten Position geschrieben.
- **XO**: Eventeingang für die horizontale Verschiebung des angezeigten Datenbereichs. **XO** steuert die Datenposition, welche in der Anzeige erscheint (gemäß der Ansichtsausrichtung). Der Wert von **XO** wird in Einheiten (Units) in den Properties spezifiziert.
- **XR**: Eventeingang für den horizontalen Anzeigebereich der Daten. **XR** legt fest, wie viele Dateneinheiten in die Anzeige passen, und ermöglicht dadurch die Vergrößerung und Verkleinerung des Ausschnitts (Zoom).
- **YO**: Eventeingang für die vertikale Verschiebung des angezeigten Datenbereichs. **YO** steuert die Datenposition, welche in der Anzeige erscheint (gemäß der Ansichtsausrichtung). Der Wert von **YO** wird in Einheiten in den Properties spezifiziert
- **YR**: Eventeingang für den vertikalen Anzeigebereich der Daten im 2D-Modus. **YR** legt fest, wie viele Dateneinheiten in die Anzeige passen, und ermöglicht dadurch die Vergrößerung und Verkleinerung des Ausschnitts (Zoom).
- **Out**: Audioausgabe der durch die Eingänge **RX**, **RY** und **R** bestimmten Zelle.



- **DX:** Eventausgang für die Länge der horizontalen Tabellenreihe in Einheiten.
- **DY:** Eventausgang für die Höhe der vertikalen Tabellenspalte in Einheiten..

## Event Processing

Event-Module können als Zähler (Counter) dienen, für logische Operationen, zum Aufteilen und Verbinden von Eventsignalen und für das Messen einer Zeit zwischen zwei Ereignissen (Timer) verwendet werden. Sie finden hier auch Module zum Formen und Randomisieren von Eventsignalen. Schliesslich gibt es ein voll ausgestattetes Table-Modul für Events - äquivalent zum Audio-Table in der Oscillator-Sektion.

---

### Accumulator

### Event Processing



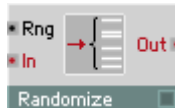
Akkumulator (Summe) für Event-Werte. Der Wert jedes einzelnen Event am Eingang wird zu der intern gehaltenen Summe hinzuaddiert. Der neue Wert der Summe wird als Event am Ausgang ausgegeben.

- **In:** Eingang für die zu summierenden Event.
- **Set:** Event-Eingang zum (zurück-)setzen der internen Summe. Der Wert eines Events and diesem Eingang bestimmt den neuen Wert der internen Summe.
- **Out:** Event-Ausgang für den Summenwert.



Zähler mit Event-Steuerung. Ein positiver Event am entsprechenden Eingang zählt den Ausgangswert um eins nach oben oder unten.

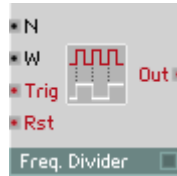
- **Up:** Eingang zum Hochzählen.  
Nur Events mit einem positiven Wert (nicht Null) haben hier eine Wirkung und erhöhen den Ausgangswert um eins.
- **Dwn:** Eingang zum Runterzählen.  
Nur Events mit einem positiven Wert (nicht Null) haben hier eine Wirkung und verringern den Ausgangswert um eins.
- **Set:** Event-Eingang zum (zurück-)setzen des Zählerwertes. Der Wert eines Events and diesem Eingang bestimmt den neuen Wert des Zählers.
- **Out:** Event-Ausgang für den Zählerwert.



Event-Zufallsmodifizierer mit einstellbarer Streubreite.

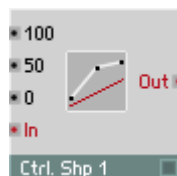
Die ankommenden Events werden mit einer zufälligen Wertänderung, deren Bereich einstellbar ist, ausgegeben (**Out** = **In** + X, wobei  $-\mathbf{Rng} \leq X \leq \mathbf{Rng}$ ).

- **Rng:** Audiosignal-Steuereingang für den Bereich der zufälligen Wertänderung.
- **In:** Event-Eingang für das zu modifizierende Steuersignal.
- **Out:** Event-Ausgang für das modifizierte Steuersignal.



Frequenzteiler für Event-Signale. Das Ausgangs-Signal kehrt nach einer von der Pulsweite **W** abhängigen Zahl von Input-Events auf Null zurück.

- **N**: Steuereingang für den Teilerfaktor ( $N < 2$  : keine Teilung, 2 ... 2.99 : Teilfaktor = 2, 3 ... 3.99 : Teilfaktor = 3, etc.).
- **W**: Steuereingang für die Pulsweite des Ausgangssignals. Wertebereich: 0 ... 1 (0% ... 100%). **W** = 0 : das Gate-Signal an **Out** geht beim nächsten Event an **In** schon wieder auf Null; **W** = 0.5 : das Gate-Signal an **Out** geht nach der halben Periode zurück auf Null; **W** = 1 : das Gate-Signal an **Out** bleibt die ganze Zeit an.
- **In**: Eingang für das in der Frequenz zu teilende Event-Signal (z. B. MIDI-Clock). Nur Events mit positivem Wert haben hier eine Wirkung.
- **Rst**: Event Eingang zum Zurücksetzen (Reset) des internen Zählers, um einen synchronen Start zu erzwingen (z. B. mit einem MIDI-Start-Signal ansteuern). Setzt auch den Wert an **Out** auf Null zurück. Events mit Wert kleiner oder gleich Null werden ignoriert, nur positive Events bewirken einen Reset.
- **Out**: Ausgang für das frequenzgeteilte Event-Signal.



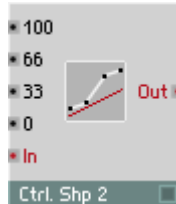
Steuersignal-Former mit stückweise linearer Kennlinie mit einem festen Breakpoint. Zwischen den eingestellten Werten wird linear interpoliert.

- **100**: Ausgangswert bei **In** = 1 (100%).
- **50**: Ausgangswert für den Breakpoint bei **In** = 0.5 (50%).
- **0**: Ausgangswert bei **In** = 0 (0%).

- **In:** Event-Eingang für das zu formende Steuersignal.
- **Out:** Event-Ausgang für das geformte Steuersignal.

## Ctrl. Shaper 2 BP

## Event Processing

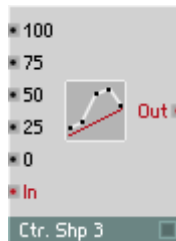


Steuersignal-Former mit stückweise linearer Kennlinie mit zwei festen Breakpoints. Zwischen den eingestellten Werten wird linear interpoliert.

- **100:** Ausgangswert bei **In** = 1 (100%).
- **66:** Ausgangswert für den oberen Breakpoint bei **In** = 0.66 (66%).
- **33:** Ausgangswert für den unteren Breakpoint bei **In** = 0.33 (33%).
- **0:** Ausgangswert bei **In** = 0 (0%).
- **In:** Event-Eingang für das zu formende Steuersignal.
- **Out:** Event-Ausgang für das geformte Steuersignal.

## Ctrl. Shaper 3 BP

## Event Processing



Steuersignal-Former mit stückweise linearer Kennlinie mit drei festen Breakpoints. Zwischen den eingestellten Werten wird linear interpoliert.

- **100:** Ausgangswert bei **In** = 1 (100%).
- **75:** Ausgangswert für den dritten Breakpoint bei **In** = 0.75 (75%).
- **50:** Ausgangswert für den zweiten Breakpoint bei **In** = 0.5 (50%).
- **25:** Ausgangswert für den ersten Breakpoint bei **In** = 0.25 (25%).

- **0**: Ausgangswert bei **In** = 0 (0%).
- **In**: Event-Eingang für das zu formende Steuersignal.
- **Out**: Event-Ausgang für das geformte Steuersignal.

---

## Logic AND

## Event Processing



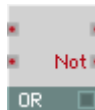
Logik Gatter für Eventsignale. Der Ausgangswert ist die logische UND-Verknüpfung der beiden Eingangssignale, d. h. der Ausgangswert ist 1, wenn beide Eingänge positiv sind, sonst ist der Ausgangswert 0.

Die Eingänge behandeln die Werte Null und darunter als den logischen Zustand "falsch" (0), Werte über Null sind logisch "wahr" (1). Der Ausgang **Not** ist die logische Negation des Ausgangs **Out**, also **A** and **B**.

---

## Logic OR

## Event Processing



Logik Gatter für Eventsignale. Der Ausgangswert ist die logische ODER-Verknüpfung der beiden Eingangssignale, d. h. der Ausgangswert ist 1, wenn einer oder beide der Eingänge positiv sind, sonst ist der Ausgangswert 0.

Die Eingänge behandeln die Werte Null und darunter als den logischen Zustand "falsch" (0), Werte über Null sind logisch "wahr" (1).

Der Ausgang **Not** ist die logische Negation des Ausgangs **Out**, also **A** nor **B**.

---

## Logic EXOR

## Event Processing



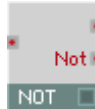
Logik Gatter für Eventsignale. Der Ausgangswert ist die logische EXOR-Verknüpfung (exklusives ODER) der beiden Eingangssignale, d. h. der Ausgangswert ist 1, wenn einer aber nicht beide Eingänge positiv sind, sonst ist

der Ausgangswert 0. Effektiv invertiert also der eine Eingang den anderen. Die Eingänge behandeln die Werte Null und darunter als den logischen Zustand "falsch" (0), Werte über Null sind logisch "wahr" (1). Der Ausgang **Not** ist die logische Negation des Ausgangs **Out**.

---

## Logic NOT

## Event Processing



Logik Gatter für Eventsignale. Der Ausgangswert ist die logische Negation des Eingangssignals, d. h. der Ausgangswert ist 0, wenn der Eingang positiv ist, sonst ist der Ausgangswert 1.

Der Eingang behandeln die Werte Null und darunter als den logischen Zustand "falsch" (0), Werte über Null sind logisch "wahr" (1).

Der Ausgang **Not** ist die logische Negation des Ausgangs **Out**, hat also logisch den gleichen Wert wie der Eingang.

---

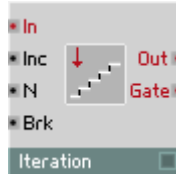
## Order

## Event Processing



Event-Reihenfolge. Ein am Eingang ankommendes Event wird unverändert an alle Ausgänge weitergegeben, aber in einer festgelegten Reihenfolge: Als erstes an den Ausgang **1**, dann an **2**, zuletzt an **3**. Das Event wandert durch ALLE Module, die mit dem Ausgang **1** verbunden sind, bevor es zu dem ersten Modul weitergegeben wird, welches an **2** angeschlossen ist etc. Siehe auch den Abschnitt *Eventsignale* (17.7)

- **In:** Eingang für Events, die in einer bestimmten Reihenfolge weitergesendet werden sollen.
- **1:** Ein Event wird an diesem Ausgang zuerst gesendet.
- **2:** Ein Event wird an diesem Ausgang als zweites gesendet.
- **3:** Ein Event wird an diesem Ausgang als drittes gesendet.



Ein am Trg-Eingang ankommendes Event wird an den Ausgang weitergeleitet und löst eine Serie von N zusätzlichen Ausgangsevents aus. Jedes folgende Event besitzt den Wert seines Vorgängers, um den an Inc anliegenden Wert inkrementiert. Alle Events werden abgearbeitet, bevor das nächste Audiosample berechnet wird. Dieses Modul kann für alle Operationen, bei denen eine sich wiederholende Event-Berechnung benötigt wird, verwendet werden. Es hilft bei der Vermeidung von Eventloops, welche zu einem instabilen Verhalten von REAKTOR führen können.

- Trg: Trigger-Eingang. Ein Event an diesem Eingang löst N+1 Events am Ausgang aus.
- Inc: Wertzuwachs für jedes folgende Event.
- N: Anzahl zusätzlicher Ausgangs-Events. Der Wert muss grösser oder gleich einer ganzen Zahl sein (Nachkommastellen werden abgeschnitten).

---

## Separator

## Event Processing



Event-Weiche. Ankommende Events werden mit dem Schwellwert (Threshold) verglichen und entweder zum einen oder zum anderen Ausgang geschickt.

Mit dem Separator kann z. B. ein Gate-Signal in ein Triggersignal gewandelt werden, indem die Null-Events herausgefiltert werden (**Thld** = 0, **Hi** = Triggerausgang).

- **Thld**: Audiosignal-Steuereingang für den Schwellwert (Threshold).
- **In**: Eingang für die auf zwei Ausgänge zu verteilenden Events.
- **Hi**: Ausgang für die Events, die größer als der Schwellwert sind.
- **Lo**: Ausgang für die Events, die kleiner oder genau so groß wie der Schwellwert sind.



Wert-Ersetzer für Events. Die Events, die am Eingang **In** ankommen, werden mit einem neuen Wert, der ihnen vom **Val** Eingang aufgeprägt wird, an **Out** ausgegeben.

Diese Modul kann auch als Event-gesteuertes Sample&Hold benutzt werden.

- **In:** Eingang für Events, deren Wert geändert werden soll.
- **Val:** Eingang zur Bestimmung des neuen Werts für die Events.
- **Out:** Ausgang für die Event mit geändertem Wert.

---

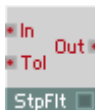
**Merge****Event Processing**

Zusammenführer für Eventsignale. Wenn mehr als ein Wire mit dem Eingang verbunden ist, wird als Ausgangswert immer der zuletzt empfangene Event ausgegeben, egal von welchem Wire er kam.

Das Modul besitzt eine dynamische Eingangsport-Verwaltung. Die Anzahl der Eingänge kann mit Min Num Port Groups auf der Function-Seite der Properties definiert werden.

Anders als in früheren REAKTOR-Version werden Folge-Events mit gleichem Wert nicht mehr herausgefiltert. Zu diesem Zweck verwenden Sie bitte das Modul Step Filter.

---

**Step Filter****Event Processing**

Ein Event wird nur dann durchgelassen und an den Ausgang weitergeleitet, wenn der Eingangswert grösser/kleiner als der vorhergehende Eingangswert ist +/- Toleranz.



- In: Eingang für die zu filternden Events.
- Tol: Eingang für den Toleranzgrad.
- Out: Eventausgang.

---

## Router M->1

## Event Processing



Router zwischen multiplen Eingängen auf einen Ausgang. Die Events am ausgewählten Eingang werden durchgelassen, alle anderen werden herausgefiltert. Die Eingänge werden mittels des Pos-Eingang ausgewählt. Wenn der **Wrap**-Modus in den Properties aktiviert ist, arbeitet **Pos** als Schleife, so daß Max +1 als 0, Max +2 als 1 usw. interpretiert wird.

Das Modul besitzt eine dynamische Eingangsport-Verwaltung. Die Anzahl der Eingänge kann mit Min Num Port Groups auf der Function-Seite der Properties definiert werden.

- Pos: Eingang zur Auswahl des durchzuschleifenden Eingangs.
- In: Signaleingang.
- Out: Ausgang für den ausgewählten Signaleingang.

---

## Router 1,2

## Event Processing



Ein-Aus-Schalter oder Umschalter für Event-Signale. Der letzte durchgeschaltete Wert wird am Ausgang gehalten.

Das Modul besitzt eine dynamische Eingangsport-Verwaltung. Die Anzahl der Eingänge kann mit Min Num Port Groups auf der Function-Seite der Properties zwischen 1 und 2 definiert werden.

- **Ctrl**: Hybrideingang zur Schaltersteuerung. Ein Wert größer 0 leitet alle Events am Eingang auf den Ausgang weiter.
- **In**: Event-Eingang für das zu schaltende Signal.
- **Out**: Event-Ausgang für das geschaltete Signal. Solange **Ctrl** kleiner 0 ist, wird am Ausgang der letzte durchgeschaltete Wert gehalten.



Router eines Eingangs auf multiple Ausgänge. Die Events am Eingang werden zum ausgewählten Ausgang durchgelassen. Der Ausgang wird mittels des Pos-Eingang ausgewählt. Wenn der **Wrap**-Modus in den Properties aktiviert ist, arbeitet **Pos** als Schleife, so daß Max +1 als 0, Max +2 als 1 usw. interpretiert wird.

Das Modul besitzt eine dynamische Ausgangsport-Verwaltung. Die Anzahl der Ausgänge kann mit Min Num Port Groups auf der Function-Seite der Properties definiert werden.

- Pos: Eingang zur Auswahl des Ausgangs, zu welchen das Eingangssignal durchgeschleift werden soll.
- In: Signaleingang.
- Out: Ausgang für das Eingangssignal.



Zeit- und Frequenzmesser.

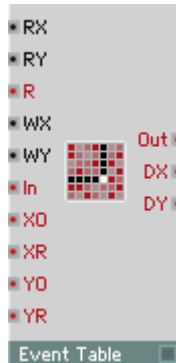
Die verstrichene Zeit zwischen den beiden zuletzt empfangenen Events wird gemessen und ausgegeben. Außerdem wird die Frequenz, deren Schwingungs-Periode dem gemessenen Zeitintervall entspricht, berechnet und ausgegeben.

- **In**: Polyphoner Event-Eingang für die Events, deren zeitlicher Abstand zu messen ist.
- **F**: Polyphoner Event-Ausgang für die Event-Frequenz in Hz.
- **T**: Polyphoner Event-Ausgang für die Zeit zwischen den Events in Millisekunden.



Halte-Hüllkurve für Events. Wenn das Modul von einem positiven Event am **T**-Eingang ausgelöst wird, wird der Wert des Events am Ausgang ausgegeben und gehalten, bis die Halte-Zeit abgelaufen ist, wonach der Ausgang auf Null zurückspringt. Das Modul kann auch während der Haltephase neu ausgelöst werden (retrigger).

- **G**: Event-Eingang zum Auslösen (trigger) der Halte-Hüllkurve. Nur Events mit einem positiven Wert haben hier eine Wirkung.
- **H**: Steuereingang für die Haltezeit (hold) in Millisekunden.
- **Out**: Event-Ausgang für das Hüllkurvensignal.



Enthält eine Tabelle von Werten. Eventwerte können aus der Tabelle ausgelesen und in der Tabelle gespeichert werden, und der Inhalt der Tabelle kann angezeigt und graphisch editiert werden. Die Tabelle kann 1-dimensional (eine Reihe von Werten, die mit X adressiert werden) oder 2-dimensional (eine Matrix von Reihen und Spalten, welche mit X und Y adressiert werden) verwendet werden.

Der ausgegebene Wert wird durch die Leseposition bestimmt, welche sich mit den Eingängen **RX** und **RY** bestimmt. Werte, die am Eingang **W** ankommen, werden in einzelnen Zellen der Tabelle entsprechend der Schreibposition, welche durch die Eingänge **WX** und **WY** vorgegeben werden, gespeichert,

X ist die horizontale Position von links nach rechts, Y die vertikale Position von oben nach unten. Die Zählung beginnt immer mit 0 für das erste Element.

Die Panelanzeige für das Modul kann alle Daten oder diejenigen eines festgelegten Bereichs anzeigen. Viele Optionen in den Properties erlauben die Festlegung des Verhaltens durch den Benutzer.

Eine vollständigen Beschreibung der Properties, Menüs und Keyboard-Shortcuts lesen Sie den Abschnitt *Die Table-Module* ab Seite 260.

- **RX**: Audioeingang für die X-Position der Tabellenzelle, von welcher Daten gelesen werden.
- **RY**: Audioeingang für die Y-Position der Tabellenzelle, von welcher Daten gelesen werden. Dieser wird im 2D-Modus verwendet oder um die Reihennummer zu adressieren, wenn mehr als eine Reihe existiert.
- **R**: Eventeingang zum Auslösen eines Lesevorgangs an der Position, die durch **RX** and **RY** bestimmt wird. Jedes Event hier erzeugt ein Event am **Out**-Ausgang
- **WX**: Audioeingang für die X-Position der Tabellenzelle, in welche Daten geschrieben werden.
- **WY**: Audioeingang für die Y-Position der Tabellenzelle, in welche Daten geschrieben werden.
- **In**: Eventeingang für das Schreiben in die Tabelle an der Position, die durch **WX** and **WY** bestimmt wird. Der Datenwert, welcher in die Tabellenzelle geschrieben wird, ist der Wert des Events, welcher am Eingang **In** ankommt.
- **XO**: Eventeingang für die horizontale Verschiebung des angezeigten Datenbereichs. **XO** steuert die Datenposition, welche in der Anzeige erscheint (gemäß der Ansichtsausrichtung). Der Wert von **XO** wird in Einheiten (Units) in den Properties spezifiziert.
- **XR**: Eventeingang für den horizontalen Anzeigebereich der Daten. **XR** legt fest, wie viele Dateneinheiten in die Anzeige passen, und ermöglicht dadurch die Vergrößerung und Verkleinerung des Ausschnitts (Zoom).
- **YO**: Eventeingang für die vertikale Verschiebung des angezeigten Datenbereichs. **YO** steuert die Datenposition, welche in der Anzeige erscheint (gemäß der Ansichtsausrichtung). Der Wert von **YO** wird in Einheiten in den Properties spezifiziert
- **YR**: Eventeingang für den vertikalen Anzeigebereich der Daten im 2D-Modus. **YR** legt fest, wie viele Dateneinheiten in die Anzeige passen, und ermöglicht dadurch die Vergrößerung und Verkleinerung des Ausschnitts (Zoom).

- **Out:** Eventausgabe von Daten aus der durch die Eingänge **RX**, **RY** und **R** bestimmten Zelle.
- **DX:** Eventausgang für die Länge der horizontalen Tabellenreihe in Einheiten.
- **DY:** Eventausgang für die Höhe der vertikalen Tabellenspalte in Einheiten..

# Auxiliary

Hier finden Sie die Tapedecks zur Aufnahme und zum Abspielen von Audio-dateien. Dann gibt es Module zur Verwaltung von Mehrstimmigkeit, zum Konver-tieren von Audiosignalen in Eventsignale, sowie für die Verbindung zu verschiedenen globalen REAKTOR-Funktionen wie Tuning oder Master-Tempo.

## Tapedeck 1-Ch

## Auxiliary



1-Kanal-Recorder zur Aufnahme und Wiedergabe von Audiosignalen. Audio-Dateien können entweder aus dem Arbeitsspeicher oder von der Festplatte wiedergegeben und auch dahin geschrieben werden, abhängig von der entsprechenden Properties-Einstellung.

Im Harddisk-Modus werden Dateien in demjenigen Ordner aufgenommen, welchen Sie in den Reaktor-Preferences zu diesem Zwecke angeben. Reaktor vollzieht dabei eine Sampleraten-Konvertierung in Echtzeit und schreibt die Datei in der von Ihrem Audiosystem verwendeten Samplerate.

Im Memory-Modus können Sie die Wellenform der geladenen Audiodatei im Panel anzeigen, indem Sie die Checkbox **Picture** auf der **Appearance**-Seite in den Properties ankreuzen. Die gesamte Wellenform wird dann automatisch auf die Größe des Displays im Panel skaliert, welche Sie unter **Appearance** mit den **Size**-Werten vorgeben.

### Memory-Modus

Wenn Sie das Kästchen **Keep audio only in memory** in den Properties ankreuzen, springt das Tapedeck in den Memory-Modus, und die Memory-Sektion mit den Buttons **Save**, **Save as...** und **Reload** wird aktiv.

Die maximale Aufnahmezeit wird im Properties-Dialog des Moduls unter **Max Recording Size (sec)** eingestellt (in Sekunden). Wie groß dieser Wert

eingestellt werden kann, hängt vom verfügbaren RAM des Rechners ab. Bei einer Abtastrate von 44,1 kHz braucht man 86 kB RAM pro Sekunde Pufferlänge, für eine Minute braucht man 5 MB. Wenn man den Speicher für das Tapedeck zu groß wählt, kann es bei der Aufnahme durch Virtual-Memory-Aktivitäten des Betriebssystems zu erheblicher Festplattenaktivität kommen, was REAKTOR bei der Audiodatenberechnung erheblich behindern kann.

Audiodateien können mittels des Buttons **Select File...** in den Properties für die Wiedergabe importiert werden. Eine bereits importierte Datei kann mit dem **Reload-Button** neu geladen werden, wenn sie z. B. in einem Sample-Editor verändert worden ist

Beim Import einer Audiodatei wird die Länge des Audiopuffers an die Daten angepaßt. Wenn man später eine Aufnahme machen will, die länger dauert, muss man erst im Properties-Dialog des Tapedecks unter **Max Recording Size (sec)** einen größeren Wert eingeben. Die geladene Audiodatei wird automatisch auf die aktuelle Samplerate von Reaktor gesetzt.

---

Hinweis: Bedenken Sie bitte, daß Reaktor im Memory-Modus die Samplerate aller Audiodateien, welche sich im Tapedeck befinden, ändert und auf die neue Samplerate setzt. Sie sollten also das Umstellen der Samplerate vermeiden, wenn sich Audiodateien im Arbeitsspeicher befinden, welche von den Tapedecks benutzt werden. Wenn sich die gleiche Audiodatei auf der Festplatte befindet, können Sie mit Hilfe des **Reload**-Buttons diese nach der Sampleraten-Änderung neu laden.

---

Eine Aufnahme wird mit dem **Save**-Button in den Properties exportiert. Wenn eine Datei unter demselben Namen bereits auf der Festplatte existiert, werden Sie gefragt, ob Sie diese überschreiben möchten. Mit **Save as...** können Sie die Datei unter einem neuen Namen speichern.

## Harddisk-Modus

Wenn Sie das Kästchen **Stream audio from/to harddisk** in den Properties ankreuzen, springt das Tapedeck in den Harddisk-Modus. Audiodateien werden dann direkt auf die Festplatte geschrieben und von dort abgespielt. Sie können eine Audiodatei für die Wiedergabe mit einem Klick auf den **Select File...**-Button auswählen. Mit dem **New File**-Button erzeugen Sie eine neue Datei auf der Festplatte mit dem Namen „untitled“ (existiert eine Datei mit diesem Namen schon auf der Festplatte, wird dem Namen „untitled“ noch eine Nummer angehängt). Sie können den Namen der Datei direkt im Namensfeld der Properties editieren.

Wenn im Properties-Dialog **Value** aktiviert ist, erscheint im Panel eine Anzeige für den Dateinamen. Über dieses Feld können auch mit dem Kontextmenü Dateien importiert und exportiert werden

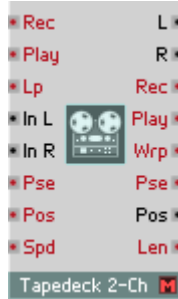
- **R:** Event-Eingang zum An- und Abschalten der Aufnahme (Record), z. B. mit einem Gate-Signal. Aufnahmestart bei Event größer 0, Aufnahmestop bei Event kleiner oder gleich 0 oder bei Erreichen der maximalen Aufnahmezeit.
- **P:** Event-Eingang zum An- und Abschalten der Wiedergabe (Play), z. B. mit Gate-Signal. Wiedergabestart bei Event größer 0, Wiedergabestop bei Event kleiner oder gleich 0.
- **Lp:** Event-Eingang zum An- und Abschalten der Endloswiedergabe (Loop Play). Wenn ein Event größer 0 empfangen wurde, wird beim Ende der Wiedergabe diese wieder von vorne begonnen, so daß eine Endlosschleife entsteht.
- **In:** Monophoner Audio-Eingang für das aufzunehmende Signal. Wenn das Signal den Wertebereich von  $\pm 1$  überschreitet, kommt es zu Clipping-Verzerrungen.
- **Pse:** Steuereingang für den Pause-Modus. Ein Wert, der grösser als 0 ist, aktiviert den Pause-Modus. Andere Werte setzen die Wiedergabe fort. Typ. range: [ 0 ... 1 ].
- **Pos:** Eingang für die Abspielposition in ms. Typ. range: [0 ... 20000].
- **Spd:** Variable Abspielgeschwindigkeit der Aufnahme. Werte müssen grösser als 0 sein. Der Wert 1 bedeutet Originalgeschwindigkeit. Typ. range: [0.8 ... 1.2].
- **Out:** Audio-Ausgang für das Wiedergabe-Signal (Playback) oder das durchgeschleifte Aufnahme-Signal (Record Monitor).
- **Rec:** 1 = Tapedeck nimmt auf, sonst 0. Typ. range: [0 ... 1].
- **Play:** 1 = Tapedeck spielt ab, sonst 0. Typ. range: [0 ... 1].
- **Wrp:** Ein Event mit dem Wert 1 wird jedes mal ausgegeben, wenn bei der Loop-Wiedergabe der Startpunkt erreicht wird.
- **Pos:** 1 = Tapedeck ist im Pause-Modus, sonst 0. Typ. range: [0 ... 1].
- **Time:** aktuelle Abspiel-/Aufnahmeposition in ms [0 ... <Samplelänge in ms>].
- **Lng:** Aufnahmelänge in ms. Typ. range: [0 ... <Aufnahmelänge in ms>].



---

## Tapedeck 2-Ch

## Auxiliary



Wie **Tapedeck 1-Ch** aber mit zwei Kanälen zur Aufnahme und Wiedergabe von Audiosignalen. Es werden Audio-Dateien in Stereo importiert und exportiert.

- **In L:** Monophoner Audio-Eingang für das auf dem linken Kanal aufzunehmende Signal. Wenn das Signal den Wertebereich von  $\pm 1$  überschreitet, kommt es zu Clipping-Verzerrungen.
- **In R:** Monophoner Audio-Eingang für das auf dem rechten Kanal aufzunehmende Signal. Wenn das Signal den Wertebereich von  $\pm 1$  überschreitet, kommt es zu Clipping-Verzerrungen.
- **L:** Audio-Ausgang für das Wiedergabesignal (Playback) oder das Aufnahmesignal (record monitor) des linken Kanals.
- **R:** Audio-Ausgang für das Wiedergabesignal (Playback) oder das durchgeschleifte Aufnahmesignal (Record Monitor) des rechten Kanals.

---

## Audio Voice Combiner

## Auxiliary



Audio-Voice-Combiner (Stimmen-Zusammenfasser). Wandelt ein polyphones Audiosignal in ein monophones um, indem es alle Stimmen durch Aufsummieren zusammenmischt.

- **In:** Polyphoner Audio-Eingang für das zusammenzufassende Signal.
- **Out:** Monophoner Audio-Ausgang für das zusammengefaßte Signal.

---

## Event V.C. All

Auxiliary



Event-Voice-Combiner (Stimmen-Zusammenfasser). Schickt alle Events eines polyphonen Signals an einen monophonen Ausgang.

- **In:** Polyphoner Event-Eingang für das zusammenzufassende Signal.
- **Out:** Monophoner Event-Ausgang für das zusammengefaßte Signal.

---

## Event V.C. Max

Auxiliary



Event-Voice-Combiner (Stimmen-Zusammenfasser) mit Maximalwert-Auswahl. Von allen aktiven Stimmen wird die mit dem größten Event-Wert bestimmt, und dieser Wert wird dann monophon ausgegeben. Ein polyphones Gate-Signal ist notwendig, um die aktiven Stimmen zu erkennen.

- **In:** Polyphoner Event-Eingang für das Signal, dessen Maximalwert ausgegeben werden soll.
- **G:** Polyphoner Event-Eingang für das Gate-Signal des polyphonen Instruments.
- **Out:** Monophoner Event-Ausgang für das Maximalwert-Signal.

---

## Event V.C. Min

Auxiliary



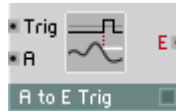
Event-Voice-Combiner (Stimmen-Zusammenfasser) mit Minimalwert-Auswahl. Von allen aktiven Stimmen wird die mit dem kleinsten Event-Wert bestimmt und dieser Wert monophon ausgegeben. Ein polyphones Gate-Signal ist notwendig, um die aktiven Stimmen zu erkennen.

- **In:** Polyphoner Event-Eingang für das Signal, dessen Minimalwert ausgegeben werden soll.
- **G:** Polyphoner Event-Eingang für das Gate-Signal des polyphonen Instruments.
- **Out:** Monophoner Event-Ausgang für das Minimalwert-Signal.



Audio-zu-Event-Konverter. Das Audiosignal wird mit der im Settings-Menü eingestellten Control Rate abgetastet und die Abtastwerte werden als Events am Ausgang ausgegeben.

---

**A to E (Trig)****Auxiliary**

Audio-zu-Event-Konverter mit Einzelabtastauslösung (Trigger). Wenn das Abtastimpuls-Signal vom Nullpunkt in positive Richtung geht (steigende Flanke), wird das Audiosignal abgetastet und als Event ausgegeben.

- **T:** Audiosignal-Steuereingang für den Abtastimpuls (Trigger). Auslösung bei positivem Nulldurchgang.
- **In:** Audio-Eingang für das abzutastende und in Form von Events auszugebende Signal.
- **Out:** Event-Ausgang für das abgetastete Signal.

---

**A to E (Perm)****Auxiliary**

Audio-zu-Event-Konverter mit regelmäßiger (permanenter) Abtastung durch internen Takt. Das Audio-Signal wird mit der eingestellten Frequenz abgetastet und in Form von Events ausgegeben.

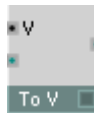
Wenn man diese Modul mit hoher Frequenz (über 1000 Hz) betreibt, kann die Prozessorlast durch die Eventverarbeitung stark steigen. Typischerweise reicht **F** = 200 Hz.

- **F:** Audiosignal-Steuereingang für die Abtastfrequenz in Hz.
- **In:** Audio-Eingang für das abzutastende und in Form von Events auszugebende Signal.
- **Out:** Event-Ausgang für das abgetastete Signal.



Audio-zu-Gate-Event-Konverter. Wenn das Abtastimpuls-Signal vom Nullpunkt in positive Richtung geht (steigende Flanke), wird das Gate-Signal mit dem aktuellen Wert des Amplitudeneingangs als Gate-Amplitude angeschaltet. Bei der fallenden Flanke (Triggersignal kleiner oder gleich 0) wird es wieder ausgeschaltet.

- **T**: Audiosignal-Steuereingang für die Gate-Steuerung.
- **A**: Audiosignal-Eingang für die Amplitude der Gate-Events.
- **Out**: Event-Ausgang für das Gate-Signal.



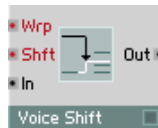
Monophonen Eingangssignale werden einer Stimme eines polyphonen Ausgangssignals zugewiesen. Die Nummer der Stimme wird durch den aktuellen Wert am **V**-Eingang festgelegt. Signale werden verworfen, wenn der Wert **V** nicht einer gültigen Stimmennummer entspricht.

- **V**: Monophoner Eingang zur Festlegung der Nummer der Stimme, zu welcher ein Signal gesendet werden soll. Typ. Range: [ 1 ... 10 ].
- **In**: Monophoner Hybrideingang für Signale, welche zu einer der polyphonen Stimmen gesendet werden sollen.
- **Out**: Polyphoner Hybridausgang. Die Eingangssignale werden (mit unverändertem Wert) auf einer Stimme des polyphonen Signals übertragen.



Eine Stimme des polyphonen Eingangssignals wird durch den am **V**-Eingang anliegenden Wert ausgewählt. Nur Signale der ausgewählten Stimme werden zum monophonen Ausgang weitergeleitet. Alle Signale der anderen Stimmen werden nicht weitergeleitet.

- **V:** Monophoner Eingang zur Bestimmung der Nummer der Stimme, von welcher die Events durchgelassen werden sollen.  
Typ. Range: [1... 10].
- **In:** Hybrideingang. Eine Stimme dieses polyphonen Signals wird zum Ausgang weitergeleitet.
- **Out:** Monophoner Hybrid Ausgang. Die Eingangssignale, die zu einer Stimme gehören, werden (mit unverändertem Wert) am monophonen Ausgang ausgegeben.



Das Modul Voice Shift können Sie verwenden, um die polyphonen Eingangswerte neu zu arrangieren, sodass die Ausgangswerte den Stimmen wie erforderlich neu zugewiesen werden. Sie können mit dem Modul Voice Shift zum Beispiel die Werte der Stimmen 1, 2, 3, 4 den ausgegebenen Stimmen 2, 3, 4, 1 oder auch 3, 4, 2, 1 zuweisen. Wenn Sie mehrere ankommende Stimmen auf dieselbe ausgegebene Stimme umsetzen, werden die ankommenden Stimmen summiert. Wenn beispielsweise die ankommenden Stimmen 1 und 2 beide auf die ausgegebene Stimme 3 versetzt werden, besteht diese Stimme 3 aus den Signalen der Stimmen 1 und 2.

**Wrp:** Monophoner Event-Wert, der das Voice-Shift-Wrapping an und aus schaltet (Default: aus). Wenn das Wrapping ausgeschaltet ist (d. h. Wrp < 0), werden Stimmen, die auf ungültige Stimmen-Nummern (d. h., kleiner als 1 oder größer als die Anzahl polyphoner Stimmen) verworfen. Wenn das Wrapping eingeschaltet ist (d. h., Wrp > 0), werden die auf ungültige Stimmen-Nummern versetzten Stimmen mit einem mathematischen (Modulo-)Verfahren auf gültige Stimmen-Nummern umgeleitet. Beispielsweise wird in einem dreistimmigen

Instrument, in dem alle Stimmen um den Wert +1 versetzt werden, die Stimme 3 auf die Stimme 1 umgeleitet.

**Shft:** Polyphoner Event-Wert, der das Voice Shifting, also das Versetzen der Stimmen, steuert. Positive Werte am Eingang Shft versetzen die Stimmen aufwärts (z. B. erzeugt ein Wert von 1 ein Versetzen der ankommenden Stimme 1 auf die ausgegebene Stimme 2). Negative Werte versetzen die Stimmen abwärts (z. B. würde durch den Wert -2 die Stimme 3 auf die Stimme 1 versetzt).

Weil Shft ein polyphoner Eingang ist, können Sie jede Stimme um einen individuellen Betrag versetzen lassen. Im Ausgangszustand hat Shft den Wert 1.

**In:** Eingang für das polyphone Signal, dessen Stimmen versetzt werden sollen.

**Out:** Ausgang für das polyphone Signal, dessen Stimmen versetzt wurden.

---

## Audio Smoother

## Auxiliary



Glätter für monophone Event-Signale mit Audioausgang. Typischerweise setzt man einen Smoother hinter einen Fader oder Button, um sanftere Übergänge zu erhalten.

Die Sprünge des ankommenden Eventsignals werden zu Rampen geglättet. Die Übergangszeit ist im Properties-Dialog unter **Transition Time** in Millisekunden einstellbar. Genau nach der Übergangszeit erreicht der Ausgang den gleichen Wert wie der Eingang, solange kein neuer Sprung am Eingang stattfindet. Je größer die **Transition Time**, desto stärker wird geglättet.

- **In:** Mono Event-Eingang für das zu glättende Event-Signal.
- **Out:** Mono Audio-Ausgang für das geglättete Signal.

---

## Event Smoother

## Auxiliary



Glätter mit Event-Ausgang, für monophone Signale. Typischerweise setzt man einen Smoother hinter einen Fader oder Button, um sanftere Übergänge zu erhalten.

Die Sprünge des ankommenden Eventsignals werden zu Rampen geglättet. Die Übergangszeit ist im Properties-Dialog unter **Transition Time** in Millisekunden

einstellbar. Genau nach der Übergangszeit erreicht der Ausgang den gleichen Wert wie der Eingang, solange kein neuer Sprung am Eingang stattfand. Je größer die **Transition Time**, desto stärker wird geglättet.

Während der Übergangszeit werden Events mit der Rate ausgegeben, die im **Settings-Menü** als **Control Rate** in Hz eingestellt ist. Je höher die Rate, desto höher die Auflösung der Glättung.

- **In:** Mono Event-Eingang für das zu glättende Signal. Das Audio-Signal wird nur mit der Control Rate abgetastet.
- **Out:** Mono Event-Ausgang für das geglättete Signal.

---

## Master Tune/Level

## Auxiliary



Steuert Master-Level und Master-Tuning.

- **Tun:** Steuereingang für das Master-Tuning. Skalierung: 1 Halbton pro Einheit. Bei 0.0 ist die Note a3 auf 440 Hz gestimmt. Typ. Range: [-1...1].
- **Lvl:** Steuereingang für den Master-Level an den Ausgangskonvertern. Skalierung: 1 dB pro Einheit. 0.0 = 0 dB. Typ. Range: [ -60 ... 0 ].
- **Tun:** Ausgang für das Master-Tuning.
- **Lvl:** Ausgang für den Master-Level.

---

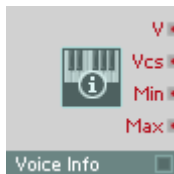
## Tempo Info

## Auxiliary



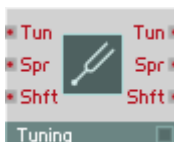
Source für die aktuelle Tempoeinstellung in Beats pro Sekunde. Um den BPM-Wert zu errechnen, multiplizieren Sie den Wert mit 60.

- **Out:** Eventausgang für das aktuelle Tempo in Beats pro Sekunde (Hz).



Voice Info-Modul.

- **V:** Polyphoner Eingang für die ID-Nummer jeder Stimme [ 1, 2, 3 ... Voices ].
- **Vcs:** Ausgang für die aktuelle Anzahl der Stimmen des Instruments.
- **Min:** Ausgang für die Min Unison Voices-Einstellung des Instruments. Dies ist die Mindestanzahl an Stimmen, welche im Unison-Modus einer Taste zugewiesen wird.
- **Max:** Ausgang für die Max Unison Voices-Einstellung des Instruments. Dies ist die Höchstanzahl an Stimmen, welche im Unison-Modus einer Taste zugewiesen wird.



Steuerungs- und Informationsmodul zu den Einstellungen der Tuning-Parameter des Instruments.

- **Tun:** Steuereingang für das Tuning des Instruments in Halbtönen.
- **Spr:** Steuereingang für die Stärke des Unison Spreads in Halbtönen.
- **Shft:** Steuereingang für die Transponierung einkommender MIDI-Noten in Halbtonschritten (note shift).
- **Tun:** Ausgang für das Tuning des Instruments in Halbtönen.
- **Spr:** Ausgang für die Stärke des Unison Spreads in Halbtönen.
- **Shft:** Ausgang für die Transponierung einkommender MIDI-Noten in Halbtonschritten (note shift).





Source für Informationen über das System: Samplingrate (in Samples/Sekunde), Controlrate (in Hz) und CPU-Verbrauch (in %).

- **SR:** Ausgang für die aktuelle Samplerate in Samples pro Sekunde.
- **CR:** Ausgang für die aktuelle Controlrate in Hz.
- **DClk:** Ausgang für die aktuelle Display-Rate in Frames pro Sekunde. Der Wert wird vor jedem Display-Update geschickt.
- **CPU:** Ausgang für den aktuellen CPU-Verbrauch in Prozent.



Note Range Info-Modul.

- **Upr:** Eingang für die Auswahl der oberen Grenze des Notenempfangs (upper range).
- **Lwr:** Eingang für die Auswahl der unteren Grenze des Notenempfangs (lower range).
- **Upr:** Ausgang für die Auswahl der oberen Grenze des Notenempfangs (upper range).
- **Lwr:** Ausgang für die Auswahl der unteren Grenze des Notenempfangs (lower range).

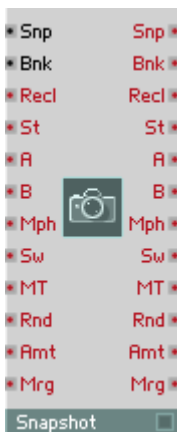


Midi Channel Info-Modul.

- **Ich:** Eingang für die Auswahl MIDI-Eingangs-Kanals.
- **Och:** Eingang für die Auswahl MIDI-Ausgangs-Kanals.
- **Ich:** Ausgang für den aktuellen MIDI-Eingangs-Kanals des Instruments.
- **Och:** Ausgang für den aktuellen MIDI-Ausgangs-Kanals des Instruments.

## Snapshot

## Auxiliary



Das Snapshot-Modul erlaubt es, aus der REAKTOR-Struktur heraus Snapshots zu wechseln und zwischen ihnen zu morphen. Das Modul besitzt ein eingebautes Panel-Bedienelement, welches exakt wie beim List-Modul funktioniert (siehe Panel-Module).

- **Snp:** Audioeingang für die Auswahl des Snapshots, welcher aufgerufen oder gespeichert werden soll. Range: 1 ... 128.
- **Bnk:** Audioeingang für die Auswahl der Snapshot-Bank. Range: 1 ... 16.
- **Recl:** Ein positiver Wert ruft den mit den Snp- und Bnk-Eingängen ausgewählten Snapshot auf.

- **St:** Ein positiver Wert speichert einen Snapshot an der mit den Snp- und Bnk-Eingängen ausgewählten Position.
- **A:** Ein positiver Wert verwendet die an den Snp- und Bnk-Eingängen anliegenden Werte, um einen Snapshot für die Morphposition A auszuwählen.
- **B:** Ein positiver Wert verwendet die an den Snp- und Bnk-Eingängen anliegenden Werte, um einen Snapshot für die Morphposition B auszuwählen.
- **Mrph:** Dieser Eingang steuert die Morphposition zwischen den Snapshots, die für A und B ausgewählt wurden. Wert  $\leq 0.0$ : A, Wert  $0.0 - 1.0$  : morpht zwischen A und B. Wert  $\geq 1.0$ : B.
- **Sw:** Negative Werte und 0 setzen die Switches/Buttons auf die von Snapshot A vorgegebene Position. Positive Werte setzen die Switches/Buttons auf die von Snapshot B vorgegebene Position.
- **MT:** Eventeingang für die Morphzeit in ms.
- **Rnd:** Ein positiver Wert löst die Zufallsfunktion für den ausgewählten Snapshot aus.
- **Amt:** Eingang für die Zufallsstärke (randomization amount). Range: 0.0 ... 1.0 (1.0 = 100%).
- **Mrg:** Ein positiver Wert löst die Random Merge-Funktion aus (es wird ein Snapshot auf Zufallsbasis erstellt, welcher gewisse Eigenschaften der zwei mit den Eingängen A und B ausgewählten Snapshots erbt).
- **Snp:** Ausgang für die Nummer des gerade ausgewählten Snapshots. Es wird jedes mal ein Wert ausgegeben, wenn ein Snapshot aufgerufen oder gespeichert wird.
- **Bnk:** Ausgang für die Nummer der gerade ausgewählten Snapshot-Bank. Es wird jedes mal ein Wert ausgegeben, wenn ein Snapshot aufgerufen oder gespeichert wird.
- **Recl:** Der Wert 1.0 wird jedes mal ausgegeben, wenn ein Snapshot aufgerufen wird.
- **St:** Der Wert 1.0 wird jedes mal ausgegeben, wenn ein Snapshot gespeichert wird.
- **A:** Ausgang für die Nummer des Snapshots der Morphposition A. Hier wird ein Wert gesendet, wenn ein Snapshot aufgerufen oder für die Position A ausgewählt wird.
- **B:** Ausgang für die Nummer des Snapshots der Morphposition B. Hier wird ein Wert gesendet, wenn ein Snapshot aufgerufen oder für die Position B ausgewählt wird.

- **Mrph:** Ausgang für die Morphposition. Wert = 0.0: A, Wert 0.0 - 1.0: morpht zwischen A und B, Wert = 1.0: B.
- **Sw:** Ausgang für die A/B-Position für Switches und Buttons. Wert = 0.0, wenn die Switches/Buttons auf die von Snapshot A vorgegebene Position gesetzt sind. Wert = 1.0, wenn die Switches/Buttons auf die von Snapshot B vorgegebene Position gesetzt sind.
- **MT:** Ausgang für die Morphzeit in ms.
- **Rnd:** Der Wert 1.0 wird ausgegeben, wenn die Zufallsfunktion ausgelöst wurde.
- **Amt:** Ausgang für die Zufallsstärke (randomization amount). Range: 0.0 ... 1.0 (1.0 = 100%).
- **Mrg:** Der Wert 1.0 wird ausgegeben, wenn die Random Merge-Funktion ausgelöst wurde.

---

## Set Random

## Auxiliary



Setzt das Random-Seed und initialisiert damit den Zufallszahlengenerator, der in allen **Event Randomize**, **Slow Random** und **Geiger** Modulen benutzt wird. Nur Module, die sich im selben Instrument befinden, werden davon beeinflusst. Jeder Wert erzeugt eine andere, einmalige Zufallszahlenfolge.

---

## Unison Spread

## Auxiliary



Polyphone Event-Quelle für einen konstanten Wert, mit dem die Parameter von Unisono-Stimmen auseinandergezogen werden können.

Im Unisono-Modus muss man die Parameter der verschiedenen Stimmen, die den gleichen Ton spielen etwas verstellen, um sie etwas unterschiedlich zu machen, damit in der Summe der Klang fetterer ist, und nicht einfach lauter. Für die Tonhöhe (Note Pitch) passiert das automatisch und wird über den **Unison Spread**-Parameter in den Instrument Properties gesteuert. Andere Parameter können auseinandergezogen werden, indem man einen Wert aus dem Unison Spread Modul addiert.

Der Wert am Eingang des Moduls bestimmt die Größe dieser Werte und damit den Grad der "Anfettung". Am Ausgang erhält man einen polyphonen Wert, der für jede der Stimmen, die zusammen einen Ton spielen, unterschiedlich ist. Der Wert ändert sich erst, wenn ein neuer Ton angeschlagen wird.



Dieses Modul speichert den Eingangswert mit einem Snapshot ab und gibt den Wert aus, wenn der Snapshot wieder aufgerufen wird.

**In:** Eingang für den in einem Snapshot zu speichernden Wert. Das Eingangssignal wird im Moment der Snapshotspeicherung abgegriffen. Wird das Modul an eine Eventquelle angeschlossen, werden Eingangsevents an den Ausgang durchgereicht.

**Out:** Ausgang für den in demjenigen Snapshot gespeicherten Wert, der zuletzt aufgerufen wurde.

---

## Snap Value Array

## Auxiliary



Speichert und lädt Arrays von Fließkomma-Werten aus dem Edit-Puffer und aus Snapshot.

Ein einziges Modul des Typs Snap Value Array kann 1-40 Arrays aufnehmen, wobei jedes Array (theoretisch) eine beliebige Anzahl von Elementen enthalten kann – die einzige Begrenzung für die Kapazität der Arrays stellt der verfügbare Arbeitsspeicher dar. Alle Arrays enthalten dieselbe Anzahl von Elementen. Sowohl die Anzahl der Arrays als auch die Anzahl der Elemente pro Array legen Sie im Dialog Properties fest.

Der Arbeitsspeicher für das Modul Snap Value Array wird dynamisch verwaltet. Das heißt, dass beim Erzeugen eines weiteren Snapshots zusätzliche Arbeitsspeicher angefordert wird; ebenso wird mit dem Löschen eines Snapshots wieder Speicher freigegeben. Dadurch bleiben die Arbeitsspeicher-Anforderungen so niedrig wie möglich.

Wenn Sie die Option Snap Isolate im Dialog Properties einschalten, wird für jedes Array-Element nur der zuletzt in das Array geschriebene Wert gespeichert (und während der Initialisierung des Ensembles wieder aufgerufen); beim Erstellen oder Laden von Snapshots werden keine Daten gespeichert bzw. geladen.

Eine typische Anwendung des Moduls Snap Value Array ist das Speichern von Sequencer-Daten in Snapshots. Zu diesem Zweck kommt das Modul oft in Verbindung mit den Modulen Event Table oder Multi Display zum Einsatz. Alle Eingänge des Moduls Snap Value Array nehmen ausschließlich monophone Signale entgegen.

**Idx:** Index des zu adressierenden Array-Elements (sowohl für Schreib- als auch für Lese-Operationen). Arrays sind 1-basiert, d. h., der Index des ersten Elements ist 1 (nicht 0). Nicht ganzzahlige Werte werden auf die nächste Ganzzahl (Integer) gerundet. Wie der Eingang Idx außerhalb des Bereichs zwischen 1 und N liegende Werte behandelt, hängt von der Einstellung der Option Index Behaviour im Dialog Properties ab.

**R:** Wenn ein Event (mit einem beliebigen Wert) am Eingang R (Read) ankommt, wird das über den Eingang Idx spezifizierte Array-Element ausgelesen und sein Wert an den Ausgangs-Port übermittelt. Mit anderen Worten: Jedes Event am Eingang R pflanzt sich als einzelnes Event bis zum Ausgangs-Port des Arrays fort. Mehrere Arrays lassen sich parallel über denselben Eingang Idx adressieren. Dadurch lösen Events, die am Eingang R ankommen, Ausgangs-Events an jedem Ausgangs-Port des Arrays (mit dem Wert des durch den Eingang Idx spezifizierten Elements) aus. Der Wert am Eingang Idx muss immer gesetzt werden, bevor Events am Eingang R ankommen.

**W:** Wenn ein Event am Eingang W ankommt, wird sein Wert in das über den Eingang Idx spezifizierte Element des Arrays geschrieben, wobei alle Daten, die zuvor in dem Element gespeichert waren, überschrieben werden. Das Modul Snap Value Array besitzt einen separaten Eingang W für jedes Array, das es enthält (anfänglich ein Array, Sie können aber im Dialog Properties weitere Arrays für das Modul erzeugen). Alle Ports können Sie nach Bedarf umbenennen. Wenn Sie im Dialog Properties die Option Events Thru einschalten, werden am Eingang W ankommende Events an die entsprechenden Ausgänge (Out) durchgereicht.

**N:** Der Ausgang N meldet die Anzahl der Elemente in einem Array.

**Gate:** Der Ausgang Gate sendet ein Event mit dem Wert 1, bevor die Ausgangs-Ports des Arrays Events senden, und sendet anschließend ein Event mit dem Wert 0.

**Idx:** Der Ausgang Idx stellt die Nummer des Elements, das gerade gelesen oder geschrieben wird, bereit (und zwar als Reaktion auf an den Eingängen R und W ankommende Events oder auf Snapshot-Operationen wie Recall und Morph hin). Mit Rücksicht auf Lese-Operationen wird die Index-Nummer vom Ausgang Idx übermittelt, bevor das Event mit dem Wert vom Ausgang Out gesendet wird.

**Out:**Der Wert des Array-Elements (das vom Eingang Idx spezifiziert wird), wird vom Ausgang Out als Antwort auf Events am Eingang R oder auf Snapshot-Operationen (Recall, Morph) gesendet. Wenn Sie im Dialog Properties die Option Self-Iteration einschalten, werden alle Array-Elemente seriell ausgegeben (d. h., zuerst das erste Element, dann das zweite Element, dann das dritte Element und so weiter), wenn eine der folgenden Operationen ausgeführt wird: Initialisierung, Aktivierung, Snapshot-Aufruf (Recall), Randomize, Random Merge und Morph. Self-Iteration ermöglicht es, einen kompletten Datensatz von Snapshot-Operationen auf den aktuellen Stand zu bringen.

# Terminals

Terminals dienen dazu, Audio- und Eventsignale in REAKTOR-Instrumente und deren Macro-Unterstrukturen hinein- und herauszuführen.

---

## In Port

## Terminal



Terminal für Audio- und Eventsignale, um Eingangsports für Instrumente und Macros zu erzeugen. Sie können Ports automatisch erzeugen, wenn Sie ein Kabel auf ein Instrument oder Macro innerhalb des Structure-Fensters ziehen, während Sie die Ctrl-Taste gedrückt halten.

---

## Out Port

## Terminal



Terminal für Audio- und Eventsignale, um Ausgangsports für Instrumente und Macros zu erzeugen. Sie können Ports automatisch erzeugen, wenn Sie ein Kabel auf ein Instrument oder Macro innerhalb des Structure-Fensters ziehen, während Sie die Ctrl-Taste gedrückt halten.

---

## Send

## Terminal



Send-Terminal für Audio- und Eventsignale zur Erzeugung einer kabellosen Verbindung innerhalb eines Instruments.

Jedes eingefügte Sendmodul erzeugt einen Eintrag in der Liste verfügbarer Signalquellen in den Properties eines Receive-Moduls.

---

## Receive

## Terminal



Eingangs-Treminal für Audio- und Eventsignale zur Erzeugung einer kabellosen Verbindung innerhalb eines Instruments.

### Properties - Function-Seite

Für jedes Send-Modul, welches in dasselbe Instrument eingefügt wird, wird ein Listeneintrag angelegt. Der Name des Listeneintrags wird ent-



sprechend dem Label des Send-Moduls angelegt. Die Up- und Down-Buttons über der Liste dienen dazu, einen ausgewählten Listeneintrag nach oben oder unten zu verschieben.

Die Liste enthält folgende Spalten, die mit einem Klick auf den entsprechenden Eintrag in der Kopfzeile sortiert werden können:

- **#:** Zeigt die Listenposition des Send-Moduls.  
Der Default-Wert bezieht sich auf diese Nummer.
- **Label:** Der Name des Send-Moduls. Wenn Sie ein Send-Modul umbenennen wird das Label in der Liste aktualisiert.
- **State:** Zeigt an, ob eine Verbindung zum entsprechenden Send-Modul möglich ist. Stellen Sie die Verbindung nur her, wenn dieses Feld OK anzeigt.,
- **Use:** Klicken Sie auf dieses Feld, um eine Verbindung zum entsprechenden Send-Modul herzustellen. Eine bestehende Verbindung wird durch ein Kreuz angezeigt.

Die Mouse Resolution-Einstellung ist nur dann von Bedeutung, wenn für das Bedienelement der Spin-Style auf der Appearance-Seite ausgewählt ist, wo man auf den Eintrag im Bedienelement klicken und die Maus nach oben oder unten ziehen kann, um den Eintrag zu ändern.

Der Default-Wert wird immer dann verwendet, wenn eine Initialisierung des Bedienelements stattfindet. In diesem Fall wird derjenige Listeneintrag ausgewählt, dessen # dem Default-Wert entspricht.

## Properties - Appearance-Seite

Die Panel-Darstellung des Bedienelements für dieses Modul ändert sich abhängig vom auf der Appearance-Seite in den Properties ausgewählten Style. Die folgenden Styles sind verfügbar:

- **Button:** Jeder Eingangsport des Moduls erzeugt einen Button. Alle Buttons werden vertikal als Buttonleiste im Instrumenten-Panel aneinandergehängt. Der gerade ausgewählte Eingang erscheint in der Indicator-Farbe des Instruments.
- **Menu:** Jeder Eingangsport des Moduls erzeugt einen neuen Eintrag in einer Dropdown-Liste.
- **Text Panel:** Jeder Eingangsport des Moduls erzeugt einen neuen Eintrag in einer Liste, welche mehrere Einträge gleichzeitig darstellt. Wenn Sie mehr Einträge erzeugen als in das Panel-Display passen, dessen Grösse Sie mit den Feldern Size X und Size Y auf der Appearance-Seite der Properties definieren, erhält das Panel-Display Scrollbalken.

- **Spin:** Jeder Eingangsport des Moduls erzeugt einen neuen Eintrag in einer Liste. Sie können mit den + und - Buttons rechts vom Panel-Display durch die Listeneinträge navigieren.

Die **Size X-** und **Size Y-Felder** regeln die Display-Grösse des Bedienelements im Panel.

---

## IC Send

## Terminal



IC Send übermittelt monophone Event-Signale an beliebige Module, die befähigt sind, Übertragungen via IC (Internal Connection) zu empfangen. Zu diesen Modulen gehören die Module des Typs IC Receive (siehe unten), aber auch verschiedene Panel-Elemente (z. B. Drehregler und Wahlschalter). Weil interne Verbindungen global (d. h., auf Ensemble-Ebene) arbeiten, können Sie diese Module verwenden, um "kabellose" Verbindungen zwischen verschiedenen Instrumenten innerhalb eines Ensembles herzustellen.

Das Modul *IC Send* besitzt eine Panel-Anzeige, die es erlaubt, Verbindungen aus dem Instrumenten-Panel heraus zu einzurichten. Alle Module innerhalb des Ensembles, die IC-Übertragungen empfangen können, erscheinen in dieser Anzeige, mit Ausnahme derjenigen Module, in deren Dialog Properties Sie die Option *No Entry in IC Menu* eingeschaltet haben. Sie können Verbindungen via IC auch im Dialog **Properties** herstellen.

---

## IC Receive

## Terminal



Empfängt und sendet monophone Event-Signale an Module, die über das Protokoll Internal Connection (IC) angeschlossen sind. Typischerweise bildet das Modul IC Send die Gegenstelle des Moduls IC Receive, Sie können IC Receive aber mit jedem zum Empfang von IC-Übertragungen befähigten Modul (z. B. Drehreglern und Wahlschaltern) verbinden. Verbindungen via IC können Sie im Dialog Properties herstellen; Verbindungen zu Modulen des Typs IC Send können Sie aber auch über das Panel des Moduls IC Send einrichten.



OSC-Messages können mit Hilfe der Module OSC Send und OSC Receive übertragen werden. Beide Module besitzen eine dynamische Port-Verwaltung, d.h. neue Ein- und Ausgangsports können hinzugefügt werden, indem Sie ein Kabel auf einen leeren Bereich neben den existierenden Ports ziehen, während Sie die Ctrl-Taste gedrückt halten.

Mehrfach-Verbindungen zum OSC Send-Modul resultieren in OSC-Messages mit multiplen Argumenten. Wenn Sie beispielsweise die Ausgänge MX und MY des XY-Moduls an ein OSC Send-Modul anschliessen, enthält jede OSC-Message sowohl den X- als auch den Y-Wert. Sie müssen auch im OSC Receive-Modul mehrere Ausgänge verkabeln, um die zusätzlichen Argumente der OSC-Message zu empfangen. Es sind so bis zu zehn Argumente innerhalb einer OSC-Message möglich.

Haben Sie mehr als einen In-Port für das OSC Send-Modul angelegt, löst immer nur der erste Port die OSC-Message aus.

OSC-Messages können auch von manchen anderen REAKTOR-Modulen gesendet und empfangen werden, zum Beispiel von den Bedienelementen. Die Sende- und Empfangs-Einstellungen sind die selben wie in den OSC-Terminal-Modulen.

Das Modul besitzt eine dynamische Eingangsport-Verwaltung. Die Anzahl der Eingänge kann mit Min Num Port Groups auf der Function-Seite der Properties definiert werden.

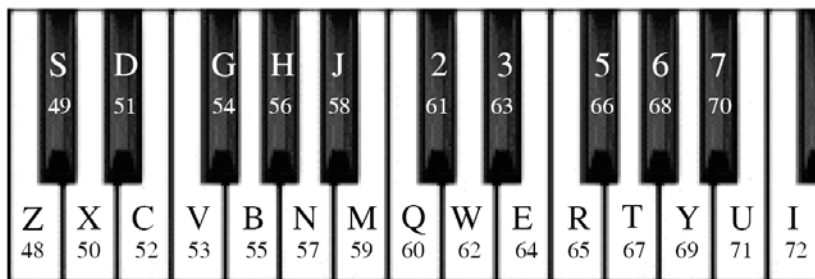
---

**OSC Receive****Terminal**

Siehe OSC Send-Modul.

# Anhang

## Transponieren ankommender MIDI-Noten

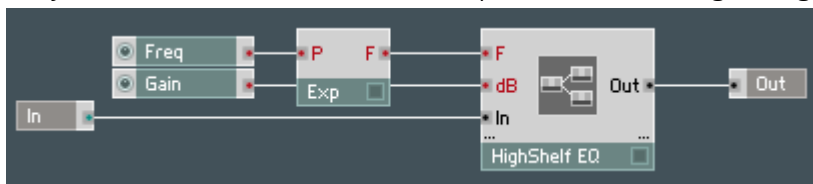


- Drücken Sie die Taste **Shift**, um alle ankommenden MIDI-Noten um zwei Oktaven nach oben zu transponieren (+24).
- Drücken Sie unter Windows XP die Tasten **Ctrl + Shift**, unter Mac OS X die Tasten **⌘ + Shift**, um alle ankommenden MIDI-Noten um zwei Oktaven nach unten zu transponieren (-24).

# Erste Schritte in REAKTOR Core

## Was ist REAKTOR Core

*REAKTOR Core* ist eine neue Funktionsschicht innerhalb von REAKTOR mit einem neuen, anderen Funktionsumfang. Weil es auch noch eine ältere Funktionsschicht gibt, werden wir diese fortan *Core Level* und *Primary Level* nennen. Wenn wir von einer “Primary-Level-Struktur” sprechen, bezeichnet dies die innere Struktur eines Instruments oder eines Macros, aber nicht die eines Ensembles. Die Funktionen von REAKTOR Core sind nicht direkt kompatibel mit denen von REAKTORs Primary Level. Daher müssen zwischen diesen Ebenen Schnittstellen geschaffen werden. Diese Arbeit übernehmen die so genannten *Core Cells*. Core Cells existieren innerhalb von Primary-Level-Strukturen und gleichen in Aussehen und Verhalten den eingebauten Modulen des Primary Level. Hier sehen Sie eine Beispiel-Struktur, die eine *HighShelf EQ Core Cell* verwendet. Diese unterscheidet sich von dem eingebauten Modul auf dem Primary Level durch die Art, wie sie auf Frequenz- und Boost-Regler reagiert:



Innerhalb der Core Cells finden sich REAKTOR-Core-Strukturen, die eine effiziente Möglichkeit zur Verfügung stellen, maßgeschneiderte Low-Level-DSP-Funktionen zu implementieren und auch größere Signalbearbeitungs-Strukturen zu entwickeln. Wir werden uns diese Strukturen später noch genauer ansehen. Obwohl eine der Spezialitäten von REAKTOR Core die Fähigkeit zum Aufbau von Low-Level-DSP-Strukturen ist, beschränkt sich das Einsatzgebiet nicht darauf. Für den Fall, dass Sie kein DSP-Experte sind (und auch nicht vorhaben, einer zu werden), stellen wir Ihnen eine Bibliothek mit vorgefertigten Modulen zur Verfügung, die Sie innerhalb der Core-Strukturen auf ähnliche Weise verbinden können, wie Sie es vom Primary Level kennen. Außerdem gibt es eine Sammlung von vorgefertigten Core Cells, die Sie sofort in Primary-Level-Strukturen einsetzen können.

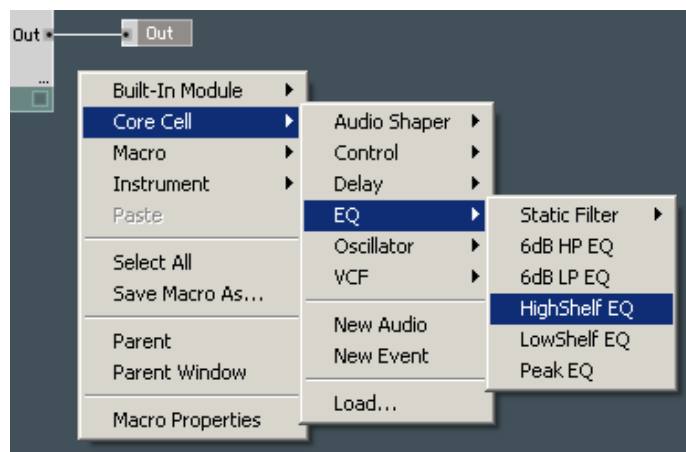
---

In Zukunft werden wir bei NATIVE INSTRUMENTS den Schwerpunkt weniger auf die Entwicklung neuer Primary-Level-Module legen. Stattdessen verwenden wir nun unsere neue REAKTOR-Core-Technik und liefern die Module in Form von Core Cells. Sie finden in der Core Cell Library bereits einen Satz neuer Filter, Hüllkurven, Effekte etc.

---

## Wie Sie Core Cells verwenden

Auf die Core Cell Library können Sie von Primary-Level-Strukturen aus zugreifen, indem Sie einen Rechts-Klick in den Hintergrund ausführen und das Untermenü *Core Cell* ausklappen:



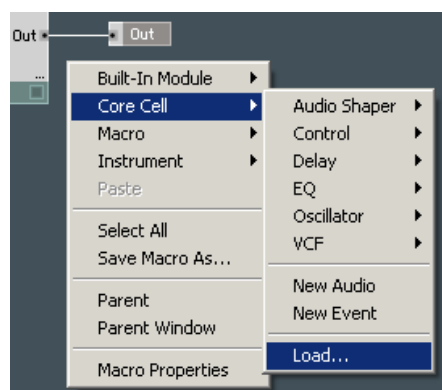
Wie Sie sehen, enthält die Liste alle möglichen Arten von Core Cells, die Sie auf dieselbe Art verwenden können wie die eingebauten Primary-Level-Module.

---

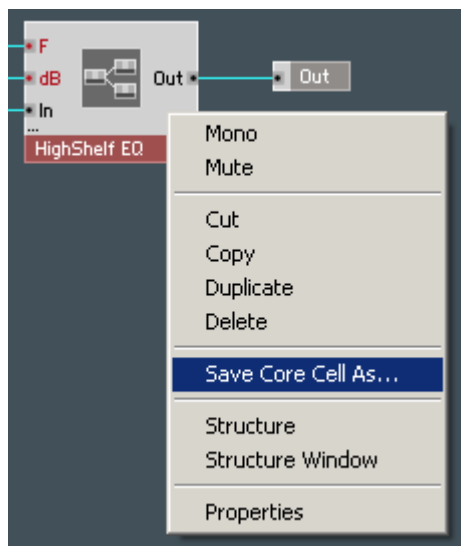
Eine wichtige Beschränkung gilt beim Einsatz von Core Cells: Sie dürfen Core Cells nicht innerhalb von Event-Loops verwenden. Jeder Event-Loop, der durch eine Core Cell entsteht, wird von REAKTOR blockiert.

---

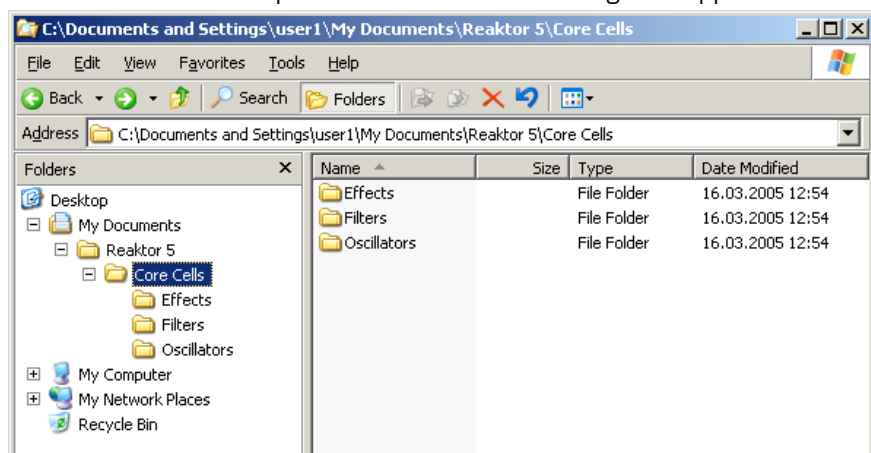
Sie können auch Core Cells einsetzen, die nicht in der Library enthalten sind. Verwenden Sie dazu den Befehl *Load...* aus dem Menü *Core Cell*:



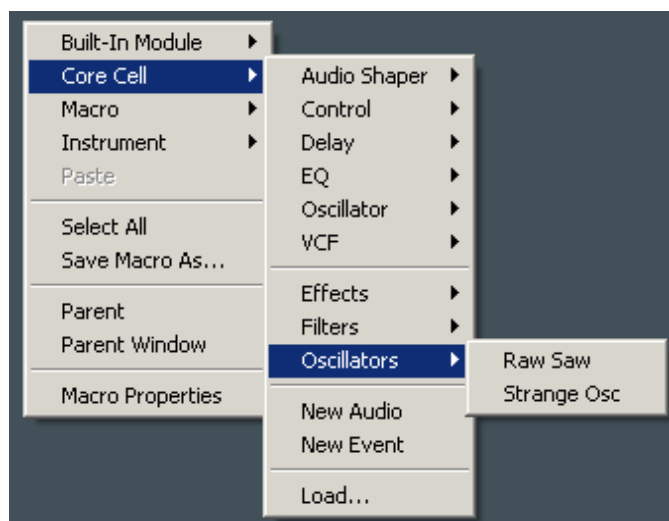
Core Cells, die Sie selbst erstellt oder verändert haben, möchten Sie vielleicht speichern. Führen Sie dazu einen Rechts-Klick auf eine Core Cell aus und wählen Sie aus dem Menü den Eintrag *Save Core Cell As*:



Sie können Ihre eigenen Core Cells auch dem Menü hinzufügen, sodass Sie nicht den Umweg über den Befehl *Load* nehmen müssen, um sie aufzurufen. Dazu legen Sie Ihre Core Cells in das Unterverzeichnis “*Core Cells*” in Ihrem User-Library-Ordner. Es empfiehlt sich aber, die Core Cells in Gruppen zu organisieren, anstatt sie einfach in das Unterverzeichnis “*Core Cells*” zu legen. Hier sehen Sie ein Beispiel für eine solche Sortierung in Gruppen:



“My Documents\REAKTOR 5” ist der User-Library-Ordner im obigen Beispiel. Auf Ihrem Computer kann dieser Ordner anders heißen, je nachdem, was Sie während der Installation von REAKTOR angegeben oder nachträglich in REAKTORs Voreinstellungen geändert haben. Innerhalb dieses User-Library-Ordners sollten Sie aber auf jeden Fall einen weiteren Ordner namens “Core Cells” finden. Falls nicht, müssen Sie diesen Ordner von Hand anlegen. Nun sehen Sie auf der Abbildung innerhalb des Ordners “Core Cells” drei weitere Ordner namens “Effects”, “Filters” und “Oscillators”. In diesen Ordnern befinden sich Core Cells, die im User-Bereich des Menüs *Core Cell* (unterhalb des obersten Trennstrichs) angezeigt werden:




---

Der Inhalt des Menüs ergibt sich aus dem Inhalt des User-Library-Ordners. Dieser wird beim Start von REAKTOR einmal gescannt. Wenn Sie neue Dateien in diesen Ordner oder in einen seiner Unterordner legen, müssen Sie REAKTOR beenden und erneut starten, damit diese neuen Core Cells im Menü erscheinen.

---



---

Leere Ordner erscheinen nicht im Menü. Damit ein Ordner im Menü angezeigt wird, muss er mindestens eine Datei enthalten.

---



---

Legen Sie unter keinen Umständen Ihre eigenen Dateien in der System Library ab! Das System-Library-Verzeichnis kann sich bei der Installation

---

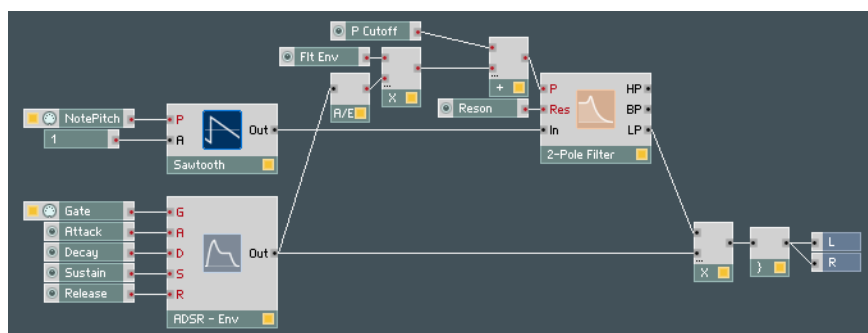


von Updates ändern oder sogar komplett ausgetauscht werden, und dann wären alle Ihre Dateien verloren. Die User Library ist der richtige Ort für alle Erweiterungen, die nicht Bestandteil der Software selbst sind.

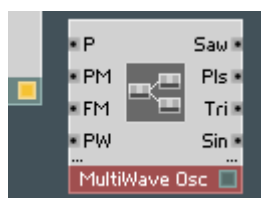
---

## Der Einsatz von Core Cells in der Praxis

Wir werden nun ein REAKTOR-Instrument nehmen, das ausschließlich aus Primary-Level-Modulen besteht, und es modifizieren, indem wir einige Core Cells einsetzen. Um das Beispiel nachzuvollziehen, finden Sie im Ordner *Core Tutorial Examples* in REAKTORs Installationsverzeichnis das Ensemble *One Osc.ens* und öffnen Sie es. Dieses Ensemble besteht aus nur einem Instrument mit der folgenden internen Struktur:

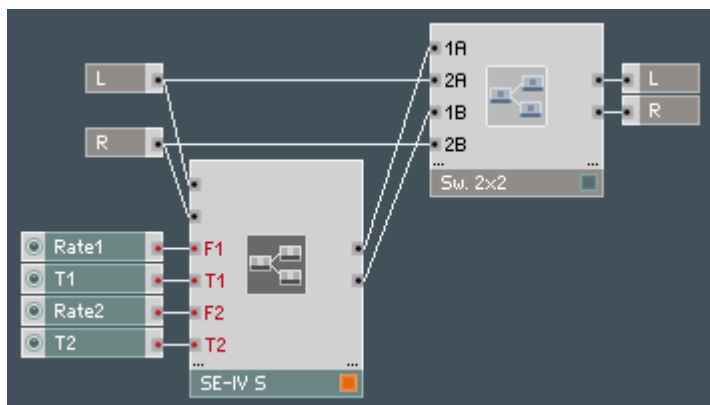


Wie Sie sehen, handelt es sich um einen sehr einfachen subtraktiven Synthesizer mit einem Oszillator, einem Filter und einer Hüllkurve. Wir werden zunächst den Oszillator durch einen anderen, leistungsfähigeren ersetzen. Dazu klicken Sie in den Hintergrund und wählen aus dem Menü *Core Cell > Oscillator > MultiWave Osc*:



Die wichtigste Funktion dieses Oszillators ist, dass er gleichzeitig analoge Wellenformen erzeugt, die in der Phase gekoppelt sind. Wir werden den Mix dieser Wellenformen anstelle des einzelnen Sägezahn-Oszillators verwenden. Glücklicherweise gibt es schon ein fertiges Mixer-Modul: *Insert Macro > Classic Modular > O2 - Mixer Amp > Mixer - Simple - Mono*:

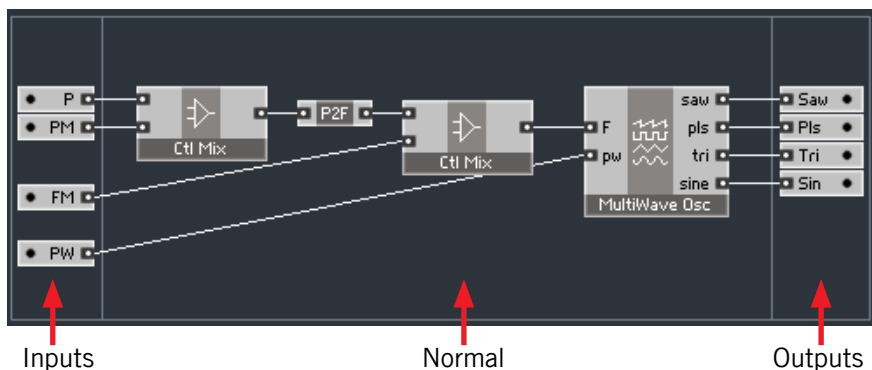




## Grundlegende Bearbeitung von Core Cells

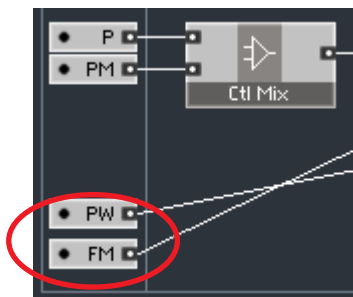
Nun erfahren Sie einige Dinge über das Bearbeiten von Core Cells. Wir fangen mit etwas Einfachem an, zum Beispiel mit der Anpassung einer existierenden Core Cell an Ihre besonderen Anforderungen.

Doppelklicken Sie zuerst das Modul *MultiWave Osc*, um sein Inneres freizulegen:

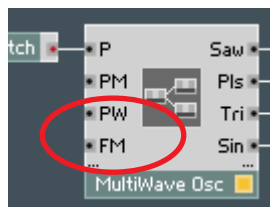


Was Sie nun sehen, ist eine REAKTOR-Core-Struktur. Die drei durch vertikale Linien voneinander abgeteilten Bereiche stehen für die unterschiedlichen Modul-Arten: Eingänge (links), Ausgänge (rechts) und normale Module (Mitte). Die Eingänge und Ausgänge können Sie nur vertikal verschieben, und ihre relative Anordnung bestimmt immer die Reihenfolge, in der sie außerhalb des Modul erscheinen. Die normalen Module dagegen können Sie in alle Richtungen verschieben. So können Sie durch einfaches Bewegen die Anordnung in

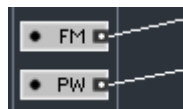
der äußeren Ansicht verändern. Probieren Sie das doch aus, indem Sie den Eingang *FM* unter den Eingang *PW* schieben:



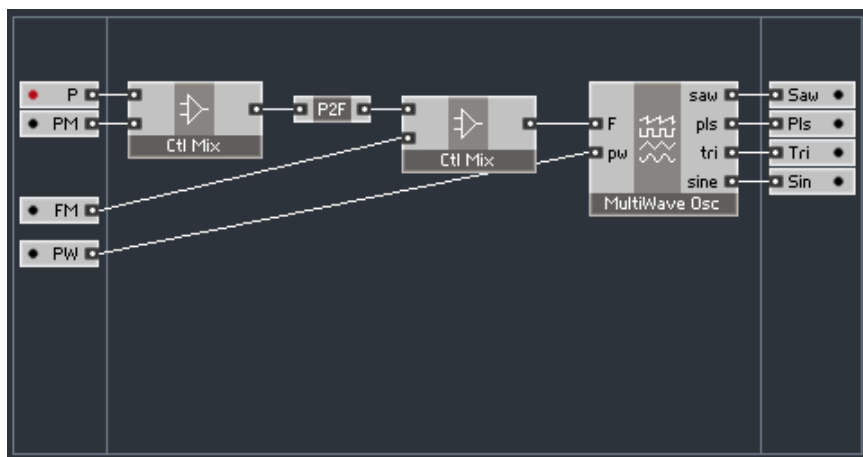
Sie können nun einen Doppelklick in den Hintergrund ausführen, um sich aufwärts zur Außenansicht der Primary-Level-Struktur zu bewegen und die geänderte Eingangskonfiguration zu sehen:



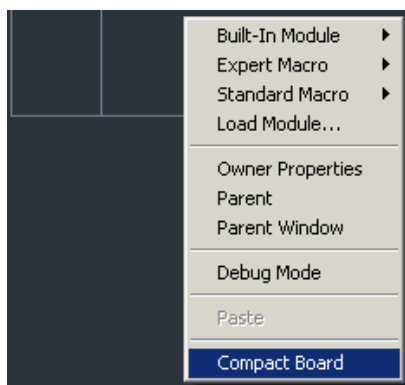
Lassen Sie uns auf die Core-Ebene zurückkehren – und vergessen Sie nicht, die ursprüngliche Anordnung der Eingänge wieder herzustellen:



Wie Sie wahrscheinlich schon bemerkt haben, erhöhen die drei Bereiche der Core Struktur automatisch die Feldgröße, um alle Module beinhalten zu können. Da Sie aber nicht wieder automatisch schrumpfen, können diese Bereiche unnötig groß werden:



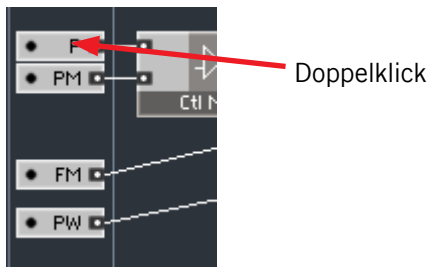
Sie können die Fläche wieder auf das passende Maß reduzieren, indem Sie einen Rechts-Klick in den Hintergrund ausführen und aus dem Menü den Eintrag Compact Board auswählen:



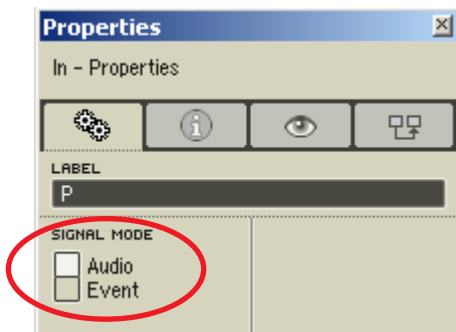
Nun, da wir gelernt haben, wie man Dinge bewegt, und sogar wissen, wie man die Port-Reihenfolge einer Core Cell verändert, wollen wir einige weitere Optionen ausprobieren.

Bei einer Core Cell, die *Audio-Ausgänge* besitzt, ist es möglich, den Typ der Eingänge zwischen Audio und Event (dazu später mehr) umzuschalten. Im obigen Beispiel haben wir das Modul *MultiWave Osc* verwendet, dessen Ein- und Ausgänge alle vom Typ Audio sind. Trotzdem brauchen wir sie in unserem speziellen Fall wirklich nicht als Audio-Ports, weil das einzige Ding, das mit dem Oszillator verbunden ist, ein Pitch-Drehregler ist. Wäre es da nicht viel günstiger, einige Ports auf den Typ Event umzuschalten und dadurch CPU-

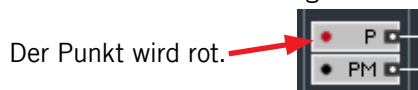
Leistung einzusparen? Die offensichtliche Antwort ist: Na klar, das wäre es. Also lassen Sie uns das tun. Sie sollten zumindest die Eingänge *P* und *PM* in den Event-Modus umschalten, denn dies bewirkt die größte Entlastung der CPU. Doppelklicken Sie dazu auf das Port-Modul *P*, um das Properties-Fenster für diesen Port zu öffnen:



Schalten Sie (falls notwendig) das Properties-Fenster in die Ansicht Function, indem Sie auf die Schaltfläche  klicken. Sie sollten nun die Eigenschaften für den Signal Mode sehen:

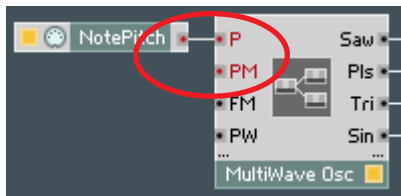


Wählen Sie im Ausklapp-Menü den Eintrag *Event*. Beachten Sie, wie daraufhin der große schwarze Punkt links im Input-Modul seine Farbe von Schwarz in Rot ändert und dadurch anzeigt, dass sich der Eingang nun im Event-Modus befindet (was besser zu erkennen ist, wenn der Port nicht mehr ausgewählt ist – klicken Sie einfach irgendwo anders hin):



Klicken Sie nun auf das Input-Modul *PM*, um es auszuwählen und auch in den Event-Modus umzuschalten. Wenn Sie möchten, können Sie die beiden verbleibenden Eingänge ebenfalls auf Event setzen. Nachdem Sie damit fertig sind, können Sie einen Doppelklick in den Hintergrund der Struktur ausführen,

um auf das Primary Level zurückzukehren und zu sehen, wie sich die Farbe der Ports geändert hat und wie die CPU-Last sinkt.



Manchmal ist es nicht sinnvoll, einen Port von einem Typ auf den anderen umzuschalten. Es ergibt zum Beispiel keinen Sinn, einen Eingang, der ein echtes Audio-Signal empfängt (das tatsächlich einen Klang enthält, nicht nur ein Kontroll-Signal wie etwa eine Hüllkurve, die mit Audio-Rate arbeitet) in den Event-Modus zu schalten. Abgesehen davon, dass das nicht nur sinnlos ist, kann es auch die Funktion des Moduls ruinieren. Ein anderer Fall, in dem es sich nicht empfiehlt, die Betriebsart der Ports zu ändern, ist gegeben, wenn ein Port tatsächlich auf Events reagieren soll. Dies kann zum Beispiel ein Event-Trigger-Signal von einer Hüllkurve sein (wie etwa Gate-Inputs von REAKTORs Primary-Level-Hüllkurven). Wenn Sie einen solchen Eingang in den Audio-Betrieb schalten, wird er nicht mehr korrekt arbeiten.

Neben den Fällen, in denen es offensichtlich sinnlos ist, den Port-Typ zu ändern, gibt es Fälle, in denen es zwar sinnvoll wäre, in denen die Module aber trotzdem nicht korrekt arbeiten, wenn Sie den Port-Typ umschalten. Diese Fälle sind allerdings entweder recht speziell oder sie sind das Ergebnis einer fehlerhaften Implementation des Moduls. Normalerweise sollte das Umschalten der Ports also funktionieren. Deshalb kommen wir nun zu folgenden Umschalt-Regel:

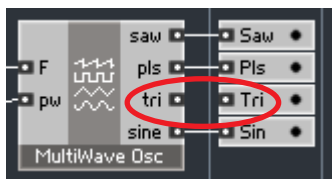
---

In einer gut gestalteten Core Cell kann ein mit Audio-Rate arbeitender Eingang ohne Probleme in den Event-Modus umgeschaltet werden. Ein Event-Eingang kann nur dann in den Audio-Modus umgeschaltet werden, wenn er nicht auf Informationen des Typs "Trigger" wartet.

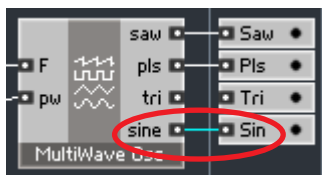
---

Ein anderer Trick, den Sie verwenden können, um CPU-Leistung zu sparen, geht so: Trennen Sie die Verbindungen von Ausgängen, die Sie nicht verwenden. Dadurch schalten Sie nicht benötigte Teile der REAKTOR-Core-Struktur ab. Auch dies nehmen Sie im Inneren vor – Verbindungen außerhalb der Struktur haben keine Auswirkungen auf die Verbindungen der Elemente innerhalb der Core-Struktur. Nehmen wir an, wir möchten in unserem Beispiel nicht alle vier Ausgänge, sondern nur Sägezahn und Puls verwenden. Wir können uns nun in das Modul *MultiWave Osc* bewegen und dort die nicht benötigten Ausgänge

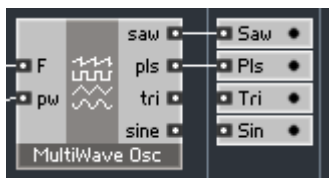
trennen. Das Trennen von Verbindungen ist in REAKTOR Core einfach: Klicken Sie auf den Eingangs-Port der Verbindung, ziehen Sie die Maus mit gedrückter Maustaste an eine Stelle, an der sich kein Ausgang befindet, und lassen Sie dann die Maustaste los. Lassen Sie uns mit dem Ausgang “Tri” beginnen – klicken Sie auf den *Eingangs*-Port des Ausgangs “Tri” und ziehen Sie die Maus mit gedrückter Taste auf einen leeren Bereich des Hintergrunds.



Es gibt noch eine weitere Möglichkeit, Verbindungen zu löschen. Klicken Sie dazu auf das Kabel zwischen dem Ausgang “sine” des Moduls *MultiWave Osc* und dem Ausgang “Sin” der Core Cell, sodass diese Verbindung ausgewählt ist (das erkennen Sie an der Farbe):



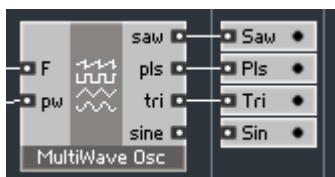
Sie können nun die Taste *Delete* drücken, um das Kabel zu löschen:



Nachdem Sie beide Kabel gelöscht haben, sollte die CPU-Anzeige noch etwas weniger Last anzeigen.

Was aber, wenn Sie es sich anders überlegen und sich entscheiden, die abgetrennten Ausgänge doch zu verwenden? Es ist einfach, die Verbindungen wieder herzustellen. Klicken Sie dazu auf einen Eingang oder Ausgang, den Sie anschließen wollen, ziehen Sie mit gedrückter Maustaste die Maus auf die “Gegenstelle”, zu der die Verbindung führen soll, und lassen Sie dort die Taste los. Klicken Sie zum Beispiel auf den Ausgang “tri” des Moduls *MultiWave Osc* und ziehen Sie die Maus zum Ausgang “Tri”. Die Verbindung ist wieder da:





Natürlich haben wir noch lange nicht alle Tuning-Möglichkeiten für Core Cells behandelt, aber das soll für den Einstieg genügen. Sie lernen im Verlauf dieses Handbuchs noch viele weitere Optionen kennen.

## Der Einstieg in REAKTOR Core

### Die Core-Cell-Typen Event und Audio

Core Cells existieren in zwei Arten: Event und Audio. Event-Core-Cells können nur Event-Signale des Primary Level an ihren Eingängen empfangen und als Antwort auf solche Signale nur Primary-Level-Event-Signale an ihren Ausgängen bereitstellen. Audio-Core-Cells sowohl Event- als auch Audio-Signale empfangen, aber nur Audio-Signale ausgeben. Zur Übersicht hier eine Tabelle:

Art	Eingänge	Ausgänge	Clock Src
Event	Event	Event	Ausgeschaltet
Audio	Event/Audio	Audio	Eingeschaltet

Daher können Audio Cells Oszillatoren, Filter, Hüllkurven, Effekte und andere Dinge enthalten, während sich Event Cells nur zur Verarbeitung von Event-Signalen eignen. Die Module *HighShelf EQ* und *MultiWave Osc*, die Sie ja schon kennen, sind Beispiele für Audio-Core-Cells (was Sie auch daran erkennen, dass sie Audio-Ausgänge besitzen):

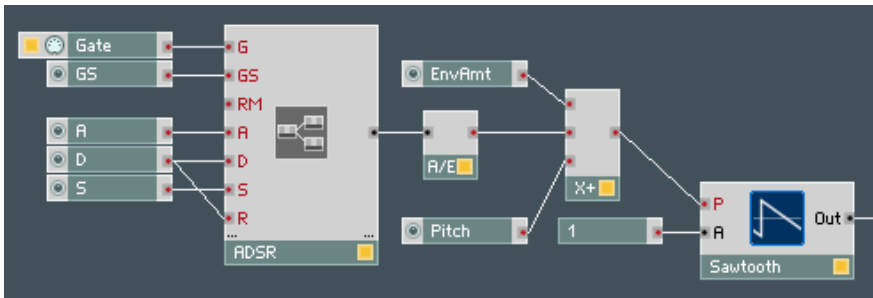


Und hier sehen Sie ein Beispiel für eine Event-Core-Cell:



Dieses Modul ist ein parabolischer Former für Kontroll-Signale, wie er zum Beispiel beim Implementieren von Velocity-Kurven oder beim Modellieren von LFO-Signalen zum Einsatz kommen kann.

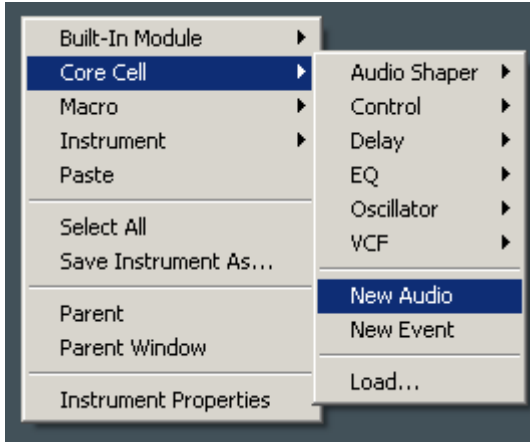
Wir haben ja bereits festgestellt, dass Core Cells vom Typ Event auf die Abarbeitung von Event-Aufgaben beschränkt sind. Weil Clock-Quellen (Clock Src) in ihnen abgeschaltet sind (siehe obige Tabelle), können sie keine eigenen Events erzeugen. Deshalb können sie nicht zum Implementieren von Module wie Event-getakteten LFOs oder Hüllkurven dienen. Wenn Sie jemals etwas derartiges vorhaben, verwenden Sie am besten eine Core Cell vom Typ Audio, deren Ausgangssignal Sie von einem Audio-nach-Event-Konverter des Primary Level auf Event-Taktung umsetzen lassen:



Die obige Struktur verwendet eine Audio-Core-Cell mit einer ADSR-Hüllkurve, deren Ausgangssignal auf Event-Takt umgesetzt wird, um einen Oszillator zu modulieren.

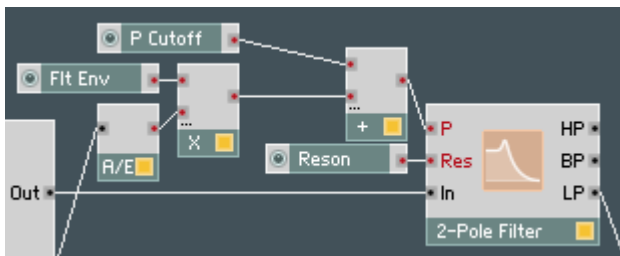
## Wie Sie Ihre erste Core Cell erstellen

Sie können neue Core Cells erzeugen, indem Sie einen Rechts-Klick in den Hintergrund einer Primary-Level-Struktur ausführen und aus dem Menü *Core Cell > New Audio* oder (für Event-Core-Cells) *Core Cell > New Event*:



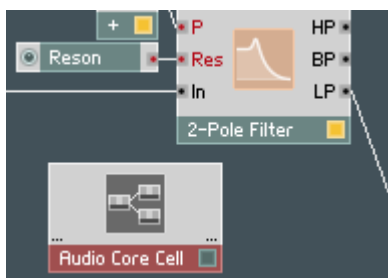
Wir werden nun eine Core Cell von Grund auf neu bauen, und zwar in demselben Ensemble *One Osc.ens*, mit dem Sie schon gespielt haben. Wir werden dabei die modifizierte Version des Ensembles mit dem neuen Oszillator und Chorus verwenden, die wir schon zusammengeschaubt haben. Wenn Sie diese Version nicht gespeichert haben, macht das auch nichts – Sie können alle Schritte auch an der Original-Version des Ensembles *One Osc.ens* nachvollziehen.

Wie Sie in diesem Ensemble sehen können, modulieren wir das Filter am Eingang P, der nur Event-Signale entgegennimmt. Wir verwenden nicht die FM-Version desselben Filters, weil diese erstens ein schlechteres Verhalten bei hohen Cutoff-Frequenzen zeigt und zweitens die lineare Skalierung am Eingang *FM* bei der Modulation durch eine Hüllkurve generell zu weniger musikalischen Ergebnissen führt (was oft nicht ganz korrekt als “langsame Hüllkurven” bezeichnet wird):

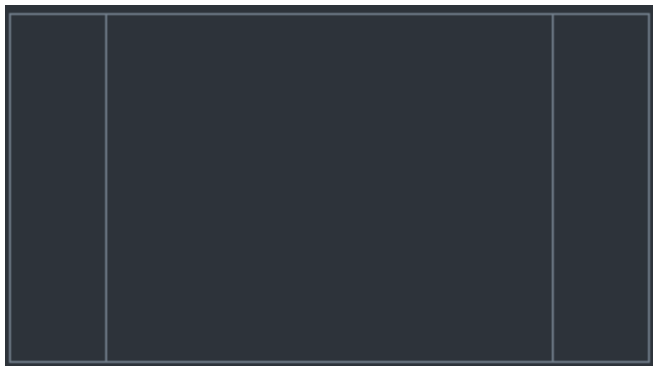


Weil wir am Event-Eingang modulieren wollen, müssen wir die Hüllkurve zunächst in ein Event-Signal umwandeln. Dazu verwenden wir einen Audio-nach-Event-Konverter (A/E). Dadurch wird die Rate unseres Kontroll-Signals ziemlich niedrig. Wir könnten natürlich auch einen Konverter mit einer deutlich höheren Taktung verwenden (der dann aber auch deutlich mehr CPU-Leistung verbraucht), aber wir werden stattdessen das Filter durch ein anderes ersetzen, das wir als Core Cell aufbauen. Alternativ hätten wir auch ein fertiges Filter aus der bestehenden Core-Cell-Library nehmen können, aber dann hätten wir uns um das Vergnügen gebracht, unsere erste eigene REAKTOR-Core-Struktur zu erstellen. Also gehen wir den nicht den einfachen Weg.

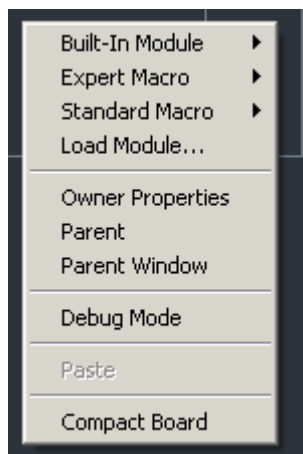
Wir beginnen mit dem Erzeugen einer neuen Core Cell vom Typ Audio, also wählen Sie *Core Cell > New Audio*. Es erscheint eine leere Audio-Core-Cell:



Doppelklicken Sie auf die neue Core Cell, um die innere REAKTOR-Core-Struktur zu sehen – die offensichtlich leer ist. Wie Sie sich bestimmt erinnern, sind die drei Bereiche für Eingänge (Input), Ausgänge (Output) und normale Module gedacht:

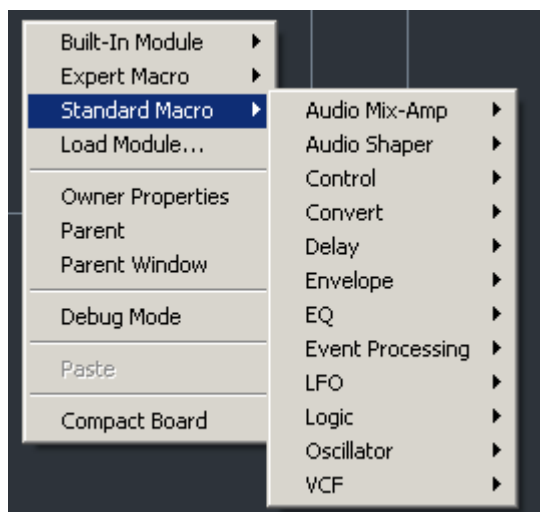


Aufgemerkt: Wir werden jetzt unsere erstes Modul in eine Core-Struktur einsetzen! Führen Sie dazu im Bereich für die normalen Module (Mitte) einen Rechts-Klick aus, um das Menü “Module Creation” aufzurufen:

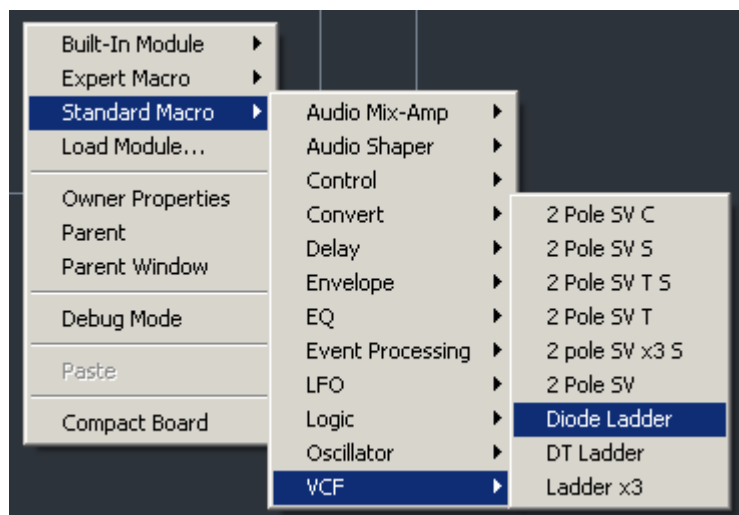


Das erste Untermenü heißt *Built In Module* und bietet Zugriff auf die eingebauten Module von REAKTOR Core, die für wirkliche Low-Level-Jobs gedacht sind und später noch behandelt werden.

Das zweite Untermenü heißt *Expert Macro* und enthält Macros, die gemeinsam mit den eingebauten Modulen für den Low-Level-Bereich zuständig sind. Das überspringen wir also auch erst einmal. Das dritte Untermenü heißt *Standard Macro* – hier scheinen wir richtig zu sein:

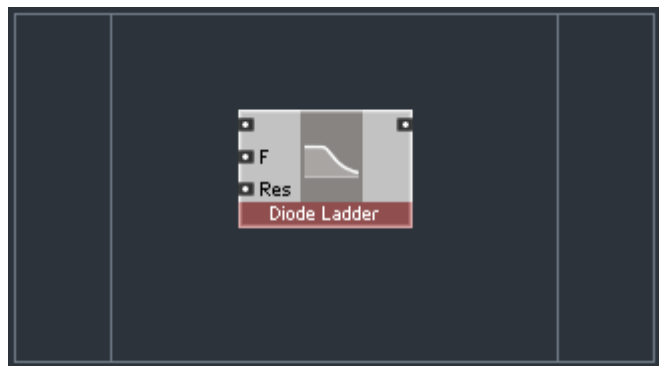


Die Abteilung *VCF* sieht ganz vielversprechend aus; schauen wir mal hinein:

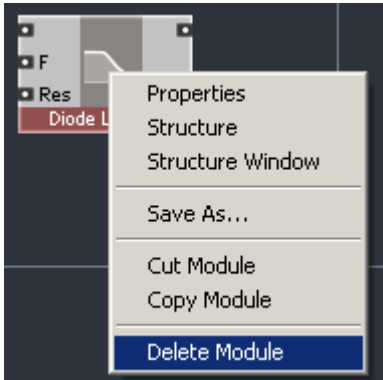


Sollten wir vielleicht mal *Diode Ladder* probieren?

Machen wir das doch einfach:

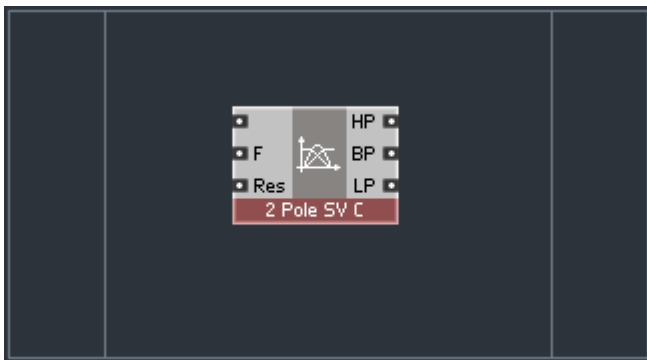


Nun, das war vielleicht nicht die allerbeste Idee, weil *Diode Ladder* wahrscheinlich ganz anders klingt als das Primary-Level-Filter, das wir ersetzen wollen. Die Dioden-Leiter *Diode Ladder* ist nämlich mindestens ein vierpoliges Filter (24 dB/ Oktave), während das Filter, das wir ersetzen wollen, ein zweipoliges Modell (mit 12 dB/ Oktave) ist. Lassen Sie uns das Modul *Diode Ladder* also wieder löschen. Dazu haben Sie zwei Möglichkeiten. Eine ist, einen Rechts-Klick auf das Modul auszuführen und aus dem Menü dem Eintrag *Delete Module* zu wählen:



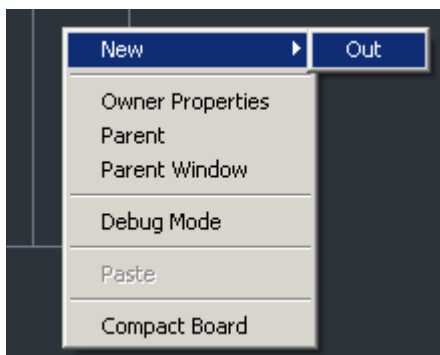
Die andere Möglichkeit ist, das Modul durch einen Klick darauf auszuwählen und dann einfach die Taste *Delete* zu drücken.

Nachdem wir die *Diode Ladder* gelöscht haben, lassen Sie uns das Filter-Modul *2 Pole SV C* aus der Abteilung *VCF* einsetzen:



Dies ist ein zweipoliges so genanntes State-Variable-Filter (Zustandsvariablenfilter), das dem Filter, das wir ersetzen wollen, ziemlich ähnlich ist (es gibt Unterschiede, aber die sind ziemlich subtil). Wichtig ist für uns vor allem, dass wir dieses Modul mit Audio-Rate modulieren können. Dazu brauchen wir offenbar noch einige Eingänge und einige Ausgänge für unsere Core Cell.

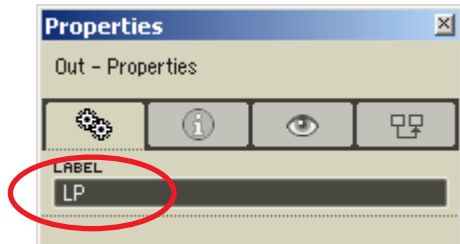
Um genau zu sein, brauchen wir wahrscheinlich nur einen Ausgang – für das *LP*-Signal. Um diesen Ausgang zu erzeugen, führen Sie einen Rechts-Klick im Ausgangs-Bereich der Core Cell aus:



Sie können hier nur eine Art von Modulen erzeugen, also wählen Sie die einfach aus. Die Struktur sollte dann so aussehen:



Doppelklicken Sie das frisch erzeugte Output-Modul, um das Properties-Fenster zu öffnen (wenn es nicht schon offen ist). Tippen Sie "LP" in das Feld "Label":



Nun verbinden Sie den Ausgang *LP* des Filters mit dem Output-Modul:



Lassen Sie uns nun mit den Eingängen beginnen. Der erste Eingang wird ein Audio-Signal-Input. Führen Sie einen Rechts-Klick in den Hintergrund des




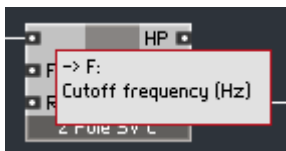
Eingangsbereich aus und wählen Sie aus dem Menü *New > In*:



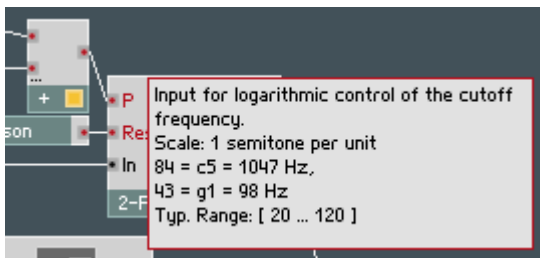
Der Eingang ist bereits vom richtigen Typ – wie Sie an dem großen schwarzen Punkt im Input-Modul sehen können, handelt es sich um einen Audio-Eingang. Sie können diesen Eingang auf dieselbe Weise, auf die Sie den Ausgang soeben in “LP” umbenannt haben, in “In” umbenennen. Verbinden Sie das Eingangs-Modul dann mit dem ersten Eingang des Filter-Moduls:



Der zweite Eingang ist eine etwas kompliziertere Angelegenheit. Wie Sie sehen können, heißt der zweite Eingang des REAKTOR-Core-Filters “F”. Das steht für “Frequency”, zu deutsch “Frequenz”. In der Tat sehen Sie, wenn Sie den Mauszeiger eine Weile auf diesem Eingang parken (und Sie den Schalter  aktiviert haben), den Text “Cutoff frequency (Hz)”:

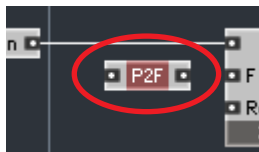


Wie wir bereits wissen, wird die Cutoff-Frequenz unseres Primary-Level-Filters von einem Eingang namens “P” kontrolliert. Wie der Info-Text erklärt, verwendet dieses Signal eine Halbton-Skala:

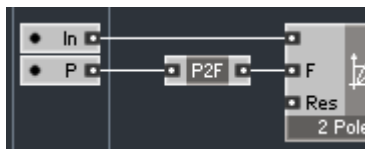


Wir müssen also offenbar eine Konvertierung von Halbtönen in Hertz durchführen. Das können wir entweder auf dem Primary Level (unter Verwend-

ung des Moduls Expon. (F)) oder innerhalb unserer REAKTOR-Core-Struktur tun. Weil wir gerade lernen, REAKTOR-Core-Strukturen zu bauen, wählen wir natürlich die zweite Möglichkeit. Führen Sie einen Rechts-Klick in den Hintergrund des normalen Bereichs aus und wählen Sie aus dem Menü *Standard Macro > Convert > P2F:structure*:



Wie der Name bereits andeutet (und auch der Info-Text verrät), konvertiert dieses Modul zwischen “P”- und “F”-Skalen – genau, was wir brauchen. Lassen Sie uns also ein zweites Eingangs-Modul namens “P” erzeugen und es unter Verwendung des Moduls *P2F* an den Eingang “F” des Filters anschließen:



Das sollte reichen – aber Moment mal, wir haben ja einen Drehregler namens “P Cutoff” in unserem Instrument, der den Basis-Cutoff-Wert des Filters bestimmt und dann dem Modulations-Signal aus der Hüllkurve hinzugefügt wird, das wir auf dem Primary Level in ein Event-Signal konvertieren müssen, um es in den Audio-Eingang “P” des Filters leiten zu können.

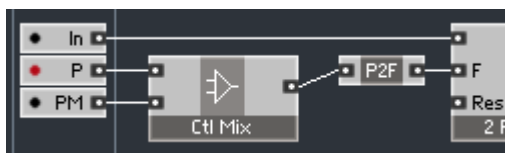
Diese Konvertierung ist jetzt natürlich nicht mehr notwendig – wir können das Modul *A/E* entfernen und das Audio-Signal direkt in den Audio-Eingang “P” unseres neuen Filters einspeisen. Mit dieser Vorgehensweise haben wir zwar kein Problem, aber wir suchen ja die sportliche Herausforderung und nehmen einen anderen Weg.

Lassen Sie uns also unseren Eingang “P” im Event-Modus belassen und einen anderen Modulations-Eingang im Audio-Modus verwenden. Wenn Sie sich an die Bemerkung über “langsame Hüllkurven” erinnern, fällt Ihnen sicher auf, warum wir dieses Modul lieber “PM” als “FM” nennen und die Modulation in einer Halbton-Skala (“Pitch”) stattfinden lassen. Genauso geschieht dies gegenwärtig in unserem Instrument – wir addieren unser Hüllkurven-Signal und das “P Cutoff”-Signal und leiten die Summe in den Eingang “P”.

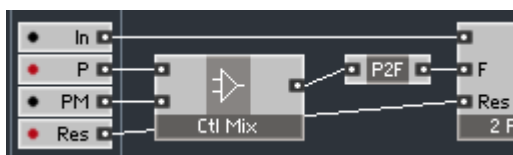
Also schalten wir unseren Eingang “P” in den Event-Modus um (Sie wissen ja schon, wie das geht) und fügen einen weiteren “PM”-Eingang hinzu, der sich offensichtlich schon im Audio-Modus befindet:



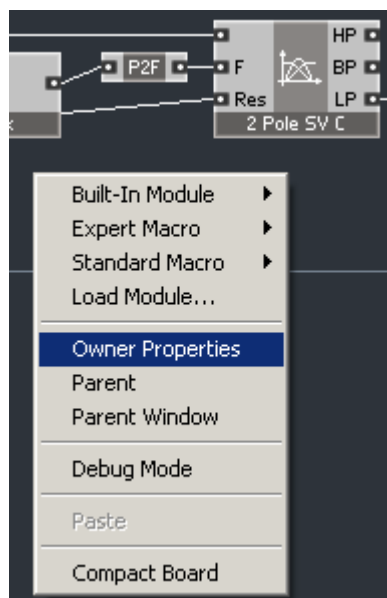
Als Anwender des Primary Level von REAKTOR würden Sie wahrscheinlich erwarten, dass wir die beiden Signale nun addieren. Das könnten wir tatsächlich tun, aber in REAKTOR Core gehört Add zu den Low-Level-Modulen und setzt die Kenntnis einiger grundlegender Arbeitsprinzipien der Low-Level-Funktionen von REAKTOR Core voraus. Diese Prinzipien sind nicht so besonders komplex und werden auch später in diesem Handbuch noch behandelt, aber im Moment brauchen Sie sie einfach nicht zu kennen. Verwenden Sie statt des Add-Moduls also einfach einen Mixer für Kontroll-Signale, zum Beispiel *Standard Macro > Control > Ctl Mix*:



Der letzte Eingang, den wir brauchen, ist ein Resonanz-Eingang, der aber nicht mit Audio-Rate arbeiten muss. Wir wählen also eine Event-Version:



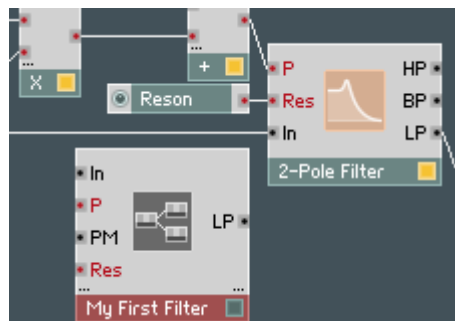
Eine weitere Sache, die wir erledigen müssen, ist, unserer Core Cell einen Namen zu geben. Um die Eigenschaften der Core Cell aufzurufen, können Sie einfach in den Hintergrund klicken, wenn das Properties-Fenster bereits geöffnet ist. Falls das Fenster nicht geöffnet ist, müssen Sie einen Rechts-Klick auf den Hintergrund ausführen und den Eintrag *Owner Properties* aus dem Menü wählen:



Sie können nun einen Namen in das Feld “Label” eintippen:

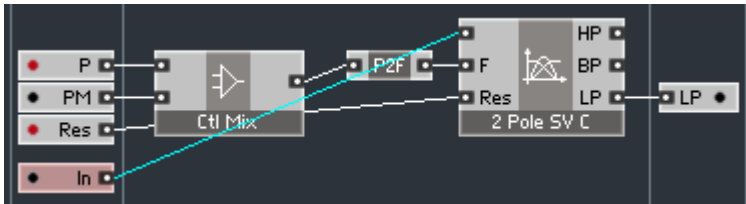


Doppelklicken Sie in den Hintergrund, um das Ergebnis zu sehen:



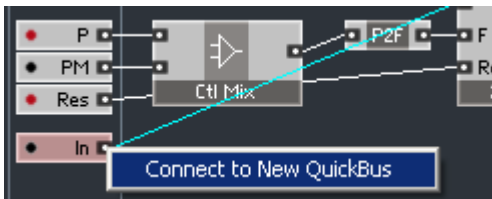
Wow, das sieht ja schon gut aus – abgesehen davon, dass der Audio-Signal-Eingang oben in der Core Cell sitzt, während er im Primary-Level-Filter unten angeordnet ist. Nun, das ist kein großes Problem, und wenn Sie es ändern wollen, ist das auch kein besonderer Aufwand; Sie wissen ja schon, wie das geht. Aber machen wir es doch einfach zusammen, dann lernen Sie unterwegs noch eine neue Funktion kennen.

Wir gehen also wieder zurück in die Core Cell und ziehen zunächst den Audio-Signal-Eingang ganz nach unten:

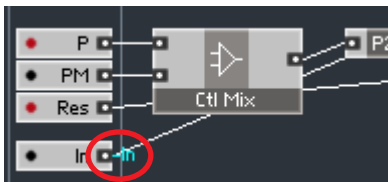


Das sollte schon die gewünschte Wirkung zeigen, abgesehen davon, dass ein diagonal über die ganze Struktur verlaufendes Kabel nicht besonders hübsch aussieht. Nun, das werden wir jetzt ändern.

Führen Sie einen Rechts-Klick auf das Eingangs-Modul “In” aus und wählen Sie aus dem Menü den Eintrag *Connect to New QuickBus*:



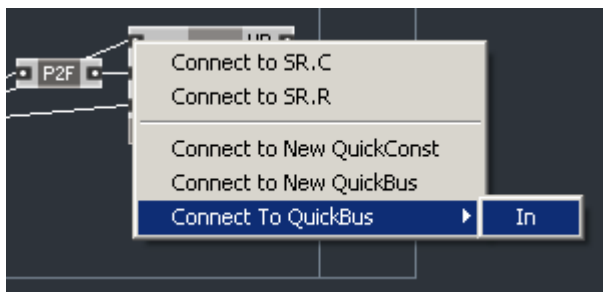
Sie sollten nun dies hier sehen:



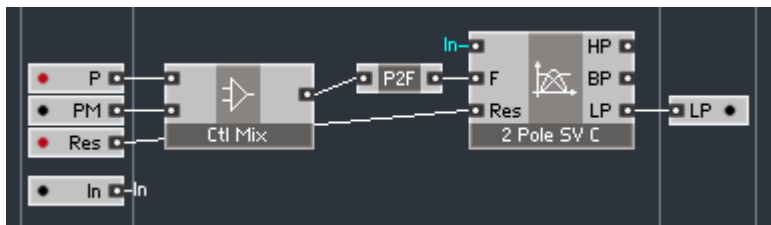
Außerdem sollte sich das Properties-Fenster öffnen und die Eigenschaften des “QuickBus” anzeigen, den Sie gerade erzeugt haben. Die nützlichste der hier angebotenen Eigenschaften der QuickBus-Verbindung ist natürlich der Name, den Sie ändern können. Die anderen Eigenschaften sind eher etwas für Fortgeschrittene, also lassen Sie jetzt erst einmal alles, wie es ist. Sie können später das Properties-Fenster öffnen, indem Sie auf den QuickBus klicken. Obwohl Sie Ihren QuickBus umbenennen könnten, ist es wahrscheinlich

günstiger, Sie behalten den Namen bei, denn der verweist auf den Eingang, der mit dem QuickBus verbunden ist. Die QuickBusse werden lokal innerhalb der Struktur verwaltet, also brauchen Sie sich keine Sorgen darüber zu machen, falls eine benachbarte oder eingebettete Struktur einen QuickBus mit demselben Namen enthält.

Als nächstes sollten Sie einen Rechts-Klick auf den oberen Eingang des Filter-Moduls 2 Pole SV C ausführen und aus dem Menü *Connect to QuickBus > In* wählen:

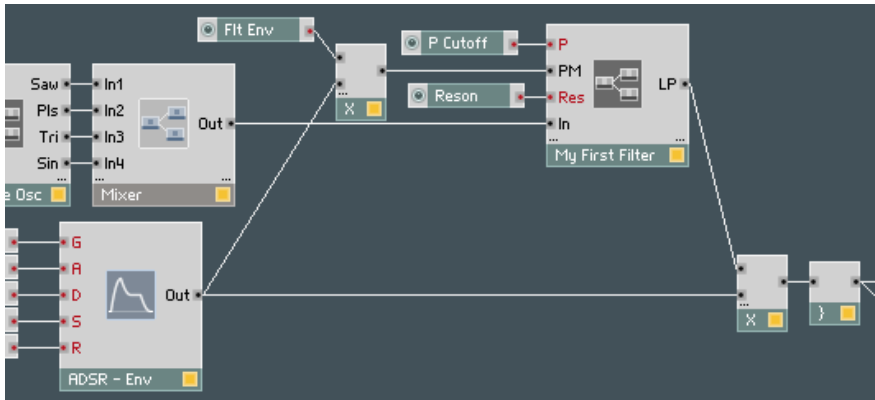


Im oben abgebildeten Menü ist "In" nichts anderes als der Name des Quick-Bus', mit dem Sie eine Verbindung herstellen möchten. Sie wollen keinen neuen QuickBus erzeugen, sondern eine Verbindung mit einem bestehenden QuickBus erzeugen, und das machen Sie hier. Danach sollte die Struktur so aussehen:



Anstelle des hässlichen diagonalen Kabels haben wir nun zwei nette Referenzen, deren Beschriftung besagt, dass sie mit einem QuickBus namens "In" verbunden sind.

Jetzt können wir auf das Primary Level zurückkehren und die Struktur für die Verwendung unseres neuen Filters, das wir gerade gebaut haben, modifizieren. Die Module *Add* und *A/E* können Sie löschen. Unser Ergebnis sollte dann so aussehen:



Verbraucht deutlich mehr CPU-Leistung, nicht wahr? Nun, vergessen Sie nicht, dass dieses Filter mit Audio-Rate in einer Tonhöhen-Skala moduliert wird. Wenn Ihnen das nicht gefällt, können Sie immer noch zu der alten Struktur zurückkehren oder das Filter-Modul *Multi 2 pole FM* auf dem Primary Level ("langsame Hüllkurven", erinnern Sie sich?) verwenden – aber wir hoffen, dass Sie es mögen. Falls nicht, gibt es noch ein paar andere Filter mit neuen Funktionen, die Ihnen vielleicht besser gefallen. Und wenn Ihnen keins der neuen REAKTOR-Core-Filter gefällt, können Sie immer noch eine ganze Ladung anderer REAKTOR-Core-Module ausprobieren.

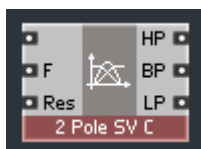
## Audio- und Kontroll-Signale

Bevor wir weitermachen, müssen wir uns zunächst eine besondere Konvention ansehen, welche die *Standard Macros* der REAKTOR-Core-Library betrifft. Die Module, die Sie in diesem Bereich finden, lassen sich am besten mit den verschiedenen Signal-Typen beschreiben: Audio, Control, Event und Logic. Wir werden Event- und Logic-Signale etwas später behandeln und uns zunächst auf die erstgenannten beiden Typen konzentrieren.

Audio-Signale sind offensichtlich Signale, die Audio-Informationen enthalten. Zu diesen gehören Signale, die an den Ausgängen von Oszillatoren, Filtern, Verstärkern und Delays abgegriffen werden. Module wie Filter, Verstärker, Sättiger (Saturators), Delays und so weiter nehmen an ihren Eingängen normalerweise Audio-Signale entgegen, die dann verarbeitet werden.

Kontroll-Signale transportieren keine Audio-Informationen, sondern werden nur verwendet, um einige Module zu kontrollieren. So geben zum Beispiel die Ausgänge von Hüllkurven und LFOs keinen Sound aus, und auch Keyboard-Pitch- und Velocity-Signale enthalten keine Audio-Informationen; sie alle können aber dazu eingesetzt werden, beispielsweise die Cutoff-Frequenz oder Resonanz eines Filters, eine Delay-Zeit oder andere Parameter zu kontrollieren. Entsprechend sind die Cutoff- und Resonanz-Eingänge eines Filters oder der Time-Input eines Delays darauf eingerichtet, Kontroll-Signale zu empfangen.

Hier sehen Sie ein Beispiel mit dem REAKTOR-Core-Filter-Modul, das Sie schon kennen:



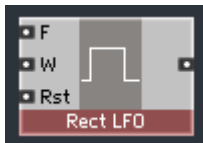
Der obere Eingang des Filters nimmt das Audio-Signal, das gefiltert werden soll, entgegen; er erwartet daher ein Signal vom Typ Audio. Die Eingänge *F* und *Res* sind offenbar Eingänge des Typs Control. An den Ausgängen des Filters liegen auf verschiedene Arten gefilterte Audio-Signale an, also sind diese Ausgänge alle vom Typ Audio.

Ein Sägezahn-Oszillator-Modul andererseits hat nur einen einzelnen Kontroll-Eingang (für die Frequenz) und einen einzelnen Audio-Ausgang:





Und wenn wir uns das Modul *Rect LFO* ansehen, stellen wir fest, dass es zwei Kontroll-Eingänge – für die Frequenz (*F*) und für die Pulsweite (*W*) – und einen Kontroll-Ausgang besitzt (weil es oft verwendet wird, um Dinge wie Filter-Cutoff oder VCA-Pegel zu kontrollieren):



---

Einige Arten der Signalverarbeitung wie zum Beispiel Mixing lassen sich sowohl auf Audio- als auch auf Kontroll-Signale sinnvoll anwenden. In solchen Fällen finden Sie gesonderte Versionen der entsprechenden Macros für beide Signal-Typen. Es gibt zum Beispiel Audio-Mixer und Control-Mixer, Audio-Verstärker und Control-Verstärker und so weiter. Im Allgemeinen ist es keine besonders gute Idee, ein Modul für die Verarbeitung des jeweils anderen Signal-Typs zu missbrauchen – Sie sollten in solchen Fällen schon genau wissen, was Sie tun.

---

---

Jetzt, da wir das geklärt haben, würden wir gerne noch anmerken, dass es oft möglich ist, Audio-Signale als Kontroll-Signale aufzubereiten. Das am häufigsten anzutreffende Beispiel für ein solches “Signal-Recycling” ist die Modulation einer Oszillator-Frequenz oder eines Filter-Cutoff-Werts durch ein Audio-Signal. Das ist absolut in Ordnung, weil Sie ja in solchen Fällen das Audio-Signal als Kontroll-Signal verwenden *wollen*. Wir wagen aber die Behauptung, dass Fälle, in denen Sie ein Kontroll-Signal als Audio-Signal verwenden wollen, ziemlich selten sein dürften.

---

Die Abgrenzung zwischen Signalen der Typen Audio, Control, Event und Logic sollten Sie aber nicht mit der Unterscheidung zwischen Audio und Event auf dem Primary Level verwechseln. Die Klassifikation der Signale auf dem Primary Level in Audio und Event bezieht sich, grob gesagt, auf die “Verarbeitungsgeschwindigkeit” und die Tatsache, dass Audio “schneller” und verarbeitet wird und die CPU stärker belastet. Wie Sie wahrscheinlich auch wissen, gelten auf dem Primary Level für Event-Signale andere Fortpflanzungsregeln als für Audio-Signale. Der Unterschied zwischen Audio-, Kontroll- und Event-Signalen in der Terminologie von REAKTOR Core ist rein semantisch und beschreibt den “Gehalt” oder die “Bedeutung” des Signals, nicht die Art seiner Verarbeitung. Es gibt also keine direkten Entsprechungen zwischen den Nomenklaturen

von Primary Level und REAKTOR Core, aber wir können trotzdem versuchen, dieses Verhältnis zu erklären.

- Ein Primary-Level-Audio-Signal entspricht normalerweise entweder einem REAKTOR-Core-Audio-Signal (z. B. dem Ausgangssignal eines Oszillators oder eines Audio-Filters) oder einem REAKTOR-Core-Kontroll-Signal (zum Beispiel dem Ausgangssignal einer Hüllkurve)
- Ein Primary-Level-Event-Signal ist in der Nomenklatur von REAKTOR Core typischerweise ein Kontroll-Signal. Ein Beispiel für ein solches Signal wäre die Ausgabe eines LFOs, eines Drehreglers oder einer Quelle für MIDI-Pitch- und Velocity-Informationen.
- Manchmal entspricht ein Primary-Level-Event-Signal einem REAKTOR-Core-Event-Signal. Das typische Beispiel dafür ist ein MIDI-Gate (wie versprochen, werden wir die Event-Signale in REAKTOR Core später noch behandeln).
- Manchmal *ähnelt* ein Primary-Level-Event-Signal einem REAKTOR-Core-Logic-Signal, obwohl diese beiden Signal-Typen nicht vollständig kompatibel sind und auf jeden Fall eine Konvertierung zwischen ihnen stattfinden muss (dieses Thema werden wir später noch behandeln). Beispiele für eine solche Ähnlichkeit sind Signale, die von Primary-Level-Modulen wie *Logic AND* verarbeitet werden.

---

Es ist wichtig, sich vor Augen zu führen, dass Sie beim Festlegen des Typs eines Core-Cell-Eingangs zwischen den Primary-Level-Signal-Typen Audio und Event wählen. nicht zwischen Audio- und Event-Signalen in REAKTOR Core. An den Core-Cell-Ports treffen die beiden Welten aufeinander, deshalb ähneln ihre Bezeichnungen der Primary-Level-Terminologie.

---

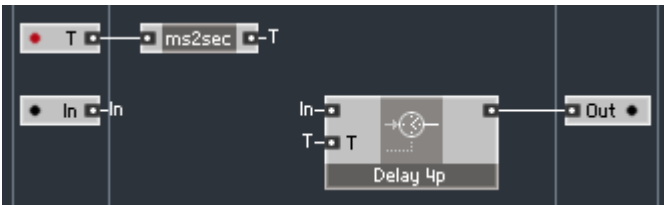
Wir werden nun etwas mehr über dieses Konzept lernen, während wir versuchen, eine Bandecho-Simulation aufzubauen. Dabei bauen zunächst wir ein einfaches digitales Echo, das wir dann erweitern, sodass es einige Funktionen eines Bandechos nachahmen kann. Lassen Sie uns zunächst eine leere Core Cell vom Typ Audio anlegen. Öffnen Sie die Core Cell und geben Sie ihr den Namen "Echo". Das erste Modul, das wir in die Struktur einbauen, ist ein Delay-Modul. Wir werden ein 4-Point-Interpolations-Delay auswählen, weil das eine bessere Qualität liefert als die 2-Point-Version, und ein nicht-interpolierendes Delay eignet sich nicht für unsere Bandecho-Simulation. Wählen Sie also aus dem Menü *Standard Macro > Delay > Delay 4p*:



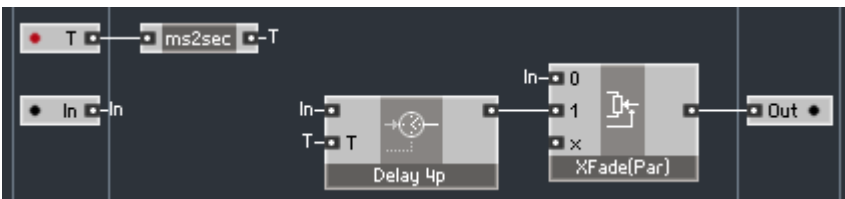
Ganz offensichtlich brauchen wir einen Audio-Eingang und einen Audio-Ausgang für unsere Delay-Core-Cell. Wir verwenden eine QuickBus-Verbindung für den Eingang und eine normale Verbindung für den Ausgang:



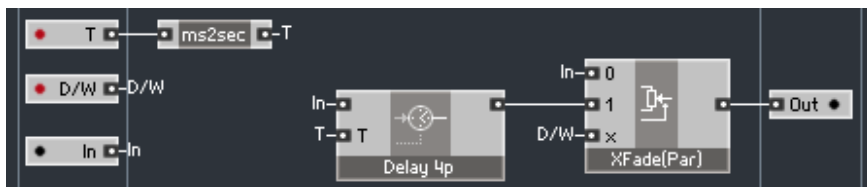
Wir brauchen außerdem einen Event-Eingang, um die Delay-Zeit zu kontrollieren. Dabei müssen wir darauf achten, dass auf dem Primary Level die Delay-Zeit normalerweise in Millisekunden ausgedrückt wird, während die Delay-Macros aus der Library von REAKTOR Core eine Angabe in Sekunden erwarten. Kein Problem für uns, denn wir haben das passende Konverter-Modul zur Hand – wählen Sie aus dem Menü *Standard Macro > Convert > ms2sec*:



Bisher haben wir aber erst ein einzelnes Echo, und dann wäre es auch noch nett, auch das Original-Signal zu hören, nicht nur den Echo-Anteil. Um das Original-Signal an den Ausgang zu leiten, müssen wir es mit dem verzögerten Signal mischen. Weil wir Audio-Signale mischen wollen, müssen wir hier einen Audio-Mixer verwenden (im Unterschied zu dem Control-Mixer, den wir beim Aufbau der Filter-Core-Cell zum Mischen der Kontroll-Signale verwendet haben). Noch besser können wir einen speziellen Audio-Mixer-Typ verwenden, nämlich einen Crossfader, der eigens dafür entworfen wurde, Kreuzblenden zwischen zwei Signalen durchzuführen: *Standard Macro > Audio Mix-Amp > XFade (par)*:



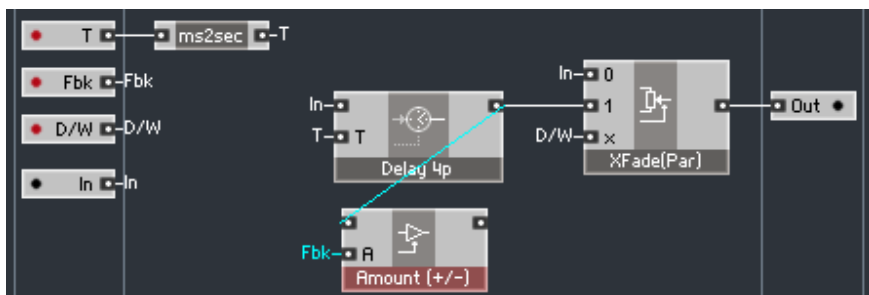
“(par)” steht für parabolisch – das erzeugt eine natürlicher klingende Kreuzblende als ein linear arbeitender Crossfader. Der Control-Input “x” des Crossfade-Macros ist noch nicht angeschlossen, wir werden ihn mit einem neuen Event-Eingang verbinden, der das Mischungsverhältnis zwischen trockenem (unbearbeitetem) und nassem (verzögerten) Signale kontrolliert. Wenn das Kontroll-Signal den Wert 0 annimmt, werden wir nur das Original-Signal hören, wenn es den Wert 1 hat, hören wir nur den verzögerten Signal-Anteil:



Das ist schon viel besser, denn wir können jetzt das Original-Signal und das Echo hören. Aber es ist immer noch nur ein Echo. Um mehrfache Echos zu erzeugen, müssen wir einen Bruchteil des verzögerten Signals wieder in den Eingang des Delays einspeisen. Dafür müssen wir das verzögerte Signal zunächst abschwächen. Den bekannten Richtlinien entsprechend – verwenden Sie einen Audio-Verstärker, um ein Audio-Signal abzuschwächen – wählen wir *Standard Macro > Audio Mix Amp > Amount*.

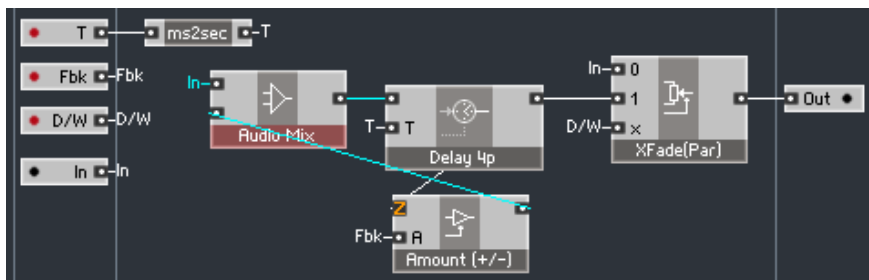


Wir nehmen das Modul *Amount*, weil wir den Betrag (Amount) des Signals kontrollieren möchten, der zurück in den Eingang des Delays geführt wird. Außerdem erlaubt uns dieser Verstärker, das Signal durch die Verwendung negativer Werte für den Betrag umzukehren. Im Gegensatz dazu wären wir mit einem Verstärker (dB), der sich eignen würde, die Lautstärke des Signals zu kontrollieren, hier nicht gut bedient, weil dieser nicht die Invertierung von Signalen erlaubt. Wir verbinden also den Amplituden-Kontroll-Eingang des Verstärkers mit einem Event-Eingang, der den Feedback-Anteil kontrolliert:

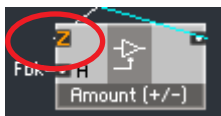


Der vernünftige Bereich für den Feedback-Anteil liegt hier etwa zwischen [ 0.9..0.9]. Sollten Sie dieses Delay bereits im nächsten Schritt ausprobieren wollen, seien Sie vorsichtig mit dem Feedback-Anteil, weil Sie leicht zu hohe Signalpegel bekommen (es gibt noch keine Sättigung in unserer Schaltung). Wir hätten auch zur Sicherheit einen Feedback Amount Clipper in unsere Core Cell einbetten können, der die Signalspitzen beschneiden würde, aber weil wir später ohnehin noch ein Sättigungs-Modul hinzufügen werden, erscheint das nicht notwendig. Im Gegenteil, Sie könnten mit hohen Feedback-Pegeln experimentieren und würden die Sättigung des Delays hören.

Wir müssen das Feedback-Signal mit dem Eingangssignal mischen. Ein Audio-Mixer (*Standard Macro > Audio Mix Amp > Audio Mix*) ist dafür die beste Wahl:



Sie fragen sich vielleicht, was mit dem oberen Eingang des Moduls Amount in der obigen Abbildung passiert ist, das nun ein großes orangefarbenes "Z" zeigt:



Tatsächlich könnte dieses “Z”, abhängig von der Software-Version und anderen Randbedingungen, auch an anderen Eingängen der Struktur auftauchen, aber das sollte Ihnen nicht allzu viel Kopfzerbrechen bereiten. Dieses Zeichen weist darauf hin, dass ein digitales Feedback in der Struktur auftritt, und ist für das Entwerfen aufwendigerer Strukturen von Bedeutung; dort kann es dem Struktur-Designer wertvolle Hinweise geben.

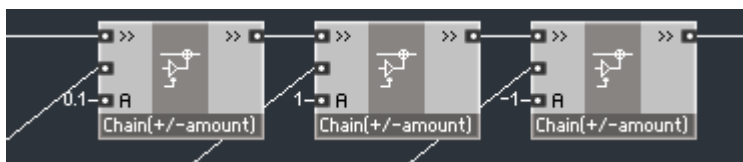
Für einfache Strukturen wie die oben abgebildete brauchen Sie diesen Hinweis normalerweise nicht zu beachten. Seine Anwesenheit bedeutet nur, dass an dem markierten Punkt eine Verzögerung von einem Sample Länge (ungefähr 0,02 ms bei einer Sampling-Rate von 44,1 kHz, noch weniger bei höheren Sampling-Raten) entsteht. Wir wagen aber zu behaupten, dass Sie es gar nicht bemerken, wenn Ihre Delay-Zeit um 0,02 Millisekunden von dem eingestellten Wert abweicht.

Lassen Sie uns zu unserer Struktur zurückkehren. Diese kann jetzt eine Serie schwächer werdender Echos erzeugen. Das sollte für ein digitales Echo in Ordnung sein, aber wir wollen die Gelegenheit nutzen und Ihnen eine andere Funktion der Library zeigen, mit der Sie Ihre Struktur verkleinern können.

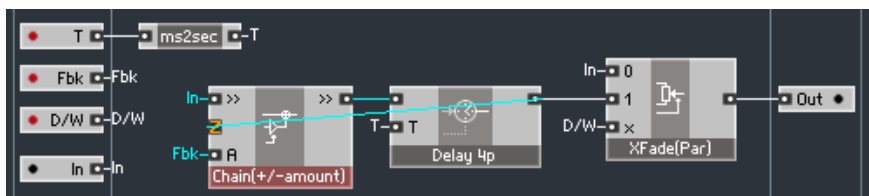
Unter den Audio-Verstärkern finden sich Verstärker, die “Chain” heißen. Diese Verstärker sind in der Lage, ein vorhandenes Signal zu verstärken und es einem anderen “verketteten” Signal hinzuzumischen. Einer dieser Verstärker ist das Modul *Audio Mix Amp > Chain(amount)*, das ähnlich wie das Modul *Amount* arbeitet, aber zusätzlich “verkettetes” Mixing beherrscht:



Das Signal am zweiten Eingang dieses Moduls wird gemäß dem am Eingang “A” anliegenden Betrag abgeschwächt und dem Signal am “Ketten”-Eingang (“>>”) hinzugemischt. Das Signal am Ketten-Eingang wird nicht abgeschwächt. Solche Verstärker können Sie verwenden, um Mixer-Ketten zu bauen, wobei die Verbindungen zwischen den mit “>>” bezeichneten Ports eine Art Mixing-Bus ergeben:



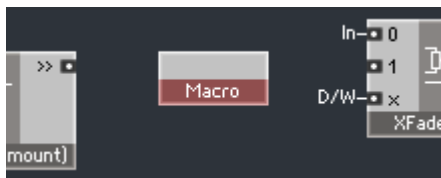
In unserem Fall brauchen wir keinen Mixing-Bus, aber wir können dieses Modul verwenden, um unsere beiden Module *Audio Mix* und *Amount* zu ersetzen. Das zurückgeführte Signal wird genau wie zuvor dem über den Eingang *Fbk* bestimmten Betrag gemäß abgeschwächt und dem Eingangssignal zugemischt:



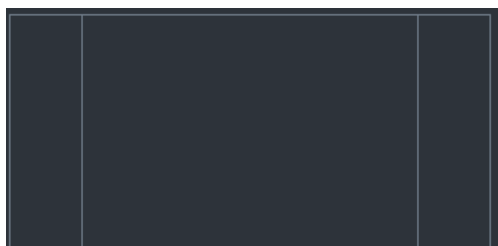
Herzlichen Glückwunsch, Sie haben einen einfachen digitalen Echo-Effekt gebaut! Im nächsten Schritt fügen wir dem etwas Bandedcho-Charakter hinzu.

## Wie Sie Ihre ersten REAKTOR-Core-Macros bauen

In dem Echo-Effekt, den wir gerade gebaut haben, haben wir das Macro *Delay 4p* aus der Library verwendet, das uns ein digitales Delay von hoher Qualität erzeugt. Hohe Qualität hin oder her, es klingt noch zu digital. Wir könnten es wärmer klingen lassen, indem wir verschiedene Funktionen hinzufügen, die man in einem Bandedcho findet, zum Beispiel Sättigung und Flattern. Dann können wir das Macro *Delay 4p* durch ein Bandedcho-Macro ersetzen, das wir jetzt bauen werden. Löschen Sie also zuerst das besagte Delay-Macro aus der Struktur. Legen Sie dann ein neues, leeres Macro an. Führen Sie dazu einen Rechts-Klick in den Hintergrund aus und wählen Sie aus dem Menü *Built In Module > Macro*:

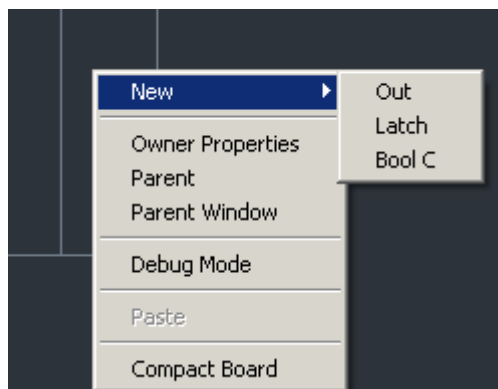


Doppelklicken Sie das neue Macro, um hineinzutauchen. Sie sehen eine leere Struktur, ähnlich der, aus der Sie abtauchen:



Es arbeitet auch ähnlich wie das vorherige, aber es gibt einige wichtige Unterschiede. Vor allem war die vorherige Struktur die einer Core Cell, während wir uns nun in der inneren Struktur eines REAKTOR-Core-Macros befinden.

Tatsächlich haben diese Unterschiede mit dem Sortiment der verfügbaren Eingangs- und Ausgangs-Module zu tun. Die sind einfach verschieden:

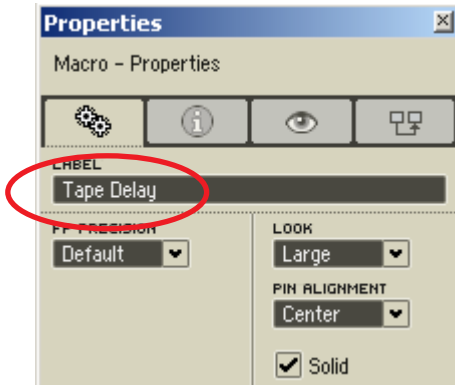


Die Port-Typen *Latch* und *Bool C* werden viel später in diesem Handbuch erklärt und kommen für fortgeschrittene Aufgaben zum Einsatz. Wir interessieren uns hier nur für den ersten Typ, der einfach “Out” heißt (oder “In”, wenn es sich um den Eingangs-Bereich handelt). Dies ist ein vielseitiger Port, der Signale der Typen Audio, Control, Event und Logic entgegennimmt. Genau genommen ist es dem Port egal, welcher Typ von Signal anliegt; der Unterschied zwischen den Signal-Typen ist nur für Sie als Anwender von Bedeutung, weil er beschreibt, wie das Signal verwendet werden soll. Für REAKTOR Core sind alle Signal-Typen gleich. Es gibt auch keinen Unterschied zwischen Audio- und



Event- Ein- und Ausgängen wie in der vorigen Struktur, weil wir es nun nicht mehr mit Signalen von REAKTORs Primary Level zu tun haben – das hier ist REAKTOR Core pur.

Zuerst werden wir unser neues Macro benennen. Das funktioniert genauso wie bei den Core Cells – führen Sie einen Rechts-Klick in den Hintergrund aus und wählen Sie aus dem Menü den Eintrag *Owner Properties*. Im Properties-Fenster tragen Sie in das Feld “Label” einen Namen ein:



Die übrigen Eigenschaften des Macros betreffen sein Aussehen und die Signalverarbeitung.

---

Obwohl es Ihnen freisteht, mit den anderen im Properties-Fenster angezeigten Parametern zu experimentieren, sollten Sie das Kästchen *Solid* auf jeden Fall angekreuzt lassen. Auch Änderungen der Einstellungen im Ausklapp-Menü *FP Precision* sollten Sie sich gut überlegen. Die Bedeutung dieser Parameter werden bei den weiterführenden Themen in diesem Handbuch behandelt.

---

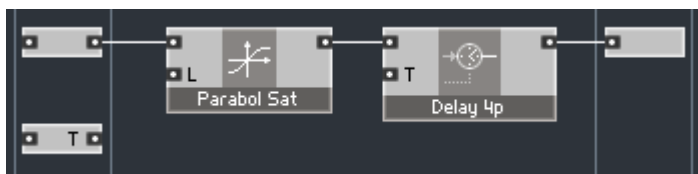
Als nächstes erzeugen wir einen Satz von Eingängen und Ausgängen für unser Macro *Tape Delay*:



Der obere Eingang wird das Audio-Eingangssignal empfangen, der untere den Zeit-Parameter. Sie haben vielleicht die zusätzlichen Ports an der linken Seite der Input-Module bemerkt; die erklären wir ein wenig später. Als zentralen Bestandteil unseres Macros verwenden wir dasselbe Delay-Modul Delay 4p




Eine einfache Emulation des Sättigungs-Effekts können wir leicht erzeugen, indem wir ein Sättigungs-Modul vor das Delay schalten. Der Sättiger (Saturator) ist eine Art Signalformer, also suchen wir ihn unter den Audio-Shaper-Modulen (weil er ein Audio-Sättiger ist). Wählen Sie also aus dem Menü *Standard Macro > Audio Shaper > Parabol Sat*:

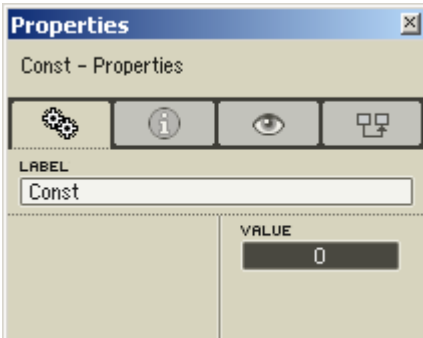


Das Eingangssignal wird nun im Bereich zwischen  $-1$  und  $+1$  gesättigt. Tatsächlich wird dieser Bereich vom Eingang "L" des Sättiger-Moduls kontrolliert; wenn dieser Eingang nicht belegt ist, liegt als Default-Wert 1 an. Das hört sich für Sie vielleicht überraschend an, weil Sie wahrscheinlich gewöhnt sind, dass nicht beschaltete Eingänge so behandelt werden, als ob sie kein Signal empfangen, oder, anders gesagt, als ob sie den Wert 0 empfangen. Nun, das ist nicht wirklich der Fall in REAKTOR-Core-Strukturen – die Module können hier für nicht belegte Eingänge eine besondere Behandlung vorsehen. So ist zum Beispiel für den Eingang "L" unseres Sättiger-Moduls festgelegt, dass er den Default-Wert 1 annehmen soll, wenn er nicht verbunden ist.

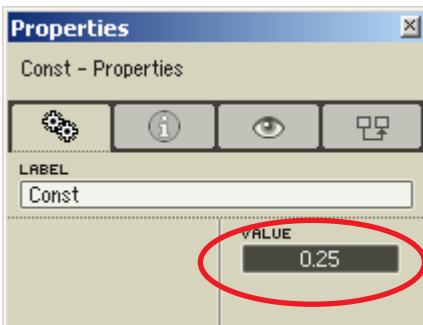
Nun wollen wir das Festlegen dieser Default-Werte am Beispiel unseres Eingang "T" lernen. Lassen Sie uns also für den Fall, dass der Eingang "T" nicht belegt ist, bestimmen, dass dieser Eingang so behandelt wird, als ob sein Eingangs-Wert bei 0,25 Sekunden läge. Das ist ganz leicht: Führen Sie einen Rechts-Klick auf diesen Port auf der linken Seite des Eingangs-Moduls "T" aus und wählen Sie aus dem Menü *Connect to New QuickConst*. Sie sollten nun dies hier sehen:



Zusätzlich sollte das Properties-Fenster die Eigenschaften dieser Konstante anzeigen (wenn es eine andere Ansicht zeigt, klicken Sie auf die Schaltfläche ):



Tippen Sie in das Feld "Value" den neuen Wert 0.25:



So sollte die QuickConst jetzt in der Struktur aussehen:



Lassen Sie uns kurz erklären, was wir gerade gemacht haben. Der Port auf der linken Seite des Eingangs-Moduls spezifiziert ein so genanntes "Default-Signal". Das heißt, dass dieses Default-Signal als Quelle für den Eingang dient, wenn der Eingang nicht verbunden ist (außerhalb des Macros). In unserem Fall wird sich der Eingang "T" des Macros Tape Delay verhalten, als ob er den konstanten Wert 0.25 empfinde, wenn er außerhalb des Macros nicht angeschlossen ist.

Eine Verbindung mit QuickConst ist natürlich nicht die einzige mögliche Verbindung für diesen Default-Signal-Eingang. Sie können ihn auch mit beliebigen anderen Modulen der Struktur verbinden, andere Eingangs-Module eingeschlossen.

Jetzt, da wir eine Sättigung und einen Default-Wert für den Eingang "T" haben, lassen Sie uns den Band-Flutter-Effekt (Flutter) emulieren. Ein einfacher Weg dafür wäre, die Delay-Zeit von einem LFO modulieren zu lassen. Man könnte nun auf der Suche nach dem besten Flutter-Effekt mit verschiedenen LFO-Wellenformen experimentieren, aber wir schlagen hier einfach vor, dass Sie einen der in der Library enthaltenen LFOs verwenden, und zwar *Standard Macro > LFO > Par LFO*:

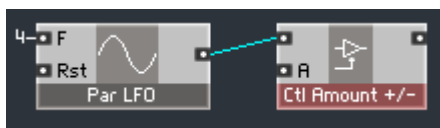


Dies ist ein parabolischer LFO, der ein Signal mit einer Form erzeugt, die einer Sinus-Welle ähnelt; er verbraucht allerdings weniger CPU-Leistung. Am Eingang "F" dieses Moduls muss ein Signal anliegen, das die Oszillations-Rate vorgibt. Wir können hier wieder QuickConst verwenden. Eine Rate von 4 Hz scheint vernünftig, also versuchen wir das mal:



Der Eingang "Rst" wird benötigt, um den LFO neu zu starten; diese Funktion brauchen wir noch nicht.

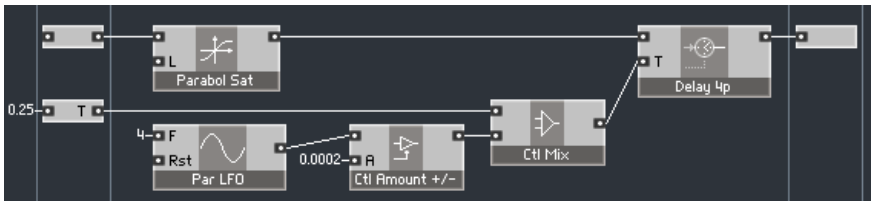
Jetzt müssen wir einen Modulations-Grad festlegen, indem wir den Ausgang des LFOs skalieren; dessen Signal variiert zurzeit im Bereich zwischen  $-1$  und  $+1$ , und das ist viel zu viel. Wenn Sie sich erinnern, dass wir es hier mit Kontroll-Signalen zu tun haben, wissen Sie natürlich, dass wir hier entsprechend ein Amount-Modul für Kontroll-Signale verwenden müssen, ähnlich dem Verstärker-Modul *Amount*, das wir für Audio verwendet haben. Wählen Sie also *Standard Macro > Control > Ctl Amount +/-*:



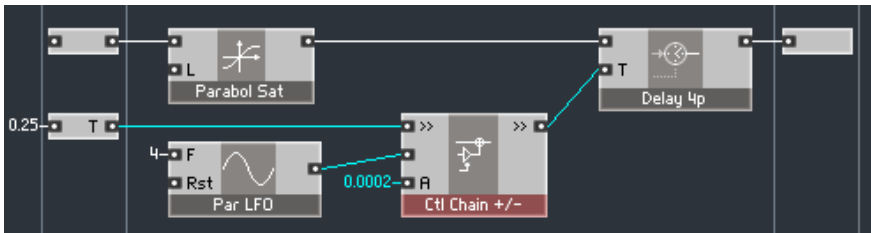
Eine Modulations-Amplitude von 0.0002 sollte gut passen, also skalieren wir das Signal auf diesen Betrag:




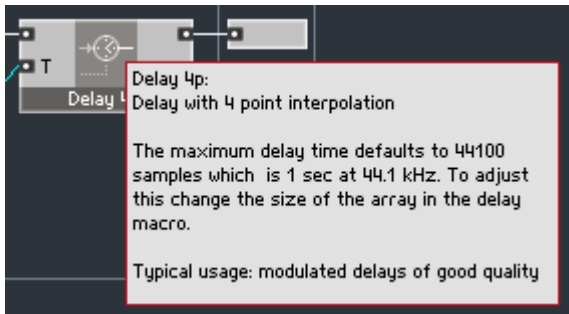
Abschließend können wir die beiden Kontroll-Signale (eins aus dem Eingang "T" und eins aus dem Modul *Ctl Amount*) mischen und sie in den Eingang "T" des Delay-Moduls einspeisen. Das bereits bekannte Modul *Ctl Mix* kann hier wieder zum Einsatz kommen:



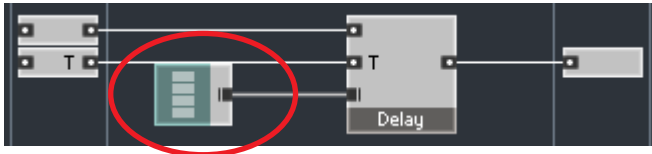
Tatsächlich haben wir aber einen ähnlichen "Ketten"-Typ von Kontroll-Mixer, wie wir ihn für Audio-Signale kennen gelernt haben. Wir könnten diesen Mixer verwenden, um die Module *Ctl Amount* und *Ctl Mix* auf ganz ähnliche Weise zu ersetzen, wie wir das im Audio-Pfad gemacht haben. Wählen Sie also *Standard Macro > Control > Ctl Chain*:




Ein letzter Schliff für unser Macro – wir werden nun die Puffergröße (Buffer Size) für unser Delay ändern, welche die maximal mögliche Delay-Zeit definiert. Wenn Sie den Mauszeiger über das Macro *Delay 4p* halten (und der Schalter  aktiv ist), können Sie im Hinweis-Text lesen, dass die Default-Puffergröße einer Sekunde Verzögerung bei 44,1 kHz entspricht:

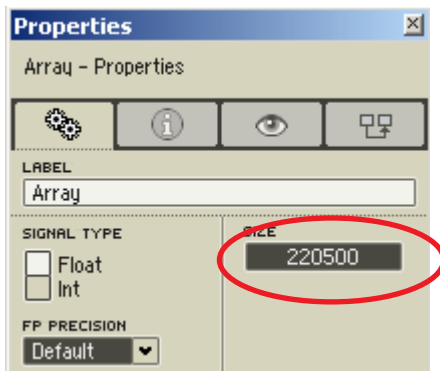


Lassen Sie uns diesen Betrag auf 5 Sekunden erhöhen. Wir multiplizieren also  $44100 \times 5 = 220500$  (nebenbei bemerkt: da jedes Sample 4 Byte verbraucht, ergibt sich daraus, dass  $220500 \text{ Samples} \times 4 \text{ Byte} = 882000 \text{ Byte}$  und damit fast 1 MByte belegen). Doppelklicken Sie auf das Macro *Delay 4p*:



Das Modul links ist das Delay-Buffer-Modul. Doppelklicken Sie es (oder führen Sie einen Rechts-Klick aus und wählen Sie aus dem Menü den Eintrag *Show Properties*), um seine Eigenschaften zu bearbeiten. Klicken Sie auf die

Schaltfläche  sodass Sie in der zugehörigen Ansicht die Eigenschaft *Size* sehen. Setzen Sie diesen Wert auf 220500 Samples:

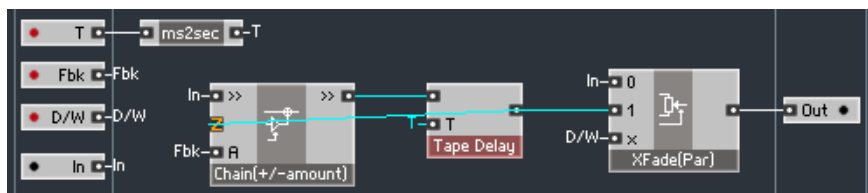


---

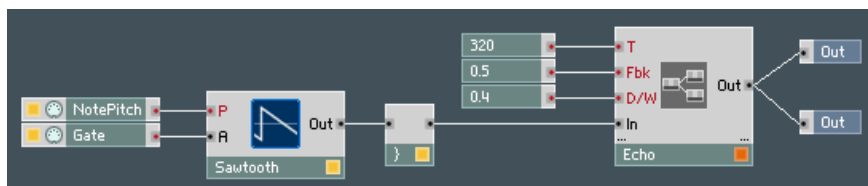
Wie Sie gerade gesehen haben, braucht ein Delay-Puffer von 5 Sekunden schon fast 1 MByte Arbeitsspeicher. Seien Sie also vorsichtig, wenn Sie die Delay-Puffergröße ändern – besonders dann, wenn Sie Delays in polyphonen Bereichen der Struktur verwenden, wo die Größe der Puffer mit der Anzahl der Stimmen multipliziert wird.

---

Jetzt können wir uns aus dem Macro *Delay 4p* hinaus bewegen, das Macro *Tape Delay* verlassen, das wir gerade erzeugt haben (doppelklicken Sie in den Hintergrund), und uns um die äußeren Verbindungen kümmern:



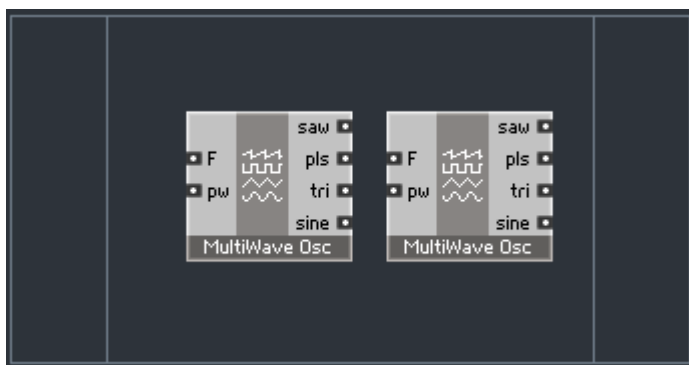
Wenn Sie das noch nicht getan haben, sollten Sie das Echo-Modul, das wir gebaut haben, jetzt ausprobieren. Hier ist eine Test-Struktur von REAKTORs Primary Level, so einfach wie möglich (beachten Sie, dass das Echo-Modul auf *mono* eingestellt ist):



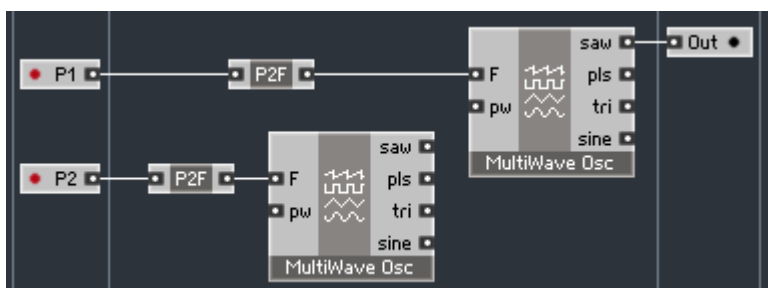
Sie können diese Struktur auf viele Arten erweitern, zum Beispiel, indem Sie Regler für die Kontrolle der Echo-Parameter vorsehen oder indem Sie einen echten Synthesizer als Signal-Quelle vorsehen.

## Wie Sie Audio als Kontroll-Signal verwenden

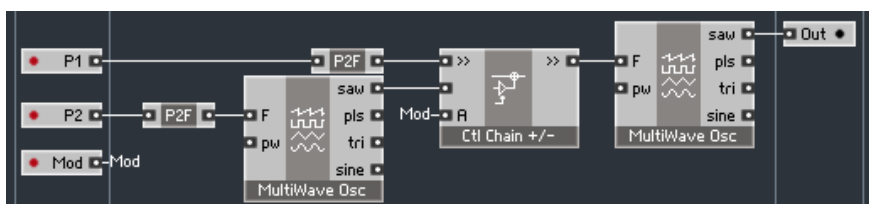
Wir haben bereits erwähnt, dass es möglich ist, Audio-Signale als Kontroll-Signale zu nutzen. Dafür sollten wir uns jetzt ein Beispiel ansehen. Dafür werden wir eine Core Cell mit einem Paar von Oszillatoren, von denen einer den anderen moduliert, erzeugen. Lassen Sie uns dafür zwei Module des Typs *Multiwave Osc* verwenden:



Wir brauchen Pitch-Kontrolle für beide Oszillatoren und einen Audio-Ausgang für den zweiten Oszillator, den wir ja hören wollen. Lassen Sie uns also die erforderlichen Eingänge und Ausgänge erzeugen:



Jetzt nehmen wir das Ausgangssignal des linken Oszillators und verwenden es, um die Frequenz des rechten Oszillators zu modulieren:

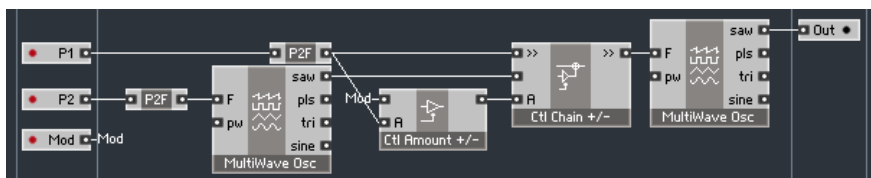


Der Eingang *Mod* kontrolliert den Grad der Modulation.

Beachten Sie, dass wir das Modulations-Signal nach dem Konverter-Modul *P2F* mischen, sodass die Modulation entlang einer Frequenz-Skala stattfindet; es ist aber auch möglich, in einer Tonhöhen-Skala (Pitch) modulieren zu lassen..)

Tatsächlich ist es aber viel günstiger, den Modulations-Grad im Verhältnis zur Grund-Frequenz der Oszillation zu skalieren:





Wenn Sie sich nun die oben abgebildete Struktur hinsichtlich der vorhandenen Kontroll- und Audio-Signale ansehen, werden Sie feststellen, dass alle Signale in dieser Struktur außer den Ausgangssignalen der Oszillatoren Kontroll-Signale sind. Die von den beiden von den Oszillatoren ausgegebenen Signale sind offensichtlich vom Typ Audio. Wie auch immer, wir “missbrauchen” das Ausgangssignal des linken Oszillators als Kontroll-Signal, und zwar an dem Punkt, an dem wir es in das Mixer-Modul *Ctl Chain* einspeisen

## Event-Signale

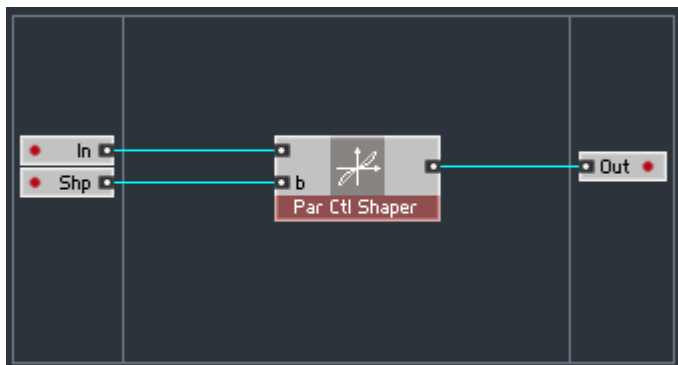
Wie wir schon festgestellt haben, hat der Begriff “Event-Signal” mehrere Bedeutungen. Sie sollten das Konzept der Event-Signale auf REAKTORs Primary Level ja schon kennen. Für ein Primary-Level-Event-Signal gibt es mehrere grundlegend verschiedene Verwendungsmöglichkeiten.

Eine Möglichkeit ist, es einfach als Kontroll-Signal zu verwenden (z. B. LFO-Ausgangssignal oder das von einem Drehregler gesendete Signal), einfach weil es weniger CPU-Last erzeugt als ein Audio-Signal auf dem Primary Level. In diesem Fall hätten Sie wahrscheinlich auch ein Audio-Signal verwenden können und damit dieselbe Wirkung erzielt. Eine andere Möglichkeit, in der Audio nicht als Alternative infrage kommt, ergibt sich dagegen, wenn Sie nicht nur an dem *Wert* interessiert sind, den das Signal transportiert, sondern auch der *Zeitpunkt*, zu dem der neue Wert geliefert wird, von Bedeutung ist – mit anderen Worten, *wann* das *Event* gesendet wird. Ein Beispiel dafür ist ein Hüllkurven-Gate-Signal (Envelope Gate) auf dem Primary Level: Die Hüllkurve wird in dem Moment getriggert, in dem das Event am Gate-Eingang *ankommt*.

Wenn wir von den Signal-Typen Audio, Control, Event und Logic in REAKTOR Core sprechen, reden wir nicht wirklich über technisch unterschiedliche Typen von Signalen (technisch sind alle Signale in REAKTOR Core gleich), sondern über unterschiedliche Einsatzbereiche der Signal-Typen. Wie Sie nun wissen, können Sie ein Primary-Level-Event-Signal als Control-, Event- oder Logic-Signal verwenden (während ein Primary-Level-Audio-Signal als Audio- oder Control-Signal verwendet werden kann).

Wir haben bereits gelernt, Primary-Level-Event-Signale in REAKTOR-Core-Strukturen einzuspeisen und sie als Kontroll-Signale zu verwenden. Die Event-Eingänge der *Audio-Core-Cell* mit dem Filter, die wir in einem früheren Kapitel

gebaut haben, sind ein gutes Beispiel dafür. Es gibt aber auch Fälle, in denen Sie eine Event-Core-Cell verwenden würden, um einige Primary-Level-Event-Signale zu verarbeiten. Hier ist ein Beispiel für eine Event-Core-Cell, in die ein Control-Shaper-Macro eingepackt ist:



Dieser Control-Shaper empfängt ein Kontroll-Signal mit Event-Rate vom Primary Level (z. B. ein MIDI-Velocity-Signal oder ein LFO-Signal), verbiegt es gemäß dem Parameter “Shp” und stellt das Ergebnis am Ausgang bereit.

---

Eine wichtige Beschränkung der Event-Core-Cells haben wir früher schon kurz angesprochen: Innerhalb von Event-Core-Cells sind alle Clock-Quellen unbrauchbar. Das heißt, dass sowohl Oszillatoren und Filter als auch Hüllkurven und LFOs innerhalb dieser Core Cells nicht arbeiten. Diese Module sind wirklich darauf beschränkt, Events von REAKTORs Primary Level zu empfangen, sie zu verarbeiten und sie nach draußen zu schicken, genau wie in dem oben beschriebenen Beispiel.

---

Alternativ können Signale, die außerhalb aus den Primary-Level-Signalen gewonnen werden, als echte Event-Signale innerhalb von REAKTOR-Core-Strukturen verwendet werden. Wir werfen im Folgenden einen Blick auf ein paar einfache Fälle, in denen Events innerhalb von REAKTOR Core genutzt werden.

Der erste Fall wäre, eine Hüllkurve in einer REAKTOR-Core-Struktur zu verwenden. Wie Sie sicher unter Berücksichtigung der oben genannten Beschränkung schon vermuten, muss dies eine Audio-Core-Cell sein. Erzeugen Sie also eine neue Core Cell vom Typ Audio und bestücken Sie sie mit dem Modul *Standard Macro > Envelope > ADSR*:

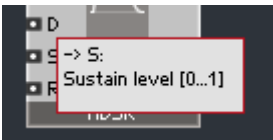


Der obere Eingang der Hüllkurve (“G”) ist ein Gate-Eingang, der ziemlich ähnlich arbeitet wie die Gates der Primary-Level-Hüllkurven. Das heißt, dieses Gate öffnet oder schließt die Hüllkurve als Reaktion auf ankommende *Events*. Kein Problem, wir erzeugen einen Event-Eingang für unsere Core Cell:

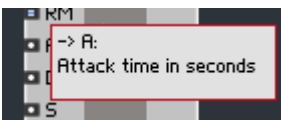


Dieser Eingang wird die vom Primary Level ankommenden Gate-Events in REAKTOR-Core-Events übersetzen.

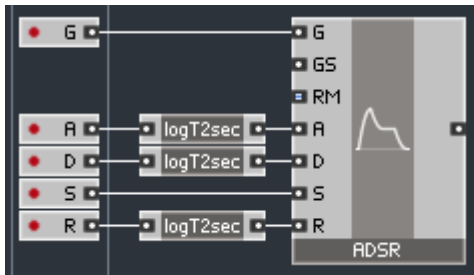
Lassen Sie uns nun die Eingänge A, D, S und R betrachten. Der Eingang “S” (Sustain-Pegel) arbeitet ähnlich wie auf dem Primary Level. Das heißt, er erwartet, dass das ankommende Signal im Bereich zwischen 0 und 1 liegt:



Die Eingänge A, D und R erwarten hingegen – im Unterschied zu den Primary-Level-Hüllkurven – als Eingangswert eine Zeitdauer in Sekunden

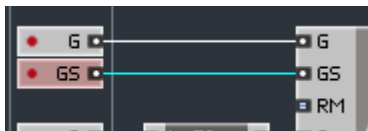


Dieses Problem können wir mit dem Konverter-Macro *Standard Macro > Convert > logT2sec* lösen, das die Hüllkurven-Dauer der Primary-Level-Hüllkurven in Sekunden konvertiert:

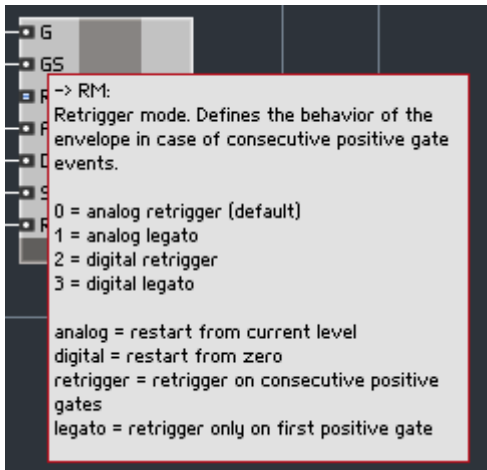


Ungeachtet der Tatsache, dass alle Eingänge in der obigen Struktur sich im Event-Modus befinden, produziert nur der erste dieser Eingänge ein Event-Signal, während die anderen Eingänge Event-Signale erzeugen.

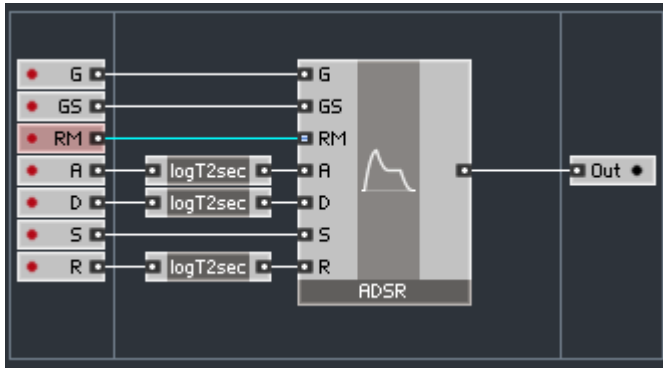
Unsere Hüllkurve hat immer noch zwei unbenutzte Ports. Der Port *GS* bestimmt über den Betrag der Empfindlichkeit (Sensitivity) des Gates. Ist der Wert 0, ignoriert die Hüllkurve den Gate-Pegel und läuft immer mit voller Amplitude. Wenn der Gate-Pegel den Wert 1 annimmt, zeigt er die maximale Wirkung, wie auf REAKTORs Primary Level. Wir können diesen Betrag von außen mit einem weiteren Eingang kontrollieren:



Der Port *RM* bestimmt über den Retrigger-Modus für die Hüllkurve:

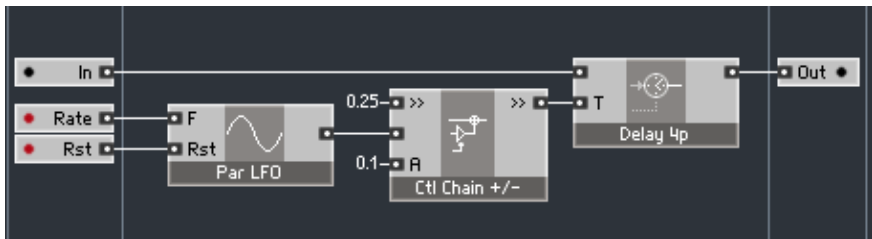


Das Aussehen des Ports “RM” unterscheidet sich von dem der anderen Ports, weil dieser Port Integer-Werte am Eingang erwartet. Wenn wir uns zum Beispiel einen typischen Eingang für eine Attack-Zeit ansehen, kann dieser eine Attack-Dauer von 1 s, von 1,5 s oder von 0,2 s erwarten. Im Gegensatz dazu erwartet der Retrigger-Modus-Port “RM” nicht wirklich Werte wie 1,2 oder 3,1. Dies wird durch das andersartige Aussehen verdeutlicht – aber es bedeutet nicht, dass wir keine normalen Signale an diesen Port anlegen dürfen. Wir können also einfach einen weiteren Event-Eingang verwenden:



Wenn die von außen kommenden Werte nicht ganzzahlig sind (und damit nicht dem Integer-Format entsprechen), werden sie auf den nächsten Integer-Wert gerundet. Das heißt zum Beispiel, ein Wert von 1,2 wird genauso behandelt wie der Wert 1.

Lassen Sie uns nun einen Blick auf ein anderes Beispiel für echte Event-Signal-Nutzung werfen:



Die obige Struktur stellt eine Art Pitch-Modulations-Effekt bereit. Der Effekt wird von einem Delay erzeugt, dessen Zeit um den Basis-Wert 250 ms um 100 ms nach oben und unten variiert. Der Rate dieser Variation wird über das Eingangs-Modul Rate kontrolliert, das die Rate des modulierenden LFOs

(in Hz) bestimmt – das ist ein reines Kontroll-Signal. Das Eingangs-Modul Rst liefert ein echtes Event-Signal, das Sie verwenden können, um den LFO neu zu starten. Der ankommende Wert spezifiziert die Restart-Phase, wobei 0 den LFO am Anfang des Durchlaufs zurücksetzen würde, 0.5 in der Mitte des Laufs und 1 am Ende. Sie können das ausprobieren, indem Sie einen Taster anschließen, der einen bestimmten Wert an dieses Eingang sendet.

## Logic-Signale

Nun, da Sie sich mit Kontroll- und Event-Signalen auskennen, ist es Zeit, eine andere Art kennen zu lernen, auf die Sie Signale in REAKTOR Core verwenden können: als *Logic-Signale*. Hier ist ein Beispiel für ein Modul, das Logic-Signale verarbeitet:



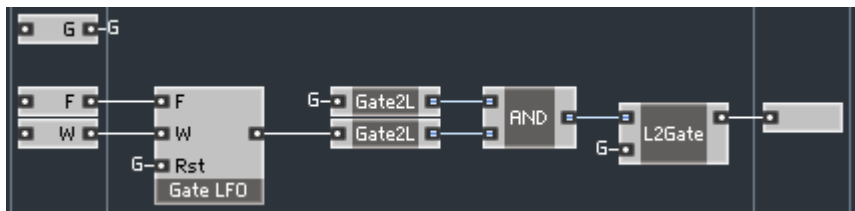
Wie Sie sehen könne, sind die Ports dieses Moduls vom Typ Integer, wie der RM-Eingang der Hüllkurve, die Sie schon kennen. Das hat damit zu tun, dass Logic-Signale generell nur Integer-Werte transportieren, ja, sie transportieren sogar nur die Werte 0 und 1.

Der Wert 1 steht dabei für den Zustand “wahr” des Logic-Signals, der Wert 0 für den Zustand “falsch”. Die Bedeutung von “wahr” und “falsch” müssen vom Anwender festgelegt werden. Es könnte zum Beispiel ein Logic-Signal geben, das Auskunft darüber gibt, ob ein bestimmtes Gate geöffnet ist:



Im obigen Fall prüft das Macro *Gate2L* das ankommende Gate-Signal und erzeugt die Ausgabe “wahr” (1), wenn das Gate offen ist, und “falsch” (0), wenn es geschlossen ist.

Sie können dann die Logic-Signale verwenden, um “logische Operationen” auszuführen. Zum Beispiel könnten Sie einen Gate-Prozessor bauen, der ein regelmäßiges “getaktetes Gate” (Clocked Gate) über ein MIDI-Gate anwendet:



Die Module *Gate2L*, *AND* und *L2Gate* sind Logic-Module, die Sie im Menü *Standard Macro > Logic* finden. Das Modul *Gate LFO* ist ein Macro, das wir für diesen Prozessor gebaut haben. Es erzeugt ein Gate-Signal, das sich in einem regelmäßigen Intervall öffnet und schließt.

Das Eingangs-Gate und der Ausgang des LFOs sind mit den Konverter-Modulen *Gate2L* verbunden, die das Gate-Signal in Logic-Signale konvertieren, indem sie offenen Gates den Wahrheitswert "wahr" und geschlossenen Gates den Wahrheitswert "falsch" zuweisen. Das Modul *AND* gibt nur dann ein Signal mit dem Wert "wahr" aus, wenn sich beide Gates zur selben Zeit im offenen Zustand befinden. Mit anderen Worten ist der Ausgangs-Wert des Moduls *AND* dann – und nur dann – "wahr", wenn der Anwender eine Taste drückt und zur selben Zeit der LFO ein offenes Gate ausgibt. Das bedeutet, dass, solange der Anwender eine Taste drückt, alternierend die Werte "wahr" und "falsch" am Ausgang des *AND*-Moduls anliegen, wobei die LFO-Rate die Geschwindigkeit der Wechsel zwischen den beiden Werten bestimmt. Die Ausgabe des Moduls *AND* wird dann zurück in das Gate-Signal konvertiert; die Amplitude des Gate-Signals wird am Gate-Eingang gewonnen, sodass der Gate-Pegel unverändert beibehalten wird.

Hier sehen Sie die Struktur unseres Macros *Gate LFO*:



Der Eingang "F" definiert die Rate der Gate-Wiederholungen, der Eingang "W" die Dauer des Zustands "Gate offen" (bei 0 sind sie 50 % der Gate-Periode geöffnet, bei -1 sind sie 0 % der der Periode offen und bei 1 sind sie 100 % offen). Der Eingang "Rst" startet den LFO auf ankommende Events hin neu (wodurch der LFO jedes Mal neu startet, wenn ein Gate-Event am Haupt-Gate-Eingang ankommt).

Das Modul, das an den Eingang "Rst" des Moduls *Rect LFO* angeschlossen ist, heißt *Value*; Sie finden es unter *Standard Macro > Event Processing*. Es stellt sicher, dass der LFO in der 0-Phase neu gestartet wird, indem es alle Werte ankommender Events durch den Wert des unteren Eingangs ersetzt, der 0 ist. Das Ausgangssignal des LFOs wird in ein Gate-Signal konvertiert, und zwar mit dem Modul *Ctl2Gate*, das Sie ebenfalls unter *Standard Macro > Event Processing* finden.

---

Wie Sie sich bestimmt erinnern, arbeiten LFOs nicht im Inneren von Event-Core-Cells; wenn Sie diese Struktur also ausprobieren wollen, müssen Sie eine Audio-Core-Cell verwenden.

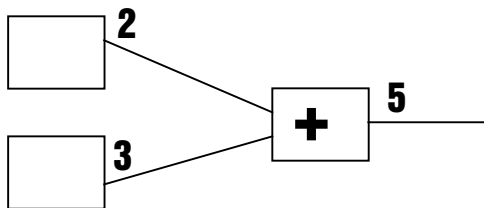
---

# Grundlagen von REAKTOR Core: Das Core-Signal-Modell

## Werte

Die meisten Ausgänge der REAKTOR-Core-Module erzeugen Werte (Values). Einen Wert “erzeugen” bedeutet, dass dem Ausgang zu jedem beliebigen Zeitpunkt ein Wert zugeordnet ist. Dieser Wert ist für die Module verfügbar, deren Eingänge mit diesem Ausgang verbunden sind.

Im folgenden Beispiel erhält ein Addierer-Modul (Adder) die Werte 2 und 3 von den anderen beiden Modulen, deren Ausgänge mit seinen Eingängen verbunden sind, und erzeugt am Ausgang den Wert 5.



---

Wenn Sie eine Parallele zur Hardware-Welt ziehen wollen, können Sie sich die Werte als Signal-Pegel (Spannungen) vorstellen, besonders, wenn Sie mit relativ großformatigen Modulen wie Oszillatoren, Filtern, Hüllkurven und so weiter hantieren. Werte sind allerdings nicht darauf beschränkt, die Aufgaben von Steuer-Spannungen zu übernehmen – sie können bei der Implementierung beliebiger Verarbeitungs-Algorithmen eingesetzt werden, nicht nur für die Modellierung von Spannungen.

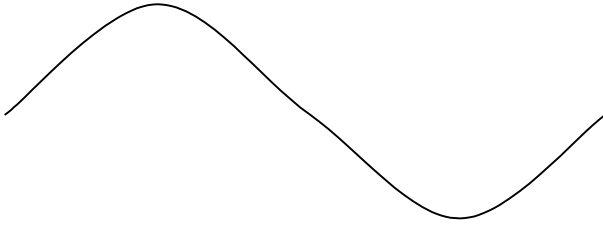
---

## Events

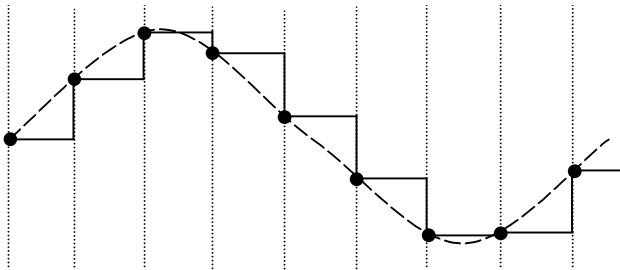
Die Zeit ist in der digitalen Welt nicht fortlaufend, sondern diskret. Das wahrscheinlich bekannteste Beispiel dafür ist, dass eine digital gespeicherte Aufnahme nicht die vollständige Information eines Audio-Signals enthält, das sich über die Zeit verändert, sondern in regelmäßigen Abständen zu diskreten Zeitpunkten aus dem Signal entnommene “Proben” speichert. Die Anzahl dieser Zeitpunkte pro Sekunde trägt den berühmten Namen *Sampling-Rate*.



Hier sehen Sie eine Darstellung eines fortlaufenden Signals:

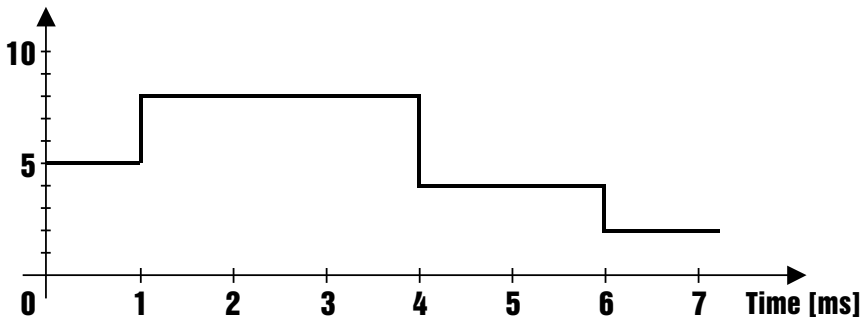


Und hier ist seine digitale Repräsentation:

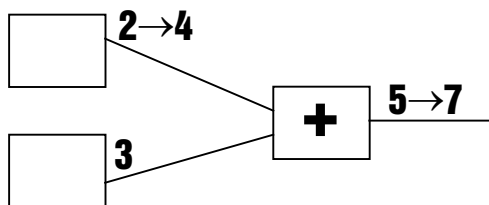


Das bedeutet für uns, dass die Ausgänge unserer Module in der digitalen Welt ihre Werte nicht fortlaufend verändern können. Auf der anderen Seite müssen wir uns auch nicht darauf beschränken, unseren Ausgängen Werteänderungen “in regelmäßigen Abständen zu diskreten Zeitpunkten” abzurufen, das heißt, wir müssen nicht in der gesamten Struktur mit derselben Sampling-Rate arbeiten. Weiterhin brauchen wir in manchen Bereichen unserer Strukturen überhaupt keine Sampling-Rate zu verwenden, was bedeutet, dass unsere Wertänderungen nicht “in regelmäßigen Abständen zu diskreten Zeitpunkten” stattfinden müssen.

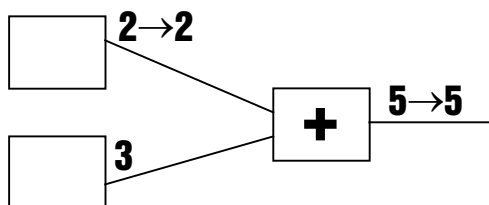
Nehmen wir zum Beispiel an, dass der Ausgang unseres Addierers zum Zeitpunkt Null den Wert 5 hat. Die erste Änderung könnte nach 1 ms stattfinden, die zweite nach 4 ms und die dritte nach 6 ms:



Auf dem obigen Bild können Sie die Veränderungen des Ausgangssignals unseres Addierers im Zeitverlauf zwischen 0 und 7 ms sehen. In dem Moment, in dem der Ausgang seinen Wert ändert, erzeugt er ein Event. *Event* bedeutet, dass der Ausgang von einer Veränderung seines Zustands “berichtet”, namentlich darüber, dass er einen neuen Werts angenommen hat. Im folgenden Beispiel hat das obere linke Modul seinen Ausgangswert von 2 auf 4 verändert, was ein Event erzeugt. Als Reaktion darauf wird auch das Addierer-Modul seinen Ausgangswert verändern und an seinem Ausgang ebenfalls ein Event erzeugen.



Alternativ dazu hätte das obere linke Modul auch ein neues Event mit demselben Wert, den das alte Event hatte, erzeugen können. Der Addierer hätte trotzdem als Reaktion darauf ein neues Event erzeugt, anscheinend ohne seinen Ausgangswert zu verändern:




---

Der neue Wert, der am Ausgang anliegt, muss sich nicht zwangsläufig von dem alten Wert unterscheiden. In jedem Fall ist aber Erzeugung eines Events die einzige Möglichkeit für einen Ausgang, seinen Wert zu ändern.

---

Wie Sie an den bisherigen Beispielen gesehen haben, wird ein Event, der an einem Ausgang eines Moduls ankommt, von den “stromabwärts” angeschlossenen Modulen wahrgenommen, die dann als Reaktion darauf weitere Events erzeugen (erinnern Sie sich daran, dass der Addierer als Reaktion auf ein ankommendes Event ein Event an seinem Ausgang erzeugt). Diese neuen Events werden dann ihrerseits wieder von den mit den Ausgängen verbundenen Modulen bemerkt und pflanzen sich immer weiter fort, bis die Vermehrung aus einem der Gründe, die wir später noch behandeln werden, zum Stillstand kommt.

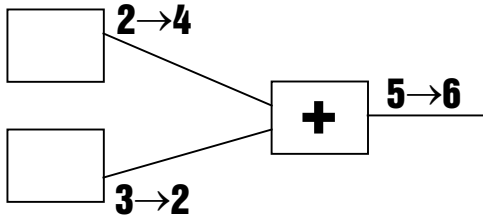
---

Die Events in REAKTOR Core sind nicht identisch mit den Events auf REAKTORs Primary Level – ihr Verhalten folgt anderen Regeln, die wir im Folgenden erklären.

---

## Gleichzeitige Events

Stellen Sie sich folgende Situation vor: Beide Module auf der linken Seite des vorigen Beispiels erzeugen *gleichzeitig* einen Event.



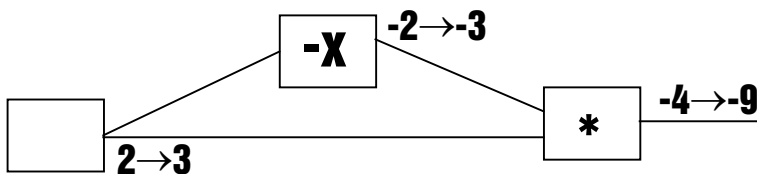
Dies ist eine der Schlüsselfunktionen des Event-Modells in REAKTOR Core – die Events können gleichzeitig an unterschiedlichen Stellen auftreten. In dieser Situation kommen die den beiden links angeordneten Modulen entstammenden Signale auch gleichzeitig an den Eingängen des Addierers an. Das führt dazu, dass der Addierer als Reaktion darauf genau *ein* Ausgangs-Event erzeugt.

---

Das ist nicht dasselbe wie auf REAKTORs Primary Level, wo Events nicht gleichzeitig stattfinden können und das Addierer-Modul *Add* (im Event-Modus) in einer solchen Situation zwei Ausgangs-Events erzeugt.

---

Natürlich werden die Events in Wirklichkeit nicht vom oberen und unteren Modul gleichzeitig erzeugt, weil beide Module von derselben CPU berechnet werden und die CPU zu einer beliebigen Zeit immer nur ein Modul verarbeiten kann. Für uns wichtig ist aber die Tatsache, dass diese Events logisch gleichzeitig stattfinden; das heißt, dass die Modul, die sie empfangen, sie als gleichzeitig behandeln. Jetzt werden wir uns ein anderes Beispiel für gleichzeitige Event-Fortpflanzung ansehen.



In dem obigen Beispiel sendet das Modul ganz links ein Event, das seinen Ausgangs-Wert von 2 auf 3 erhöht. Das Event wird gleichzeitig sowohl an den Inverter (-x) als auch an den Multiplizierer (\*) gesendet. Als Reaktion auf das ankommende Event wird der Inverter den neuen Ausgangs-Wert -3 erzeugen. Beachten Sie hier, dass *beide Events logisch gleichzeitig* sind, obwohl das Ausgangs-Event des Inverters ja als Reaktion auf das vom Modul ganz links gesendete Event und daher später als das ankommende Event erzeugt wurde. Dennoch kommen diese Events gleichzeitig im Multiplizierer an, und dieser erzeugt nur einen einzigen Ausgangs-Wert von -9.

---

Auf dem Primary Level hätten Sie es hier wieder mit zwei Events am Ausgang des Multiplizierer-Moduls *Event Mult* zu tun. Sie könnten außerdem nicht definieren, ob das Event vom Ausgang des Moduls ganz links zuerst an den Inverter oder an den Multiplizierer geschickt würde (was für unsere Beispiel-Struktur allerdings nicht wichtig ist).

---

Im Allgemeinen können Sie die folgende Regel verwenden, um zu bestimmen, ob zwei bestimmte Events gleichzeitig sind oder nicht:

---

Alle Events, die ihren Ursprung im selben Event haben (also als Reaktion auf denselben Event gesendet wurden), sind gleichzeitig. Alle als Reaktion auf eine beliebige Anzahl von an unterschiedlichen Ausgängen, aber zur selben Zeit gesendeten (und somit gleichzeitigen) Events gesendeten Folge-Events sind ebenfalls gleichzeitig.

---

Das letzte Beispiel zeigt den Sinn der Gleichzeitigkeit von Events. In diesem Fall eliminieren wir die überflüssige Verarbeitung des zweiten Events durch den Multiplizierer, die zusätzlich die CPU belastet hätte. Ohne solche Maßnahmen würde die Anzahl der Events in größeren Strukturen unkontrolliert anwachsen, es sei denn, der Entwickler der Struktur hätte besonders darauf geachtet, die Anzahl doppelter Events niedrig zu halten.

Neben der Einsparung von Rechenzeit führt dieses Konzept zu einem anderen Ansatz beim Entwurf von Strukturen und besonders bei der Entwicklung von Low-Level-DSP-Algorithmen. Sie werden diese Unterschiede besser verstehen und nachvollziehen können, sobald Sie mit der Konstruktion eigener Strukturen angefangen haben.

## Verarbeitungsreihenfolge

Wie Sie in den vorigen Beispielen schon gesehen haben, reagieren die “stromabwärts” im Signalfluss angeordneten Module, sobald ein Modul ein Event sendet. Daraus könnte man schließen, dass die Module definitiv nicht gleichzeitig verarbeitet werden, obwohl sie logisch gleichzeitige Events erzeugen. Man könnte weiter annehmen, dass es für jede mögliche Verbindung sinnvoll wäre, das weiter “stromaufwärts” gelegene Modul vor dem “stromabwärts” gelegenen zu verarbeiten. Wenn Sie nun diese Vermutungen angestellt haben – nun, dann haben Sie Recht.

Die allgemeine Regel für die Verarbeitungsreihenfolge von Modulen lautet:

---

Wenn zwei verbundene Module “gleichzeitige” Events verarbeiten, wird das im Signalfluss weiter “stromaufwärts” gelegene Modul zuerst verarbeitet. Wenn die Events nicht gleichzeitig sind, entspricht die Verarbeitungsreihenfolge der Module natürlich der Reihenfolge, in der die Events verarbeitet werden.

---

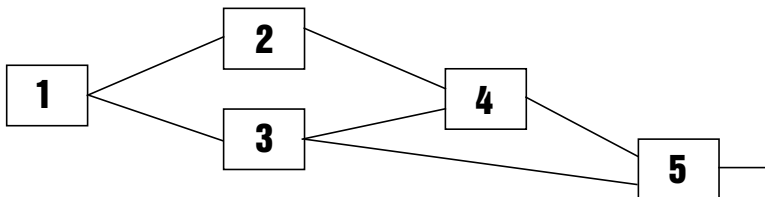
Aus der obigen Regel folgt, dass bei “Einbahnstraßen”-Verbindungen (immer aufwärts oder immer abwärts) von einem Modul zum anderen eine definierte Verarbeitungsreihenfolge existiert, nach der das aufwärts sendende Modul zuerst verarbeitet wird.

---

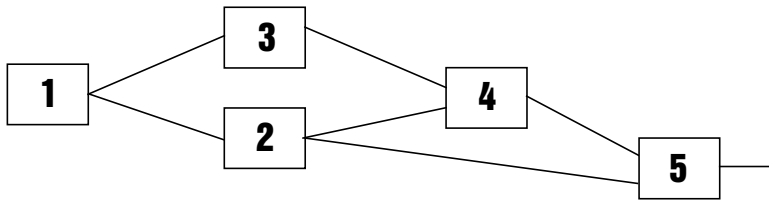
Wenn zwischen den Modulen kein Einbahnstraßen-Pfad besteht, ist die relative Verarbeitungsreihenfolge der Module untereinander undefiniert (für gleichzeitige Events). Das bedeutet, dass diese Reihenfolge beliebig ist und sich jederzeit ändern kann. Beim Entwerfen von Strukturen sollten Sie also darauf achten, dass solche Situationen nur für Module eintreten, deren Verarbeitungsreihenfolge unwichtig ist. Das gilt normalerweise automatisch, solange keine OBC-Verbindungen (siehe unten) beteiligt sind.

---

Hier sehen Sie ein Beispiel; die Ziffern zeigen die Verarbeitungsreihenfolge der Module an:



Eine alternative, ebenso gültige Verarbeitungsreihenfolge sehen Sie hier:

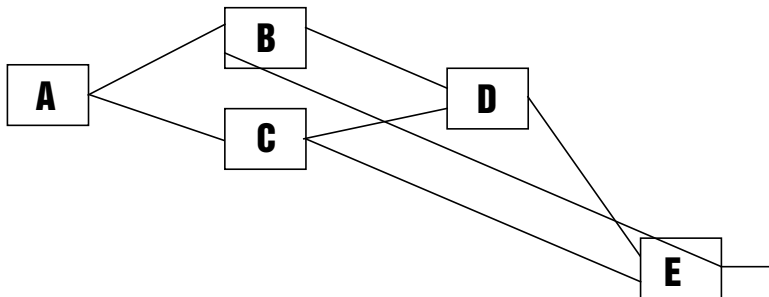


Es gibt keine Möglichkeit, vorherzusehen, welche dieser beiden Varianten die Software wählen wird. Glücklicherweise ist die relative Reihenfolge der Module wirklich unwichtig, solange Sie keine der OBC-Verbindungen verwenden, von denen unten die Rede ist.

---

Die oben genannten Regeln für die Verarbeitungsreihenfolge gelten nicht, wenn in Strukturen Feedback vorkommt, weil sich in diesem Fall für kein Paar von Modulen in der Feedback-Schleife bestimmen lässt, welches der beiden Module "stromaufwärts" des anderen liegt. Auf die Probleme bei der Behandlung solcher Feedback-Situationen auf die zugehörige Verarbeitungsreihenfolge gehen wir später noch genauer ein.

---



Für die oben abgebildete Struktur ist es nicht möglich, zu bestimmen, ob beispielsweise das Modul B stromaufwärts des Modul liegt oder umgekehrt. Anscheinend existiert eine aufwärts gerichtete Verbindung von D nach B, aber es besteht ebenso eine Aufwärts-Verbindung von B nach D (über E).

## Event-Core-Cells im Rückblick

Lassen Sie uns unter Berücksichtigung des eben beschriebenen Event-Konzepts von REAKTOR Core einen Blick auf die Event-Core-Cells werfen.

Wie Sie wissen, besitzen Event-Core-Cells Event-Eingänge und Event-Ausgänge. Diese Eingänge und Ausgänge sind die Schnittstellen zwischen dem Primary Level und der REAKTOR-Core-Ebene. Sie erfüllen diesen Zweck, indem sie Konvertierungen zwischen den Primary-Level-Events und den REAKTOR-Core-Events und umgekehrt durchführen. Die Regeln dieser Konvertierungen lauten wie folgt:

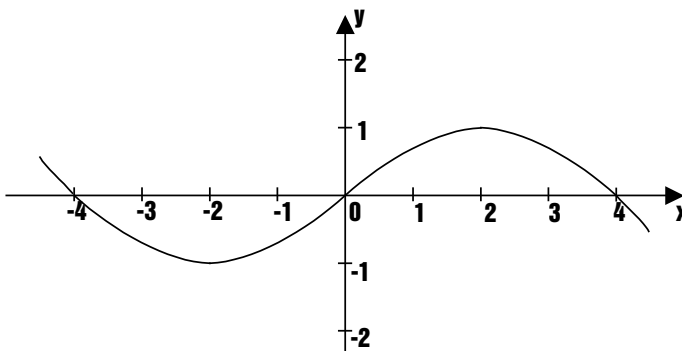
**Event Inputs** senden als Reaktion auf von außen kommende Primary-Level-Events REAKTOR-Core-Events an das Innere der Struktur. Weil die Primary-Level-Events von außen nicht gleichzeitig an den Eingängen ankommen können, finden die intern erzeugten Events ebenfalls nicht gleichzeitig statt.

**Event Outputs** senden als Reaktion auf interne REAKTOR-Core-Events Primary-Level-Events aus der Struktur heraus. Obwohl REAKTOR-Core-Events gleichzeitig an mehreren Ausgängen stattfinden können, können die korrespondierenden Primary-Level-Events nicht gleichzeitig gesendet werden. Deshalb werden im Fall von gleichzeitigen REAKTOR-Core-Events die zugehörigen Primary-Level-Events nacheinander gesendet, wobei *die oberen Ausgänge immer vor den darunter gelegenen Ausgängen senden..*

Als nächstes werden wir das in der Praxis nachvollziehen.

Lassen Sie uns mal versuchen, ein Event-Verarbeitungs-Modul zu bauen, das eine Signal-Formung gemäß der Formel  $y = 0.25 \cdot x \cdot (4 - |x|)$  durchführt.

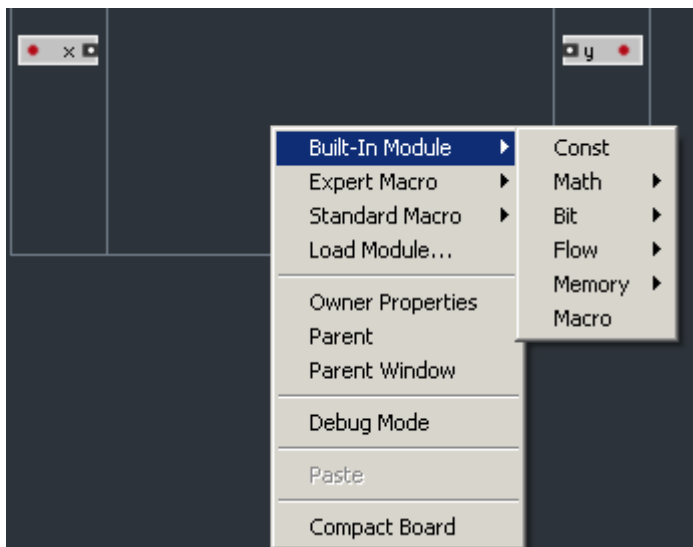
Der Graph dieser Funktion sieht folgendermaßen aus:



Wir beginnen mit dem Erzeugen einer neuen Event-Core-Cell mit einem Eingang und einem Ausgang, die wir gemäß unserer Formel “x” und “y” nennen:



Lassen Sie uns nun die Struktur entwerfen, die diese Formel berechnet. Wir müssen zunächst Module für “|x|” (absoluter Wert) und “-” (Subtraktion) sowie zweimal “\*” (Multiplikation) im normalen Bereich anlegen. Hierbei handelt es sich nicht um REAKTOR-Core-Macros, sondern um echte, eingebaute Module von REAKTOR Core. Um eingebaute Module in REAKTOR-Core-Strukturen einzusetzen, führen Sie einen Rechts-Klick in den Hintergrund des normalen Bereichs aus und wählen aus dem Menü das Untermenü *Built In Module*:

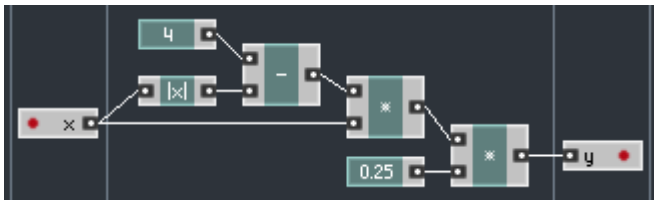




Sie finden alle oben genannten Module im Untermenü  
*Built In Module > Math:*

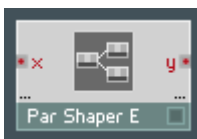


Wir brauchen zwei konstante Werte: 0.25 und 4. Wir könnten natürlich genau wie früher QuickConsts verwenden, aber es gibt auch die Möglichkeit, über Built In Module > Const ein echtes Konstanten-Modul einzusetzen (dessen Wert Sie wie bei QuickConst im Properties-Fenster festlegen können):

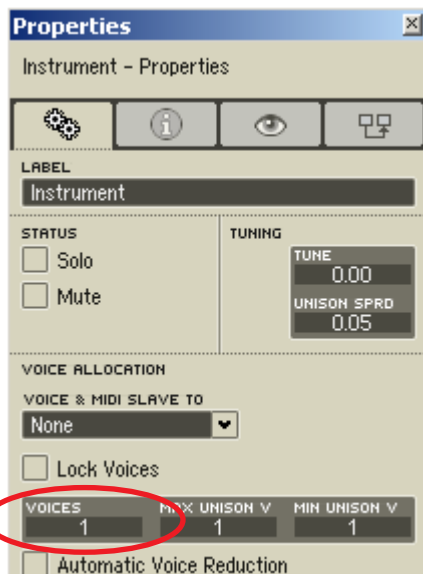


Natürlich bringt der Einsatz von Const-Modulen in unserem besonderen Fall keinen Vorteil gegenüber der Verwendung von QuickConsts, aber manchmal wollen Sie es vielleicht doch nutzen. Wenn zum Beispiel dieselbe Konstante an mehrere Eingänge übergeben werden soll, kann es vorteilhaft sein, das Modul Const zu verwenden, weil Sie dann nur eins davon brauchen und den Wert an einer zentralen Stelle kontrollieren können.

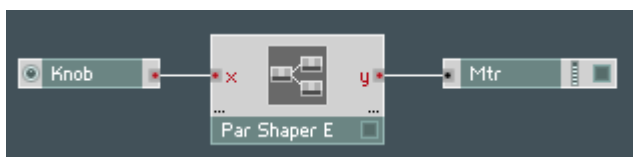
Wie auch immer, die oben abgebildete Struktur erledigt den Job, das Signal in der beschriebenen Weise zu formen, ziemlich gut. Wir können also unserem Modul einen Namen geben und auf das Primary Level zurückkehren:



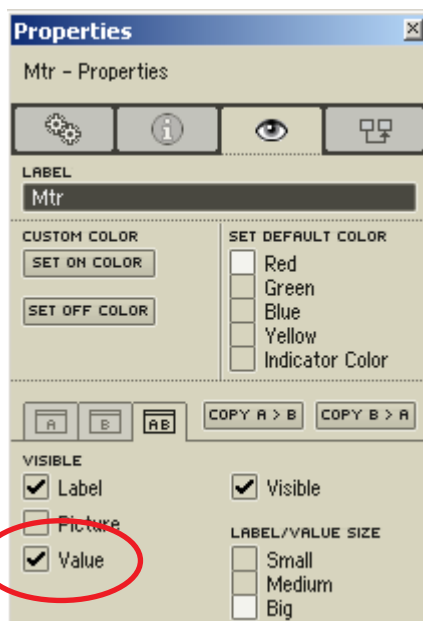
Jetzt lassen Sie 's uns ausprobieren. Setzen Sie den Wert für die Stimmenzahl des REAKTOR-Instruments auf 1, sodass wir leichter eine Werte-Anzeige (Meter-Modul) verwenden können.



Erzeugen Sie einen Drehregler und eine Werte-Anzeige und verbinden Sie diese mit dem Eingang und dem Ausgang Ihres Moduls:



Legen Sie die Eigenschaften für den Drehregler und die Werte-Anzeige fest. Vergessen Sie dabei nicht, die Werte-Anzeige so einzustellen, dass sie ihren Wert anzeigt,...



...und das Kästchen “Always Active” anzukreuzen:



Bewegen Sie nun den Drehregler und beobachten Sie, wie sich die Ausgangswerte verändern.



Shaper-Struktur, die wir gebaut haben, sollte sich perfekt für das Formen von Kontroll-Signalen eignen, aber sie hat noch einen kleinen Makel, der die Event-Verarbeitung betrifft. Wir werden später auf dieses Problem zurückkommen und es dann auch gleich beheben.

# Strukturen mit internem Zustand

## Clock-Signale

Die Art, wie eine Core Cell ein ankommendes Event behandelt, bleibt vollkommen dem Modul überlassen. Normalerweise verarbeitet ein Modul den ankommenden Wert auf irgendeine Weise, aber es könnte ihn auch einfach ignorieren. Der typische Fall einer solchen Behandlung sind *Clock Inputs* (Takt-Eingänge).

Ein Beispiel für ein Modul mit einem Clock-Eingang ist das Modul *Latch*. Dies ist kein eingebautes Modul, sondern ein Macro, aber es eignet sich trotzdem perfekt, um daran das Clock-Prinzip zu erklären.

*Latch* hat zwei Eingänge: einen für den Wert und einen für die Clock.



Der Werte-Eingang (oben) schreibt als Reaktion auf einen ankommenden Event den anliegenden Wert in den internen Speicher des Latch-Moduls; es wird nichts an den Ausgang gesendet. Der Clock-Eingang schickt als Reaktion auf ein ankommendes Event den letzten gespeicherten Wert an den Ausgang.

---

Der Clock-Eingang ignoriert normalerweise (außer, wenn es ausdrücklich anders festgelegt ist) den Wert des ankommenden Events und reagiert nur auf die Tatsache, dass ein Event ankommt.

---

Weil wir uns hier gerade mit Clock-Signalen befassen, nicht mit Latches, folgen die Beispiele zur Verwendung des Moduls *Latch* später an anderer Stelle.

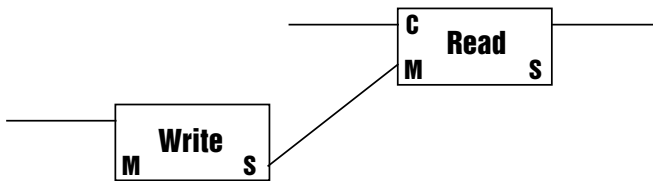
Weil es Module mit Clock-Eingängen gibt, sollte klar sein, dass einige Signale in der Struktur keine Nutz-Werte (oder in diesem Fall “nützlichen” Werte) transportieren. Einige Signale werden sogar eigens für den Einsatz als Clock-Quelle erzeugt. Wir können diese Signale *Clock-Signale* nennen.

Ein Beispiel für ein Clock-Signal ist die Sampling-Rate-Clock. Diese produziert für jedes neue Audio-Sample, das erzeugt werden soll, ein Event, sodass die “Uhr” bei einer Sampling-Rate von 44,1 kHz eben 44100 Mal pro Sekunde “ticken” würde. Der Wert dieses Signals hat keine Bedeutung, ist nicht für eine Auswertung gedacht und (in der derzeitigen Implementation) immer Null.

## Object Bus Connections

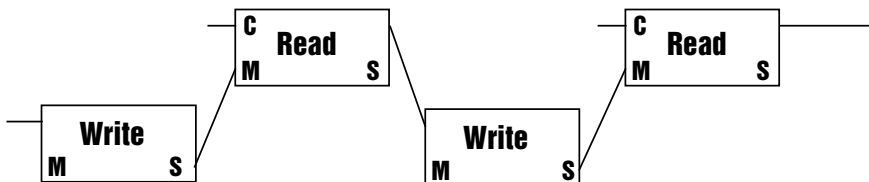
Die bereits oben erwähnten *Object Bus Connections* (OBC, Objekt-Bus-Verbindung) sind eine spezielle Art von Verbindungen zwischen den Modulen. Eine OBC zwischen zwei Modulen besagt, dass sich diese beiden Module irgendein internes Objekt teilen. Der häufigste Fall von Modulen, die eine OBC nutzen, sind die Speicher-Module *Read* und *Write*, die sich gemeinsamen Speicher teilen, wenn sie via OBC verbunden sind.

Die Funktion des Moduls *Write* besteht darin, einen an seinem Eingang ankommenden Wert in den über OBC gemeinsam genutzten Speicher zu schreiben. Die Aufgaben des Moduls *Read* ist es, auf ein am Eingang "C" eingehendes Clock-Signal hin den OBC-Speicher auszulesen; den ausgelesenen Wert stellt das Modul dann an seinem Ausgang bereit.



Die obige Struktur stellt die Funktionalität des Macros Latch dar (und es handelt sich in der Tat um die interne Struktur des Latch-Macros). Die Anschlüsse *M* und *S* der Module *Read* und *Write* sind vom Typ *Latch OBC*. Der Anschluss *M* ist der Master-Connection-Eingang, der Anschluss *S* der Slave-Connection-Eingang. Der Master-Eingang des Moduls *Read* ist mit dem Slave-Ausgang des Moduls *Write* verbunden (das andere Master-Slave-Paar ist nicht belegt). Deshalb teilen sich in dieser Struktur die Module *Write* und *Read* den gemeinsamen Speicher.

In der nächsten Struktur sehen Sie zwei Paare von Modulen der Typen *Write* und *Read*. Jedes Paar besitzt seinen eigenen Speicher. Beachten Sie, dass die Verbindung in der Mitte (vom Ausgang von *Read* zum Eingang von *Write*) nicht vom Typ *OBC* ist.



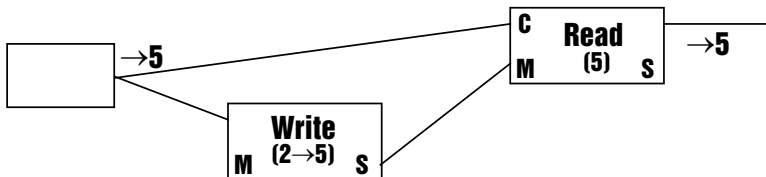
Sie könnten sich nun fragen, worin der Unterschied zwischen Master- und Slave-Betrieb besteht. Nun, hinsichtlich der Besitzanteile am gemeinsamen Objekt (in diesem Fall Speicher) gibt es keine Unterschiede. Allerdings besagt eine Regel, die Sie in einem früheren Kapitel gelernt haben, dass bei der Verarbeitung “gleichzeitiger Events” “stromaufwärts” gelegene Module vor den weiter “stromabwärts” im Signalfluss gelegenen Modulen verarbeitet werden. Deshalb werden in den beiden letztgenannten Beispielen die Write-Module vor den als Slave angeordneten Read-Modulen verarbeitet, was ganz offensichtlich nicht dasselbe ist wie die umgekehrte Reihenfolge.

---

Die relative Reihenfolge der Verarbeitung von via OBC verbundenen Modulen folgt denselben Regeln wie die Verarbeitung anderer Module: Die weiter “stromaufwärts” gelegenen Module werden zuerst verarbeitet.

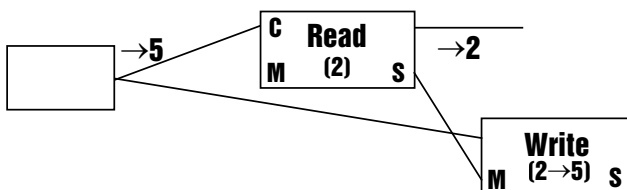
---

Lassen Sie uns nun zwei verschiedene Fälle betrachten. In beiden Fällen ist der Ausgangszustand des Speichers der Wert 2, und dasselbe Event von der Wertigkeit 5 wird sowohl an das *Write*- als auch an das *Read*-Modul geschickt. In dem einen Fall ist das Modul *Write* der Master, im anderen Fall ist *Read* der Master.



Oben sehen Sie die Struktur für den ersten Fall. Das Modul auf der linken Seite schickt einen Event mit dem Wert 5, der zuerst im *Write*-Modul eintrifft und dieses veranlasst, den neuen Wert 5 in den gemeinsamen Speicher des Modul-Paars *Write* und *Read* zu schreiben. Nun kommt der Event im *Read*-Modul an, wird als Clock-Event ausgewertet und triggert den Lese-Vorgang, im Verlauf dessen der gerade gespeicherte Wert 5 ausgelesen und zum Ausgang geschickt wird. Dies ist die Funktion, die das Macro *Latch* aus der Macro-Library von REAKTOR Core zur Verfügung stellt.

Nun betrachten Sie die zweite Struktur:



Hier haben wir die entgegengesetzte Situation. Das Clock-Event trifft zuerst im *Read*-Modul ein und sendet den Wert 2 an den Ausgang. Erst nach dem Lesen kommt das Event am Eingang des Moduls *Write* an und setzt den gespeicherten Wert auf 5. Diese Struktur implementiert die Funktion eines  $Z^{-1}$ -Blocks (ein Sample Verzögerung), die in der DSP-Theorie viel verwendet wird. In der Tat liegt in diesem Fall der Ausgangs-Wert immer einen Schritt hinter dem Eingangs-Wert zurück

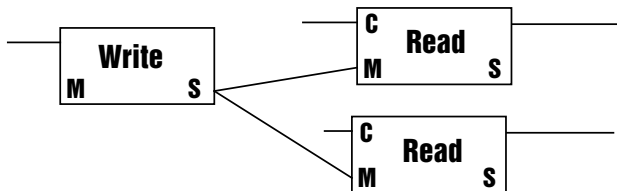
---

Wie erwähnt, implementiert die obige Struktur die  $Z^{-1}$ -Funktionalität. Bevor Sie aber selbst solche Strukturen aufbauen können, müssen Sie einige andere wichtige Dinge wissen – also lesen Sie weiter.

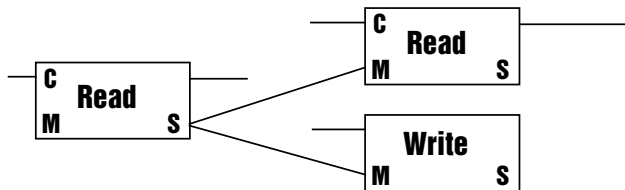
---

Wenn Sie mehr als zwei Module via OBC verbunden haben, bedeutet dies, dass alle diese Module dasselbe Objekt teilen. Hier ist es sehr wichtig zu wissen, ob eine bestimmte Reihenfolge von Schreib- und Lese-Vorgängen von Bedeutung ist, und wie diese Reihenfolge gegebenenfalls aussehen sollte.

In der folgenden Struktur beispielsweise ist die relative Reihenfolge der beiden Lese-Vorgänge nicht definiert, aber sie erfolgen beide nach dem Schreib-Vorgang, sodass alles in Ordnung sein sollte:

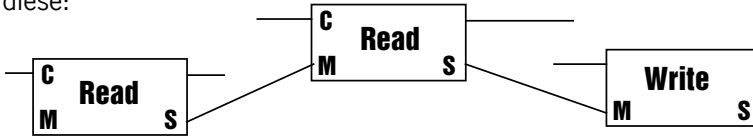


In der nächsten Struktur ist die relative Reihenfolge des Schreib-Vorgangs und des zweiten Lese-Vorgangs undefiniert, sodass dies eine möglicherweise gefährliche Struktur ist, die Sie auf jeden Fall vermeiden sollten:

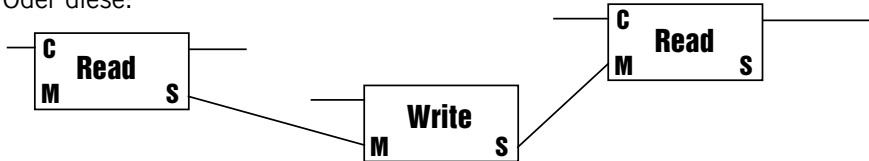




Eine bessere Möglichkeit, die obige Struktur umzusetzen, wäre wahrscheinlich diese:



Oder diese:



Auch wenn Sie annehmen, dass an manchen Stellen die relative Reihenfolge der Schreib- und Lese-Vorgänge irrelevant ist, schadet es nicht, eine bestimmte Reihenfolge festzulegen – im Gegenteil, es macht die Sache sicherer.

---

Die relative Reihenfolge der Schreib-Vorgänge ist wichtig. Die relative Reihenfolge der Lese-Vorgänge ist nicht von Bedeutung, solange ihre Anordnung im Verhältnis zu den Schreib-Vorgängen definiert bleibt.

---

---

Verbindungen des Typs OBC sind nicht mit normalen Signal-Verbindungen kompatibel. Weiterhin sind Verbindungen des Typs OBC, die mit unterschiedlichen Typen von Objekten korrespondieren (z. B. verschiedene Fließkomma-Genauigkeiten bei der Speicherverwaltung) nicht miteinander kompatibel. Anschlüsse inkompatiblen Typs lassen sich nicht miteinander verbinden, d. h., Sie können einen normalen Signal-Ausgang nicht mit einem OBC-Eingang verbinden.

---

## Initialisierung

Wenn wir anfangen, mit Objekten zu arbeiten, die einen internen Zustand annehmen (im Fall von *Read* und *Write* ist der gemeinsame Speicher dieser Objekte ihr interner Zustand), ist es wichtig, den *Anfangs-Zustand* (Initial State) einer gegebenen Struktur zu kennen. Wenn wir zum Beispiel einen Wert aus dem Speicher lesen wollen (wozu wir das *Read*-Modul verwenden), bevor etwas in den Speicher geschrieben wurde, müssen wir uns fragen, welcher Wert wohl in diesem Fall übergeben würde. Und wenn uns dieser Default-Wert nicht passt, wie können wir ihn ändern?

Diesen Fragen widmet sich der Initialisierungs-Mechanismus von REAKTOR Core. Die Initialisierung von REAKTOR-Core-Strukturen wird wie folgt durchgeführt:

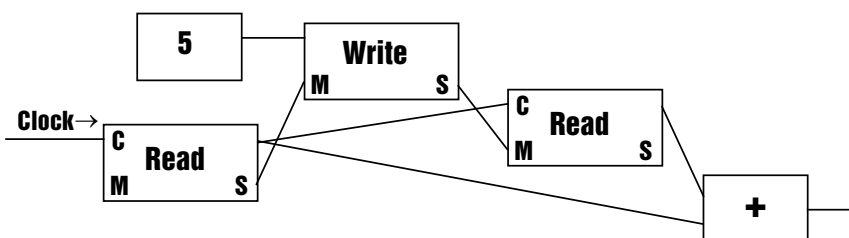
- Zuerst werden alle Zustands-Elemente auf Default-Werte initialisiert, normalerweise auf Nullen. Besonders alle gemeinsam genutzten Speicher und alle Ausgangs-Werte der Module werden auf Null gesetzt, außer in den im Handbuch beschriebenen Ausnahmefällen.
- Anschließend wird ein *Initialisierungs-Event* gesendet, und zwar gleichzeitig von allen *Initialisierungs-Quellen*. Die Initialisierungs-Quellen umfassen die meisten Module, die keinen Eingang besitzen: Const-Module (einschließlich *QuickConsts*), Core-Cell-Eingänge (typischerweise) und einige andere. Die Quellen senden ihre Anfangs-Werte während des Initialisierungs-Events, z. B. würden Konstanten-Module ihre Werte schicken und Core-Cell-Eingänge würden die Anfangswerte senden, die sie von der umgebenden Primary-Level-Struktur empfangen haben.

---

Wenn ein Modul eine Initialisierungs-Event-Quelle ist, finden Sie Informationen darüber im Modul-Referenz-Abschnitt zu diesem Modul. Wenn das Modul keine Initialisierungs-Event-Quelle ist, behandelt es das Initialisierungs-Event genau wie jedes andere ankommende Event. Die *allermeisten* Initialisierungs-Quellen sind Module, die keine Eingänge haben.

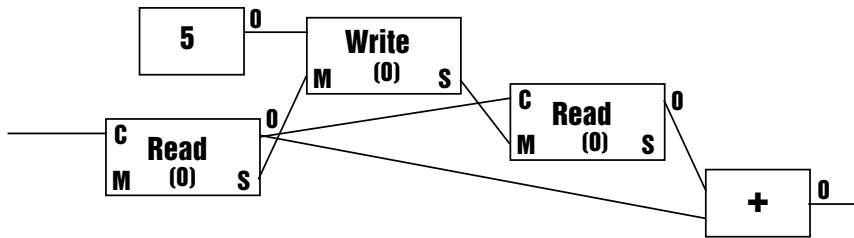
---

Lassen Sie uns am folgenden Beispiel betrachten, wie die Initialisierung vor sich geht:

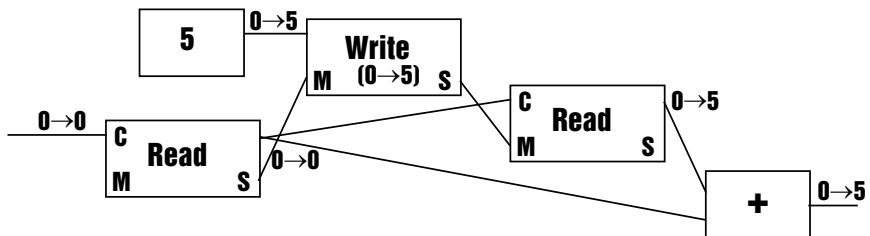


Dies ist ein Teil der Struktur; das *Read*-Modul links ist mit einer Clock-Quelle verbunden, die auch ein Initialisierungs-Event sendet (wie es Clock-Quellen normalerweise tun sollten).

Anfänglich sind alle Signal-Ausgänge und der interne Zustand der Lesen-Schreiben-Lesen-Kette (Read-Write-Read) auf Null gesetzt



Dann senden die Clock-Quelle und das Konstanten-Modul 5 gleichzeitig ein Initialisierungs-Event.

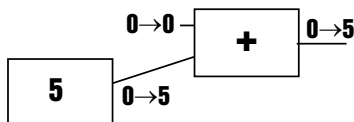


Das Modul *Read* (links) wird vor dem Modul *Write* verarbeitet, sodass das Clock-Event dort eintrifft, bevor der neue Wert in den Speicher geschrieben wurde; also ist die Ausgabe dieses Moduls Null. Dann wird der Wert vom Modul *Write* in den Speicher geschrieben. Nun wird das zweite *Read*-Modul angetriggert, was einen Wert von 5 an seinem Ausgang erzeugt. Schließlich wird das Addierer-Modul verarbeitet und ermittelt aus den anliegenden Werten eine Summe von 5.

---

Wie Sie sich erinnern, werden nicht angeschlossene Eingänge in REAKTOR Core behandelt, als hätten sie den Wert Null (sofern dies nicht von einem bestimmten Modul anders festgelegt wird). Genauer gesagt werden unbeschaltete Eingänge wie Null-Konstanten behandelt. Das bedeutet, dass diese Eingänge genauso Initialisierungs-Events empfangen wie Eingänge, an denen ein echtes Konstanten-Modul mit dem Wert Null angeschlossen ist.

---



Oben sehen Sie einen Addierer mit einem nicht verbundenen Eingang und einem Eingang, der mit einem Konstanten-Modul verbunden ist. Dieser Addierer empfängt zwei gleichzeitige Initialisierungs-Events, und zwar eins von der an unbeschalteten Eingängen anliegenden Default-Konstante Null und eins über eine echte Verbindung zu einem Konstanten-Modul.

---

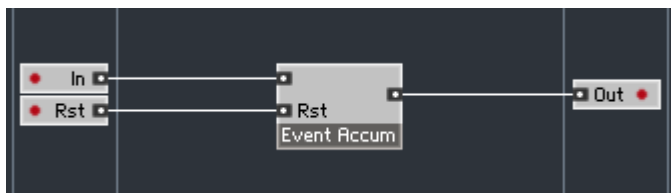
Nicht angeschlossene Eingänge, die keine Signal-Eingänge sind (und die man deshalb ohnehin nicht mit einer Null-Konstante verbinden kann), können eine besondere Bedeutung bekommen. Zum Beispiel kann ein nicht beschalteter Master-Eingang eines Write-Moduls bedeuten, dass die gemeinsam genutzte Speicher-Kette hier beginnt und sich in den Modulen, die an den Slave-Ausgang angeschlossen sind, fortsetzt.

---

## Wie Sie einen Event-Akkumulierer bauen

Ein Event-Akkumulierer, wie wir ihn nun bauen werden, braucht zwei Eingänge, und zwar einen für die Event-Werte, die akkumuliert werden sollen, und einen zum Rücksetzen des Akkumulierer-Moduls auf Null. Es gibt außerdem einen Ausgang, der die Gesamtsumme der akkumulierten Events ausgibt.

Wir werden dieses Modul in Form eines Core-Macros anlegen. Ein solches Macro wäre auch leicht innerhalb einer Event-Core-Cell zu verwenden:



Uns so sieht das Innere unseres Macros aus:

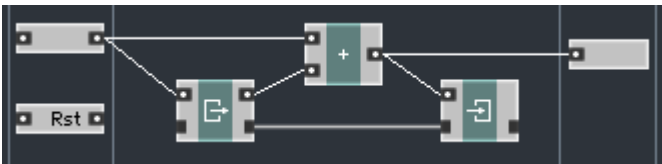


Offensichtlich braucht das Akkumulierer-Modul einen internen Zustand, in den es den aktuellen akkumulierten Wert schreiben kann. Wir werden also die Module *Read* und *Write* verwenden, um die Akkumulierer-Schleife zu bauen. Sie finden diese Module unter *Built In Module > Memory*:



Das Modul, das Sie links im Bild sehen (mit einem Pfeil in Auswärts-Richtung), ist das Modul *Read*, das Modul rechts (mit dem nach innen weisenden Pfeil) ist das Modul *Write*

Als Reaktion auf ein ankommendes Event soll die Akkumulierer-Schleife den alten Wert nehmen und den neuen Wert hinzufügen. Wir müssen also ein *Read*-Modul einsetzen, um den alten Zustand auszulesen, dem ermittelten Wert mit einem Addierer den neuen Wert hinzufügen und ein *Write*-Modul verwenden, um den Wert wieder zu speichern.



Beachten Sie, dass das Modul *Read* vom ankommenden Event getaktet wird und dass das via OBC angeschlossene *Write*-Modul natürlich stromabwärts angeordnet ist, weil wir ja erst schreiben wollen, nachdem wir gelesen haben. Im Normalfall sollte die oben abgebildete Struktur sofort arbeiten, und zwar sollte sie die ankommenden Werte sammeln (akkumulieren) und die Gesamtsumme am Ausgang ausgeben. Was noch fehlt, sind eine Reset-Funktion zum Zurücksetzen und eine Schaltung, die den korrekten Anfangs-Zustand sicher stellt.

Lassen Sie uns zunächst die Reset-Schaltung bauen. Weil wir uns in der REAKTOR-Core-Welt bewegen, können die Eingänge "In" und "Rst" gleichzeitig Events senden, also müssen wir das im Allgemeinen (wenn wir ein allgemein verwendbares REAKTOR-Core-Macro bauen wollen) berücksichtigen.

Lassen Sie uns also annehmen, dass die Eingänge "In" und „Rst“ gleichzeitig jeweils ein Event erzeugen. Was soll in diesem Fall geschehen? Ist es logisch betrachtet günstiger, den Reset durchzuführen, bevor das akkumulierte Event verarbeitet wird, oder danach? (Dieser Unterschied ist ganz ähnlich wie der Unterschied zwischen den Funktionen *Latch* und  $Z^1$ , die sich nur in ihrer relativen Verarbeitungsreihenfolge von Signal-Eingang und Clock-Eingang unterscheiden.)

Wir schlagen die Latch-Variante vor, weil dieses Modul sehr viel in REAKTOR-Core-Strukturen, verwendet wird und deshalb ein derartiges Verhalten eher der Intuition entspricht. In einem Latch kommt das Clock-Signal logisch später an als das Werte-Signal. In unserem Fall sollte das Reset-Signal logisch nach dem akkumulierten Signal eintreffen (und Zustand und Ausgang auf Null setzen). Dies bedeutet, dass wir irgendwie das Ausgangssignal des Akkumulierers mit einem Anfangs-Wert überschreiben müssen. Um das zu erreichen, brauchen wir ein neues Konzept, das wir nun diskutieren wollen.

## Event Merging

Sie haben schon verschiedene Methoden gesehen, mit denen sich zwei verschiedene Signale in REAKTOR Core kombinieren lassen, darunter arithmetische Operationen und einige andere. Was uns aber bislang fehlt, ist eine Möglichkeit, Signale sortiert zu mischen.

Dieses im Englischen *Merging* genannte sortierende Mischen ist kein Addieren. Merging bedeutet, dass wir von allen ankommenden Signalen den jeweils letzten Wert nehmen, anstatt sie zu summieren. Um die Signale zu mischen, die Sie mischen müssen, verwenden Sie das Modul *Merge*.

Lassen Sie uns mal schauen, wie das funktioniert. Stellen Sie sich vor, wir haben ein *Merge*-Modul mit zwei Eingängen. Der anfängliche Ausgangs-Wert (vor dem Initialisierungs-Event) ist natürlich Null, wie für die meisten Module:



Nun kommt ein Event mit dem Wert 4 am zweiten Eingang des Moduls an:



Das Event passiert das Modul und erscheint am Ausgang. Nun hat der Ausgang des *Merge*-Moduls den Wert 4.

Dann kommt ein weiteres Event mit dem Wert 5 am ersten Eingang an:



Das Event passiert ebenfalls das Modul und erscheint am Ausgang, wodurch sich dessen Wert in 5 ändert. Nun kommen zwei Events mit den Werten 2 und 8 gleichzeitig an beiden Eingängen an.



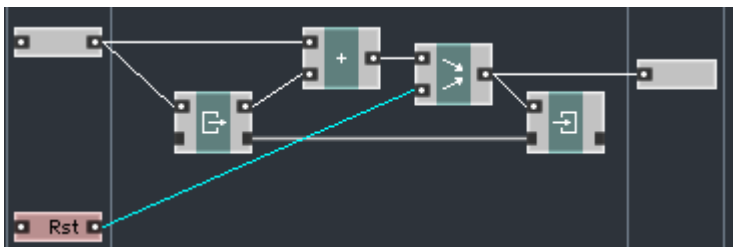
Und hier kommen wir zu einer besonderen Regel für unser Merge-Modul:

Events, die gleichzeitig an den Eingängen eines Merge-Moduls ankommen, werden in der Reihenfolge der Eingangs-Nummerierung verarbeitet. Trotzdem wird nur ein Ausgangs-Event erzeugt, weil ein Ausgang in REAKTOR Core nicht mehrere Events gleichzeitig erzeugen kann.

Im oben beschriebenen Fall bedeutet dies, dass das Event am zweiten Eingang nach dem Event am ersten Eingang verarbeitet wird, wodurch der Wert 2 mit dem Wert 8 überschrieben wird. Dieser Wert 8 liegt dann am Ausgang an.

## Event-Akkumulierer mit Reset und Initialisierung

Um also die gewünschte Reset-Funktion zu erreichen, müssen wir die Ausgaben des Addierers mit einem Anfangs-Wert überschreiben. Dafür können wir ein *Merge*-Modul verwenden (das Sie im Untermenü *Built In Module > Flow* finden). Die einfachste Möglichkeit wäre, den zweiten Eingang des *Merge*-Moduls mit dem Eingangs-Modul "Rst" zu verbinden:



Jetzt wird der Reset-Event direkt an das *Merge*-Modul übergeben und über-

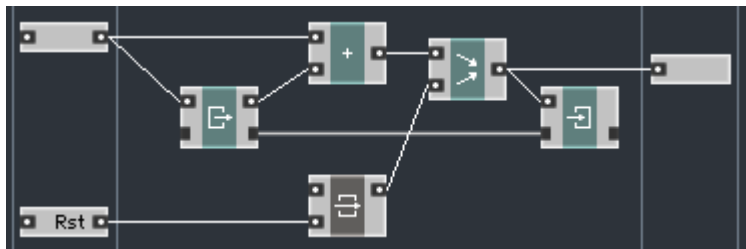
schreibt die Ausgabe des Addierers, falls das akkumulierte Event zur selben Zeit eintreffen sollte. Von da aus gelangt das Reset-Event an den Ausgang und in den internen Zustand des Akkumulierers.

In der oben abgebildeten Struktur wird der am Eingangs-Modul “Rst” auftretende Wert als neuer Wert des Akkumulierers verwendet. Vielleicht ist das keine schlechte Idee, aber dann ist es eigentlich keine “Reset”-Funktion, sondern eher eine “Set”-Funktion, wie sie im Standard-Event-Akkumulierer-Modul von REAKTOR verwirklicht wurde. Wenn wir eine echte Reset-Funktion entwickeln wollen, sollten wir ausschließlich Null-Werte in den Zustand schreiben, unabhängig davon, welcher Wert am Eingangs-Modul “Rst” anliegt. Wir müssen also immer dann einen Null-Wert an das Modul *Write* schicken, wenn am Eingang “Rst” ein Event auftritt.

Das Senden eines Events mit einem bestimmten Wert als Reaktion auf einen ankommenden Event ist eine ziemlich normale Sache in REAKTOR Core, und wir würden zu diesem Zweck das Macro *Latch* aus der REAKTOR-Core-Library verwenden. Sie finden es unter *Expert Macro > Memory > Latch*:



Wie bereits erwähnt, besitzt das Latch-Modul einen Werte-Eingang (oben) und einen Clock-Eingang (unten). Wir müssen das Eingangs-Modul “Rst” mit dem Clock-Eingang verbinden, um das Senden von Events an den Ausgang des Latch-Moduls zu triggern. Außerdem müssen wir ein Konstanten-Modul mit dem Wert Null an den Werte-Eingang des Latch-Moduls anschließen, weil wir die ganze Zeit Events mit dem Wert Null an diesen Eingang senden wollen. Oder wir erinnern uns daran, dass nicht angeschlossene Eingänge so behandelt werden, als wäre ihr Wert konstant Null (sofern nicht anders angegeben) und lassen den Werte-Eingang des Moduls Latch einfach offen:

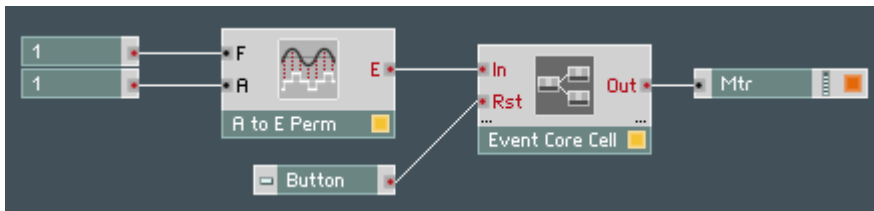


Jetzt sollte die Reset-Funktion arbeiten wie gewünscht.

Schließlich müssen wir noch für eine korrekte Initialisierung sorgen. Es ist auch







Die Stimmen-Anzahl des Instruments sollten Sie auf 1 setzen; das Anzeige-Modul Mtr sollten Sie so einstellen, dass es einen Wert anzeigt und immer aktiv ist, wie im vorigen Beispiel beschrieben. Der Schalter sollte sich im Trigger-Modus befinden.



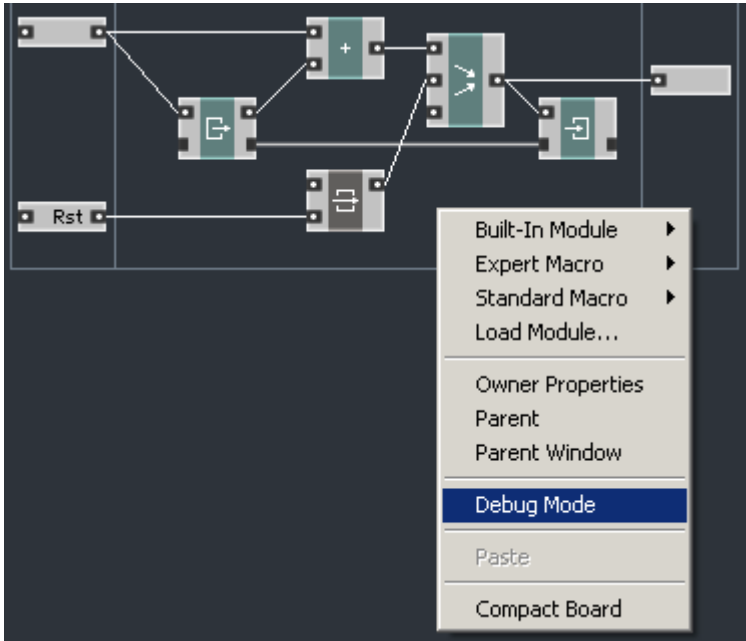
Schalten Sie nun in die Panel-Ansicht und schauen Sie zu, wie sich die Werte in Schritten von 1 pro Sekunde erhöhen und wie sie auf Knopfdruck zurückgesetzt werden.




Wir werden diese Gelegenheit nutzen, um Ihnen den ebug-Modus von REAKTOR Core vorzustellen. Wie Sie wahrscheinlich schon bemerkt haben, können Sie in REAKTOR Core nicht wie auf dem Primary Level den Wert am Ausgang eines Moduls durch Parken des Mauszeigers auf diesem Ausgang anzeigen lassen. Das ist ein unglücklicher Seiteneffekt der internen Optimierung von REAKTOR Core, der dazu führt, dass die Werte von REAKTOR-Core-Strukturen nicht von außen verfügbar sind

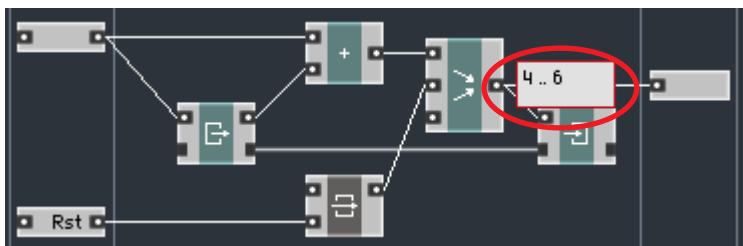
Weil wir Sie schon über diesen Umstand klagen hören, haben wir uns für einen Kompromiss entschieden: Sie können die Optimierung für eine bestimmte Core-Struktur abschalten und dann die Ausgangs-Werte sehen. Lassen Sie uns diesen Debug-Modus an der Struktur ausprobieren, die wir gerade gebaut haben.


Führen Sie einen Rechts-Klick in den Hintergrund aus und wählen Sie aus dem Menü den Eintrag *Debug Mode*:

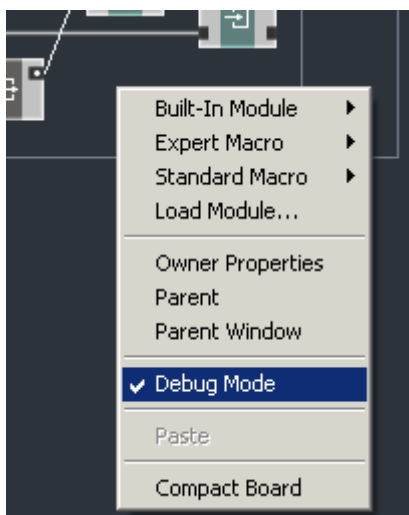


(Sie können dieselbe Funktion auch über die Schaltfläche  in der Werkzeugleiste erreichen).

Wenn Sie nun den Mauszeiger über einen bestimmten Ausgang führen, werden Sie die Werte (oder den Wertebereich) dieses Ausgangs sehen:

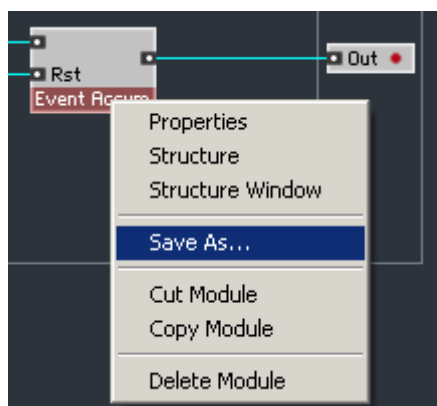


Sie können den Debug-Modus wieder abschalten, indem Sie denselben Eintrag aus dem Menü noch einmal wählen (oder denselben Schalter  ein zweites Mal drücken):

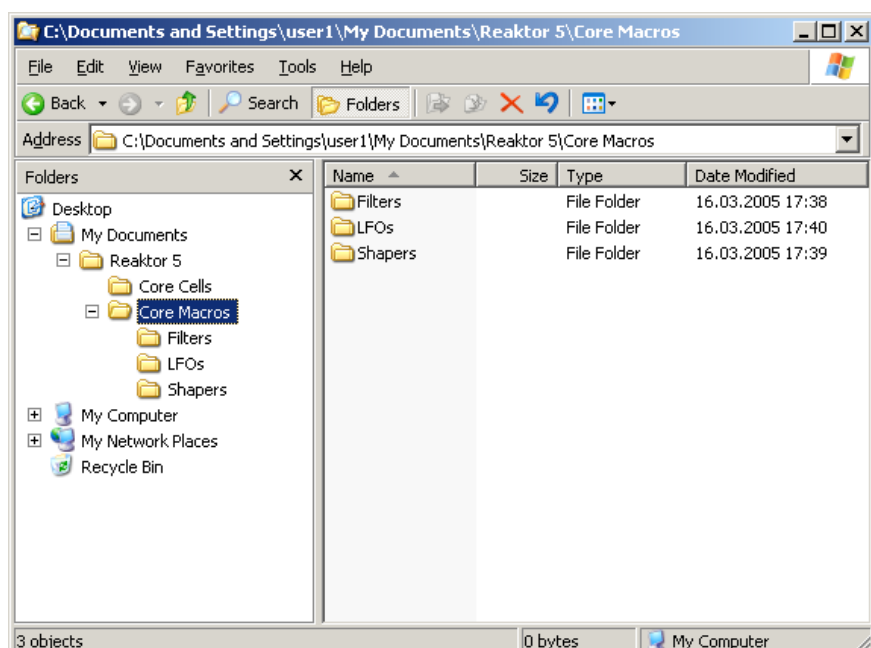


Alternativ dazu wird der Debug-Modus automatisch abgeschaltet, wenn Sie diese Struktur verlassen, sodass Sie ihn für andere Strukturen bei Bedarf wieder einschalten müssen.

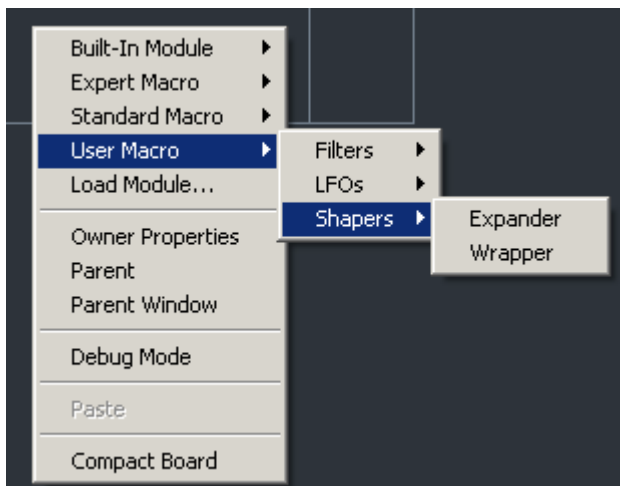
Nach dem “Debugging” unseres REAKTOR-Core-Macros können wir uns überlegen, es für zukünftige Einsätze als separate Datei zu speichern. Dazu führen Sie einen Rechts-Klick af das Macro aus und wählen aus dem Menü den Eintrag “Save As...”:



Wie bei Core Cells haben Sie die Möglichkeit, Ihre eigenen Macros in das Menü aufzunehmen. Dazu müssen Sie die Macros in das Unterverzeichnis “Core Macros” im User-Library-Verzeichnis von REAKTOR legen:



Wenn in diesem Ordner “Core Macros” oder in einem seiner Unter-Ordner irgendwelche Dateien gefunden werden, erscheint ein neues Untermenü im “Rechts-Klick-Menü”:

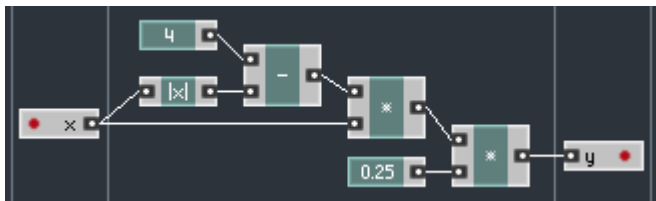


Wie für den Ordner “Core Cells” gelten auch für den Ordner “Core Macros” einige Beschränkungen:

- Leere Ordner werden im Menü nicht angezeigt.
- Legen Sie niemals Ihre eigenen Dateien in die System-Library von REAKTOR, sondern bewahren Sie sie in Ihrem User-Library-Ordner auf.

## Die Reparatur des Event-Shaper-Moduls

Jetzt können wir genauer darauf eingehen, was an der Struktur des Event-Shaper-Moduls, das wir in einem früheren Kapitel gebaut haben, falsch ist:



Das Problem ist das Initialisierungs-Event. Wenn Sie sich ansehen, wie die Initialisierung der obigen Struktur erfolgt, werden Sie folgendes bemerken:

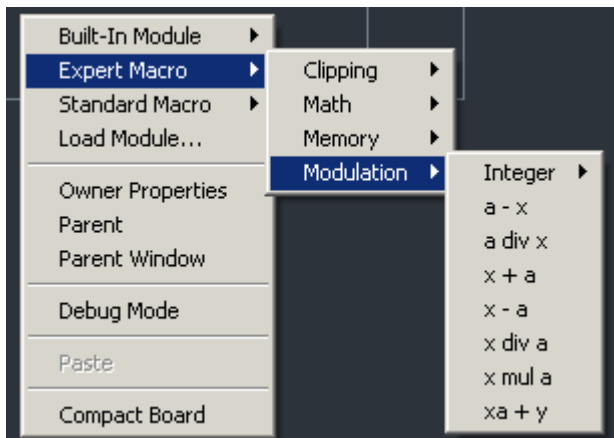
- Der Eingang “x” sendet seine Initialisierungs-Event ab oder sendet sie nicht, abhängig davon, ob er von der umgebenden Primary-Level-Struktur ein Initialisierungs-Event empfängt (dies ist die Initialisierungs-Event-Regel für die Core-Cell-Event-Eingänge)

- Die Konstanten 4 und 0.25 senden immer ein Initialisierungs-Event. Dadurch wird in dem Fall, dass am Eingang des Event-Shapers kein Initialisierungs-Event stattfindet (aus welchen Gründen auch immer), der Ausgang des Shapers immer noch das Event von der letzten Multiplikations-Stufe empfangen und dieses Event an die umgebende Primary-Level-Struktur weitergeben.

Obwohl das für die Verarbeitung von Kontroll-Signalen ganz in Ordnung sein mag (weil im Fall einer fehlenden Eingangs-Initialisierung der Eingang den Wert Null annimmt und das Event für die Initialisierung des Ausgangs immer noch gesendet wird), ist es nicht gerade das, was man intuitiv von einem Event-Verarbeitungs-Modul erwarten würde. Ein etwas besser nachvollziehbares Verhalten wäre, wenn das Modul ein Ausgangs-Event nur als Reaktion auf ein ankommendes Event schicken würde.

Unser Problem besteht also darin, dass die beiden Konstanten-Module Events zur falschen Zeit senden dürfen (das heißt, wenn kein Eingangs-Event anliegt). Als Lösung schlagen wir vor, die Subtraktions- und Multiplikations-Module, die Konstanten an ihren Eingängen empfangen, durch ihre Gegenstücke vom Typ *Modulation* zu ersetzen.

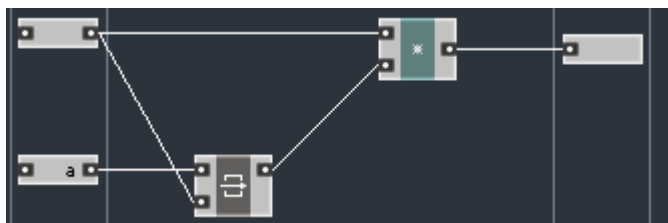
Die “Modulations-Macros” sind eine Gruppe von Modulen in der REAKTOR-Core-Library, die Sie unter *Expert Module > Modulation* finden:



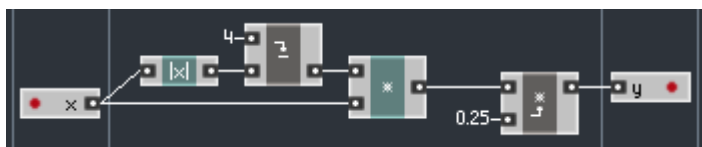
Der Name “Modulation” ist eigentlich nicht ganz korrekt, aber er spiegelt die Fähigkeit dieser Macros wider, ein Signal zu verwenden, um ein ande-

res zu modulieren (was Sie am leichtesten verstehen, wenn wir später in den Low-Level-Strukturen Kontroll-Signale verwenden, um Audio-Signale zu manipulieren). Die meisten dieser Macros kombinieren zwei Signale, von denen eines der “Träger” (oder Carrier) und das andere der “Modulator” ist. Anders als die eingebauten arithmetischen REAKTOR-Core-Module erzeugen die “Modulations-Macros” nur als Reaktion auf ein Event am “Carrier”-Eingang hin eine Ausgabe. Die Events am “Modulator”-Eingang triggern den Rechenvorgang nicht an.

Die interne Implementation der Modulations-Macros ist sehr einfach: Sie speichern einfach das Modulator-Signal zwischen, wobei das Latch-Modul vom Carrier getaktet wird. Hier sehen Sie ein Beispiel für die Macro-interne Struktur eines “Modulations-Multiplizierers” (wobei der Eingang “a” der Modulator ist):



Wir ersetzen also das Subtrahierer-Modul durch das Modulations-Macro  $a - x$  und das zweite Multiplizierer-Modul durch das Modulations-Macro  $x \text{ mul } a$ . So sieht die Struktur nach dem Ersetzen aus (wir haben auch die Const-Module durch QuickConst ersetzt, aber das ist unwichtig):



Sie können die Modulations-Eingänge von Modulations-Macros normalerweise an ihren Icons (Symbole auf dem Modul) erkennen. Eine Modulator-Eingangs-Position erkennen Sie an der Position des Pfeils auf dem Modul. Im Fall des Subtrahierer-Moduls ist der Pfeil oben, deshalb ist der Modulations-Eingang auch oben. Beim Multiplizierer-Modul ist der Pfeil unten, entsprechend liegt auch der Modulations-Eingang unten. Sie werden auch bemerken, dass der Ausgang dieser Module jeweils dem Carrier-Eingang gegenüber liegt, was einen zusätzlichen Anhaltspunkt bietet.



Schließlich können Sie noch den Mauszeiger über das Modul und seine Eingänge bewegen und die zugehörigen Hinweis-Texte lesen.

In der obigen Struktur werden keine Events gesendet, bis ein Event am Eingang der Core Cell auftritt:

- Das Modul `lxl` wird direkt vom Core-Cell-Eingangs-Event getriggert.
- Das nachfolgende Subtrahierer-Modul wird nur vom Ausgang des Moduls `lxl` getriggert, der nur als Reaktion auf das Eingangs-Event ein Event sendet; `QuickConst` hat keine Trigger-Funktion.
- Der erste Multiplizierer wird entweder vom Ausgang des Subtrahierer-Moduls oder vom Core-Cell-Eingangs-Event angetriggert, aber wir haben bereits gesehen, dass diese beiden Events nur gleichzeitig auftreten.
- Der zweite Multiplizierer wird nur vom ankommenden Event und nicht von `QuickConst` getriggert.

So, nun legt unsere Struktur ein etwas besser nachvollziehbares Verhalten an den Tag.

# Audio-Verarbeitung im Kern

## Audio-Signale

Es gibt keinen besonderen Signal-Typ für Audio-Signale in REAKTOR Core. Audio wird in REAKTOR Core als Event behandelt, das sich hinsichtlich seiner Struktur nicht von beliebigen anderen Events unterscheidet. Der Unterschied besteht vielmehr darin, dass entlang von Audio-Signalwegen die Events normalerweise “in regelmäßigen Abständen zu diskreten Zeitpunkten” erzeugt werden, und zwar abhängig von der “Sampling-Rate”.

Um Events in regelmäßigen Abständen (oder, was diese Sache angeht, irgendwelche Events) zu erzeugen, brauchen wir eine Event-Quelle. Ähnlich wie bei Event-Core-Cells, bei denen die Eingänge der Module die Event-Quellen waren, sind auch bei Audio-Core-Cells die Eingänge die Event-Quellen. Allerdings steht uns hier ein zusätzlicher Eingangs-Typ zur Verfügung:

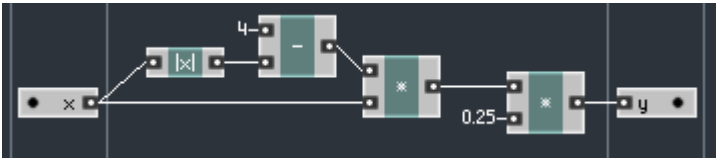
**Audio Inputs** senden regelmäßig und mit der in den Sampling-Rate-Einstellungen der umgebenden Primary-Level-Struktur eingestellten Rate REAKTOR-Core-Events an das Innere der Struktur. Die Events werden gleichzeitig von allen Audio-Eingängen einer Core-Cell-Struktur gesendet.

Die Audio-Eingänge senden auch das Initialisierungs-Event an die Core-Cell-Struktur. Dieses Event wird unabhängig davon gesendet, was gerade in der umgebenden Primary-Level-Struktur passiert. Auf jeden Fall hängt der Wert, den diese Eingänge während der Initialisierung senden, vom äußeren Initialisierungs-Vorgang ab.

Es gibt auch einen Ausgangs-Typ, der *anstelle* von Event-Ausgängen verwendet werden muss.

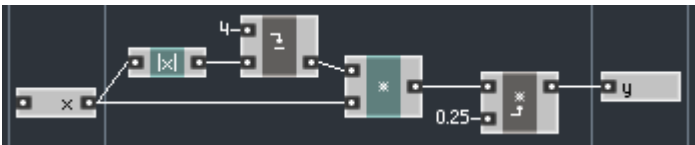
**Audio Outputs** liefern den letzten aus dem Inneren der REAKTOR-Core-Struktur empfangenen Wert an die umgebende Primary-Level-Struktur. Weil die Audio-Ausgänge auf dem Primary Level keine Events senden, werden keine Events nach außen geschickt.

Jetzt werden wir denselben Shaper, den wir für Events gebaut haben, für Audio-Signale nachbauen. Dafür erzeugen Sie zuerst eine Audio-Core-Cell. Grundsätzlich können wir genau dieselbe Struktur verwenden, nur dass wir anstelle von Event-Eingängen und –Ausgängen die Audio-Variante wählen müssen:

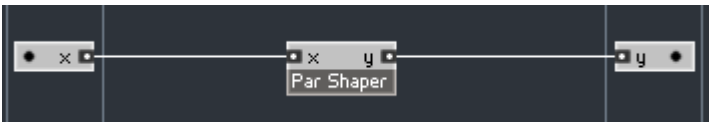


Sie fragen sich vielleicht, warum wir in diesem Fall nicht die Modulations-Macros verwenden. Nun, weil wir hier ein Audio-Signal verarbeiten und Audio-Signale immer ein Initialisierungs-Event senden, sind wir hier auf der sicheren Seite. Wenn Sie wollen, können Sie aber auch Modulations-Macros verwenden, das macht keinen Unterschied.

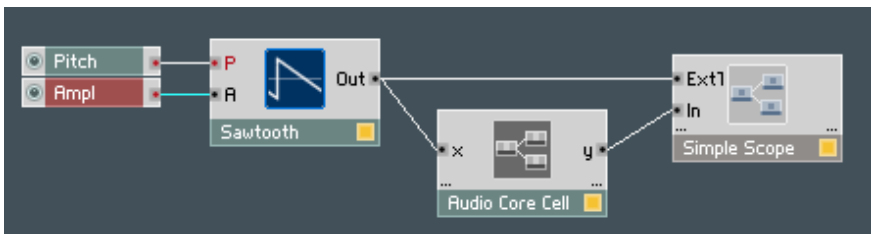
Wir könnten auch versuchen, die oben abgebildete Struktur in ein Macro zu verpacken, das wir in verschiedenen REAKTOR-Core-Struktur sowohl für Audio- als auch für Event-Verarbeitung verwenden können. In diesem Fall sollten wir besser Modulations-Macros verwenden, weil wir ja nicht im Voraus wissen, welche Art von Signalen von diesem Modul verarbeitet wird:



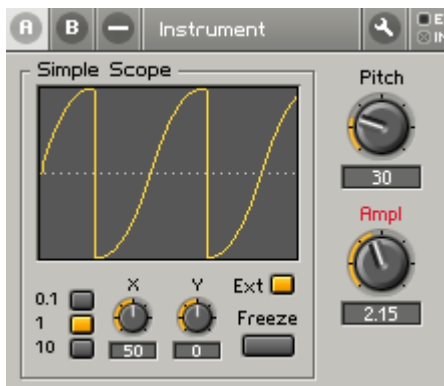
Und das ist die innere Struktur der Audio-Core-Cell in diesem Fall:



Um sie auszuprobieren, schließen wir einen Sägezahn-Oszillator und ein Oszilloskop an. Ein Oszilloskop finden Sie unter *Insert Macro > Classic Modular > OO Classic Modular – Display > Simple Scope* (aus einer Primary-Level-Struktur). Vergessen Sie auch nicht, die Anzahl der Stimmen für das Instrument auf 1 zu begrenzen.



Wir verwenden den externen Trigger für das Oszilloskop, um bei hohen Verzerrungsgraden eine bessere Synchronisation zu erzielen (der Schalter *Ext* auf dem Oszilloskop-Panel muss aktiv sein, damit das funktioniert). Ändern Sie den Bereich des Reglers *Ampl* auf einen Wert zwischen 0 und 5, damit Sie die formende Wirkung unserer Schaltung sehen können.



## Sampling Rate Clock Bus

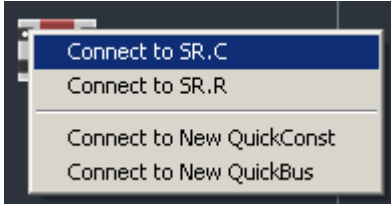
Es sieht ganz so aus, als würden zumindest einige Funktionen zum Aufbau von Audio-Strukturen noch fehlen. Eine könnte uns zum Beispiel erlauben, Audio-Core-Cells ohne Audio-Eingänge zu entwerfen. Wir können solche Core Cells natürlich bauen, aber woher bekommen wir dann die Quelle für die Audio-Events? Die zweite Funktion brauchen wir, weil viele DSP-Algorithmen Kenntnis über die derzeitige Sampling-Rate haben müssen. Wir werden Ihnen diese fehlenden Funktionen im Folgenden vorstellen.

In jeder REAKTOR-Core-Struktur ist eine besondere Art von Verbindungen möglich, die “Sampling Rate Clock Bus” heißt. Dieser Bus überträgt zwei Signale: die Clock und die Rate.

**Clock** ist eine Signal-Quelle, die Events regelmäßig mit Audio-Sampling-Rate sendet. Wie bei allen Standard-Audio-Signalen der Fall, sendet sie außerdem ein Initialisierungs-Event.. Der Wert dieser Events ist zurzeit immer Null, aber im eigentlich sollte jede Struktur, die dieses Signal verwendet, die Werte ignorieren, weil diese sich in Zukunft möglicherweise ändern.

**Rate** ist eine Signal-Quelle, deren Wert immer der aktuellen Audio-Sampling-Rate in Hz entspricht. Die Events werden von dieser Quelle während der Initialisierung und bei jeder Änderung der Sampling-Rate gesendet.

Sie können auf den Sampling-Rate-Bus zugreifen, indem Sie auf einem Signal-Eingang einen Rechts-Klick ausführen und aus dem Menü den Eintrag *Connect to SR.C* auswählen, um den Eingang mit einem Clock-Signal zu verbinden. Um den Eingang mit einem Rate-Signal zu verbinden, wählen Sie den Eintrag *Connect to SR.R*.



Die Verbindung wird direkt neben dem Eingang angezeigt:



---

Der Sampling Rate Clock Bus arbeitet innerhalb von Event-Core-Cells nicht.

---

## Verbindungs-Feedback

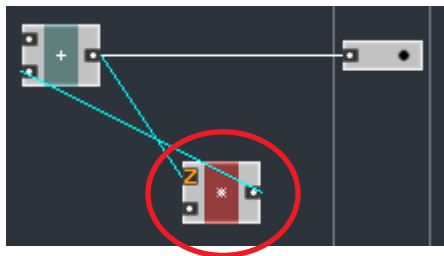
Wie Sie bereits gesehen haben, lassen sich die Regeln für die Verarbeitungsreihenfolge nicht anwenden, wenn innerhalb der Struktur Feedback-Verbindungen existieren. Deshalb müssen wir ein paar weitere Regeln aufstellen, die über den Umgang mit solchen Rückkopplungen bestimmen.

Die Hauptregel ist: REAKTOR-Core-Strukturen können nicht mit Feedback umgehen.

Das stimmt natürlich nicht ganz – Sie können Feedback-Verbindungen in REAKTOR Core anlegen. Aber weil die Engine von REAKTOR Core keine Strukturen mit Feedback verarbeiten kann, wird sie das Feedback *auflösen*. “Feedback auflösen” bedeutet, dass Ihre Struktur so verändert wird (und zwar intern, Sie werden es auf dem Bildschirm nicht sehen), dass keine Rückkopplung mehr auftritt.

Nur für den Fall, dass Sie es noch nicht wussten: In der digitalen Welt ist ein Feedback ohne Verzögerung nicht möglich. Normalerweise entsteht eine Verzögerung von mindestens einem Sample im digitalen Audio-Signalweg. Und

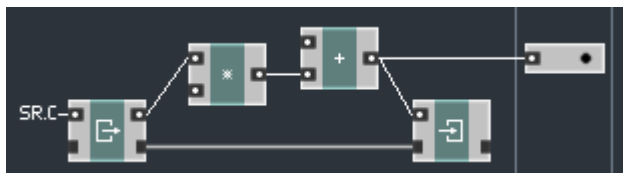
deshalb setzt die Engine von REAKTOR Core beim Auflösen der Feedback-Situation ein Modul ( $Z^{-1}$ ) ein, das einen Versatz von einem Sample Länge in den Feedback-Pfad einfügt. Bekanntlich wird eine Stelle, an der das Modul  $Z^{-1}$  implizit seine Arbeit verrichtet, durch das große orangefarbene Z gekennzeichnet, das Sie an dem betreffenden Port sehen:



Wir haben schon eine Struktur auf der Grundlage der Module *Read* und *Write* gesehen, in der die  $Z^{-1}$ -Funktion zum Einsatz kam. Lassen Sie uns eine solche Konstruktion mit *Read* und *Write* in unsere Struktur einbauen. Wir werden sie an das Kabel anschließen, an dem die Feedback-Auflösung stattfindet:



Also, erst schreiben, dann lesen (und beachten Sie, dass das Modul *Read* via SR.C getaktet wird, um sicherzustellen, dass der Lese-Vorgang einmal pro "Audio-Tick" stattfindet). Deshalb liegt der gelesene Wert immer ein Audio-Sample hinter dem geschriebenen zurück. Und schon ist da kein Feedback mehr in der Struktur. Sehen Sie's nicht? Okay, lassen Sie uns die Module ein bisschen zurechtrücken (und dabei keine einzige Verbindung verändern):



Sehen Sie's jetzt? Na klar.

Also beseitigt das Einfügen eines  $Z^{-1}$ -Moduls formal die Rückkopplung aus der Struktur, belässt sie aber logisch betrachtet darin (durch die Verzögerung von einem Audio-Sample).

---

Tatsächlich ist die interne Struktur eines Macros des Typs  $Z^{-1}$  etwas komplizierter und besteht nicht nur aus einem Paar von *Read*- und *Write*-Modulen. Wie es genau beschaffen ist und warum das so ist, lernen Sie im nächsten Abschnitt.

---

Sie haben keine Kontrolle darüber, an welcher Stelle die automatische Feedback-Auflösung stattfindet. Sie tritt an einer beliebigen Signal-Verbindung innerhalb der Feedback-Schleife auf. Es ist nicht einmal gesagt, dass die Auflösung immer an dieser bestimmten Verbindung auftritt – in der nächsten Version der Software kann das anders sein, aber es kann auch schon ausreichen, die Struktur an einer anderen Stelle zu ändern oder sie neu von der Festplatte zu laden.

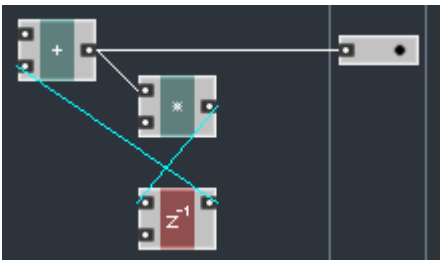
Diese automatische Feedback-Auflösung ist für Strukturen gedacht, bei denen es nicht von Bedeutung ist, wo genau die Auflösung stattfindet. Es kann nämlich vorkommen, dass solche Feedback-Strukturen von Anwendern erstellt werden, die sich noch nicht gut genug mit digitaler Signal-Verarbeitung auskennen, um das Rückkopplungs-Problem zu verstehen. Die automatische Feedback-Auflösung erlaubt diesen Anwendern immer noch, brauchbare Ergebnisse zu erzielen.

---

Wenn Sie die Punkte, an denen in Ihrer Struktur eine Feedback-Auflösung stattfinden soll, genau bestimmen wollen, können Sie dafür gezielt  $Z^{-1}$ -Module in die Struktur einsetzen. Diese Module werden die Rückkopplung formal entfernen, eine Auflösung durch REAKTOR Core ist dann nicht mehr notwendig.

---

Hier sehen Sie eine Version der oben abgebildeten Struktur mit einem  $Z^{-1}$ -Macro (das Sie im Untermenü *Expert Macro > Memory* finden):



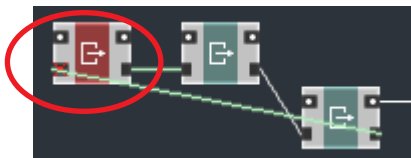
Wie Sie sehen können, ist das große orangefarbene Z nun verschwunden. Beachten Sie auch, dass der Punkt, an dem der Versatz von einem Sample erzeugt wird, anders angeordnet ist als bei der automatischen Feedback-Auflösung: Die automatische Auflösung fand in der Verbindung zwischen dem Ausgang des Addierers und dem Eingang des Multiplizierers statt, unsere neue Version löst die Rückkopplung zwischen dem Ausgang des Multiplizierers und dem Eingang des Addierers auf.

---

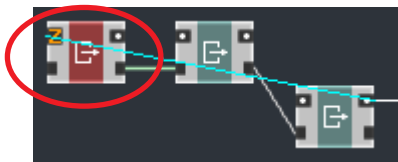
Die Bedeutung des zweiten Eingangs am Modul  $Z^{-1}$  wird später noch erklärt. Im Normalfall lassen Sie diesen Eingang einfach unbeschaltet.

---

Feedback in OBC und anderen Typen von Nicht-Signal-Verbindungen (die wir noch kennen lernen werden) ist vollkommen sinnlos und deshalb nicht erlaubt. Sofern Feedback-Schleifen auftreten, in denen keine Kabel-Verbindungen vorkommen, wird eine dieser Verbindungen als ungültig markiert und als nicht vorhanden betrachtet. Die Markierung "ungültig" ist ein großes rotes X auf dem betreffenden Port:



Auf der anderen Seite sind Feedback-Schleifen mit gemischten Arten von Verbindungen, die normale Signal-Kabel enthalten, vollkommen in Ordnung. Diese Rückkopplungen werden normal aufgelöst, wobei die Auflösung in einem der normalen Signal-Kabel stattfindet:



---

Alles in allem bedeutet dies, dass Nicht-Signal-Verbindungen nie von der Feedback-Auflösung betroffen sind, außer wenn Sie komplette Nicht-Signal-Rückkopplungen erzeugen, was nun wirklich überhaupt keinen Sinn ergibt.

---



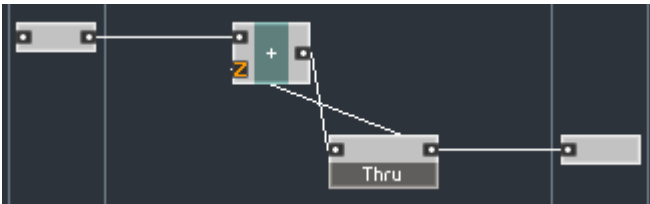
## Feedback um Macros herum

Macros werden hinsichtlich der Feedback-Auflösung im Allgemeinen genauso behandelt wie die eingebauten Module.

Lassen Sie uns ein Macro betrachten, das einfach das ankommende Signal durchlässt. Das hier ist die interne Struktur eines solchen Macros, das wir einfach *Thru* nennen:



Nun nehmen wir an, wir bauen eine Feedback-Struktur, in der wir dieses Macro verwenden:



Die Feedback-Schleife verläuft durch zwei Kabel in der obigen Struktur und durch ein weiteres Kabel *innerhalb* des Macros. Nur, wo findet jetzt die Auflösung statt? Okay, Sie können auf dem Bild oben erkennen, dass sie *in diesem speziellen Fall* am Eingang des Addierer-Moduls stattfindet, aber wie wir wissen, hätte sie auch irgendwo anders stattfinden können.

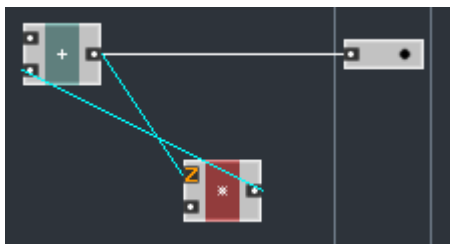
Stellen Sie sich für einen Moment vor, dass Thru kein Macro, sondern ein eingebautes Modul wäre. In diesem Fall ist es offensichtlich, dass die Feedback-Auflösung nicht innerhalb dieses Moduls stattfinden könnte, sondern außerhalb geschehen müsste. Nun, wir versuchen unser Bestes, den Macros das Aussehen und Verhalten der eingebauten Module beizubringen. Aus diesem Grund findet *standardmäßig* die Auflösung von Feedback-Schleifen außerhalb des Macros statt. Es ist zwar nicht festgelegt, wo genau sie stattfindet, aber die Stelle liegt auf jeden Fall außerhalb unseres Macros *Thru*.

---

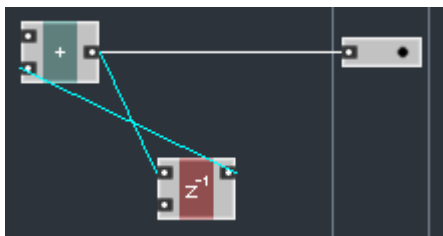
Die allgemeine Regel lautet: Die Feedback-Auflösung erfolgt auf der höchsten Struktur-Ebene der Feedback-Schleife

---

Trotzdem können Sie dieses Verhalten ändern – namentlich können Sie Feedback-Auflösung innerhalb von Macros erlauben. Sie hätten sich tatsächlich schon fragen sollen, wie das Macro  $Z^{-1}$  die Rückkopplung auflöst, wenn Macros genauso behandelt werden wie eingebaute Module. Schauen Sie sich einmal die folgende Struktur an:



Wenn Macros und eingebaute Module dasselbe sind, sollte sich doch eigentlich nichts ändern, wenn wir den Multiplizierer durch das Macro  $Z^{-1}$  ersetzen:



Aber es *gibt* einen Unterschied, weil das implizite Feedback jetzt verschwunden ist. Es muss also etwas Besonderes an diesem  $Z^{-1}$  Macro sein. Und in der Tat gibt es eine Besonderheit daran. Wenn wir uns dieses Macro von innen ansehen, erkennen wir zur Verwirklichung der  $Z^{-1}$ -Funktion fast dieselbe Struktur wie die bereits früher erwähnte:



Wie Sie sehen, ist der Clock-Eingang des Macros mit dem internen *Read*-Modul verbunden. Die Default-Verbindung für diesen Eingang ist keine Null-Konstante, sondern die *Audio-Clock*, und genau das wollen wir in den meisten Fällen. Das Modul, das zwischen dem oberen Eingang und dem Modul *Write* angeschlossen ist, sehen wir uns später noch an; vorerst ignorieren Sie es einfach.

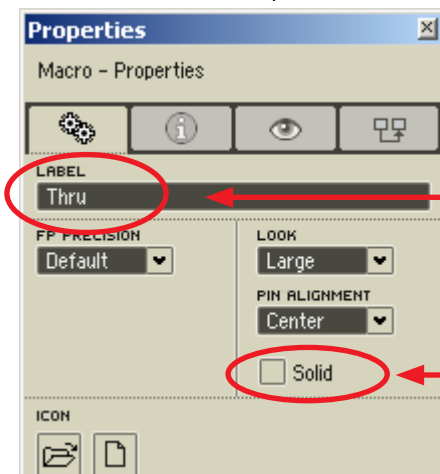
Bisher also nichts Besonderes an der Struktur, außer dass sie die  $Z^{-1}$ -Struktur zu enthalten scheint, die wir schon behandelt haben. Trotzdem, woher weiß die Engine von REAKTOR Core, dass diese Struktur dazu da ist, Feedback-Schleifen aufzulösen? Es ist ja offensichtlich, dass die Engine Feedback-Schleifen auflösen kann, aber wie erkennt sie, ob das beabsichtigt ist?

Diese Eigenschaft kontrollieren Sie im Properties-Fenster des Macros, genauer gesagt mit der Einstellung *Solid*:



Diese Eigenschaft verrät der REAKTOR-Core-Engine, ob das Macro für den Zweck der Feedback-Auflösung wie ein “festes” (solid) eingebautes Modul oder transparent erscheinen soll. In 99 % der Fälle werden Sie diese Eigenschaft eingeschaltet lassen. Das sollten Sie tun, weil Sie normalerweise keine implizite Feedback-Auflösung im Inneren Ihrer Macros stattfinden lassen wollen.

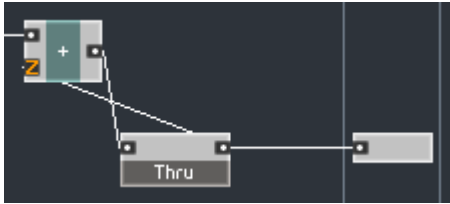
Ein Grund dafür ist, dass die Auflösung im Inneren eines Macros unsichtbar wäre, bis Sie in das Macro hineinschauen, sodass einige der impliziten Verzögerungen durch die  $Z^{-1}$ -Funktion unbemerkt geschehen. Zum Beispiel können wir unsere frühere Struktur mit dem Macro *Thru* nehmen und die Eigenschaft *Solid* abschalten (achten Sie darauf, dass Sie die *Solid*-Einstellung für das richtige Macro bearbeiten; das erkennen Sie an dem Text “Thru” im Feld *Label* des Properties-Fensters):



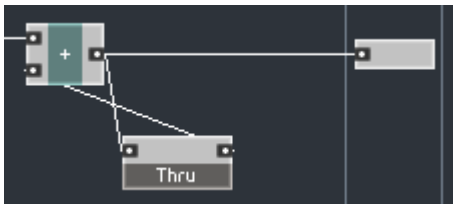
Stellen Sie sicher das Sie das richtige Macro bearbeiten.

Schalten Sie Solid aus.

Jetzt sieht die äußere Struktur wahrscheinlich genauso aus (wir sagen “wahrscheinlich”, weil man nie sicher sein kann, wo genau die automatische Feedback-Auflösung stattfinden wird):



Aber wenn Sie Ihre Struktur ein wenig verändern, indem Sie den Ausgang mit einem anderen Modul verbinden. könnte sie so aussehen:



Unsere Feedback-Auflösungs-Verzögerung scheint verschwunden zu sein. In einer größeren und unübersichtlicheren Struktur könnten wir leicht übersehen, dass eine implizite Verzögerung stattfindet. Wo ist unser  $Z^{-1}$ -Delay hin? Natürlich befindet es sich jetzt innerhalb unseres Macros *Thru* – an dem einzigen Ort in der Struktur, an dem wir es nicht sehen können.:



Ein anderer Grund dafür, die Eigenschaft *Solid* eingeschaltet zu lassen, ist, dass sich in manchen Fällen die Funktion eines Macros ändern kann, wenn es in den Feedback-Pfad eingesetzt wird. Also tun Sie sich selbst den Gefallen und schalten Sie diese Eigenschaft nur dann ab, wenn Sie Macros bauen, die Feedback auflösen sollen. Das wird aber nicht oft vorkommen.

Lassen Sie uns nun zum Modul  $Z^{-1}$  zurückkehren. Wenn die Eigenschaft *Solid* für dieses Macro abgeschaltet ist, werden die Grenzen des Macros für die Feedback-Auflösung vollkommen durchsichtig. Dadurch wird das Macro  $Z^{-1}$  nicht wirklich wie ein eingebautes Modul behandelt und ist daher in der Lage, Rückkopplungen in der beschriebenen Weise aufzulösen.

## Denormale Werte

Die Signal-Werte in den Strukturen, die wir während der bisherigen Abschnitte aufgebaut haben, werden im Computer durch einen binären Daten-Typ repräsentiert, den man auf Deutsch *Fließkomma-Zahlen*, auf Englisch *floating point numbers* oder kurz *floats* nennt. Diese Art von Daten erlaubt die effiziente Darstellung eines großen Spektrums von Werten. Die Bezeichnung *Fließkomma-Zahlen* gibt keinen genauen Aufschluss darüber, wie die Zahlen dargestellt werden, sondern beschreibt nur den Ansatz, der für ihre Darstellung gewählt wird, und lässt bei der Umsetzung eine Menge Freiraum für die Details der Implementation.

Die CPUs heutiger Personal Computer verwenden den “IEEE Floating Point Standard”. Dieser Standard definiert genau, wie die Fließkomma-Zahlen repräsentiert werden sollten und was bei Rechenoperationen mit diesen Zahlen geschehen soll (z. B., wie mit dem Thema der begrenzten Genauigkeit umgegangen werden soll). Insbesondere besagt dieser Standard, dass für eine Gruppe von besonders kleinen Fließkomma-Werten, die wegen der begrenzten Fließkomma-Genauigkeit nicht auf die “übliche” Weise dargestellt werden können, eine andere Darstellungsform gewählt werden soll. Diese andere Darstellungsform nennen wir “denormale Repräsentation” oder auch kurz “Denormale”.

---

Der Bereich der “denormalen” Werte für 32-Bit-Fließkomma-Werte reicht ungefähr von  $10^{-38}$  bis  $10^{-45}$  und von  $-10^{-38}$  bis  $-10^{-45}$ . Kleinere Werte als  $10^{-45}$  sind überhaupt nicht darstellbar und werden als Nullen angesehen.

---

Weil diese Zahlen eine etwas andere Darstellungsform haben als “normale” Zahlen, haben manche CPUs bestimmte Probleme im Umgang mit solchen Zahlen. Besonders Rechenoperationen mit diesen Zahlen können nur sehr viel langsamer durchgeführt werden und dauern bis zu zehnmal solange wie mit “normalen” Zahlen.

---

Eine typische Situation, in der denormale Zahlen über *längere Zeiträume hinweg* auftreten, ergibt sich durch exponentiell abfallende Werte, wie sie in Filtern, in manchen Hüllkurven oder in Feedback-Strukturen vorkommen. In solchen Strukturen fallen die Signale *asymptotisch* bis auf Null ab, nachdem das Eingangssignal auf den Wert Null geschaltet wurde. *Asymptotisch* bedeutet, dass das Signal versucht, den Wert Null zu erreichen, das aber nie schafft. In dieser Situation können denormale Zahlen auftreten und für relativ lange Zeit in der Struktur verweilen, wodurch sich die CPU-Belastung deutlich erhöht.

---

---

Eine andere Situation, in der denormale Zahlen auftreten können, entsteht, wenn Sie die Genauigkeit eines Fließkomma-Werts von einer höheren Genauigkeit (64 Bit) auf eine niedrigere Genauigkeit (32 Bit) umschalten. Auslöser ist hier, dass ein Wert von  $10^{-41}$  bei einer Fließkomma-Genauigkeit von 64 Bit nicht denormal ist, bei 32-Bit-Genauigkeit hingegen schon. Das Verändern der Fließkomma-Genauigkeit diskutieren wir übrigens später.

---

Lassen Sie uns nun die Modellierung eines 1-Pol-Tiefpassfilters mit einer Cutoff-Frequenz von 20 Hz betrachten. Unsere digitalen Signal-Werte werden analogen Spannungen in Volt entsprechen. Wir stellen uns vor, dass der Pegel des Eingangssignals über einen hinreichend langen Zeitraum gleichwertig mit einer Spannung von 1 Volt wäre. Dann ist die Spannung am Ausgang des Filters ebenfalls gleichwertig mit 1 V. Jetzt ändern wir abrupt die Eingangsspannung auf Null. Die Ausgangsspannung wird gemäß dem folgenden Gesetz abfallen:

$$V_{out} = V_0 e^{-2\pi f_c t}$$

wobei  $f_c$  der Filter-Cutoff in Hz ist,  $t$  die Zeit in Sekunden und  $V_0 = 1\text{ V}$  (Anfangs-Spannung).

Dann wird sich die Ausgangsspannung wie folgt verändern:

nach 0.5 sec	$V_{out} \approx 10^{-29}$ volt
nach 0.6 sec	$V_{out} \approx 10^{-33}$ volt
nach 0.7 sec	$V_{out} \approx 10^{-38}$ volt
nach 0.8 sec	$V_{out} \approx 10^{-44}$ volt

Oha, die Zahlen zwischen  $10^{-38}$  und  $10^{-45}$  liegen im denormalen Bereich. Also wird in der Zeitspanne ungefähr zwischen 0,7 bis 0,8 Sekunden unsere Spannung durch einen denormalen Wert dargestellt. Und das nicht nur im Inneren des Filters – das Ausgangssignal des Filters wird wahrscheinlich von einigen stromabwärts in der Struktur liegenden Modulen weiter verarbeitet, sodass es zumindest diese paar folgenden Module ebenfalls mit denormalen Werten zu tun bekommen.

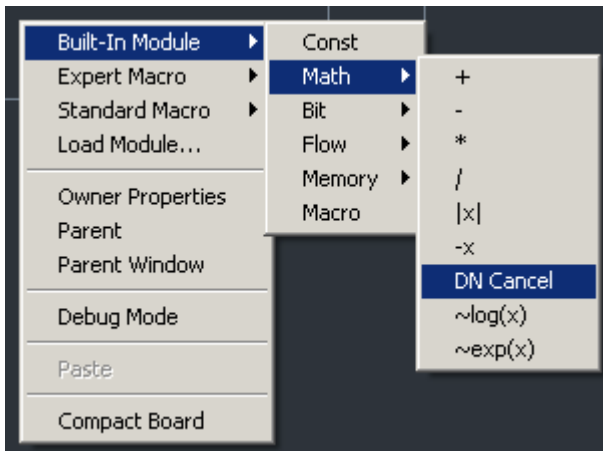
Bei einer Sampling-Rate von 44,1 kHz entspricht ein Zeitintervall von 0,1 Sekunden 4410 Samples. Wenn wir voraussetzen, dass die typische ASIO-Puffergröße einige Hundert Samples beträgt, müssen wir bei deutlich höherer CPU-Last mehrere Puffer erzeugen. Falls die CPU-Belastung (pro Puffer-Berechnung) nah genug an 100 % heranreicht oder diesen Wert überschreitet, führt das zu Audio-Aussetzern.

---

Aus dem obigen Text müssen Sie vor allem einen Schluss ziehen: Denormale Werte sind schlecht für die Echtzeit-Verarbeitung von Audio.

---

Die Module auf REAKTORs Primary Level sind so programmiert, dass sie auf denormale Werte achten, die in ihrem Inneren auftreten. Dieser Vorsorge-Mechanismus beruht auf Modifikationen an den verwendeten DSP-Algorithmen; diese wurden so verändert, dass sie generell keine denormalen Werte erzeugen. Wenn Sie Ihre eigenen Low-Level-DSP-Strukturen in REAKTOR Core entwerfen, müssen Sie denormale Werte ebenfalls vermeiden. Um Ihnen dabei zu helfen, haben wir das Modul *Denormal Cancel* eingeführt, das Sie im Untermenü *Built In Module > Math* finden:



Das Modul *Denormal Cancel* besitzt einen Eingang und einen Ausgang und versucht, den ankommenden Wert vorsichtig so zu verändern, dass keine Denormalen den Ausgang erreichen:



Die Art, wie dieses Modul das Signal modifiziert, ist nicht dauerhaft festgelegt und kann sich von einer Software-Version zur andern verändern oder sogar an zwei Stellen in der Struktur unterschiedlich sein. Zurzeit wird bei der Modifikation dem Eingangs-Wert eine sehr kleine Konstante hinzugefügt. Wegen der damit einhergehenden Einbußen bei der Genauigkeit verändert diese Addition keine Werte, die groß genug sind (so wird ein Wert von beispielsweise  $10^{-10}$  überhaupt nicht verändert). Dieselben Genauigkeits-Verluste machen es andererseits sehr unwahrscheinlich, dass das Ergebnis der Addition ein denormaler Wert ist (und in den meisten Fällen ist das sogar unmöglich).

---

Wenn aus irgendeinem Grund das Modul *Denormal Cancel* (DN Cancel) in Ihrer Struktur nicht funktioniert, können Sie selbstverständlich Ihre eigene Technik zum Unterdrücken denormaler Werte verwenden. Allerdings kann dabei das Problem auftreten, dass diese Technik, die auf einer Plattform arbeitet, auf der anderen nicht arbeitet, während wir versucht haben, unseren eingebauten *DN-Cancel*-Algorithmus an jede unterstützte Plattform anzupassen. Versuchen Sie also nach Möglichkeit, das Modul zu verwenden. Wir denken sogar darüber nach, alternative Algorithmen in dieses Modul einzubauen – fühlen Sie sich ermutigt, in unserem Support-Forum Ihre Meinung zu diesem Thema zu sagen!

---

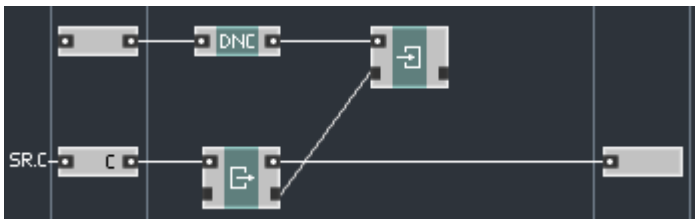
---

Einige CPUs bieten die Möglichkeit, den IEEE-Standard zu verletzen und die Erzeugung denormaler Zahlen zu unterdrücken, wobei die denormalen Werte zwangsweise auf Null gesetzt werden. Diese Option ist manchmal verfügbar, manchmal nicht. Weil REAKTOR-Core-Strukturen eigentlich plattformunabhängig arbeiten sollen, raten wir Ihnen sehr, bereits in Ihren Strukturen auf eine Unterdrückung denormaler Werte zu achten, selbst wenn Ihr eigener Rechner nicht unter den Auswirkungen dieser Werte leidet.

---

Weil eine der häufigsten Situationen, in denen denormale Werte auftreten, exponentiell abfallende Feedback-Schleifen sind, und weil die meisten Feedback-Schleifen in der Audio-Verarbeitung exponentiell abfallen (was Filter und Feedback-Schleifen mit Delays belegen), haben wir uns entschlossen, die Denormalen-Unterdrückung in das Standard-Macro  $Z^{-1}$  einzubauen.

Wie Sie sich erinnern, sieht das Innere dieses Macros folgendermaßen aus:



Es gibt noch eine andere Version dieses Macros namens  $Z^{-1} \text{ ndc}$ , die keine Denormalen-Unterdrückung durchführt (ndc = no denormal cancel).



Sie können diese alternative Version in Strukturen verwenden, von denen Sie sicher wissen, dass sie keine Denormalen erzeugen (z. B. FIR-Filter):



## Andere böse Zahlen

Denormale Zahlen sind nicht die einzige schlechte Art von Zahlen, die in Strukturen mit internem Zustand und besonders in Feedback-Schleifen kleben bleibt. Es gibt noch ein paar andere Sorten schlechter Zahlen: INFs, NaNs und QNaNs. Wir werden darauf jetzt nicht im Detail eingehen, die entsprechenden Informationen sind aber anderweitig verfügbar, zum Beispiel im Internet. Für uns wichtig ist, wie wir das Auftauchen dieser Zahlen in unseren Strukturen verhindern.

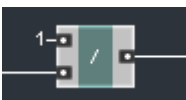
Im Allgemeinen erscheinen solche Zahlen als Ergebnis ungültiger Operationen. Das Teilen durch Null ist der einfachste Fall. Andere Fälle sind Zahlen, die zu groß sind, um in die Fließkomma-Darstellung zu passen (das wäre bei Werten oberhalb von  $10^{38}$  in vollem Ausmaß der Fall), oder die außerhalb des erlaubten Bereichs für eine bestimmte Operation liegen.

Solche Zahlen neigen dazu, in den Strukturen haften zu bleiben, und auf bestimmte Weise sind sie noch viel klebriger als Denormale. Wenn Sie nämlich einen denormalen Wert und einen anderen, nicht denormalen Wert addieren, wird das Ergebnis nicht denormal sein (es sei denn, der andere Wert ist auch extrem klein und liegt dicht an der Grenze zur Denormalität). Wenn Sie andererseits einen Wert zu einem INF addieren, wird das Ergebnis immer noch ein INF sein.

Abgesehen von ihrer Neigung, sich in Strukturen auf ewig (das heißt, bis die Struktur zurückgesetzt wird) festzusetzen, haben diese Zahlen auch die schlechte Angewohnheit, auf manchen CPUs extrem viel Rechenzeit zu fordern. Deshalb sollten wir vorsichtig sein und uns nach Kräften bemühen, die Entstehung dieser Zahlen zu verhindern.

Vorsichtig sein heißt zum Beispiel, dass Sie immer, wenn zwei Zahlen dividiert werden sollen, überprüfen müssen, ob eine Division durch Null möglich ist. Die Initialisierung ist hier von besonderer Bedeutung.

Betrachten Sie zum Beispiel folgendes Element einer Struktur:



Wenn aus irgendeinem Grund das Initialisierungs-Event nicht am unteren Eingang des Dividierer-Moduls ankommt, wird während der Verarbeitung der Initialisierung eine Division durch Null erfolgen. In diesem Fall können Sie darüber nachdenken, ein Delay-Macro aus der Abteilung “Modulation” zu verwenden, oder nach einer zu Ihren jeweiligen Anforderungen passenden Alternative suchen.

## Wie Sie ein 1-Pol-Tiefpassfilter bauen

Ein einfaches 1-Pol-Tiefpassfilter können Sie nach der folgenden rekursiven Gleichung aufbauen:

$$y = b * x + (1 - b) * y_{-1}$$

wobei

$x$  das Eingangs-Sample ist,

$y$  das neue Ausgangs-Sample,

$y_{-1}$  das vorige Ausgangs-Sample ist und

$b$  die Koeffiziente, die den Filter-Cutoff festlegt.

Den Wert der Koeffizienten  $b$  nehmen wir als gleichwertig zur normalisierten zyklischen Cutoff-Frequenz an, die sich nach folgender Formel errechnen lässt:

$$F_c = 2 * \pi * f_c / f_{SR}$$

wobei

$f_c$  die gewünschte Cutoff-Frequenz in Hz ist,

$f_{SR}$  die Sampling-Rate in Hz ist,

$\pi$  etwa 3,14159... entspricht und

$F_c$  der normalisierte zyklische Cutoff (in Radianen, falls Sie das interessiert) ist.

---

Tatsächlich entspricht die Koeffiziente  $b$  dem normalisierten Cutoff nur ungefähr, wobei sich die Abweichung bei hohen Cutoff-Werten vergrößert. Für unsere Zwecke sollte das aber mehr oder weniger in Ordnung sein, besonders, wenn wir keine genaue Einstellung der Cutoff-Frequenz für unser Filter brauchen

---

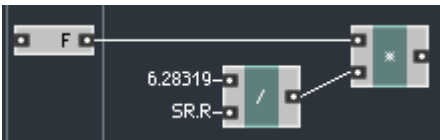
Wir beginnen, indem wir eine Audio-Core-Cell mit zwei Eingängen erzeugen: einen für den Audio-Eingang und eine für den Cutoff. Wir werden in dieser Version des Moduls einen Event-Eingang für den Cutoff verwenden.



Tatsächlich glauben wir, dass es eine gute Angewohnheit ist, REAKTOR-Core-Strukturen als Macros aufzubauen, um ihre leichte Wiederverwendung zu ermöglichen. Deshalb werden wir unser Filter auch als Macro anlegen. Wir erzeugen also ein neues Macro in der Core Cell und bestücken es mit derselben Konstellation von Eingängen und Ausgang:



Lassen Sie uns nun die Schaltung für die Konvertierung der Cutoff-Frequenz in den normalisierten zyklischen Cutoff bauen:

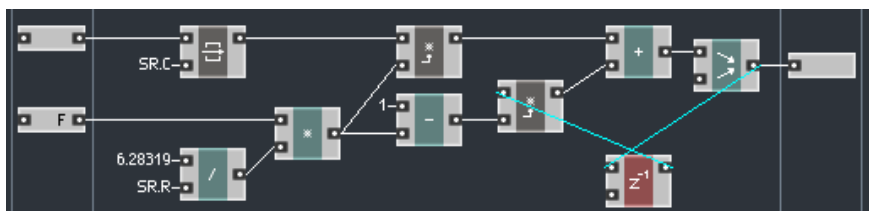


6.28319 ergibt sich aus  $2 \cdot \pi$ , dieser Wert wird dann durch die Sampling-Rate geteilt, woraus sich der Wert ergibt, der mit der Cutoff-Frequenz multipliziert wird. Wir brauchen keinen Modulations-Multiplizierer, weil "F" logisch ein Kontroll-Signal-Eingang ist, sodass wir die anfängliche Multiplikation auch dann durchführen dürfen, wenn kein Initialisierungs-Event am Eingang "F" anliegt.

---

Wir führen die Division vor der Multiplikation durch, weil die Division die CPU relativ stark belastet und sich die Sampling-Rate auch nicht so oft ändert. Wenn sich nur die Cutoff-Frequenz ändert, werden keine Events an das Dividierer-Modul geschickt und die Division wird nicht durchgeführt. Dies ist eine der Standard-Optimierungen, die Sie als Designer von REAKTOR-Core-Strukturen vornehmen können.

---



Das Audio-Eingangssignal wird für den Fall zwischengespeichert, dass Events asynchron zur Standard-Audio-Clock eintreffen. Das wäre in einer Core-Cell-Struktur, in der Audio-Eingänge bekanntlich ihre Events immer zur richtigen Zeit senden, nicht notwendig, aber in einem allgemein nutzbaren Core-Macro ist das eine sehr gute Maßnahme.

Zwei Modulations-Multiplizierer verhindern, dass Events am Eingang (die dort, allgemein gesagt, jederzeit eintreffen können), die Berechnung in der Feedback-Schleife triggern. Hier sollte auch klar werden, warum diese Macros “Modulations-Macros” heißen; in diesem Fall wird das aus dem Cutoff gewonnene Signal verwendet, um die Verstärkung im Feedback-Pfad zu modulieren.

---

Die Zwischenspeicherung (Latching) ist eine Standard-Technik in REAKTOR Core, die dafür sorgt, dass ankommende Events nicht zur unpassenden Zeit Berechnungen triggern. Latching ist auch in Form von Modulations-Macros und anderen Situationen weit verbreitet.

---

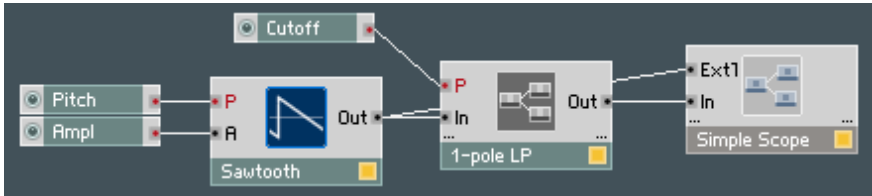
Das Modul  $Z^{-1}$  speichert der vorigen Ausgangs-Wert und sendet automatisch auf jeden Audio-Tick hin ein Event mit dem vorigen Ausgangs-Wert als Inhalt. Es achtet auch auf mögliche denormale Werte, die andernfalls auftreten könnten. Wenn Sie sich mit DSP auskennen, sollte Ihnen auffallen, dass die Struktur ziemlich ähnlich aussieht wie die herkömmlichen DSP-Filter-Diagramme.

Das Merge-Modul am Ausgang des Addierers stellt sicher, dass der Zustand des Filters nach der Initialisierung immer noch Null ist, auch wenn das Eingangssignal zu diesem Zeitpunkt einen anderen Wert als Null führt.

Vergessen Sie nicht, den Tonhöhe-nach-Frequenz-Konverter  $P2F$  in die Core Cell einzubauen; wenn Sie dieses Modul platziert haben, sind wir bereit für den Test:



For testing we suggest using the following structure (don't forget about the 1 voice setting for the instrument):



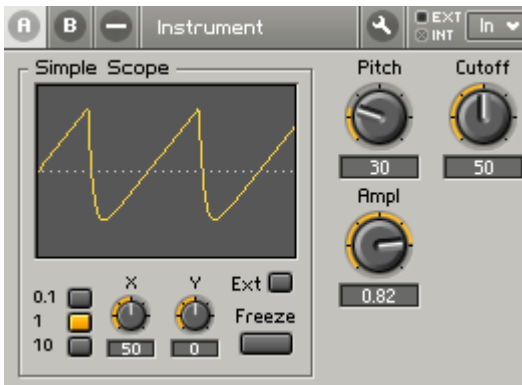
Den Cutoff-Regler sollten Sie auf einen Bereich zwischen 0 und 100 oder so ähnlich einstellen. Vorsicht vor zu hohen Cutoff-Werten! Wegen der zunehmenden Abweichung der Filter-Koeffizienten bei hohen Cutoff-Werten wird das Filter bei solchen hohen Werten instabil.

---

Ein besseres Filter-Design sollte wenigstens die Cutoff-Werte auf den Bereich beschneiden, in dem das Filter stabil arbeitet. Für unseren Fall hätten wir das durch Beschneiden der Koeffizienten  $b$  auf den Bereich zwischen 0 und 0,99 oder ähnliche Werte erreicht. Die Technik für das Beschneiden von Werten wird später im Text noch beschrieben.

---

Das hier sollten Sie nun im Panel sehen:



Bewegen Sie den Cutoff-Regler und beobachten Sie, wie sich die Form des Signals verändert.

# Bedingte Verarbeitung

## Event-Routing

Die Events in REAKTOR Core müssen nicht immer dieselben festgelegten Pfade benutzen – es gibt auch die Möglichkeit, diese Pfade dynamisch zu verändern. Sie können diese Änderung der Signalwege mit dem Modul *Router* erreichen, das Sie unter *Built-In Module > Flow > Router* finden:



Das Modul *Router* nimmt Events an seinem Signal-Eingang (unten) entgegen und leitet entweder auf seinen Ausgang 1 (oben) oder seinen Ausgang 0 (unten). Auf welchen der Ausgänge die Events geleitet werden, hängt vom aktuellen Zustand des Router-Moduls ab, der über den Eingang *Ctl* (oben) gesteuert wird.

Der Eingang *Ctl* akzeptiert eine Verbindung eines neuen Typs, die weder mit normalen Signalen noch mit OBCs kompatibel ist. Dieser Signal-Typ heißt “Boolean Control” (*BoolCtl*). Das *BoolCtl*-Signal kann einen von zwei Zuständen annehmen: wahr oder falsch (an oder aus, 1 oder 0). Wenn das *BoolCtl*-Signal den Zustand “Falsch” hat, werden die Events auf den Ausgang 0 geleitet.

---

Die *BoolCtl*-Signale (und andere Kontroll-Signale) unterscheiden sich deutlich von den normalen Signalen in REAKTOR Core: Sie übertragen keine Events und können deshalb von sich aus keine Verarbeitung triggern.

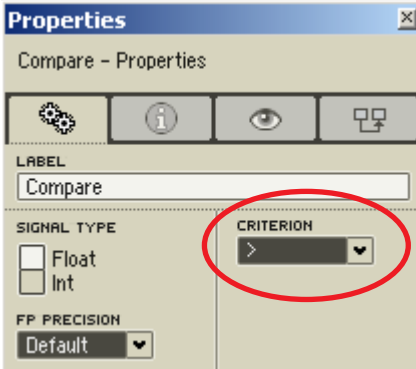
---

Um einen Router zu kontrollieren, braucht man offensichtlich eine Kontroll-Signal-Quelle. Die gebräuchlichste wäre das Vergleichs-Modul (*Compare*), das Sie unter *Built In Module > Flow > Compare* finden:



Dieses Modul vergleicht zwei ankommende Signale und gibt das Ergebnis in Form eines *BoolCtl*-Signals aus. Das obere Eingang wird als links des Vergleichs-Symbols befindlich angenommen, der untere Eingang befindet sich logisch rechts davon. So erzeugt ein Modul mit dem Symbol “>” ein *BoolCtl*-Signal mit der Wertigkeit “wahr”, wenn der Wert am oberen Eingang höher ist als der Wert am unteren Eingang. Sie können das Vergleichs-Kriterium im

Properties-Fenster des Moduls ändern:



Die verfügbaren Kriterien sind:

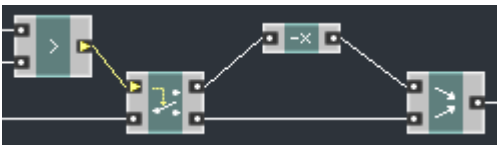
- = gleich
- != nicht gleich ( $\neq$ )
- <= kleiner gleich ( $\leq$ )
- < kleiner
- >= größer gleich ( $\geq$ )
- > größer

---

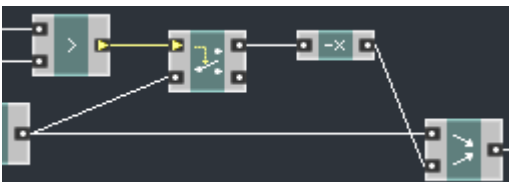
Es ist natürlich möglich, mehrere Router an dasselbe Vergleicher-Modul anzuschließen. Die Router werden in diesem Fall zeitgleich ihre Zustände wechseln.

---

Das Router-Module spaltet den Event-Pfad in zwei Zeige. Ziemlich oft werden diese Pfade wieder zusammengeführt:



Abhängig vom Ergebnis des Vergleichs wird die obige Struktur das Signal entweder invertieren oder es unberührt lassen. Eine alternative Implementation dieser Struktur ist möglich:



In dieser Version ist der Ausgang 0 des Routers nicht angeschlossen. Deshalb arbeitet der Router wie ein Gate, das die Events nur dann durchlässt, wenn es sich im Zustand “wahr” befindet. Das invertierte Signal kommt dann am zweiten Eingang des Moduls *Merge* an, wobei es den nicht invertierten Wert überschreibt, der immer am ersten Eingang ankommt. Wenn der Router sich im Zustand “falsch” befindet, empfängt der Inverter kein Event und sendet demnach auch kein Signal an den zweiten Eingang des Moduls *Merge*, sodass das Original-Signal unverändert den Ausgang des Moduls *Merge* erreicht.

---

Die Zweige werden meistens mit einem Merge-Module wieder vereint. Theoretisch würden sich dafür aber auch andere Module eignen, z. B. arithmetische Module wie Addierer, Multiplizierer und so weiter.

---

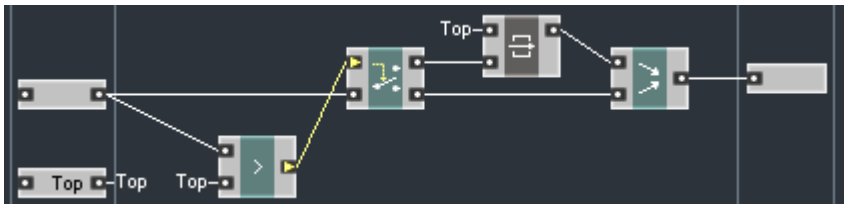
---

Router behandeln das Initialisierungs-Event genau wie jedes andere Event. Deshalb kann man durch die Verwendung von Router-Modulen das Initialisierungs-Event ausfiltern, sodass es in bestimmten Bereichen der Struktur nicht auftaucht.

---

## Wie Sie einen Signal-Beschneider bauen

Lassen Sie uns nun eine REAKTOR-Core-Macro-Struktur bauen, die das ankommende Audio-Signal oberhalb eines bestimmten Pegels abschneidet:



Wenn das Eingangssignal nicht oberhalb der festgelegten Schwelle liegt, wird es auf den Ausgang 0 des Routers geleitet und gelangt durch das Modul *Merge* unverändert zum Ausgang der Struktur. Wenn das Signal die Schwelle überschreitet, wird es auf den Ausgang 1 geleitet, wo es das Latch-Modul triggert, das den Schwellwert an das Merge-Modul sendet. Dasselbe passiert bei der Initialisierung.

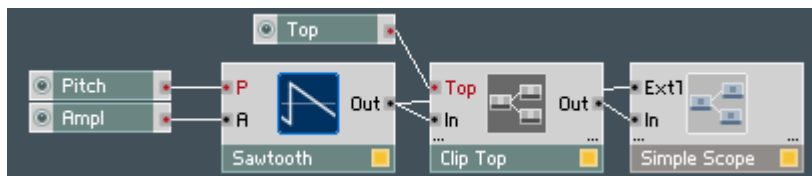
---

Beachten Sie, dass diese Struktur ihre Ausgabe nicht als Reaktion auf Änderungen des Schwellwerts ändert. Stattdessen wird der neue Schwellwert für das nächste am Signal-Eingang ankommende Event

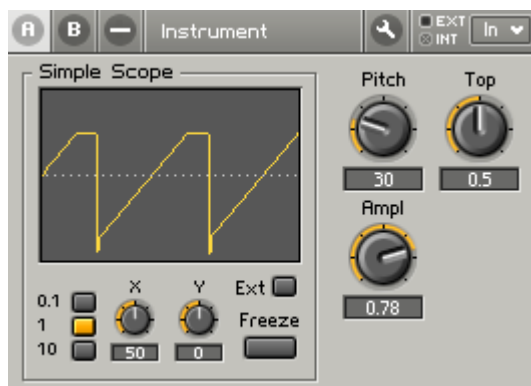


und alle nachfolgenden Events verwendet.. Dieses Verhalten ähnelt in gewisser Weise dem Verhalten der Modulations-Macros, bei denen Veränderungen am Modulator ebenfalls keine Ausgangs-Events erzeugen. similar to a modulation macros behavior, where modulator changes do not result in output events.

Hier sehen Sie eine Test-Struktur für das Beschneider-Modul, das wir unter Verwendung einer Audio-Core-Cell gebaut haben:



Und das hier sollten Sie in der Panel-Ansicht sehen:



Tatsächlich finden Sie eine Anzahl solcher “Modulations”-Beschneider-Macros im Menü *Expert Macro > Clipping*.

## Wie Sie einen einfachen Sägezahn-Oszillator aufbauenr

Wir wollen nun einen einfachen Sägezahn-Oszillator bauen, der eine Sägezahn-Wellenform mit der Amplitude 1 und einer festgelegten Frequenz erzeugt. Wir werden den folgenden Algorithmus verwenden: “Erhöhe den Ausgangssignal-Pegel mit konstanter Geschwindigkeit und lasse ihn in dem Moment, in dem er größer als 1 wird, um 2 abfallen.”

Man könnte nun einwenden, dass wir auch den Pegel auf  $-1$  zurücksetzen könnten, anstatt das Signal um 2 abfallen zu lassen, aber das

ist im Allgemeinen eine schlechte Idee, weil wir dann die geforderte Oszillator-Frequenz nicht genau beibehalten können.

Die Geschwindigkeit des Anwachsens definiert die Oszillator-Frequenz durch die folgende Gleichung:

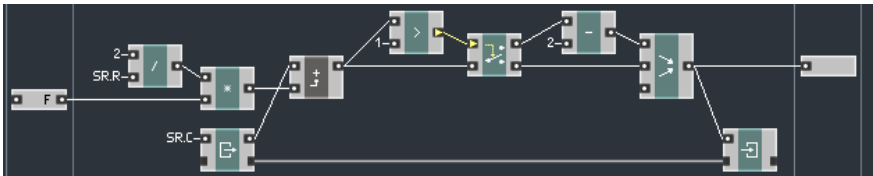
$$d = 2f / f_{\text{SR}}$$

wobei  $d$  Pegel-Erhöhung mit jedem Audio-Sample ist,  
 $f$  die Oszillator-Frequenz und  $f_{\text{SR}}$  die Sampling-Rate.

Zuerst bauen wir die Schaltung für die Berechnung der Geschwindigkeit des Anwachsens:



Jetzt brauchen wir eine Erhöher-Schleife (Increment Loop). Es ist also Zeit, ein Modul-Paar aus *Read* und *Write* zu verwenden, genau wie wir es im Akkumulierer gemacht haben:



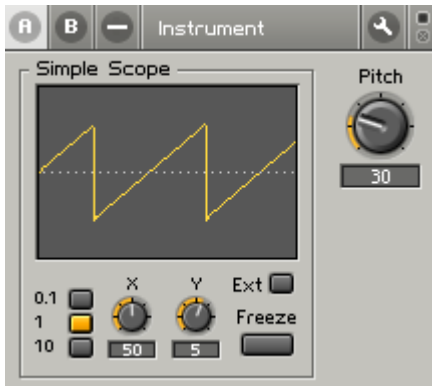
Das Modul *Read* triggert die Pegel-Erhöhung während jedes Audio-Events. Die Summe der alten Pegel und die Erhöhung werden dann mit dem Schwellwert 1 verglichen und je nach Ergebnis entweder direkt an den Ausgangs-Zwischenspeicher geleitet oder in den umhüllenden Schaltkreis geschickt.

Der dritte Eingang des Moduls *Merge* stellt die Initialisierung des Oszillators mit dem Wert Null sicher. In der Theorie hätte das Modul, das 2 vom Signal-Pegel subtrahiert, eigentlich ein Modulations-Macro sein sollen, aber das soll uns hier nicht weiter stören, denn *Merge* überschreibt das Ergebnis der Initialisierung sowieso.

Hier ist die vorgeschlagene Test-Struktur (vergessen Sie nicht das Konverter-Modul *P2F* innerhalb der Core Cell):



Und hier ist die Panel-Ansicht:



## Weitere Signal-Typen

### Fließkomma-Signale

Der Signal-Typ, der bei der digitalen Signalverarbeitung auf modernen Personal Computern am häufigsten vorkommt, ist der Typ Fließkomma (Floating Point, kurz Float). Mit Fließkomma-Zahlen lassen sich Werte bis  $10^{38}$  (im 32-Bit-Modus) oder sogar  $10^{308}$  (im 64-Bit-Modus) ausdrücken. So praktisch das ist, einen Haken gibt es: die begrenzte Genauigkeit. Im 64-Bit-Modus ist die Genauigkeit zwar höher, aber immer noch beschränkt.

---

Die Begrenzung der Genauigkeit von Fließkomma-Werten hat technische Gründe: Wenn diese Werte nicht begrenzt würden, wäre eine unendlich große Menge Speicher für ihre Verarbeitung erforderlich. Sie kennen das Problem vielleicht vom Umgang mit Zahlen wie  $\pi$ , die Sie auch nicht in voller Länge auf ein endliches Blatt Papier schreiben können. Selbst wenn Ihnen die komplette Ziffernfolge bekannt wäre (was natürlich bei einer unendlichen Zahl von Stellen unmöglich ist, aber mal angenommen...), geht Ihnen an irgendeiner Stelle das Papier aus. Abgesehen davon würde das Verarbeiten solcher Zahlen auch eine unendlich schnelle CPU erfordern.

---

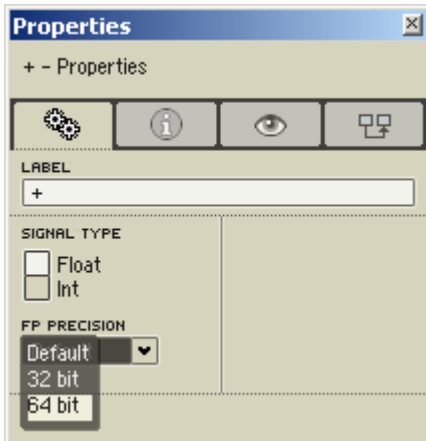
Für die Darstellung der Signale und für die Speicherverwaltung, die wir bisher kennengelernt haben, kommen 32-Bit-Fließkomma-Zahlen zum Einsatz. REAKTOR Core bietet Ihnen die Möglichkeit, 64-Bit-Fließkomma-Zahlen zu verwenden, wenn Sie eine höhere Genauigkeit oder einen größeren Wertebereich benötigen (wobei man sich kaum vorstellen kann, dass der Bereich zwischen  $10^{-38}$  und  $10^{38}$  nicht ausreichen soll).

---

Standardmäßig werden alle Berechnungen in REAKTOR Core mit 32-Bit-Fließkomma-Genauigkeit durchgeführt. Das heißt nicht zwingend, dass die Signale wirklich als 32-Bit-Floats berechnet werden, sondern dass mindestens 32-Bit-Fließkomma-Zahlen bei der Verarbeitung verwendet werden (obwohl mitunter auch 64-Bit-Floats für Zwischenergebnisse verwendet werden).

---

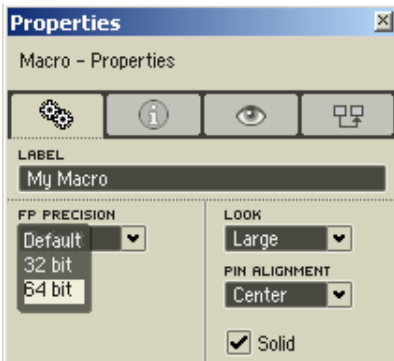
Sie können die Fließkomma-Genauigkeit sowohl für einzelne Module als auch für ganze Macros festlegen. Um die Fließkomma-Genauigkeit für ein einzelnes Modul zu ändern, öffnen Sie das Properties-Fenster des Moduls und wählen Sie einen Eintrag aus dem Menü *FP Precision* (Floating Point Precision):



- |                |  |
|----------------|--|
| <i>default</i> | bedeutet, dass die Default-Genauigkeit der aktuellen Struktur verwendet wird, welche das auch immer ist. |
| <i>32 bit</i>  | bedeutet, dass mindestens eine 32-Bit-Genauigkeit verwendet wird.  |
| <i>64 bit</i>  | bedeutet, dass mindestens eine 64-Bit-Genauigkeit verwendet wird.  |

Das Verändern der Genauigkeit für ein Modul führt dazu, dass die Verarbeitung innerhalb dieses Moduls fortan mit der angegebenen Genauigkeit durchgeführt wird und dass der Ausgangs-Wert ebenfalls unter Verwendung des angegebenen Genauigkeit erzeugt wird.

Die Standard-Genauigkeit für ganze Strukturen ändern Sie, indem Sie einen Rechts-Klick in den Hintergrund ausführen und aus dem Menü den Eintrag *Owner Properties* auswählen, der die Eigenschaften des Eigentümer-Moduls aufruft:



Die hier getroffene Einstellung legt die Genauigkeit für alle Module einschließlich Macros innerhalb der betreffenden Struktur fest, sofern für die Module keine abweichende Genauigkeit spezifiziert oder (im Fall von Macros) keine Standard-Genauigkeit für die eingeschlossenen Strukturen festgelegt ist.

---

Die normalen Fließkomma-Signale mit 32 und 64 Bit Genauigkeit sind voll miteinander kompatibel und können entsprechend beliebig untereinander verbunden werden. OBC-Signale mit unterschiedlicher Genauigkeit sind nicht miteinander kompatibel (weil es keinen Speicherinhalt geben kann, der gleichzeitig in 32 Bit und in 64 Bit vorliegt). Außerdem werden im Fall von OBC-Signalen die Einstellungen "Default", "32 bit" und "64 bit" als unterschiedlich und deshalb inkompatibel behandelt, weil die effektive Standard-Genauigkeit durch Ändern der entsprechenden Eigenschaft für eins der übergeordneten Macros verändert werden kann.

---



---

Die Eingangs- und Ausgangs-Module der Top-Level-Strukturen von Core Cells senden und empfangen immer im 32-Bit-Fließkomma-Format, weil dieser Signal-Typ für die Event- und Audio-Verbindungen auf REAKTORs Primary Level verwendet wird.

---

## Integer-Signale

Es gibt noch einen anderen Signal-Typ, der von modernen CPUs umfassend unterstützt wird, und dieser Typ spielt in der digitalen Welt eine noch größere Rolle als Fließkomma-Zahlen. Es handelt sich dabei um den Typ Integer (Ganzzahl). Integer-Zahlen werden mit unendlicher Genauigkeit dargestellt und verarbeitet.

Obwohl die Genauigkeit von Integer-Zahlen unendlich ist, ist der Darstellungsbereich auch für Werte dieses Typs begrenzt. Die Obergrenze für 32-Bit-Integer-Werte liegt oberhalb von  $10^9$ .

---

Unendliche Genauigkeit für die Speicherung und Verarbeitung von Integer-Werten ist möglich, weil Werte dieses Typs keine Dezimalstellen nach dem Komma besitzen, sodass man sie mit einer endlichen Anzahl von Stellen beschreiben kann. Schreiben Sie zum Beispiel die Anzahl der Sekunden einer Stunde auf: 3, 6, 0, 0 – fertig. So leicht ist das. Wenn Sie dagegen die Zahl  $\pi$  aufschreiben wollen, schaffen Sie das niemals in voller Länge – 3, 1, 4, 1, Stopp. Nicht ganz vollständig? Okay, lassen Sie uns noch ein paar Stellen aufschreiben: 5, 9, Stopp. Immer noch nicht vollständig. Dieses Spiel können Sie beliebig fortsetzen.. Eine Integer-Zahl dagegen können Sie komplett und präzise aufschreiben: 3600, das reicht.

---

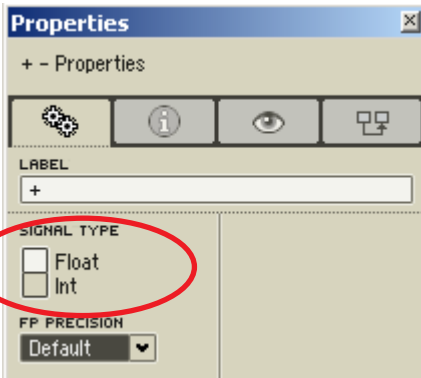
Während der Typ Fließkomma eine natürliche Wahl für Werte ist, die sich fortlaufend verändern (beispielsweise Audio-Signale), sind Integer-Werte für sich diskret ändernde Werte (z. B. Zähler) die besser passende Wahl. Viele der Module in REAKTOR Core können Sie in den Integer-Modus schalten, in dem die Module Integer-Signale am Eingang erwarten, diese als Integer (das heißt mit unendlicher Genauigkeit) verarbeiten und am Ausgang das Ergebnis als Integer-Wert bereitstellen. Zu diesen Modulen gehören beispielsweise arithmetische Module wie Addierer, Subtrahierer und Multiplizierer. Es gibt sogar einige Module, die sich nur mit Integer-Werten verwenden lassen.

---

Die minimale Wortlänge für Integer-Werte in REAKTOR Core beträgt 32 Bit.

---

Zwischen den Typen Float und Integer umschalten können Sie (sofern das Modul dies unterstützt) in der Eigenschaft Signal Type im Properties-Fenster des Moduls:



Ein Modul, das in den Integer-Modus geschaltet ist, wird die ankommenden Werte als Integer-Zahlen verarbeiten und Integer-Ausgangs-Werte erzeugen. Dass sich ein Modul im Integer-Modus befindet, erkennen Sie daran, dass seine Signal-Ein- und Ausgänge anders aussehen:



Es gibt keinen Standard-Signal-Typ für Macros. Der Grund dafür ist, dass Sie Strukturen, die Integer-Werte verarbeiten sollen, normalerweise nicht genauso aufbauen würden wie Strukturen, die zur Verarbeitung von Fließkomma-Zahlen gedacht sind, und umgekehrt – das kommt eigentlich nur bei einigen sehr einfachen Strukturen vor.

---

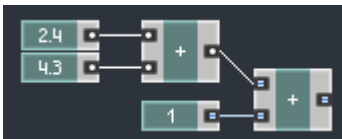
Integer-Signale können Sie frei mit Fließkomma-Signalen verbinden, allerdings bei solchen Verbindungen zwischen unterschiedlichen Typen eine Signal-Konvertierung durchgeführt, die CPU-Last erzeugt. Diese Konvertierung macht sich auf den zur Zeit der Erstellung dieses Handbuchs aktuellen PCs etwas, auf aktuellen Macs deutlich stärker bemerkbar. OBC-Anschlüsse unterschiedlichen Typs sind überhaupt nicht miteinander kompatibel.

Durch die Signal-Konvertierung können Informationen verloren gehen. Besonders große Integer-Zahlen können nicht exakt durch Fließkomma-Zahlen ausgedrückt werden; ebenso lassen sich Fließkomma-Werte nicht genau als Integer-Zahlen wiedergeben. Große Fließkomma-Werte (größer als der größte darstellbare Integer-Wert) lassen sich gar nicht im Integer-Format darstellen, weshalb die Konvertierung in solchen Fällen undefiniert ist. Während der Konvertierung von Float nach Integer

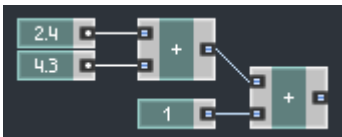
werden die Werte *näherungsweise auf den nächsten Integer-Wert* gerundet. “Näherungsweise” bedeutet, dass das Ergebnis einer Rundung des Float-Werts 0,5 entweder 0 oder 1 sein kann, wobei Sie ziemlich sicher sein können, dass 0,49 auf 0 abgerundet, 0,51 dagegen auf 1 aufgerundet wird.

---

Es ist wichtig, sich vor Augen zu führen, dass das Umschalten des Verarbeitungs-Modus‘ auf Integer nicht zu denselben Ergebnissen führt wie die Konvertierung des Fließkomma-Ergebnisses derselben Operation. Lassen Sie uns zur Verdeutlichung ein Beispiel ansehen. Hier addieren wir zwei Fließkomma-Werte, nämlich 2,4 und 4,3. Das Ergebnis ist natürlich 6,7, was nach der Konvertierung in Integer den Wert 7 ergibt. Die Ausgabe der folgenden Struktur lautet also 8:



Wenn wir nun den Modus des ersten Addierers auf Integer umschalten, werden statt der Werte 2,4 und 4,3 die gerundeten Werte 2 und 4 addiert, wodurch sich als Ergebnis 6 ergibt. Die Ausgabe der Struktur lautet also 7:



Clock-Eingänge ignorieren die ankommenden Werte vollständig, deshalb sind sie normalerweise immer vom Typ Float. Die Signal-Konvertierung wird nicht für Signale durchgeführt, die nur als Clock dienen:



Hier befindet sich der Clock-Eingang des Moduls Read immer noch im Float-Modus, obwohl das Modul in den Integer-Modus geschaltet wurde (die OBC-Ports sehen immer gleich aus, unabhängig davon, ob sie vom Typ Float oder Integer sind).



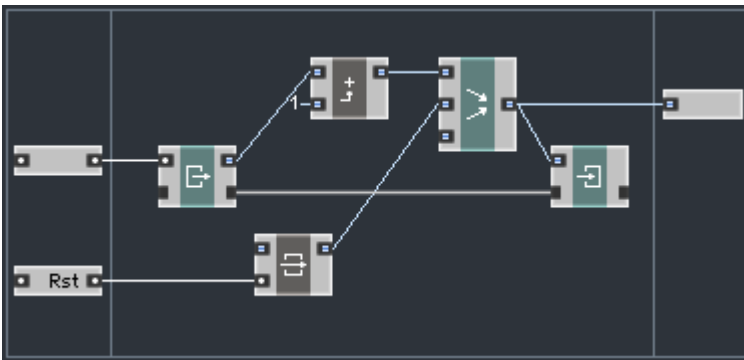
---

Integer-Feedback wird automatisch auf dieselbe Art aufgelöst wie Fließkomma-Feedback – indem ein Integer-Mode-Modul des Typs  $Z^{-1}$  eingesetzt wird (wobei hier natürlich keine Denormalen-Unterdrückung notwendig ist).

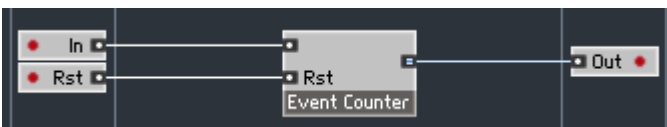
---

## Wie Sie einen Event-Zähler bauen

Lassen Sie uns ein Event-Zähler-Modul bauen. Die Funktion dieses Moduls wird der des Event-Akkumulierers sein, aber anstatt die Werte der Events zu summieren, werden wir die Events diesmal nur zählen. Der Signal-Typ Integer scheint eine logische Wahl für den Zählvorgang zu sein:

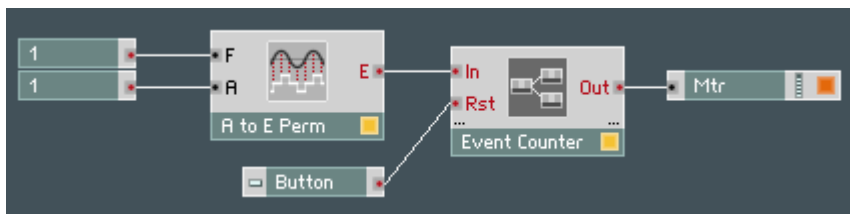


Ausgangs-Modul und alle eingebauten Module befinden sich im Integer-Modus. Das Macro *lLatch* setzt anstelle eines normalen Latch-Macros die Schaltung zurück. Es macht genau dasselbe wie ein Latch (und ist auch im selben Menü zu finden), arbeitet aber mit Integer-Signalen. Außerdem kommt ein Integer-Modulations-Macro zum Einsatz (das Sie unter *Expert Macro > Modulation > Integer* finden). Die beiden Eingänge müssen Sie nicht in den Integer-Modus schalten, sie liefern nur Clock-Signale. Sehen Sie sich nun die Struktur unserer Event-Core-Cell an, die dieses Macro enthält:



Wie Sie sehen, befindet sich der Ausgang dieses Moduls nicht im Integer-Modus (und lässt sich auch gar nicht in diesen Modus versetzen). Das kommt

daher, dass die Core Cell nach außen ein Primary-Level-Modul ist und ein normales Primary-Level-Event vom Typ Float liefern muss. Hier sehen Sie eine Test-Struktur für unser Zähler-Modul:



Und das Panel wird voraussichtlich so aussehen:



## Wie Sie einen Flanken-Zähler bauen

Nun lernen Sie eine Technik für den *Vorzeichen-Vergleich* kennen, die Sie manchmal beim Aufbau von REAKTOR-Core-Strukturen brauchen werden. “Vorzeichen-Vergleich” beschreibt eine besondere Art des Vergleichens von zwei Zahlen, bei der Sie den Wert der Zahlen ignorieren und nur die jeweiligen Vorzeichen beachten (Plus oder Minus). Natürlich ist auch hier Plus höherwertig als Minus. Hier ist ein Beispiel:

- 3.1 ist vorzeichengrößer als -1.4
- 2.1 ist vorzeichengleich mit 5.0
- 4.5 ist vorzeichengleich mit -2.9

---

Beachten Sie, dass das Vorzeichen von Null undefiniert ist. Das bedeutet, dass das Ergebnis jedes Vorzeichen-Vergleichs, an dem der Wert Null beteiligt ist, beliebig sein kann.

---

Natürlich hätten Sie den Vorzeichen-Vergleich auch unter Verwendung einiger Vergleichs-Module und einiger Router implementieren können, aber die Lösung hier ist viel effizienter. Sie können den Vorzeichen-Vergleich in REAKTOR-Core-Strukturen nämlich mit dem Modul Compare Sign durchführen (*Built In Module > Flow > Compare Sign*):

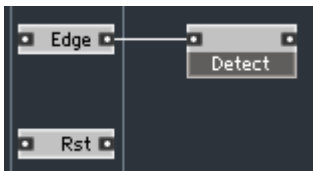


Das Modul erzeugt an seinem Ausgang ein Signal des Typs *BoolCtl*, sodass Sie es mit einem Router verbinden können.

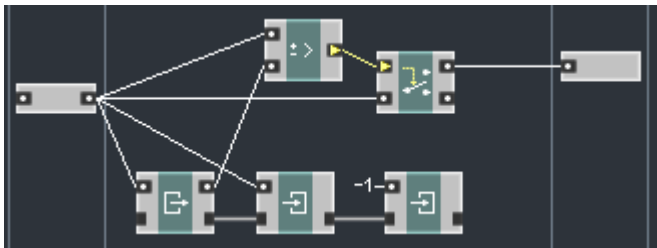
Ein mögliches Einsatzgebiet für ein solches Modul wäre, die steigenden Flanken eines ankommenden Signals zu erkennen. Einen Zähler für diese steigenden Flanken werden wir nun als REAKTOR-Core-Macro aufbauen:



Achten Sie darauf, dass sich der Ausgang im Integer-Modus befindet, weil die Zählung in Ganzzahlen stattfindet. Zuerst brauchen wir ein Flanken-Detektor-Macro, das die erkannten Flanken in ein Event konvertiert:



So könnte unser Detektor-Macro aussehen:



Die OBC-Kette unten enthält den vorigen Wert des Eingangssignals. Wie Sie sehen können, wird der neue Wert gespeichert, nachdem der alte Wert gelesen wurde. Das letzte Write-Modul in der Kette übernimmt die Initialisierung des vorigen Werte-Speichers. Wir initialisieren den Speicher mit -1, sodass der erste positive Wert als steigende Flanke gezählt wird.

---

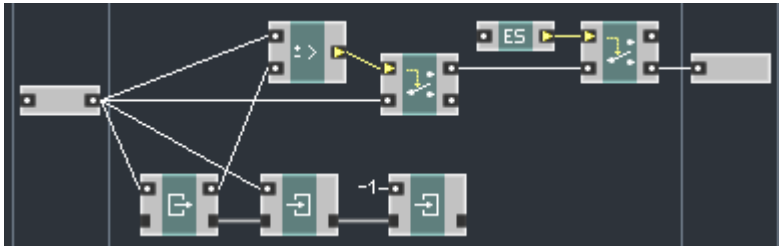
Ein Write-Modul am Ende der OBC-Kette ist eine andere Möglichkeit (als Merge), den Speicher zu initialisieren. Dabei muss das Write-Modul als letztes Glied der Kette angeordnet sein, um die von den stromaufwärts liegenden Write-Modulen gespeicherten Ergebnisse überschreiben zu können.

---

Der vom Modul Sign Comparison kontrollierte Router arbeitet als Event-Gate und lässt nur die Events durch, bei denen ein Vorzeichenwechsel von negativ in positiv erfolgt.

IES ist nicht eindeutig, ob ein solches Modul während der Initialisierung ein

Event sendet oder nicht. Das liegt daran, dass der Speicher zum Zeitpunkt der Verarbeitung des Initialisierungs-Events immer noch den Wert Null hat und das Vorzeichen von Null undefiniert ist. Wir können die Struktur modifizieren, um das Senden eines Events während der Initialisierung zu verhindern:



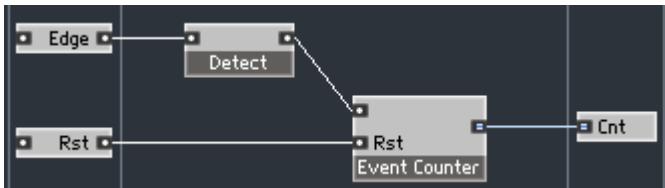
Das Modul *ES Ctl* ist ein “Event-empfindlicher Regler”. Das von diesem Modul erzeugte Kontroll-Signal ist nur dann “wahr”, wenn am Eingang des Moduls ein ankommendes Event vorhanden ist. Weil dieser Eingang in der obigen Struktur nicht angeschlossen ist und deshalb eine Null-Konstante empfängt, nimmt sein Signal nur zum Zeitpunkt der Initialisierung den Wert “wahr” an. Der zweite Router wird also alle während der Initialisierung auftretenden Events abblocken und alle anderen durchlassen.

---

Beachten Sie, dass wir es hier mit einem Modul zu tun haben, dessen Ausgang während der Initialisierung kein Event sendet.

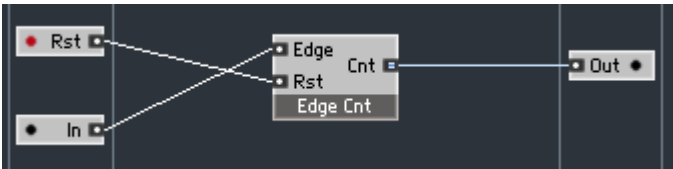
---

Nun haben wir ein Detektor-Modul, das wir mit der Zähler-Schaltung verbinden können, die wir schon aufgebaut haben:

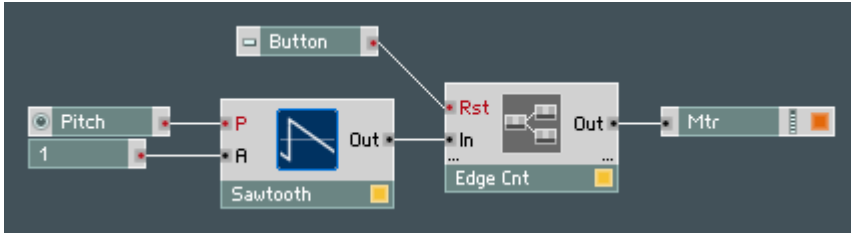


Die Funktion der obigen Schaltung sollte einleuchten. Das Modul *Detect* sendet jedes Mal ein Event, wenn es eine steigende Flanke erkennt; diese Events zählt das Modul *Evt Cnt*.

Um unsere Schaltung zu testen, fügen wir dieses Macro in eine Audio-Core-Cell ein und zählen die steigenden Flanken einer Sägezahn-Wellenform. Die interne Struktur der Core Cell wird folgendermaßen aussehen:



Und hier die Primary-Level-Test-Struktur:



Vergessen Sie nicht, die Eigenschaften für das Modul *Mtr* (Meter) wie in den vorangegangenen Beispielen einzustellen.

Das hier sollten Sie nun in der Panel-Ansicht sehen:



Die Geschwindigkeit der Werteänderungen im Anzeigefeld korrespondiert mit der Frequenz des Oszillators, die Sie mit dem Regler *Pitch* einstellen können. Bei einem *Pitch*-Wert von Null beträgt die Oszillator-Frequenz ungefähr 8 Hz, sodass sich der Wert im Anzeigefeld mit einer Rate von acht Schritten pro Sekunde ändern sollte.

# Arrays

## Einführung in das Thema “Arrays”

Lassen Sie uns annehmen, Sie wollen einen Audio-Signal-Wahlschalter bauen, der abhängig von dem am Kontroll-Eingang anliegenden Wert das Signal von einem der vier Audio-Signal-Eingänge entgegennimmt:



Diese Funktion ließe sich mit Router-Module umsetzen, aber es gibt auch noch eine andere Möglichkeit. Wir verwenden nämlich lieber die in REAKTOR Core verfügbaren *Arrays*.

Ein *eindimensionales Array* ist eine *geordnete Sammlung* von Daten-Einträgen *desselben Typs*, die sich über ihre Position in dieser Ordnung (oder diesem *Index*) adressieren lassen. Nehmen wir zum Beispiel die folgende Gruppe von Fließkomma-Zahlen:

5.2      16.1      -24.0      11.9      -0.5

In REAKTOR Core sind die Array-Element-Indizes Null-basiert; das bedeutet, dass das erste Element des Arrays einen Index von 0 hat. Deshalb ist für unsere Zahlengruppe ein Index von 0 dem Wert 5,2 zugeordnet, ein Index von 1 entspricht 16,1, der Index 2 adressiert -24,0, der Index 3 ist dem Wert 11,9 zugewiesen und Index 4 verweist auf den Wert -0,5.

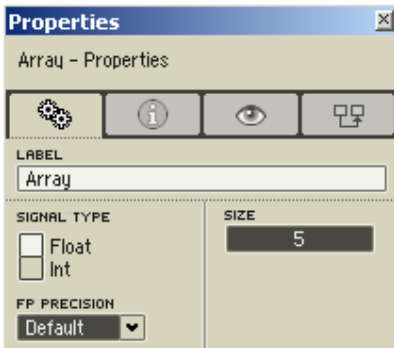
Hier sehen Sie eine tabellarische Darstellung des beschriebenen Arrays:

Index	0	1	2	3	4
Value	5.2	16.1	-24.0	11.9	0.5

Arrays erzeugen Sie in REAKTOR Core durch Verwendung der *Array-Module* (*Built In Module > Memory > Array*):



Ein Array-Modul hat einen einzelnen Ausgang vom Typ *Array OBC*. Die Größe des Arrays (die Anzahl der Elemente) und den Typ der in dem Array aufbewahrten Daten können Sie im Properties-Fenster des Array-Moduls angeben:



Zum Beispiel müssen wir für die oben abgebildete Tabelle mit fünf Elementen im Feld *Size* eine Größe von 5 eintragen. Da es sich bei den Elementen unseres Arrays um Fließkomma-Zahlen handelt, müssen wir außerdem den Signal-Typ *Float* auswählen

---

Weil Array-Indizes in REAKTOR Core Null-basiert sind, umfasst der Index-Bereich für ein Array der Größe 5 von 0 bis 4 (wie Sie auch in der oben abgebildeten Tabelle sehen können).

---

---

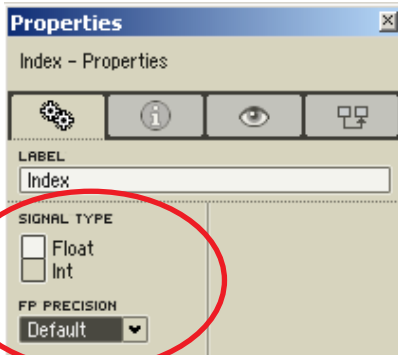
Array-OBC-Signale, die mit Elementen unterschiedlichen Daten-Typs korrespondieren, sind natürlich nicht miteinander kompatibel.

---

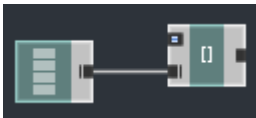
Um ein Array-Element zu adressieren, müssen Sie einen Index anlegen. Dazu stellt REAKTOR Core das Modul *Index* (*Built In Module > Memory > Index*) zur Verfügung:



Der Master-OBC-Eingang des Index-Moduls (unten) sollte mit dem Slave-Ausgang eines Array-Moduls verbunden sein. Der Verbindungs-Typ des Master-Eingangs sollte mit dem Array-Typ übereinstimmen; Sie können ihn im Properties-Fenster des Moduls *Index* einstellen:



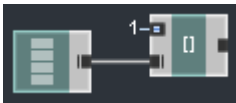
Und nun die Verbindung:



Der obere Eingang des Index-Moduls ist immer vom Typ Integer und nimmt den Index-Wert entgegen. Hier sehen Sie, wie wir das Array-Element mit dem Index von 1 adressieren:



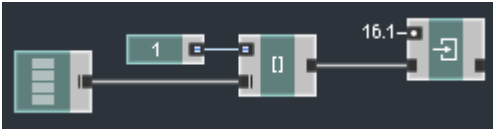
Beachten Sie, dass sich das Konstanten-Modul ebenfalls im Integer-Modus befindet (was Sie am Aussehen des Ausgangs-Ports erkennen). Das ist zwar nicht unbedingt notwendig, weil andernfalls eine automatische Konvertierung nach Integer durchgeführt werden würde, aber es sieht besser aus. Alternativ hätten wir auch eine QuickConst verwenden können:



Der Ausgang des Index-Moduls hat den Typ *Latch OBC*. Das bedeutet, dass Sie Module der Typen *Read* und *Write* (und davon sogar mehrere) mit diesem Ausgang verbinden können. Dabei müssen Sie natürlich darauf achten, dass die Read- und Write-Module auf denselben Daten-Typ gesetzt sind wie die Module *Array* und *Index*.



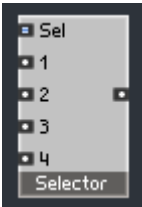
Hier sehen Sie, wie das Array-Element mit dem Index von 1 auf den Wert 16,1 initialisiert wird:



Wenn ein außerhalb des definierten Bereichs liegender Index an das Index-Modul gesendet wird, ist das Ergebnis des Zugriffs auf das Array undefiniert. Die Struktur wird nicht abstürzen, aber es ist in diesem Fall unklar, auf welches Element des Arrays in diesem Fall zugegriffen wird und ob überhaupt ein Zugriff stattfindet. Das bedeutet, dass Sie im Zweifelsfall den Wertebereich des Index' unter Verwendung von Router-Modulen oder Macros aus der Library beschneiden sollten.

## Wie Sie einen Audio-Signal-Wahlschalter bauen

Lassen Sie uns zum Aufbau des Audio-Signal-Wahlschalters zurückkehren:



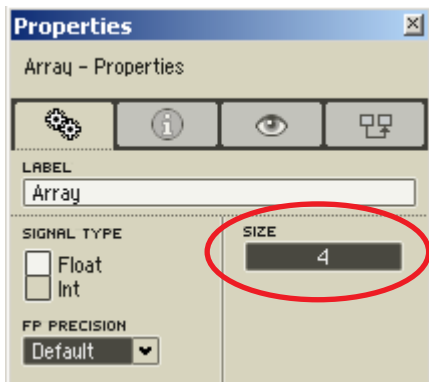
Hier sehen Sie eine leere interne Struktur für dieses Modul:



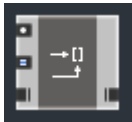
Wir werden ein Array mit vier Fließkomma-Elementen zum Speichern unserer Audio-Signale verwenden:



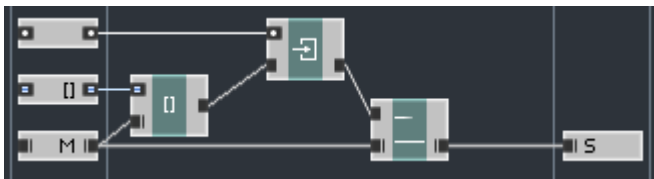
Hier ist das Properties-Fenster des Array-Moduls:



Um die Eingangs-Werte in das Array zu schreiben, sollten wir das Standard-Macro *Write []* (*Expert Macro > Memory > Write []*) verwenden:



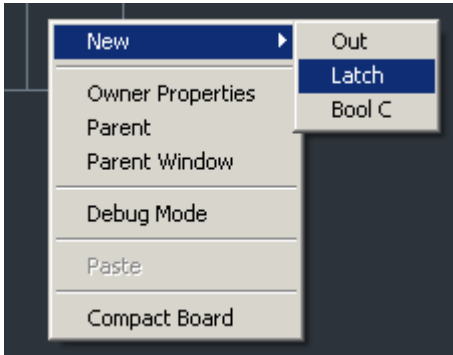
Dieses Macro besitzt ein internes Index-Modul und ein Write-Modul, das mit einem festgelegten Index in das Array-Element schreibt:



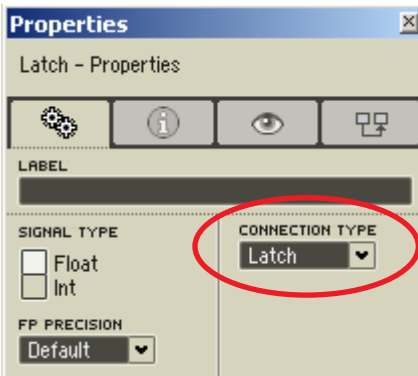
Der obere Eingang nimmt natürlich den Wert entgegen, der geschrieben werden soll. Der Eingang “[]” empfängt den Index, an dem der Schreibvorgang stattfinden soll. Der Eingang “M” stellt via OBC die Verbindung zu einem Float-Array mit Standard-Genauigkeit her, der Ausgang “S” ist eine “durchgehende” Verbindung, ähnlich wie bei den anderen OBC-Modulen, etwa *Read* und *Write*.

Der Eingang “M” und der Ausgang “S” gehören zu einer anderen Sorte von Macro-Anschlüssen, die sich von der unterscheidet, die wir bisher verwendet

haben. Solche Ports können Sie über den Eintrag “Latch” aus dem Port-Menü erzeugen (der dritte Eintrag erzeugt einen Macro-Port vom Typ *Bool/Ctl*):



Die Latch-Ports können Sie für Latch-OBC-Verbindungen (zwischen Modulen der Typen Read und Write) oder für Array-OBC-Verbindungen verwenden. Diese Auswahl können Sie im Properties-Fenster des Ports treffen:



Indem Sie die Verbindungs-Typ auf “Latch” oder “Array” setzen, definieren Sie den OBC-Modus entsprechend. Für die Ports des Macros Write [] müssen Sie hier offensichtlich “Array” einstellen.

Das Modul mit den beiden horizontalen Linien heißt *R/W Order (Built In Module > Memory > R/W Order)*:



Es lässt nur die Verbindung an seinem Master-Eingang (unten) zu seinem

Slave-Ausgang durch, weiter tut es nichts. Der obere Eingang hat absolut keine Wirkung, weil aber an diesem Eingang eine Verbindung anliegt, beeinflusst er doch die Verarbeitungsreihenfolge der Module. Deshalb *wird alles, was an den Ausgang "S" des Macros angeschlossen ist, nach dem Write-Modul verarbeitet*, was nicht der Fall wäre, wenn das Modul *R/W Order* in dieser Struktur fehlen würde.

---

Wenn das Modul *R/W Order* fehlen würde, wäre die Funktion des Macros *Write []* nicht sehr zuverlässig oder nachvollziehbar, weil Sie als Anwender natürlich erwarten, dass alles an den Ausgang "S" des Macros *Write []* Angeschlossene nach diesem Macro verarbeitet wird. Im Allgemeinen entsteht dieses Problem nur bei Verbindungen des Typs OBC; setzen Sie in diesem Fall R/W-Order-Module an den entsprechenden Stellen in den Macros, die Sie entwerfen, ein.

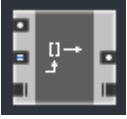
---

Das Modul *R/W Order* besitzt, ähnlich wie die OBC-Ports, eine Eigenschaft namens *Connection Type*. Bei diesem Modul kontrolliert die Eigenschaft *Connection Type* nur den Typ der Ports "M" und "S", der "Sidechain"-Eingang befindet sich immer im Modus "Latch". Weitere Details finden Sie im Anhang in der Beschreibung des Moduls *R/W Order*.

Lassen Sie uns nun eine Schaltung aufbauen, die die Eingangssignale in das Array schreibt:



Die vier Module des Typs *Write []* sorgen dafür, dass die ankommenden Audio-Werte in das Array geschrieben werden. Nun brauchen wir einen Schaltkreis, um einen der vier Werte auszulesen. Dafür schlagen wir das Macro *Read []* (zu finden unter *Expert Macro > Memory > Read []*) vor:

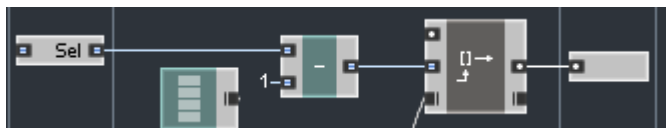


Dieses Macro liest ein Array-Element aus, dessen Index über den Integer-Eingang in der Mitte bestimmt wird. Der obere Eingang ist der Clock-Eingang für den Lese-Vorgang – er sendet als Reaktion auf ein ankommendes Event den gelesenen Wert an den oberen Ausgang des Moduls. Die unteren Ports sind natürlich Master- und Slave-Array-Verbindungen.

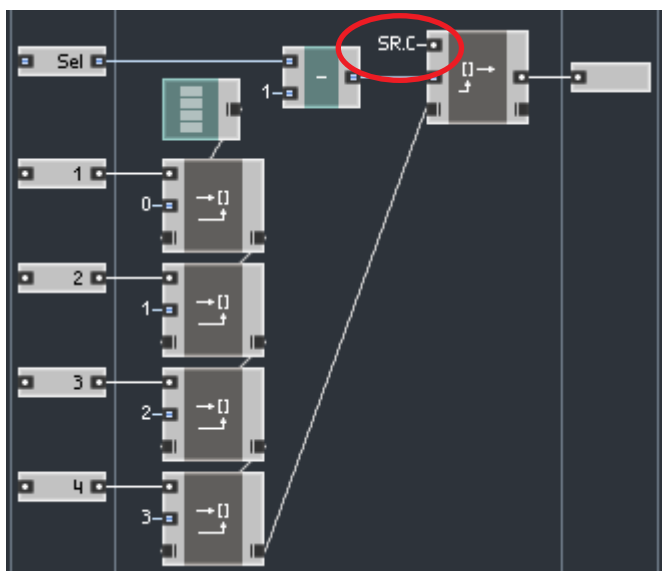
Aber womit verbinden wir nun den Master-Eingang? Anscheinend können wir ihn nicht direkt mit dem Array-Modul verbinden, weil der Lese-Vorgang nach allen Schreib-Vorgängen erfolgen muss (weil sonst eine Verzögerung von einem Sample entstehen kann oder auch nicht, alles in allem ist das jedenfalls ziemlich unzuverlässig). Wir können den Master-Eingag aber auch nicht direkt mit einem der Module des Typs *Write []* verbinden, weil das unser Problem nicht lösen würde. Daher schlagen wir vor, dass Sie die *Write []*-Module in Reihe schalten, sodass Sie das Modul *Read []* mit dem Ausgang des letzten Moduls vom Typ *Write []* verbinden können.



Und wo schließen wir den Index-Eingang des Moduls *Read []* an? Wenn wir wollen, dass unser Auswahl-Wert im Bereich zwischen 1 und 4 liegt, müssen wir den Wert des Eingangs “Sel” um 1 verringern. Beachten Sie, dass wir hier eine Integer-Subtraktion durchführen (weil “Sel” nur ein Kontroll-Eingang ist, brauchen wir hier wirklich kein Modulations-Macro):

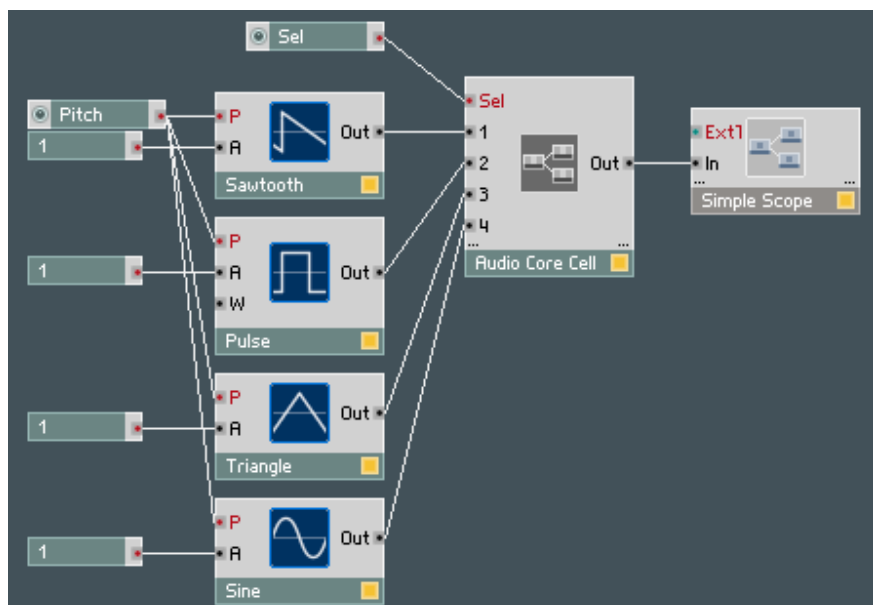


Zu guter Letzt müssen wir noch das Read-Modul durch die Sampling-Rate-Clock takten lassen (weil das hier ja ein Audio-Signal-Wahlschalter sein soll):



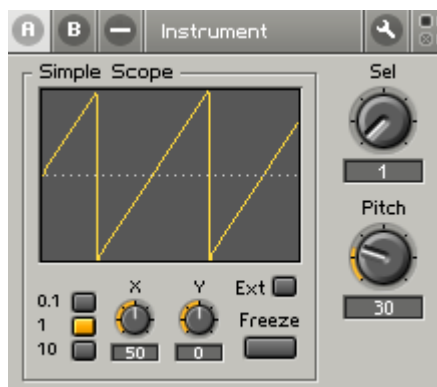
Normalerweise hätten wir auch auf das Beschneiden des Eingangs-Werts am Eingang “Sel” auf den korrekten Bereich achten sollen, aber der Einfachheit halber verzichten wir hier darauf.

Hier sehen Sie eine geeignete Test-Struktur (das Macro ist in eine Audio-Core-Cell verpackt):



Der Drehregler “Sel” ist so eingestellt, dass er zwischen den vier Werten von 1 bis 4 umschaltet.

Nun schalten Sie in die Panel-Ansicht um und sehen Sie sich an, wie sich die Wellenform abhängig von der Stellung des Drehreglers ändert:



## Wie Sie ein Delay aufbauen

Nun, da wir etwas Erfahrung mit Arrays haben, lassen Sie uns ein einfaches Audio-Delay-Macro aufbauen. Das Modul sollte wie folgt aussehen:

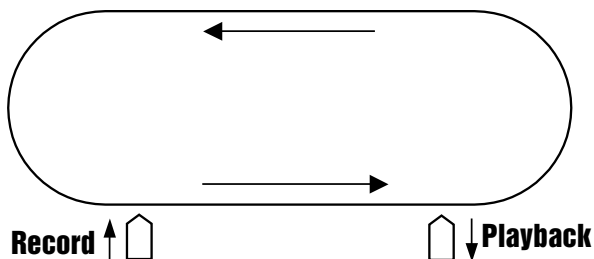


Noch besser ist, wenn es so aussieht (wofür wir die Eigenschaft *Port Alignment* des Macros in “top” ändern müssen):



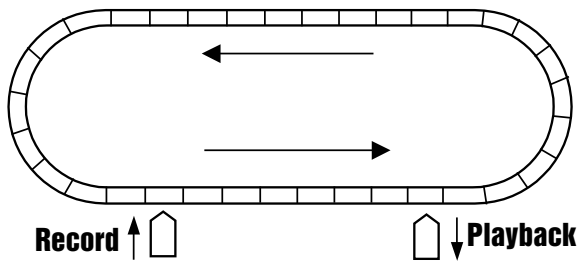
Der Eingang “T” nimmt die Delay-Zeit in Sekunden entgegen.

Wenn Sie sich einmal ein analoges Bandocho-Gerät öffnen, sehen Sie darin eine Tonbandschleife in Kombination mit einem Aufnahmekopf (Record) und einem Wiedergabekopf (Playback). Genau genommen gibt es auch noch einen Löschkopf, aber um unser Beispiel einfach zu halten, nehmen wir an, dass der Aufnahmekopf sowohl für das Aufnehmen als auch für das Löschen zuständig ist.



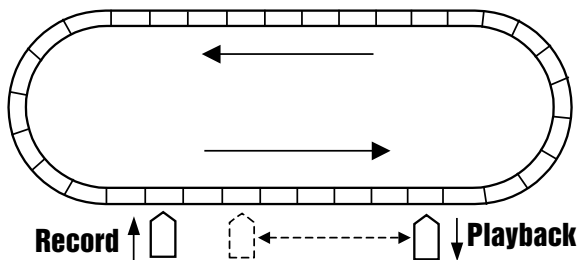
Wenn wir nun diese Konstruktion in digitaler Form simulieren wollen, brauchen wir eine Art digitaler Bandschleife. Wegen der diskreten Natur der digitalen Welt nimmt die digitale “Bandschleife” eine endliche Zahl von Audio-Samples auf. Diese Samples werden mit der Audio-Sampling-Rate aufgenommen und wiedergegeben. In dem folgenden Bild nimmt jedes Segment der Bandschleife ein Audio-Sample auf:





Die nächstliegende Wahl für eine digitale Bandsschleife ist ein Array, wobei die Größe des Arrays der Gesamtanzahl der in der Bandschleife aufgenommenen Samples entsprechen muss.

Bei einem analogen Bandecho hängt die Delay-Zeit von zwei Parametern ab: von dem (unveränderlichen) Abstand zwischen dem Aufnahme- und dem Wiedergabekopf und von der (veränderlichen) Bandgeschwindigkeit. Diese Lösung hat technische Gründe – es ist viel einfacher, die Laufgeschwindigkeit des Tonbands zu verändern, als den Abstand zwischen den Köpfen. Im Fall unserer digitalen Simulation ist es allerdings umgekehrt, weil eine Variation der Bandgeschwindigkeit einer Sampling-Raten-Konvertierung zwischen dem “digitalen Tonband” und dem Ausgang entsprechen würde, während wir den Abstand zwischen den “Köpfen” relativ einfach verändern können – also machen wir das:



Es gibt noch einen weiteren Unterschied zur analogen Welt: In der analogen Welt bewegt sich die Bandschleife. Wenn wir unser digitales Tonband bewegen wollen würden, müssten wir *mit jedem Audio-Takt-Signal* alle Array-Elemente auf ihre Nachbarpositionen kopieren, was ziemlich aufwendig wäre. Wir werden also stattdessen die Köpfe bewegen.

Aus dem oben Gesagten können wir ableiten, dass wir für unsere Bandecho-Simulation folgende Bauteile brauchen:

- array – um unsere “digitale Bandschleife” zu simulieren
- write index – dies ist unser Aufnahmekopf
- read index – dies ist unser Wiedergabekopf

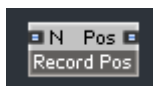
Schreib- und Lese-Index werden sich Sample für Sample durch das Array bewegen. In dem Moment, in dem einer von beiden das Ende des Arrays erreicht, soll der betreffende Index auf den Anfang des Arrays zurückgesetzt werden (was dem Zusammenkleben der offenen Enden eines Tonband-Streifens zu einer Schleife entspricht). Der Versatz zwischen der Schreib- und der Lese-Position ist von der Delay-Zeit (in Samples) abhängig.

---

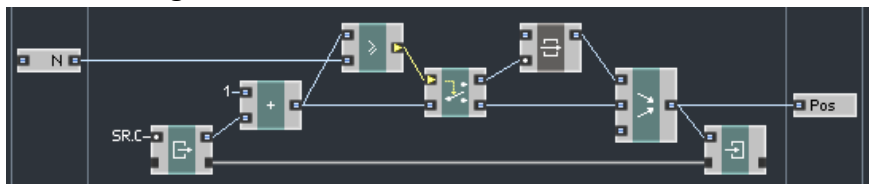
Diese Technik ist beim Programmieren ziemlich gebräuchlich und heißt dort “Kreispuffer” oder “Ringpuffer”.

---

Wir beginnen mit der Konstruktion unseres “Aufnahmekopfs”. Diese Funktion ähnelt stark dem Sägezahn-Oszillator, den wir schon kennen, abgesehen davon, dass die Berechnungen im Integer-Modus stattfinden. Werte werden mit jedem Audio-Tick um 1 erhöht; der Bereich der Ausgangs-Werte ist mit 0 bis  $N-1$  festgelegt, wobei  $N$  die Größe des Arrays ist. Lassen Sie uns den Schaltkreis, der den Schreib-Index berechnet, in ein Macro namens “Record-Pos” verpacken:

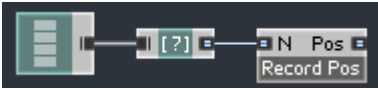


Der Eingang “N” sollte die Anzahl der Elemente im Array entgegennehmen, der Ausgang “Pos” gibt die aktuelle Schreib-Position (Index) aus. Dies ist eine Möglichkeit, dieses Macro aufzubauen (sehen Sie sich zum Vergleich noch einmal den Sägezahn-Oszillator an):

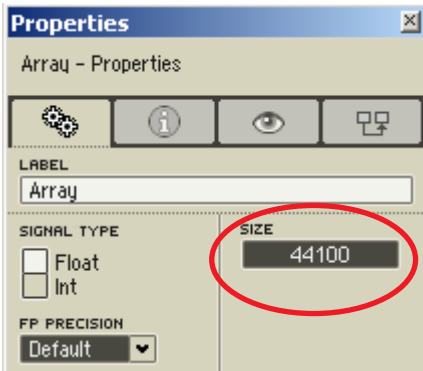


Beachten Sie, dass das Vergleichs-Modul auf “>=“ eingestellt ist. Das war für den Sägezahn-Oszillator nicht von Bedeutung (da konnten wir sowohl “>=“ als auch “>“ verwenden), aber in Integer-Berechnungen ist das normalerweise wichtig. Verwenden Sie also die Bedingung “>=“ um sicherzustellen, dass der Schreib-Index niemals den Wert  $N$  erreicht.

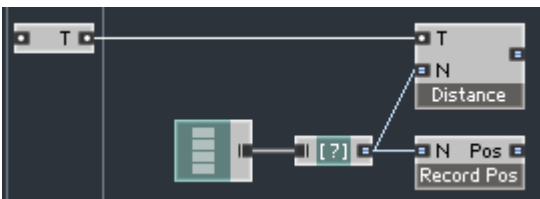
Auf der oberen Ebene erzeugen Sie ein Array-Modul und verbinden es über das Modul *Size []* (das Sie unter *Built In Module > Memory > Size []* finden und das die Größe des Arrays meldet) mit dem Macro “RecordPos”:



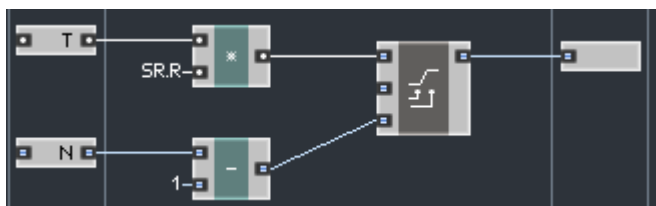
Die Eigenschaft *Size* des Arrays können Sie im Properties-Fenster auf den Wert 44100 einstellen. Dies ermöglicht uns ein Delay mit einer Länge von bis zu einer Sekunde (genau genommen ein Sample weniger) bei einer Sampling-Rate von 44,1 kHz:



Jetzt müssen wir den Schreib-Index berechnen. Das werden wir machen, indem wir zwei neue Macros aufbauen. Das erste Macro wird die gewünschte Delay-Zeit in eine Entfernung in Samples konvertieren:



Das geschieht, indem die Zeit in Sekunden mit der Sampling-Rate in Hz multipliziert wird. Wir sollten auch nicht vergessen, das Ergebnis passend zu beschneiden; dafür verwenden wir das Macro *Expert Macro > Clipping > IClipMinMax*:



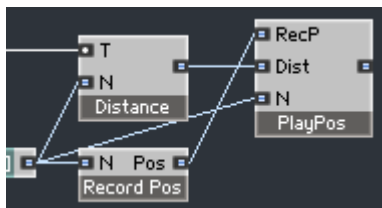
Wir stützen das Ergebnis auf N-1, weil dies die maximale Distanz zwischen zwei verschiedenen Array-Elementen ist. Beachten Sie, dass die Konvertierung nach Integer hinter der Multiplikation stattfindet.

---

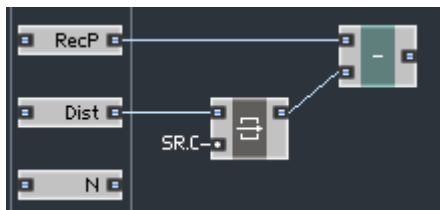
Alternativ hätten wir den Eingangs-Wert auf einen bestimmten Bereich beschneiden können, was meistens etwas günstiger ist, weil Fließkommawerte, die außerhalb des Bereichs der Integer-Darstellungsraums liegen, beliebige Integer-Werte erzeugen können, sodass wir nicht länger eine wirkliche Werte-Beschneidung durchführen.

---

Jetzt verwenden wir ein anderes Macro, um den Lese-Index aus den Werten von *RecordPos* und *Distance* zu ermitteln:



Offenbar muss die Abspiel-Position um die von *Distance* gespeicherte Anzahl von Samples hinter der Aufnahme-Position liegen, deshalb ziehen wir die eine von der anderen ab:

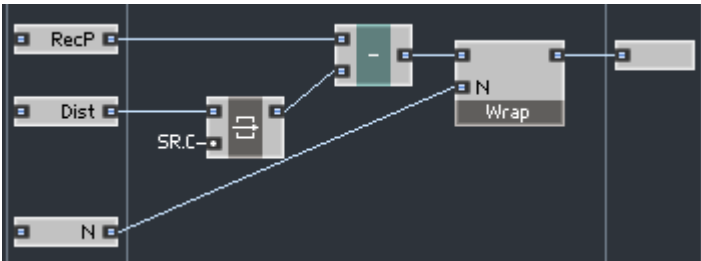


Der Entfernungswert wird zwischengespeichert, weil er einem Kontroll-Signal-Eingang entspringt, der prinzipiell jederzeit Events empfangen kann, und wir wollen ja nicht, dass die Subtraktion zu einem anderen Zeitpunkt stattfindet als beim Eintreffen eines Audio-Clock-Events.

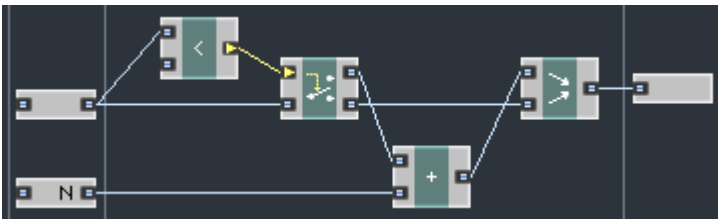
Wenn wir einfach subtrahieren, kann die Differenz allerdings leicht weniger als Null sein. Das kommt daher, dass unser Array keine Schleife ist – seine “Enden” sind nicht miteinander verbunden. Wir müssen das Ergebnis also “umbiegen”:

- 1 muss zu  $N-1$  werden,
- 2 muss zu  $N-2$  werden,
- 3 muss zu  $N-3$  werden,
- etc.

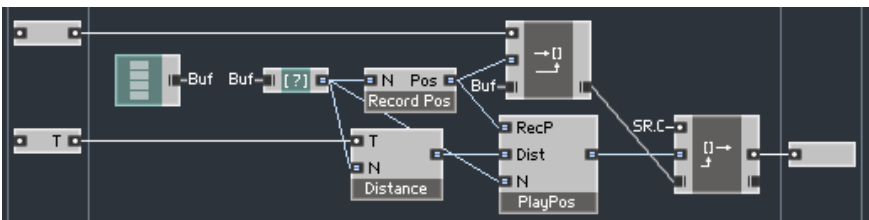
Also setzen wir für dieses “Umbiegen” (Wrapping) ein anderes Macro ein:



Weil wir wissen, dass die Differenzen nicht kleiner als  $-N+1$  werden können (weil *RecordPos* immer zwischen 0 und  $N-1$  liegt und *Distance* sich zwischen 0 und  $N-1$  bewegt), können wir das Wrapping als einfache Addition von  $N$  implementieren:

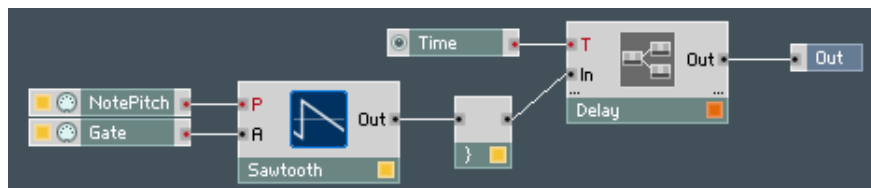


Lassen Sie uns nun zu unserer Top-Level-Struktur zurückkehren. Es scheint, als wir nun Schreib- und Lese-Indizes hätten, also müssen wir nur noch das Schreiben und Lesen durchführen lassen:



Beachten Sie, dass das Lesen nach dem Schreiben erfolgt und dass es von der Sampling-Rate-Clock getaktet wird.

Hier sehen Sie eine geeignete Test-Struktur. Vergessen Sie nicht, den Konverter *ms2sec* in die Delay-Core-Cell einzusetzen und auf monophonen Betrieb einzustellen:



---

Es ist tatsächlich eine gute Idee, das Delay so früh wie möglich in den monophonen Modus zu schalten, weil es für jede Stimme ungefähr 200 KByte Arbeitsspeicher belegen würde. 44100 Samples, von denen jedes 4 Byte (32 Bit) verwendet:  
 $44100 \times 4 = 176400$  Byte, etwas mehr als 172K (ein Kilobyte (KByte) hat 1024 Byte)

---

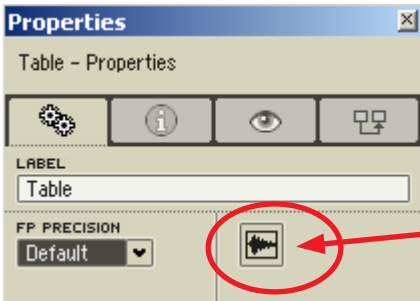
Um die obige Struktur zu testen, spielen Sie Noten auf Ihrem MIDI-Keybord und hören Sie, wie diese Noten um die mit dem Regler “Time” eingestellte Zeitspanne verzögert werden.

## Tabellen

Es gibt noch ein anderes Modul, das dem Array ähnlich ist. Dieses Modul heißt *Table* (Tabelle); Sie finden es unter *Built In Module > Memory > Table*:

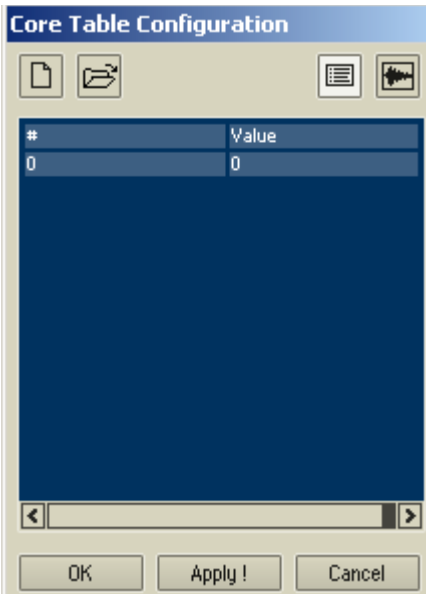


Die Ähnlichkeit ist, dass Tabellen Arrays einer bestimmten Art sind. Der Unterschied ist, dass Sie aus Tabellen nur lesen, aber nichts in die Tabellen hinschreiben können. Die Werte in einer Tabelle werden im Properties-Fenster des Table-Moduls vorinitialisiert. Um auf die Liste der Werte zuzugreifen, klicken Sie auf den Schalter im Properties-Fenster:



Klicken Sie hier um die Werte zu bearbeiten.

Ein neues Fenster sollte erscheinen:

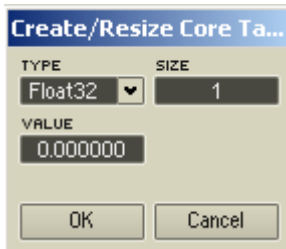


Was Sie hier sehen, ist eine leere Tabelle. Sie besteht im Moment nur aus einem einzigen Element mit dem Wert Null. Sie können nun entweder manuell (über Ihre Computertastatur) neue Werte eingeben oder Werte aus einer Datei importieren.

Wenn Sie den manuellen Weg gehen wollen, klicken Sie auf die Schaltfläche




button. Es erscheint das folgende Dialogfenster:



Hier müssen Sie den Daten-Typ angeben, der in der Tabelle gespeichert werden soll. Außerdem müssen Sie die Größe der Tabelle bestimmen (die Anzahl der Elemente in der Tabelle) und einen Wert angeben, um alle Elemente der Tabelle zu initialisieren.

Alternativ können Sie die Tabelle aus einer Datei importieren. Die Datei kann in den Audio-Formaten WAV und AIFF, als reiner Text (TXT/ASC) oder als Native Table File (NTF) vorliegen.

Um eine Tabelle aus einer Datei zu importieren, klicken Sie auf die Schaltfläche .

Es erscheint ein Dateiauswahl-Dialog, in dem Sie eine Datei auswählen können. Anschließend geben Sie in einem weiteren Dialogfenster den Daten-Typ für die Tabellenwerte an.

Lassen Sie uns mal versuchen, eine Tabelle zu verwenden. Wir werden ein Sinus-Oszillator-Macro bauen, das auf einer Tabelle beruht:



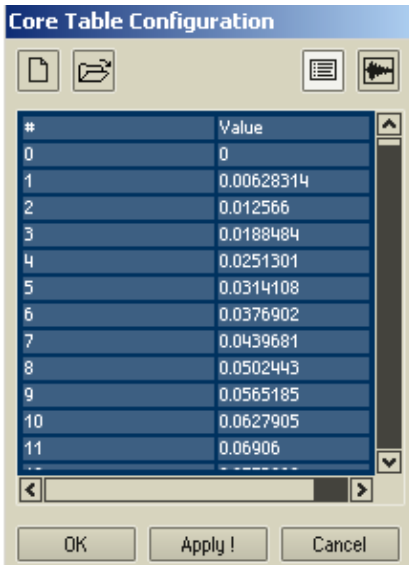
In diesem Macro erzeugen Sie ein Table-Modul:





Und dann initialisieren Sie die Tabelle mit dem Inhalt der Datei *sinetable.txt*, die wir für Sie vorbereitet und im Ordner “Core Tutorial Examples” in Ihrem REAKTOR-Installationsverzeichnis abgelegt haben. Es handelt sich dabei um eine Text-Datei, die Werte für eine Periode und ein Sample einer Sinus-Funktion enthält.



Importieren Sie diese Datei als Werte vom Typ *Float32*:

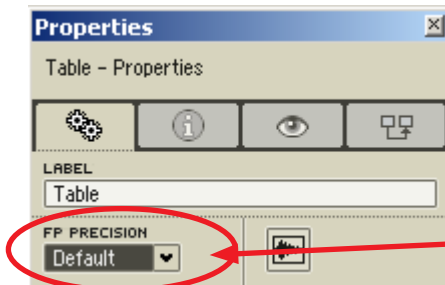


Sie können sich die geladenen Werte auch in einer Wellenform-Anzeige ansehen.

Mit den Schaltflächen  und  können Sie zwischen der Listen- und der Wellenform-Ansicht umschalten.

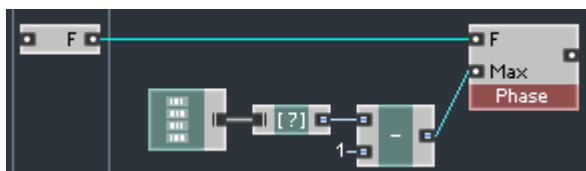
Klicken Sie nun auf "OK", um den Dialog zu schließen und die geladenen Werte an die Tabelle zu übergeben.

Im Properties-Fenster des Table-Moduls finden Sie außerdem die Eigenschaft *FP Precision*. Diese Eigenschaft kontrolliert nicht wirklich die Genauigkeit der Werte in der Tabelle (denn die sollten Sie ja schon beim Importieren oder manuellen Eingeben der Werte festlegen), sondern die "formale" Genauigkeit der Ausgangs des Table-Moduls. Normalerweise behalten Sie hier die Einstellung "Default" bei:

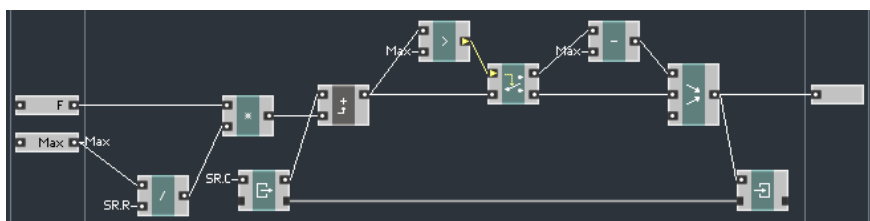


Formale Ausgangsgenauigkeit

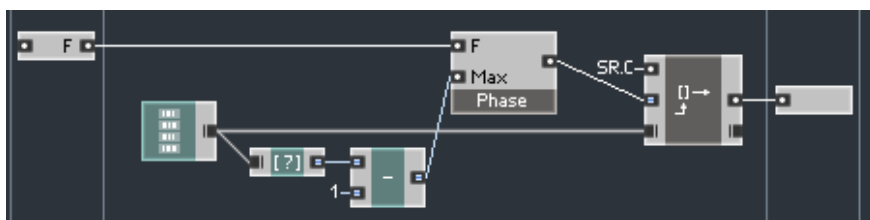
Jetzt haben wir also eine Tabelle und können den Aufbau des Oszillators fortsetzen. Im Kern wird es sich um einen “Phasen-Oszillator” handeln, der ein steigendes Sägezahn-Rampen-Signal von 0 bis zur Größe der Tabelle minus 1 erzeugt



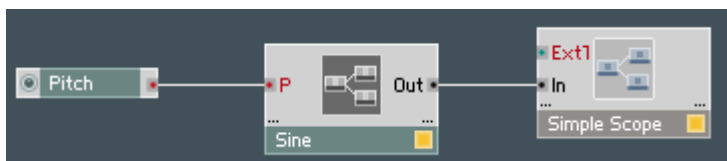
Der Phasen-Oszillator ist ziemlich ähnlich aufgebaut wie ein Sägezahn oder die Aufnahme-Position unseres Delays:



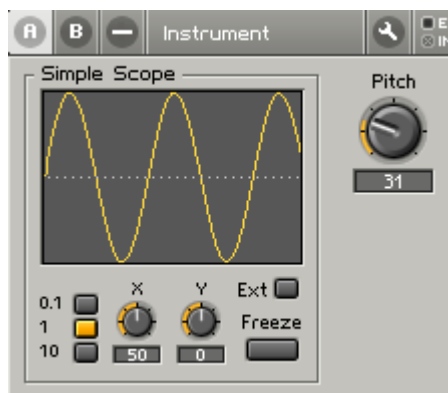
Ein Modul des Typs *Read []*, das mit dem Phasen-Oszillator verbunden ist und von der Sampling-Rate-Clock getaktet wird, greift auf das zugehörige Tabellen-Element zu und gibt seinen Wert aus.



Hier sehen Sie die passende Test-Struktur (und vergessen Sie nicht, ein Konverter-Modul des Typs *P2F* in die Core Cell einzusetzen):



Und hier ist die entsprechende Panel-Ansicht:



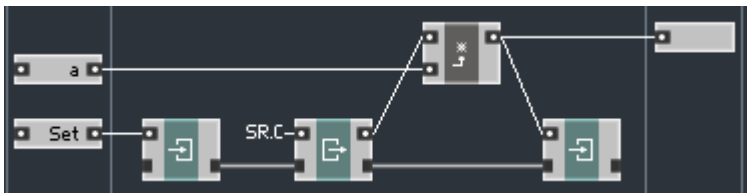
Das ist natürlich kein besonders sauber klingender Sinus, weil wir keine Interpolation zwischen den Werten vorgesehen haben. Wir überlassen es Ihnen, eine interpolierte Version zu bauen.

# Wie Sie optimale Strukturen aufbauen

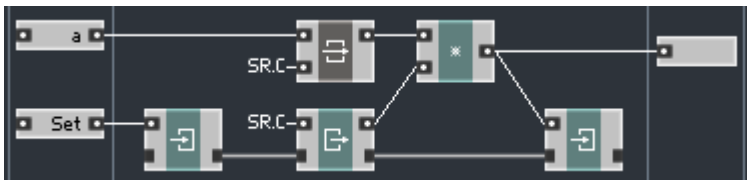
Es gilt die Regel, dass kein Werkzeug von sich aus ideal ist. REAKTOR Core bildet da keine Ausnahme. Das bedeutet natürlich nicht, dass es schlecht ist, ganz im Gegenteil – wir finden REAKTOR Core natürlich klasse. *Nicht ideal* bedeutet einfach, dass Sie einige Dinge darüber wissen müssen, wie man aus dieser Technik die besten Ergebnisse herausholt. Nennen Sie's "Tipps und Tricks" oder wie Sie sonst wollen, hier befassen wir uns jedenfalls mit diesen Dingen.

## Latches und Modulations-Macros

Verwenden Sie Latches und Modulations-Macros an allen Stellen, an denen es geboten erscheint, um sicherzustellen, dass die Events verzögert werden, bis die Werte, die sie transportieren, wirklich verarbeitet werden müssen. Hier sehen Sie eine Struktur, die ein Modulations-Macro für die Multiplikation in der Audio-Iterationsschleife verwendet. Durch den Einsatz des Modulations-Macros wird verhindert, dass am Eingang "a" ankommende Events die Verarbeitung triggern:



Alternativ dazu könnte man einen expliziten Latch in dieser Struktur verwenden:



Wir haben in früheren Kapiteln auch verschiedene andere Beispiele für diese Technik kennen gelernt. Die Verwendung von Latches hat sowohl mit der Optimierung der Performance als auch mit der *Korrektheit* Ihrer Strukturen zu tun. Einige typische Fehler beim Entwerfen von Strukturen haben mit dem Senden von Events an bestimmte Module zu ungeeigneten Zeitpunkten zu tun.

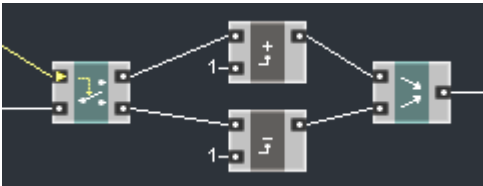
Haben Sie keine Angst, dass die Latches Ihre Strukturen ausbremsen – Latches

sind nicht sehr anspruchsvoll in Hinblick auf Prozessorleistung, und in manchen Fällen brauchen sie auch *überhaupt keine* Rechenzeit.

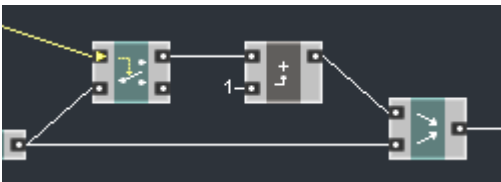
Latches sollten Sie beim Filtern von Events generell dem Routing vorziehen, weil sie weniger CPU-Last erzeugen. Verwenden Sie Router nur dann, wenn die Verarbeitungslogik Routing vorschreibt.

## Routing und Merging

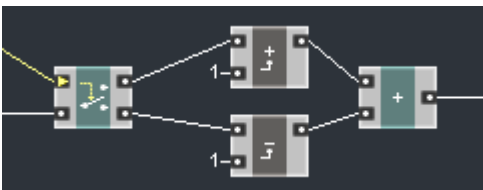
Das Routing kann die CPU je nach Situation und Plattform mehr oder weniger belasten. Wenn Sie Routing vermeiden können, ohne sich dafür andere aufwendige Berechnungen einzuhandeln, dann tun Sie's. Manchmal lässt sich das ES-Ctl-Routing durch die Verwendung von Latches umgehen. Wenn möglich, nehmen Sie diese Gelegenheit wahr. Wenn Sie den Event-Pfad unter Verwendung eines Routers in zwei Zweige aufspalten, ist es eine gute Idee, die an den Ausgängen des Router-Moduls entspringenden Zweige zu mischen:



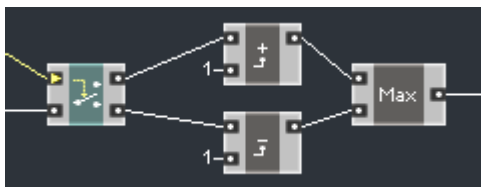
Es ist außerdem klug, das ankommende (ungeteilte) Event dem Router hinzuzumischen:



Das sortierende Mischen (Merging) muss nicht notwendigerweise mit einem *Merge*-Modul durchgeführt werden. Jedes arithmetische oder vergleichbare Modul kann diese Aufgabe übernehmen



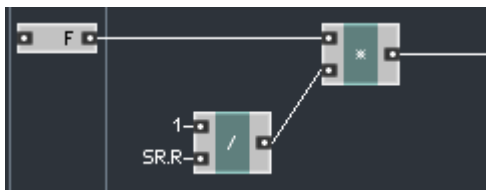
Das Merging kann auch innerhalb eines Macros stattfinden (abhängig von seiner internen Struktur):



Es kann sinnvoll oder sogar notwendig sein, die beiden von verschiedenen Router-Modulen erzeugten Zweige zu mischen, aber behalten Sie in diesem Fall die CPU-Last im Auge.

## Numerische Operationen

Addition, Multiplikation, Subtraktion, Absoluter Wert und Negation sind normalerweise die am wenigsten anspruchsvollen Fließkomma-Operationen. Beim Verarbeiten von Integer-Zahlen belasten Addition, Subtraktion und Negation die CPU am wenigsten. Auch der Absolute Wert ist bei Integer-Berechnungen mehr oder weniger okay. *DN Cancel* entspricht zurzeit einer einfachen Addition, wie Sie sich erinnern werden. Die Division von Float-Werten und die Multiplikation und Division von Integer-Zahlen belasten die CPU allerdings deutlich mehr als die durchschnittlichen Rechenoperationen. Es ist ratsam, dass Sie Ihre Rechenoperationen so gruppieren, dass die anspruchsvollste so selten wie möglich in Zahlen ausgedrückt wird. Wenn Sie zum Beispiel eine normalisierte Frequenz durch Teilen der Frequenz in Hz durch die Sampling-Rate berechnen wollen, ist es sinnvoll, dass Sie zuerst den Kehrwert der Sampling-Rate berechnen und dann das Ergebnis mit der Frequenz multiplizieren:



In der obigen Struktur wird die Division nur dann durchgeführt, wenn sich die Sampling-Rate ändert, was ziemlich selten vorkommen sollte. Veränderungen der Frequenz würden nur die Multiplikation triggern.

Betrachten Sie zum Vergleich die simple Implementation derselben Formel:

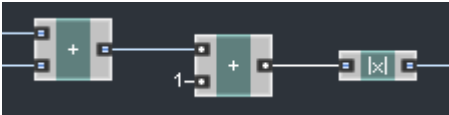


Hier wird die Division auch bei jeder Änderung der Frequenz durchgeführt.

## Konvertierungen zwischen Float und Integer

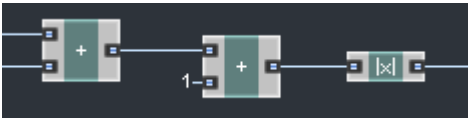
Vermeiden Sie nach Möglichkeit alle unnötigen Konvertierungen zwischen Fließkomma- und Integer-Zahlen. Abhängig von der Plattform erfordern diese Konvertierungen nämlich viel CPU-Leistung. Notwendige Konvertierungen dürfen Sie natürlich durchführen lassen, keine Frage.

Obwohl die folgende Struktur vielleicht wie vorgesehen arbeitet, finden hier tatsächlich zwei unnötige Konvertierungen zwischen den Zahlen-Formaten Float und Integer statt:

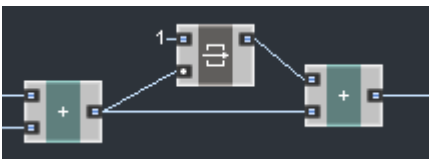


Die erste Konvertierung erfolgt am Eingang des Addierer-Moduls in der Mitte. Das Modul befindet sich im Fließkomma-Modus, empfängt aber ein Integer-Eingangssignal. Deshalb wird eine Konvertierung von Integer nach Float durchgeführt. Die zweite Konvertierung findet am Eingang des Absoluter-Wert-Moduls statt, das im Integer-Modus arbeitet, aber ein Fließkomma-Signal empfängt. Deshalb ist eine Konvertierung von Float nach Integer erforderlich.

Hier sehen Sie eine viel bessere Lösung:



Alle Module sind in den Integer-Modus geschaltet, deshalb finden keine Konvertierungen statt. Clock-Signale sollten normalerweise im Fließkomma-Format vorliegen, aber wenn ein Integer-Signal als Clock auftaucht, ist das auch kein Problem:



Obwohl der Clock-Eingang des Moduls ILatch sich im Fließkomma-Modus befindet, wird er von einem Integer-Signal getaktet. Weil der Wert des Clock-Signals allerdings unerheblich ist, findet keine Konvertierung statt.

# Anhang A. REAKTOR Cores Bedienoberfläche

## A.1. Core cells

Eine Core Cell erzeugen Sie aus einer Primary-Level-Struktur heraus (außer aus Ensemble-Strukturen), indem Sie einen Rechts-Klick in den Hintergrund ausführen und aus dem Menü *Core Cell > New Audio* oder *Core Cell > New Event* auswählen.

Library-Core-Cells (sowohl aus der System- als auch aus der User-Library) finden Sie im selben Menü *Core Cell*. Sie können Core Cells auch über den Menü-Eintrag *Core Cell > Load...* laden.

Um eine Core Cell zu löschen, wählen Sie die Core Cell aus und drücken die Taste *Delete*. Alternativ können Sie einen Rechts-Klick auf die Core Cell ausführen und aus dem Menü den Eintrag *Delete* auswählen. Sie können auch mehrere ausgewählte Core Cells gleichzeitig löschen.

Um eine Core Cell als Datei zu speichern, führen Sie einen Rechts-Klick auf die Core Cell aus und wählen aus dem Menü den Eintrag *Save Core Cell As...* aus. Um die interne Struktur einer Core Cell zu bearbeiten, doppelklicken Sie auf die Core Cell. Um zurück auf die höhere Ebene zu gelangen, klicken Sie in den Hintergrund.

Um die äußeren Eigenschaften einer Core Cell zu bearbeiten, führen Sie einen Rechts-Klick auf die Core Cell aus und wählen Sie den Eintrag *Properties* aus dem Menü. Wenn das Properties-Fenster bereits geöffnet ist, müssen Sie nur auf die Core Cell klicken, deren Eigenschaften Sie bearbeiten wollen.

Um die inneren Eigenschaften einer Core Cell zu bearbeiten, müssen Sie die innere Struktur der Core Cell freilegen, einen Rechts-Klick in den Hintergrund ausführen und den Eintrag *Owner Properties* aus dem Menü auswählen. Wenn das Properties-Fenster bereits geöffnet ist, müssen Sie nur in den Hintergrund klicken.

## A.2. Core-Module und -Macros

Um ein normales Core-Modul oder -Macro zu erzeugen, führen Sie einen Rechts-Klick in den zentralen (und größten) Bereich der Core-Struktur aus und wählen Sie einen der Einträge aus den Menüs *Built In Module*, *Expert Macro*, *Standard Macro* oder *User Macro* aus. Sie können Module und Macros auch laden, indem Sie einen Rechts-Klick in den Hintergrund ausführen und aus dem Menü den Eintrag *Load Module...* auswählen.

Ein leeres Macro erzeugen Sie, indem Sie im Menü *Built In Module* den entsprechenden Eintrag auswählen. Um ein Core-Modul oder Core-Macro als Datei zu speichern, führen Sie einen Rechts-Klick auf das betreffende Modul



oder Macro aus und wählen aus dem Menü den Eintrag *Save As...* aus.

Um ein Core-Modul oder Core-Macro zu löschen, wählen Sie es aus und drücken die Taste *Delete*. Alternativ können Sie einen Rechts-Klick auf das Modul oder Macro ausführen und aus dem Menü den Eintrag *Delete* auswählen. Sie können auch mehrere ausgewählte Module oder Macros gleichzeitig löschen.

Um die interne Struktur eines Core-Macros zu bearbeiten, doppelklicken Sie auf die Core Cell. Um zurück auf die höhere Ebene zu gelangen, klicken Sie in den Hintergrund.

Um die Eigenschaften eines Core-Moduls oder –Macros zu bearbeiten, legen Sie die innere Struktur frei, führen einen Rechts-Klick in den Hintergrund aus und wählen den Eintrag *Owner Properties* aus dem Menü. Wenn das Properties-Fenster bereits geöffnet ist, müssen Sie nur in den Hintergrund klicken. Sie können die Eigenschaften des Moduls oder Macros auch von außen erreichen, indem Sie einen Rechts-Klick auf das Modul oder Macro ausführen und aus dem Menü den Eintrag *Properties* auswählen. Wenn das Properties-Fenster bereits geöffnet ist, müssen Sie nur auf das Modul oder Macro klicken.

### **A.3. Core ports**

Um einen Core-Port zu erzeugen führen Sie einen Rechts-Klick in den Eingangs-Bereich (links) oder in den Ausgangs-Bereich (rechts) einer Core-Struktur aus. Wählen Sie aus dem Untermenü *New* einen der verfügbaren Port-Typen aus.

Um einen Core-Port zu löschen, wählen Sie den Port aus und drücken die Taste *Delete*. Alternativ können Sie einen Rechts-Klick auf den Port ausführen und aus dem Menü den Eintrag *Delete* auswählen. Sie können auch mehrere ausgewählte Core-Ports gleichzeitig löschen (einschließlich gemischter Modul-/Port-Auswahlen).

### **A.4. Core-Strukturen bearbeiten**

Um ein Core-Modul zu bewegen, klicken Sie darauf und verschieben Sie das Modul mit gedrückter Maustaste an den gewünschten Platz. Ports können Sie nur in vertikaler Richtung verschieben; die vertikale Reihenfolge bestimmt über ihre Reihenfolge in der äußeren Ansicht.

Um eine Verbindung zwischen dem Eingang eines Moduls und dem Ausgang eines anderen Moduls herzustellen, klicken Sie auf einen der beiden Ports und ziehen Sie das virtuelle “Kabel” zum anderen Port.



Um eine Verbindung zu entfernen, wählen Sie zunächst das Verbindungskabel aus, indem Sie darauf klicken. Drücken Sie dann die Taste *Delete*, um die ausgewählte Verbindung zu löschen. Alternativ können Sie auch das Verbindungskabel aus dem Eingang ziehen und über dem Hintergrund der Struktur loslassen.

Um eine QuickConst zu erzeugen, führen Sie einen Rechts-Klick auf den Eingang eines Moduls aus und wählen Sie den Eintrag *Connect to New QuickConst* aus. Um das Properties-Fenster der QuickConst zu öffnen und auf die Eigenschaften zuzugreifen, klicken Sie auf die QuickConst.

Um einen QuickBus zu erzeugen, führen Sie einen Rechts-Klick auf einen Eingang oder einen Ausgang eines Moduls aus und wählen aus dem Menü den Eintrag *Connect to New QuickBus* aus. Um einen Eingang oder einen Ausgang eines Moduls mit einem bestehenden QuickBus zu verbinden, führen Sie einen Rechts-Klick auf diesen Eingang oder Ausgang aus und wählen aus dem Menü *Connect to QuickBus* einen der verfügbaren Busse aus.

# Anhang B. Konzepte von REAKTOR Core

## B.1. Signale und Events

In REAKTOR Core kommen Signale der Typen Float und Integer vor. Float-Ports sehen so aus: . Integer-Ports sehen so aus: . Die Signale pflanzen sich durch die Verbindungen zwischen Ausgängen und den angeschlossenen Eingängen in Form von Events fort. Ein Event ist eine elementare Aktion, die infolge einer Änderung des Werts eines Ausganges an diesem Ausgang auftritt (wobei in Sonderfällen der Wert auch auf denselben Wert geändert werden kann). Alle Events, die aus derselben Event-Quelle stammen, werden als "gleichzeitig" behandelt. "Gleichzeitig" bedeutet, dass zwei solche Events, die an unterschiedlichen Eingängen desselben Moduls ankommen, dort "gleichzeitig" ankommen.

"Dieselbe Event-Quelle" bedeutet, dass die Events von demselben Ausgang gesendet werden. Unter bestimmten Umständen können aber auch mehrere Ausgänge als "dieselbe Event-Quelle" angesehen werden. Z. B. werden alle Audio-Eingänge und alle Standard-Sampling-Rate-Clock-Verbindungen als eine Event-Quelle behandelt. Während der Initialisierung werden alle Ausgänge, die Events senden, als dieselbe Event-Quelle betrachtet. "Dieselbe Event-Quelle" bedeutet in diesem Zusammenhang allerdings nicht "derselbe Wert", sondern bezieht sich nur auf die "Gleichzeitigkeit". Wenn ein Modul nicht selbst eine Event-Quelle ist, kann es nur durch das Eintreffen eines oder mehrerer Events an seinem Eingang zum Verarbeiten der anliegenden Werte angeregt werden. Im Fall des Eintreffens mehrerer Events wird nur ein Ausgangs-Event erzeugt, weil die Eingangs-Events gleichzeitig eintreffen.

## B.2. Initialisierung

Die Initialisierung der Strukturen geht folgendermaßen vor sich: Zuerst werden alle Werte auf Null zurückgesetzt. Dann senden alle Initialisierungs-Quellen gleichzeitig das Initialisierungs-Event. Normalerweise sind diese Quellen Konstanten-Module, Core-Cell-Eingänge (nicht immer) und Clock-Quellen. Das ist es schon.

## B.3. Verbindungs-Typ OBC

Der Verbindungs-Typ OBC (Object Bus Connections) bezeichnet Verbindungen zwischen Modulen, die keine Signale senden, sondern belegen, dass die Module einen gemeinsamen Zustand (Speicherplatz) teilen. Am häufigsten kommt der Typ OBC bei Verbindungen zwischen Modulen der Typen *Read* und *Write*, die auf denselben gespeicherten Wert zugreifen, zum Einsatz.

## B.4. Routing

Sie können Module des Typs *Router* verwenden, um den Strom von Events zwischen zwei möglichen Pfaden aufzuteilen. Wenn der *Router* einen Ausgangs-Pfad für das ankommende Event auswählt, empfängt der andere Pfad keine Events (was sich besonders darin äußert, dass sich der Wert dieses anderen Ausgangs nicht ändern kann). Das Modul *Router* wird von einer Eingangs-Verbindung des Typs *BoolCtl* kontrolliert. Auf der anderen Seite dieser Verbindung befindet sich typischerweise ein Vergleichs-Modul (*Compare*); manchmal sind einige vermittelnde Modulen wie Macro-Ports des Typs *BoolCtl* zwischen die Module *Compare* und *Router* geschaltet. Durch Verwenden von Modulen des Typs *Router* können Sie Werteänderungen in Teilen Ihrer Struktur gezielt ermöglichen und verhindern.

In den allermeisten Fällen werden Sie die beiden durch das Einsetzen eines Router-Moduls entstehenden Signal-Pfade wieder mischen; hierzu können Sie ein Modul des Typs *Merge* oder auch ein anderes Modul einsetzen. Oftmals können Sie auch einen Pfad dem vor der Aufteilung anliegenden Original-Signal hinzumischen. Aber Sie können hier im Allgemeinen machen, was Sie wollen – behalten Sie nur die Performance Ihrer Struktur im Auge!

## B.5. Latching

Latching ist wahrscheinlich die am weitesten verbreitete Technik in REAKTOR Core. Das Prinzip dahinter ist, dass Sie Latch-Module verwenden, um zu verhindern, dass Events zur falschen Zeit gesendet werden. Zum Beispiel wollen Sie wahrscheinlich nicht, dass ein Kontroll-Signal eine Berechnung in der Audio-Schleife der Struktur triggert. Alternativ können Sie auch Macros verwenden, die Sie im Menü unter *Expert Macros > Modulation* finden; es handelt sich bei diesen Macros im Prinzip um die häufigsten Kombinationen von Latches mit einigen arithmetischen Modulen.

## B.6. Clocking

Clocks sind Quellen von Events. Das Clock-Event tritt üblicherweise in regelmäßigen Intervallen abhängig von der Clock-Rate auf. Sie brauchen Clocks normalerweise, um eine Vielzahl von Modulen anzutreiben, zum Beispiel Oszillatoren, Filter und so weiter. Die meisten dieser Module brauchen keine Taktung von außen, sondern verwenden eine in Core-Strukturen verfügbare Standard-Clock-Quelle. Diese Quelle ist die Sampling-Rate-Clock, die mit der Standard-Audio-Rate läuft. Beachten Sie, dass in Event-Core-Cells das Clock-Signal nicht verfügbar ist, obwohl eine Verbindung zur Sampling-Rate-Clock möglich ist. Deshalb arbeiten die meisten Oszillatoren, Filter und ähnliche Module nicht in Event-Core-Cells.

# Anhang C. Core-Macro-Ports

## C.1. In



Empfängt ein ankommendes Event von der Außenseite und leitet es unverändert an seinen inwärts gerichteten Ausgang weiter.

Die Eingangs-Verbindung auf der Innenseite können Sie zum Überschreiben des Default-Werts dieses Ports verwenden.

## C.2. Out



Empfängt ein ankommendes Event an seinem innenseitigen Eingang und leitet es unverändert an seinen nach außen gerichteten Ausgang weiter.

## C.3. Latch (Eingang)



Empfängt ein ankommendes Event an seinem innenseitigen Eingang und leitet es unverändert an seinen nach außen gerichteten Ausgang weiter.

## C.4. Latch (Ausgang)



Schleust eine OBC-Verbindung von der Innenseite eines Macros zur Außenseite des Macros durch.

## C.5. Bool C (Eingang)



Schleust eine BoolCtl-Verbindung von der Außenseite eines Macros zur Innenseite des Macros durch.

Die Eingangs-Verbindung auf der Innenseite können Sie zum Überschreiben des Default-Werts dieses Ports verwenden.

## C.6. Bool C (Ausgang)



Schleust eine BoolCtl-Verbindung von der Innenseite eines Macros zur Außenseite des Macros durch.

# Anhang D. Core-Cell-Ports

## D.1. In (Audio-Modus)



Ermöglicht den Zugriff auf das Audio-Signal von außerhalb des Moduls. Sendet regelmäßig (mit der Sampling-Rate), synchron zur globalen Sampling-Rate-Clock.

**INITIALIZATION EVENT:** sendet ein Initialisierungs-Event. Der Wert wird von der äußeren Initialisierung bestimmt.

## D.2. Out (Audio-Modus)



Stellt den am inneren Eingang empfangenen Wert an der Außenseite des Moduls zur Verfügung. Zu jedem Zeitpunkt wird der letzte empfangene Wert an die Außenseite übergeben.

## D.3. In (Event-Modus)



Konvertiert von außen eintreffende Primary-Level-Events in REAKTOR-Core-Events und leitet sie an die Innenseite weiter.

**INITIALIZATION EVENT:** sendet ein Initialisierungs-Event, wenn auf der Außen-seite ein Initialisierungs-Event empfangen wird.

## D.4. Out (Event-Modus)



Konvertiert REAKTOR-Core-Events, die aus dem Inneren ankommen, in Primary-Level-Events und leitet diese an die Außenseite weiter. Wenn mehrere im Event-Modus befindliche Ausgänge gleichzeitig REAKTOR-Core-Events empfangen, werden die korrespondierenden Primary-Level-Events in der vertikalen Reihenfolge der Ausgänge (von oben nach unten) gesendet.

## Anhang E. Built-in buses

### E.1. SR.C

Sendet regelmäßige Clock-Events mit der Sampling-Rate.

**INITIALIZATION EVENT:** sendet immer ein Initialisierungs-Event.

### E.2. SR.R

Stellt die aktuelle Sampling-Rate in Hz zur Verfügung. Sendet als Reaktion auf Veränderungen der Sampling-Rate Events mit neuen Werten.

**INITIALIZATION EVENT:** sendet immer ein Initialisierungs-Event mit der anfänglichen Sampling-Rate.



# Anhang F. Built-in modules

## F.1. Const



Erzeugt ein Signal mit einem konstanten Wert.

Der Wert wird im Modul angezeigt.

**INITIALIZATION EVENT:** sendet während der Initialisierung ein Event mit dem festgelegten Wert an den Ausgang. Dies ist das einzige Mal, dass dieses Modul ein Event sendet.

**EIGENSCHAFTEN:**

**Value** der Wert, der an den Ausgang gesendet werden soll

## F.2. Math > +



Erzeugt die Summe aus den ankommenden Signalen am Ausgang. Das Ausgangs-Event wird jedes Mal gesendet, wenn ein Event an einem oder beiden Eingängen gleichzeitig ankommt.

## F.3. Math > -



Erzeugt die Differenz aus den ankommenden Signalen am Ausgang (das Signal des unteren Eingangs wird vom Signal des oberen Eingangs subtrahiert). Das Ausgangs-Event wird jedes Mal gesendet, wenn ein Event an einem Eingang oder an beiden Eingängen gleichzeitig ankommt.

## F.4. Math > \*



Erzeugt das Multiplikations-Produkt aus den ankommenden Signalen am Ausgang. Das Ausgangs-Event wird jedes Mal gesendet, wenn ein Event an einem Eingang oder an beiden Eingängen gleichzeitig ankommt.

## F.5. Math > /



Erzeugt den Quotienten aus den ankommenden Signalen am Ausgang (das Signal am oberen Eingang wird durch das Signal am unteren Eingang geteilt). Im Integer-Modus führt dies zu einer Division mit Rest, wobei der Rest verworfen wird. Das Ausgangs-Event wird jedes Mal gesendet, wenn ein Event an einem Eingang oder an beiden Eingängen gleichzeitig ankommt.

## F.6. Math > |x|



Erzeugt den absoluten Wert des ankommenden Signals. Das Ausgangs-Event wird jedes Mal gesendet, wenn ein Event am Eingang ankommt.

## F.7. Math > -x



Erzeugt den invertierten Wert (durch Ändern des Vorzeichens) des ankommenden Signals am Ausgang. Das Ausgangs-Event wird jedes Mal gesendet, wenn ein Event am Eingang ankommt.

## F.8. Math > DN Cancel



Modifiziert das ankommende Signal so, dass das Auftreten denormaler Zahlen verhindert wird. Zurzeit geschieht dies durch Addieren einer sehr kleinen Konstante. Funktioniert nur bei Fließkomma-Zahlen. Das Ausgangs-Event wird jedes Mal gesendet, wenn ein Event am Eingang ankommt.

## F.9. Math > ~log



Berechnet eine Annäherung an den Logarithmus des ankommenden Werts. Das Ausgangs-Event wird jedes Mal gesendet, wenn ein Event am Eingang ankommt.

## EIGENSCHAFTEN:

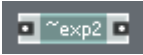
**Base**

die Logarithmus-Basis

**Precision**

die Genauigkeit der Näherung (höhere Genauigkeit erfordert mehr CPU-Leistung)

## F.10. Math > ~exp



Berechnet eine Annäherung an den Exponenten des ankommenden Werts. Das Ausgangs-Event wird jedes Mal gesendet, wenn ein Event am Eingang ankommt.

### EIGENSCHAFTEN

**Base**

die Exponenten-Basis

**Precision**

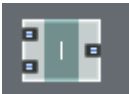
die Genauigkeit der Näherung (höhere Genauigkeit erfordert mehr CPU-Leistung)

## F.11. Bit > Bit AND



Führt eine bitweise Zusammenführung (Konjunktion) der ankommenden Signale durch. Arbeitet nur mit Integer-Signalen. Das Ausgangs-Event wird jedes Mal gesendet, wenn ein Event an einem Eingang oder an beiden Eingängen gleichzeitig ankommt.

## F.12. Bit > Bit OR



Führt eine bitweise Trennung (Disjunktion) der ankommenden Signale durch. Arbeitet nur mit Integer-Signalen. Das Ausgangs-Event wird jedes Mal gesendet, wenn ein Event an einem Eingang oder an beiden Eingängen gleichzeitig ankommt.

## F.13. Bit > Bit XOR



Führt eine bitweise ausschließende Trennung (exklusive Disjunktion) der ankommenden Signale durch. Arbeitet nur mit Integer-Signalen.

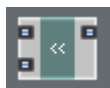
Das Ausgangs-Event wird jedes Mal gesendet, wenn ein Event an einem Eingang oder an beiden Eingängen gleichzeitig ankommt.

## F.14. Bit > Bit NOT



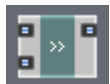
Führt eine bitweise Umkehrung (Invertierung) der ankommenden Signale durch. Arbeitet nur mit Integer-Signalen. Das Ausgangs-Event wird jedes Mal gesendet, wenn ein Event am Eingang ankommt.

## F.15. Bit > Bit <<



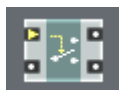
Versetzt den Wert an oberen Eingang bitweise nach links in Richtung der signifikanteren Bits (More Significant Bits). Die Anzahl von Bits, um die der Wert versetzt werden soll, wird vom unteren Eingang bestimmt. Das Ergebnis für  $N < 0$  und  $N > 31$  ist undefiniert (das heißt, Sie sollten diese Funktion nur verwenden, wenn gilt  $0 \leq N \leq 31$ ). Arbeitet nur mit Integer-Signalen. Das Ausgangs-Event wird jedes Mal gesendet, wenn ein Event an einem Eingang oder an beiden Eingängen gleichzeitig ankommt.

## F.16. Bit > Bit >>



Versetzt den Wert des oberen Eingangs bitweise nach links in Richtung der weniger signifikanten Bits (Less Significant Bits). Es wird keine Vorzeichen-Erweiterung durchgeführt. Das Ergebnis für  $N < 0$  und  $N > 31$  ist undefiniert (das heißt, Sie sollten diese Funktion nur verwenden, wenn gilt  $0 \leq N \leq 31$ ). Arbeitet nur mit Integer-Signalen. Das Ausgangs-Event wird jedes Mal gesendet, wenn ein Event an einem Eingang oder an beiden Eingängen gleichzeitig ankommt.

## F.17. Flow > Router



Leitet das am Signal-Eingang (dem unteren Eingang) anliegende Signal abhängig vom Zustand des Kontroll-Signals (am oberen Eingang) an einen der

beiden Ausgänge weiter. Wenn das Kontroll-Signal den Wert “wahr” hat, wird der Ausgang 0 (der obere Ausgang) beschickt; nimmt das Kontroll-Signal den Zustand “falsch” an, landet das Signal am Ausgang 0 (dem unteren Ausgang). Das Ausgangs-Event wird an genau einen der Ausgänge gesendet, wenn ein Event am Signal-Eingang ankommt.

## F.18. Flow > Compare

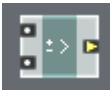


Erzeugt ein BoolCtl-Signal am Ausgang, welches das Ergebnis des Vergleichs der Eingangs-Werte enthält. Der Wert am oberen Eingang steht auf der linken Seite des Vergleichs-Symbols, der Wert des unteren Eingangs steht rechts (sodass das abgebildete Modul prüft, ob der obere Wert größer ist als der untere Wert).

### EIGENSCHAFTEN:

**Criterion** die Vergleichs-Operation, die durchgeführt werden soll

## F.19. Flow > Compare Sign



Erzeugt ein BoolCtl-Signal am Ausgang, welches das Ergebnis des Vergleichs der Eingangs-Werte enthält. Der Wert am oberen Eingang steht auf der linken Seite des Vergleichs-Symbols, der Wert des unteren Eingangs steht rechts (sodass das abgebildete Modul prüft, ob das Vorzeichen des oberen Werts größer ist als das Vorzeichen des unteren Werts).

Der Vorzeichen-Vergleich ist wie folgt definiert:

- + ist gleich +
- ist gleich –
- + ist größer als –

Das Vorzeichen des Werts Null ist undefiniert, sodass beliebige Ergebnisse auftreten können, wenn einer der verglichenen Werte Null ist.

### EIGENSCHAFTEN:

**Criterion** die Vergleichs-Operation, die durchgeführt werden soll

## F.20. Flow > ES Ctl



Erzeugt ein BoolCtl-Signal am Ausgang, das die gegenwärtige Anwesenheit eines Events am Eingang anzeigt (was bedeutet, dass das Kontroll-Signal den Wert “wahr” annimmt, wenn zu einem bestimmten Zeitpunkt ein Event am Eingang dieses Moduls auftritt).

## F.21. Flow > ~BoolCtl



Erzeugt ein BoolCtl-Signal am Ausgang, das eine Umkehrung des BoolCtl-Signals ist (“wahr” wird zu “falsch” und vice versa).

## F.22. Flow > Merge

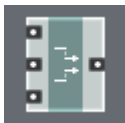


Sendet jedes Mal ein Ausgangs-Event, wenn an einem der Eingänge oder an mehreren Eingängen gleichzeitig ein Event ankommt. Wenn nur ein Eingang zu einem bestimmten Zeitpunkt das Event empfängt, ist der Ausgangs-Wert gleich dem Wert dieses Eingangs-Events. Wenn mehrere Eingänge gleichzeitig ein Event empfangen, wird der Wert des untersten Eingangs (der in diesem Moment empfangenden Eingänge) ausgewählt. Wenn also sowohl der zweite als auch der dritte (von oben aus gesehen) Eingang ein Event empfangen, wird der Wert am dritten Eingang ausgewählt.

### EIGENSCHAFTEN:

**Input Count** Anzahl der Eingänge des Moduls

## F.23. Flow > EvtMerge



Die Funktion ähnelt der des Moduls Merge, allerdings werden alle Eingangs-Werte ignoriert. Der Wert des Ausgangs-Events ist undefiniert. Dieses Modul ist dafür gedacht, Signale zu erzeugen, die als Clock verwendet werden. Arbeitet nur im Fließkomma-Modus, weil die Werte ohnehin nicht für eine

Weiterverwendung vorgesehen sind.

**EIGENSCHAFTEN:**

**Input Count** Anzahl der Eingänge des Moduls

## F.24. Memory > Read



Liest den gespeicherten Wert aus dem Speicher, der der OBC-Kette zugeordnet ist, zu der dieses Modul gehört. Der Lese-Vorgang findet als Reaktion auf ein Event am oberen Eingang (Clock) statt, der gelesene Wert wird an den oberen Ausgang geschickt. Die unteren Ports sind Master- und Slave-Anschlüsse für die OBC-Anbindung.

## F.25. Memory > Write



Schreibt den Wert, der am oberen Eingang ankommt, in den Speicher, der der OBC-Kette zugeordnet ist, zu der dieses Modul gehört. Der Schreib-Vorgang findet als Reaktion auf ein Event am oberen Eingang (Clock) statt. Die unteren Ports sind Master- und Slave-Anschlüsse für die OBC-Anbindung.

## F.26. Memory > R/W Order



Dieses Modul führt gar keine Aktion aus. Sie können es in einen Struktur einsetzen, um die Verarbeitungsreihenfolge der via OBC verbunden Module zu kontrollieren. Die OBC-Ports (unten) sind OBC-Master- und Slave-Verbindungen, die intern einfach “durchgeschleift” sind. Der OBC-Eingang (oben) stellt die “Sidechain”-Verbindung her, die es ermöglicht, das Modul logisch hinter dem Modul zu platzieren, das an den Sidechain-Eingang angeschlossen ist.

Die Sidechain-Verbindung können Sie nur an normale Module des Typs Latch OBC anschließen. Die Master- und Slave-Ports dagegen können Sie mit Latch- oder Array-OBC-Modulen verbinden, abhängig von den für das Modul R/W Order im Properties-Fenster getroffenen Einstellungen. Auf jeden Fall müssen Signal-Typ und Genauigkeit für alle Verbindungen zu diesem Modul gleich sein (d. h., Sie können nicht den Sidechain-Eingang mit einem Integer-Read-Modul verbinden und gleichzeitig Master und Slave an Fließkomma-Module anschließen).

## EIGENSCHAFTEN:

**Connection Type**      Art der “durchgeschleiften” Port-Verbindung  
(Latch oder Array)

## F.27. Memory > Array



Definiert ein Array-Speicher-Objekt. Das Modul selbst führt keine Aktionen aus. Alle Operationen über dem Inhalt des Arrays werden von den Modulen durchgeführt, die an den Ausgang des Arrays angeschlossen sind. Dieser Ausgang ist eine OBC-Slave-Verbindung vom Typ Array.

### EIGENSCHAFTEN:

**Size**      Anzahl der Elemente im Array

## F.28. Memory > Size [ ]



Übermittelt die Größe des Array-Objekts, das mit dem Eingang verbunden ist. Die Größe ist ein konstanter Integer-Wert

**INITIALIZATION EVENT:** sendet während der Initialisierung ein Event mit dem Wert der Array-Größe an den Ausgang. Dies ist das einzige Mal, dass dieses Modul ein Event sendet.

## F.29. Memory > Index



Ermöglicht den Zugriff auf ein einzelnes Array-Element. Der Zugang erfolgt in Form einer Latch-OBC-Verbindung, die dem Array-Element zugeordnet ist. Die Zuordnung wird durch das Senden eines Events an den oberen Eingang (Index) des Index-Moduls hergestellt und/oder verändert, der *Null-basiert* ist und sich immer im Integer-Modus befindet. Der untere Eingang ist die Master-OBC-Verbindung zum Array. Der Ausgang stellt die Latch-OBC-Verbindung mit dem über den Index-Eingang ausgewählten Array-Element her. Der Basis-Werte-Typ und die Genauigkeit müssen am Eingang und Ausgang der OBC-Verbindung identisch sein; Sie können diese Parameter im Properties-Fenster des Index-Moduls festlegen



## F.30. Memory > Table



Definiert ein vorinitialisiertes Read-Only-Array (Tabelle). Alle Operationen über dem Inhalt der Tabelle werden von den Modulen durchgeführt, die an den Ausgang des Table-Moduls angeschlossen sind. Dieser Ausgang ist eine OBC-Slave-Verbindung vom Typ Array.

### EIGENSCHAFTEN:



ruft den Editor zum Bearbeiten der Werte der Tabelle auf

#### FP Precision

kontrolliert die formale Genauigkeit der Ausgangs-Verbindung

## F.31. Macro



Bildet einen Container für eine im Inneren befindliche Struktur. Die Anzahl der Eingänge und Ausgänge ist nicht festgelegt und wird von der internen Struktur bestimmt.

### EIGENSCHAFTEN:

#### FP Precision

kontrolliert die formale Genauigkeit der Ausgangs-Verbindung

#### Look

schaltet zwischen den Ansichten *Large* (Label und Port-Namen sichtbar) und *Small* (Label und Port-Namen unsichtbar) um

#### Pin Alignment


kontrolliert die Ausrichtung der Ports in der äußeren Ansicht des Macros

#### Solid

kontrolliert die Behandlung des Macros durch die Core-Engine. Wenn diese Option ausgeschaltet ist, sind die Grenzen des Macros transparent für die Feedback-Auflösung und andere Dinge. Lassen Sie diese Option eingeschaltet, außer, Sie wissen wirklich ganz genau, was Sie tun!

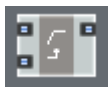
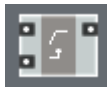
#### Icon



lädt ein neues Icon für das Macro,  löscht das Icon (kein Icon zugewiesen)

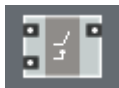
## Anhang G. Expert macros

### G.1. Clipping > Clip Max / IClip Max



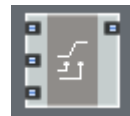
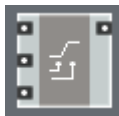
Das Signal am oberen Eingang wird von oben ausgehend gemäß dem am unteren Eingang anliegenden Schwellwert beschnitten. Veränderungen des Schwellwerts erzeugen keine Events.

### G.2. Clipping > Clip Min / IClip Min



Das Signal am oberen Eingang wird von unten ausgehend gemäß dem am unteren Eingang anliegenden Schwellwert beschnitten. Veränderungen des Schwellwerts erzeugen keine Events.

### G.3. Clipping > Clip MinMax / IClipMinMax



Das Signal am oberen Eingang wird von oben ausgehend gemäß dem am mittleren Eingang anliegenden Schwellwert und von oben ausgehend gemäß dem Schwellwert am unteren Eingang beschnitten. Veränderungen der Schwellwerte erzeugen keine Events.

### G.4. Math > 1 div x



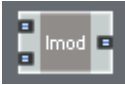
Berechnet den Kehrwert des Eingangs-Werts.

### G.5. Math > 1 wrap



Faltet die ankommenden Werte in den Bereich [-0.5 bis 0.5] (die Faltungsperiode ist 1).

## G.6. Math > Imod



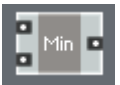
Berechnet den Rest, der bei der Division des oberen Werts durch den unteren Wert übrig bleibt. Das Ausgangs-Event wird jedes Mal gesendet, wenn ein Event an einem Eingang oder an beiden Eingängen gleichzeitig ankommt.

## G.7. Math > Max / IMax



Berechnet das Maximum der Eingangs-Werte. Das Ausgangs-Event wird jedes Mal gesendet, wenn ein Event an einem Eingang oder an beiden Eingängen gleichzeitig ankommt.

## G.8. Math > Min / IMin



Berechnet das Minimum der Eingangs-Werte. Das Ausgangs-Event wird jedes Mal gesendet, wenn ein Event an einem Eingang oder an beiden Eingängen gleichzeitig ankommt.

## G.9. Math > round



Rundet den ankommenden Wert auf die nächste Ganzzahl (Integer). Das Ergebnis der Rundung ist für Werte, die genau in der Mitte zwischen zwei Ganzzahlen liegen, nicht definiert. Das bedeutet, dass z. B. 1,5 entweder auf 1 oder auf 2 gerundet wird.

## G.10. Math > sign +/-



Gibt entweder 1 oder -1 aus, abhängig vom Vorzeichen des Eingangs (positive Zahlen erzeugen den Wert 1, negative den Wert -1; es wird niemals Null ausgegeben).

## G.11. Math > sqrt (>0)



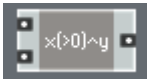
Berechnet näherungsweise die Quadratwurzel des Eingangs-Werts. Arbeitet nur mit Werten größer als 0.

## G.12. Math > sqrt



Berechnet näherungsweise die Quadratwurzel des Eingangs-Werts.

## G.13. Math > x(>0)^y



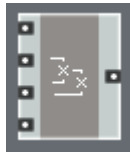
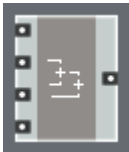
Angenäherte Bestimmung von  $x^y$ , wobei gelten muss  $x > 0$ . Das Ausgangs-Event wird jedes Mal gesendet, wenn ein Event an einem Eingang oder an beiden Eingängen gleichzeitig ankommt.

## G.14. Math > x^2 / x^3 / x^4



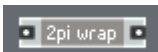
Berechnen die zweite, dritte und vierte Potenz von x.

## G.15. Math > Chain Add / Chain Mult



Addiert bzw. multipliziert die Signale in der Reihenfolge von oben nach unten. Das Ausgangs-Event wird gesendet, wenn an einem der Eingänge ein Event auftritt bzw. mehrere Events auftreten.

## G.16. Math > Trig-Hyp > 2 pi wrap



Faltet die ankommenden Werte in den Bereich  $[-\pi$  bis  $\pi]$  (die Faltungsperiode ist  $2\pi$ ).

### G.17. Math > Trig-Hyp > arcsin / arccos / arctan



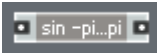
Arcsinus-/Arccosinus-/Arctangens-Approximation.

### G.18. Math > Trig-Hyp > sin / cos / tan



Sinus-/Cosinus-/Tangens-Approximation.

### G.19. Math > Trig-Hyp > sin -pi..pi / cos -pi..pi / tan -pi..pi



Sinus-/Cosinus-/Tangens-Approximation (arbeitet nur im Bereich  $[-\pi$  bis  $\pi$ ])

### G.20. Math > Trig-Hyp > tan -pi4..pi4



Tangens-Approximation (arbeitet nur im Bereich  $[-\pi/4.. \pi/4]$ ).

### G.21. Math > Trig-Hyp > sinh / cosh / tanh



Hyperbolische Sinus-/Cosinus-/Tangens-Approximation.

### G.22. Memory > Latch / ILatch



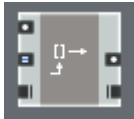
Verzögert das Signal am oberen Eingang, bis ein Clock-Event am unteren Eingang ankommt. Wenn beide Events gleichzeitig ankommen, wird das ankommende Signal sofort durchgelassen.

## G.23. Memory > $z^{-1}$ / $z^{-1}$ ndc



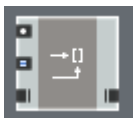
Senden als Reaktion auf ein Clock-Event den letzten Wert, den der obere Eingang empfangen hat, bevor das Clock-Event ankommt. Wenn der Clock-Eingang nicht angeschlossen ist, verwenden beide Versionen des Moduls stattdessen die Standard-Audio-Clock (SR.C) und verzögern das Signal effektiv um ein Sample. Beide Module können automatisch Feedback-Schleifen auflösen, allerdings beherrscht nur das Modul  $z^{-1}$  die Denormalen-Unterdrückung. Die Version  $z^{-1}$  ndc sollten Sie nur an Stellen verwenden, an denen keine Denormalen zu erwarten sind.

## G.24. Memory > Read []



Liest als Reaktion auf ein (am oberen Eingang) ankommendes Clock-Event einen Wert aus einem Array mit einem bestimmten Index (der vom mittleren Eingang spezifiziert wird). Der untere Eingang (OBC) stellt die Verbindung zum Array her. Verwenden Sie den OBC-Ausgang des Moduls, um OBC-Ketten für serielle Array-Zugriffe zu erstellen.

## G.25. Memory > Write []



Schreibt einen Wert (der am oberen Eingang empfangen wird) in ein Array an einen bestimmten Index (der vom mittleren Eingang spezifiziert wird). Der Schreib-Vorgang wird von einem ankommenden Wert getriggert. Der untere Eingang (OBC) stellt die Verbindung zum Array her. Verwenden Sie den OBC-Ausgang des Moduls, um OBC-Ketten für serielle Array-Zugriffe zu erstellen.

## G.26. Modulation > $x + a$ / Integer > $lx + a$



Fügt als Reaktion auf ein ankommendes Event einen Parameter (vom unteren Eingang) dem Signal des oberen Eingangs hinzu. Parameter-Änderungen erzeugen keine Events.

## G.27. Modulation > $x * a$ / Integer > $lx * a$



Multipliziert als Reaktion auf ein ankommendes Event das Signal (am oberen Eingang) mit einem Parameter (am unteren Eingang). Parameter-Änderungen erzeugen keine Events.

## G.28. Modulation > $x - a$ / Integer > $lx - a$



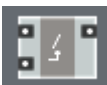
Subtrahiert als Reaktion auf ein ankommendes Event einen Parameter (am unteren Eingang) vom Signal (am oberen Eingang). Parameter-Änderungen erzeugen keine Events.

## G.29. Modulation > $a - x$ / Integer > $la - x$



Subtrahiert als Reaktion auf ein ankommendes Event das Signal (am unteren Eingang) von einem Parameter (am oberen Eingang). Parameter-Änderungen erzeugen keine Events.

## G.30. Modulation > $x / a$



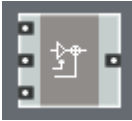
Dividiert als Reaktion auf ein ankommendes Event das Signal (am oberen Eingang) durch einen Parameter (am unteren Eingang). Parameter-Änderungen erzeugen keine Events.

### G.31. Modulation > a / x



Dividiert als Reaktion auf ein ankommendes Event einen Parameter (am oberen Eingang) durch ein Signal(am unteren Eingang). Parameter-Änderungen erzeugen keine Events.

### G.32. Modulation > xa + y



Multipliziert das Signal am oberen Eingang mit dem Gain-Parameter (am mittleren Eingang) und addiert das Ergebnis zum Signal am unteren Eingang. Events an einem Signal-Eingang oder an beiden Signal-Eingängen erzeugen einen neuen Ausgangs-Wert, Events am Parameter-Eingang nicht,



# Anhang H. Standard macros

## H.1. Audio Mix-Amp > Amount



Ermöglicht eine lineare, invertierbare Kontrolle der Amplitude eines Audio-Signals.

- |          |                              |
|----------|------------------------------|
| $A = 0$  | schaltet das Signal stumm    |
| $A = 1$  | lässt das Signal unverändert |
| $A = -1$ | invertiert das Signal        |

Typische Verwendung: Kontrolle der Stärke eines Audio-Feedbacks

## H.2. Audio Mix-Amp > Amp Mod



Moduliert die Amplitude des Audio-Signals mit einer bestimmten Stärke (AM) entlang einer linearen Skala.

- |           |                              |
|-----------|------------------------------|
| $AM = 1$  | verdoppelt die Amplitude     |
| $AM = 0$  | lässt das Signal unverändert |
| $AM = -1$ | schaltet das Signal stumm    |
- Typische Verwendung: Tremolo, Amplitudenmodulation.

## H.3. Audio Mix-Amp > Audio Mix



Mischt die beiden Audio-Signale zusammen.

## H.4. Audio Mix-Amp > Audio Relay



Schaltet zwischen zwei anliegenden Audio-Signalen um. Wenn  $x > 0$ , wird Signal 1 durchgelassen, andernfalls wird Signal 0 durchgelassen.

## H.5. Audio Mix-Amp > Chain (amount)

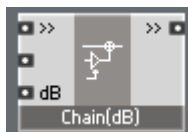


Verändert die Amplitude eines Audio-Signals um eine bestimmte lineare Größe (A) und mischt das Signal dem verketteten Audio-Signal (>>) hinzu.

- A = 0 schaltet das Signal stumm
- A = 1 lässt das Signal unverändert
- A = -1 invertiert das Signal

Typische Verwendung: Audio-Mixer-Ketten, Audio-Feedback-Regelung

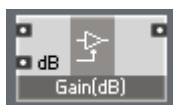
## H.6. Audio Mix-Amp > Chain (dB)



Verändert die Amplitude eines Audio-Signals um einen bestimmten Wert in dB und mischt das Signal dem verketteten Audio-Signal (>>) hinzu.

Typische Verwendung: Audio-Mixer-Ketten

## H.7. Audio Mix-Amp > Gain (dB)



Verändert die Amplitude eines Audio-Signals um einen bestimmten Wert in dB.

- +6 dB verdoppelt die Amplitude
- 0 dB lässt das Signal unverändert
- 6 dB halbiert die Amplitude

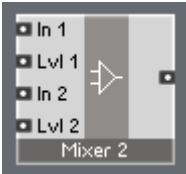
Typische Verwendung: Lautstärke-Kontrolle entlang einer dB-Skala

## H.8. Audio Mix-Amp > Invert



Kehrt die Polarität des Audio-Signals um.

## H.9. Audio Mix-Amp > Mixer 2 ... 4



Mischt die anliegenden Audio-Signale (In 1, In 2, ...) und schwächt dabei ihre Pegel um die angegebenen Werte (Lvl 1, Lvl 2, ...) in dB ab.

## H.10. Audio Mix-Amp > Pan



Bewegt die ankommenden Audio-Signale im Stereo-Panorama (mit parabolischem Kurvenverlauf). Die Panorama-Position ist folgendermaßen definiert:

- 1 ganz links
- 0 Mitte
- 1 ganz rechts

## H.11. Audio Mix-Amp > Ring-Amp Mod

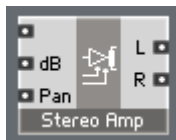


Das Carrier-Audio-Signal (am oberen Eingang) wird vom Signal am Eingang "Mod" moduliert. Die Art der Modulation kontrolliert der Eingang "R/A", der sanft zwischen Ringmodulation und Amplitudenmodulation überblendet.

- R/A = 0 Ringmodulation
- R/A = 1 Amplitudenmodulation

(Für echte Amplitudenmodulation sollte die Amplitude des Modulators den Wert 1 nicht überschreiten).

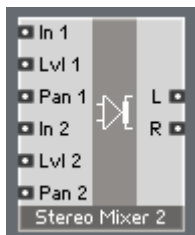
## H.12. Audio Mix-Amp > Stereo Amp



Verstärkt ein monophones Audio-Signal um einen bestimmten Wert in dB und platziert es an der angegebenen Position im Stereo-Panorama. Die Panorama-Position ist folgendermaßen definiert:

- 1 ganz links
- 0 Mitte
- 1 ganz rechts

## H.13. Audio Mix-Amp > Stereo Mixer 2 ... 4



Mischt die anliegenden Audio-Signale (In 1, In 2, ...), schwächt dabei ihre Pegel um die angegebenen Werte (Lvl 1, Lvl 2, ...) in dB ab und platziert sie auf den Positionen im Stereo-Panorama (Pan 1, Pan 2, ...). Die Panorama-Position ist folgendermaßen definiert:

- 1 ganz links
- 0 Mitte
- 1 ganz rechts

## H.14. Audio Mix-Amp > VCA



Audio-Verstärker mit direkter linearer Kontrolle über die Amplitude.

- A = 0 schaltet das Signal stumm
- A = 1 lässt das Signal unverändert

Typische Verwendung: Verbinden Sie die Amplituden-Hüllkurve mit dem Eingang "A".

Anmerkung: Verwenden Sie für umkehrbare Verstärkung das Modul Audio Amount.

## H.15. Audio Mix-Amp > XFade (lin)

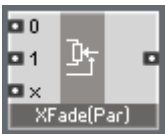


Audio-Crossfader mit linearem Kurvenverlauf

- |         |  |
|---------|--|
| x = 0   | nur das Signal 0 ist zu hören                    |
| x = 0.5 | beide Signale sind gleich stark im Mix vertreten |
| x = 1   | nur das Signal 1 ist zu hören                    |

Anmerkung: Mit einem parabolischen Crossfade erzielen Sie meistens besser klingende Ergebnisse.

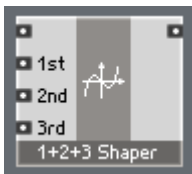
## H.16. Audio Mix-Amp > XFade (par)



Audio-Crossfader mit parabolischem Kurvenverlauf. Erzeugt meistens besser klingende Ergebnisse als ein linearer Crossfader.

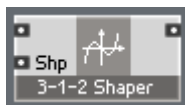
- |         |  |
|---------|--|
| x = 0   | nur das Signal 0 ist zu hören                    |
| x = 0.5 | beide Signale sind gleich stark im Mix vertreten |
| x = 1   | nur das Signal 1 ist zu hören                    |

## H.17. Audio Shaper > 1+2+3 Shaper



Stellt kontrollierbares Audio-Signal-Shaping zweiter und dritter Ordnung zur Verfügung. Der Eingang "1st" spezifiziert den Anteil des Original-Signals am Ausgangssignal (1= unverändert, 0=kein Anteil). Die Eingänge "2nd" und "3rd" legen den Anteil der Verzerrungen 2. respektive 3. Ordnung fest.

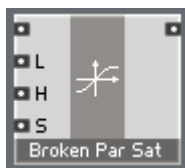
## H.18. Audio Shaper > 3-1-2 Shaper



Audio-Signal-Shaper mit veränderlichem Anteilen von Verzerrungen 2. und 3. Ordnung. Der Anteil und die Art der Verzerrung wird über den Eingang "Shp" kontrolliert:

Shp = 0	kein Shaping
Shp > 0	Shaping 3. Ordnung
Shp < 0	Shaping 2. Ordnung

## H.19. Audio Shaper > Broken Par Sat



Kaputter parabolischer Sättiger (Saturator). Hat ein lineares Segment um den Nullpegel herum.

Der Eingang "L" bestimmt den Ausgangspegel für die "volle Sättigung" (Default = 1).

Der Eingang "H" bestimmt die Härte (Bereich 0...1). Größere Werte bedingen ein größeres lineares Segment um den Nullpegel herum.

Der Eingang "S" kontrolliert die Symmetrie der Shaping-Kurve (Bereich -1...1). Am Wert 0 ist die Kurve symmetrisch.

## H.20. Audio Shaper > Hyperbol Sat



Einfacher hyperbolischer Sättiger. Der Eingang "L" bestimmt den Ausgangspegel für die "volle Sättigung" (Default = 1). Allerdings erreicht dieser Typ Sättiger niemals die "volle Sättigung".

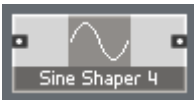
## H.21. Audio Shaper > Parabol Sat



Einfacher parabolischer Sättiger. Der Eingang “L” bestimmt den Ausgangspegel für die “volle Sättigung” (Default = 1).

Anmerkung: Die volle Sättigung wird bei einem Eingangspegel erreicht, der doppelt so hoch ist wie der Wert am Eingang L.

## H.22. Audio Shaper > Sine Shaper 4 / 8



Sinus-Shaper 4./8. Ordnung. Der Shaper 8. Ordnung hat eine bessere Sinus-Approximation, braucht aber mehr CPU-Leistung.

## H.23. Control > Ctl Amount



Ermöglicht lineare, umkehrbare Kontrolle der Amplitude des Kontroll-Signals.

- |        |                              |
|--------|------------------------------|
| A = 0  | schaltet das Signal ab       |
| A = 1  | lässt das Signal unverändert |
| A = -1 | invertiert das Signal        |

Typische Verwendung: Kontrolle des Modulationsgrads

## H.24. Control > Ctl Amp Mod



Moduliert die Amplitude des Kontroll-Signals um einen bestimmten Wert (AM) entlang einer linearen Skala.

- |         |                           |
|---------|---------------------------|
| AM = 1  | verdoppelt die Amplitude  |
| AM = 0  | keine Veränderung         |
| AM = -1 | schaltet das Signal stumm |

## H.25. Control > Ctl Bi2Uni



Formt ein bipolares Signal mit einem Wertebereich von  $-1$  bis  $1$  in ein unipolares Signal um. Der Eingang "a" kontrolliert den Grad der Umformung: Bei  $0$  findet keine Veränderung statt, bei  $1$  wird das Signal zu  $100\%$  geändert (Default =  $1$ ). Typische Verwendung: Schalten Sie dieses Modul direkt hinter einen LFO, um die Polarität der Modulation einzustellen.

## H.26. Control > Ctl Chain



Ändert die Amplitude des Kontroll-Signals um einen bestimmten Wert (A) und mischt das Signal dem verketteten Kontroll-Signal (>>) hinzu.

- |        |                              |
|--------|------------------------------|
| A = 0  | schaltet das Signal ab       |
| A = 1  | lässt das Signal unverändert |
| A = -1 | invertiert das Signal        |

Typische Verwendung: Verwenden Sie dieses Modul zur Kontrolle von Mixer-Ketten

## H.27. Control > Ctl Invert



Invertiert die Polarität des Mixer-Signals.

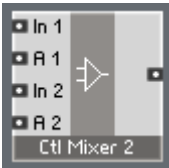
## H.28. Control > Ctl Mix



Mischt zwei Kontroll-Signale.



## H.29. Control > Ctl Mixer 2



Mischt zwei Kontroll-Signale (In 1, In 2) unter Verwendung bestimmter Gain-Werte (A 1, A 2) zusammen.

- A = 0      kein Signal
- A = 1      unverändertes Signal
- A = -1     invertiertes Signal

## H.30. Control > Ctl Pan



Platziert ein Kontroll-Signal im "Stereo-Panorama"; verwendet eine parabolische Kurve.

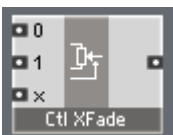
- Pos = -1    ganz links
- Pos = 0    Mitte
- Pos = 1    ganz rechts

## H.31. Control > Ctl Relay



Schaltet zwischen zwei Kontroll-Signalen um. Wenn  $x > 0$ , wird Signal 1 weitergeleitet; andernfalls wird Signal 0 weitergeleitet.

## H.32. Control > Ctl XFade



Führt eine Kreuzblende (Crossfade) zwischen zwei Kontroll-Signalen aus; verwendet eine lineare Kurve.

$x = 0$	nur Signal 0 kann passieren
$x = 0.5$	beide Signale sind gleich stark im Mix vertreten
$x = 1$	nur Signal 1 kann passieren

### H.33. Control > Par Ctl Shaper



Wendet eine doppelte parabolische Kurve auf ein Kontroll-Signal an. Das Eingangssignal muss im Bereich zwischen  $-1$  und  $1$  liegen. Das Ausgangssignal liegt ebenfalls im Bereich zwischen  $-1$  und  $1$ . Der Grad der Beugung des Signals wird über den Eingang "b" kontrolliert (dessen Bereich ebenfalls von  $-1$  bis  $1$  reicht).

$b = 0$	keine Beugung (linearer Kurvenverlauf)
$b = -1$	maximal mögliche Beugung auf die x-Achse zu
$b = 1$	maximal mögliche Beugung auf die y-Achse zu

Sie können diesen Shaper auch für Signale verwenden, deren Wertebereich von  $0$  bis  $1$  reicht; in diesem Fall wird nur eine Hälfte der Kurve genutzt. diesem Fall wird nur eine Hälfte der Kurve genutzt.

Typische Verwendung: Shaping von Velocity-Werten und anderen Kontroll-Signalen

### H.34. Convert > dB2AF



Konvertiert ein Kontroll-Signal aus der dB-Skala in einen linearen Amplituden-Verstärkungsfaktor.s

$0 \text{ dB}$	$\rightarrow$	$1.0$
$-6 \text{ dB}$	$\rightarrow$	$0.5$
etc.		

### H.35. Convert > dP2FF



Konvertiert ein Kontroll-Signal aus einem Tonhöhen-Intervall (in Halbtönen) in ein Frequenzverhältnis.

$12 \text{ Halbtöne}$	$\rightarrow$	$2$
$-12 \text{ Halbtöne}$	$\rightarrow$	$-2$
etc.		

### H.36. Convert > logT2sec



Konvertiert die auf REAKTORs Primary Level für Hüllkurven verwendete logarithmische Zeiteinteilung in Sekunden.

0 → 0.001 sec  
60 → 1 sec  
etc.

### H.37. Convert > ms2Hz



Konvertiert einen Zeitabschnitt (in Millisekunden) in die entsprechende Frequenz in Hz. Z. B. 100ms → 10 Hz.

### H.38. Convert > ms2sec



Konvertiert eine in Millisekunden angegebene Zeitdauer in Sekunden. Z. B. 500ms → 0,5 sec.

### H.39. Convert > P2F



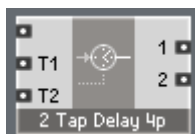
Konvertiert ein Kontroll-Signal aus der Tonhöhen-Skala in die Frequenz-Einteilung. Z. B. Note 69 → 440 Hz.

### H.40. Convert > sec2Hz



Konvertiert eine Zeitdauer in Sekunden in die entsprechende Frequenz in Hz. Z. B. 0.1sec → 10 Hz.

## H.41. Delay > 2 / 4 Tap Delay 4p



2/4-stufiges Delay mit 4-Punkt-Interpolation. Die Eingänge T1 bis T4 bestimmen die Delay-Zeit in Sekunden für jede der Stufen.

Die maximale Delay-Zeit liegt standardmäßig bei 44100 Samples, beträgt also 1s bei einer Sampling-Rate von 44,1kHz. Um die Delay-Zeit einzustellen, ändern Sie die Größe des Arrays im Delay-Macro.

## H.42. Delay > Delay 1p / 2p / 4p



Delay, das wahlweise als nicht interpoliertes 1-Punkt-Delay oder als interpoliertes 2-Punkt- respektive 4-Punkt-Delay arbeitet. Der Eingang "T" bestimmt die Delay-Zeit in Sekunden.

Die maximale Delay-Zeit liegt standardmäßig bei 44100 Samples, beträgt also 1s bei einer Sampling-Rate von 44,1kHz. Um die Delay-Zeit einzustellen, ändern Sie die Größe des Arrays im Delay-Macro.

Verwenden Sie die interpolierten Delay-Versionen für modulierte Delays. Für nicht modulierte (mit festen Zeiten versehene) Delays eignen sich nicht interpolierte Delay-Varianten normalerweise besser.

## H.43. Delay > Diff Delay 1p / 2p / 4p



Diffusions-Delay, das wahlweise als nicht interpoliertes 1-Punkt-Delay oder als interpoliertes 2-Punkt- respektive 4-Punkt-Delay arbeitet. Der Eingang "T" bestimmt den Diffusionsfaktor.

Die maximale Delay-Zeit liegt standardmäßig bei 44100 Samples, beträgt also 1s bei einer Sampling-Rate von 44,1kHz. Um die Delay-Zeit einzustellen, ändern Sie die Größe des Arrays im Delay-Macro.

## H.44. Envelope > ADSR

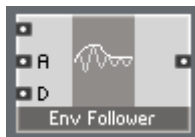


Erzeugt eine ADSR-Hüllkurve.

- A, D, R bestimmen die Zeiten für Attack, Decay und Release in Sekunden
- S bestimmt den Sustain-Pegel (im Bereich von 0 bis 1; bei 1 ist der Sustain-Pegel gleich dem Spitzenpegel)
- G Gate-Eingang. Positive ankommende Events starten die Hüllkurve (neu). Events mit dem Wert 0 oder mit negativen Werten schließen die Hüllkurve.
- GS Gate-Empfindlichkeit. Bei einer Empfindlichkeit von 0 hat der Spitzenpegel der Hüllkurve immer eine Amplitude von 1. Bei einer Empfindlichkeit von 1 ist der Spitzenpegel gleich dem positiven Gate-Pegel.
- RM Retrigger-Modus. Wählt zwischen analogem und digitalem Modus und zwischen Retrigger- und Legato-Modus. Im "digitalen" Modus startet die Hüllkurve immer bei Null neu, während sie im "analogen" Modus immer bei ihrem aktuellen Ausgangspegel startet. Im "Retrigger"-Modus starten nachfolgende positive Gate-Events die Hüllkurve neu, während die Hüllkurve im "Legato"-Modus nur dann neu startet, wenn das Gate seinen Wert von "negativ" oder "Null" in "positiv" ändert. Die erlaubten Werte für den Parameter "RM" sind:

- RM = 0 Analog-Retrigger (Default)
- RM = 1 Analog-Legato
- RM = 2 Digital-Retrigger
- RM = 3 Digital-Legato

## H.45. Envelope > Env Follower



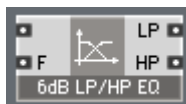
Erzeugt ein Kontroll-Signal, das der Hüllkurve des ankommenden Audio-Signals “folgt”. Die Eingänge “A” und “D” bestimmen die Attack- und Decay-Zeiten des “Verfolger-Signals” in Sekunden.

## H.46. Envelope > Peak Detector



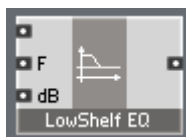
Gibt den letzten Spitzenpegel (Peak) des ankommenden Audio-Signals als Kontroll-Signal aus. Der Eingang “D” bestimmt die Decay-Zeit für den Ausgangspegel in Sekunden.

## H.47. EQ > 6dB LP/HP EQ



1-Pol-Tiefpass-/ Hochpass-EQ (6 dB/Oktave). Der Eingang “F” bestimmt die Cutoff-Frequenz (in Hz) sowohl für den Ausgang “LP” als auch für den Ausgang “HP”.

## H.48. EQ > 6dB LowShelf EQ



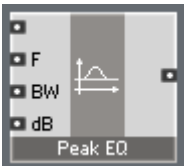
1-Pol-Low-Shelving-EQ. Der Eingang “dB” bestimmt die Stärke der Anhebung tiefer Frequenzen in dB (negative Werte beschneiden die tiefen Frequenzen), der Eingang “F” bestimmt die Mitten-Übergangsfrequenz in Hz.

## H.49. EQ > 6dB HighShelf EQ



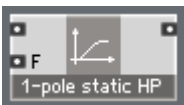
1-Pol-High-Shelving -EQ. Der Eingang “dB” bestimmt die Stärke der Anhebung hoher Frequenzen in dB (negative Werte beschneiden die hohen Frequenzen), der Eingang “F” bestimmt die Mitten-Übergangsfrequenz in Hz.

## H.50. EQ > Peak EQ



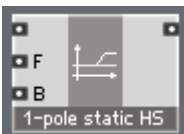
2-Pol-Peak- bzw. Notch-EQ. Der Eingang “F” bestimmt die Mittenfrequenz in Hz, der Eingang “BW” bestimmt die Bandbreite des Equalizers in Oktaven und der Eingang “dB” bestimmt die Höhe des Spitzenpegels (Peak). Negative Werte am Eingang “dB” erzeugen eine Kerbe (Notch).

## H.51. EQ > Static Filter > 1-pole static HP



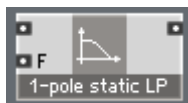
1-poliges statisches Hochpassfilter. Der Eingang “F” bestimmt die Cutoff-Frequenz in Hz.

## H.52. EQ > Static Filter > 1-pole static HS



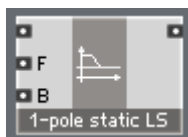
1-poliges statisches High-Shelving-Filter. Der Eingang “F” bestimmt die Cutoff-Frequenz in Hz, der Eingang “B” bestimmt die Stärke der Anhebung der hohen Frequenzen in dB.

### H.53. EQ > Static Filter > 1-pole static LP



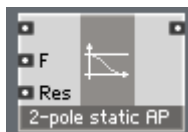
1-poliges statisches Tiefpassfilter. Der Eingang “F” bestimmt die Cutoff-Frequenz in Hz.

### H.54. EQ > Static Filter > 1-pole static LS



1-poliges statisches Low-Shelving-Filter. Der Eingang “F” bestimmt die Cutoff-Frequenz in Hz, der Eingang “B” bestimmt die Stärke der Anhebung der tiefen Frequenzen in dB.

### H.55. EQ > Static Filter > 2-pole static AP



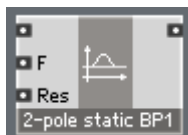
2-poliges statisches Allpassfilter. Der Eingang “F” bestimmt die Cutoff-Frequenz in Hz, der Eingang “Res” bestimmt die Resonanz (im Bereich von 0 bis 1).

### H.56. EQ > Static Filter > 2-pole static BP



2-poliges statisches Bandpassfilter. Der Eingang “F” bestimmt die Cutoff-Frequenz in Hz, der Eingang “Res” bestimmt die Resonanz (im Bereich von 0 bis 1).

### H.57. EQ > Static Filter > 2-pole static BP1





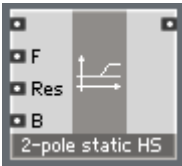
2-poliges statisches Bandpassfilter. Der Eingang "F" bestimmt die Cutoff-Frequenz in Hz, der Eingang "Res" bestimmt die Resonanz (im Bereich von 0 bis 1). Die Verstärkung an der Cutoff-Frequenz ist immer 1, ungeachtet der Resonanz.

### H.58. EQ > Static Filter > 2-pole static HP



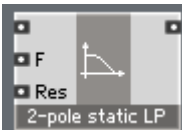
2-poliges statisches Hochpassfilter. Der Eingang "F" bestimmt die Cutoff-Frequenz in Hz, der Eingang "Res" bestimmt die Resonanz (im Bereich von 0 bis 1).

### H.59. EQ > Static Filter > 2-pole static HS



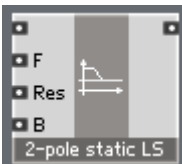
2-poliges statisches Hochpassfilter. Der Eingang "F" bestimmt die Cutoff-Frequenz in Hz, der Eingang "Res" bestimmt die Resonanz (im Bereich von 0 bis 1).

### H.60. EQ > Static Filter > 2-pole static LP



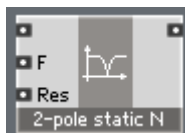
2-poliges statisches Tiefpassfilter. Der Eingang "F" bestimmt die Cutoff-Frequenz in Hz, der Eingang "Res" bestimmt die Resonanz (im Bereich von 0 bis 1).

### H.61. EQ > Static Filter > 2-pole static LS



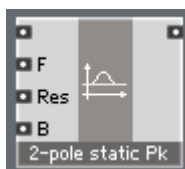
22-poliges statisches Low-Shelving-Filter. Der Eingang “F” bestimmt die Cutoff-Frequenz in Hz, der Eingang “Res” bestimmt die Resonanz (im Bereich von 0 bis 1). Der Eingang “B” bestimmt die Stärke der Anhebung der tiefen Frequenzen in dB.

## H.62. EQ > Static Filter > 2-pole static N



2-poliges statisches Notch-Filter. Der Eingang “F” bestimmt die Cutoff-Frequenz in Hz, der Eingang “Res” bestimmt die Resonanz (im Bereich von 0 bis 1).

## H.63. EQ > Static Filter > 2-pole static Pk



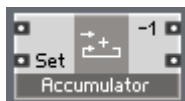
2-poliges statisches Peak-Filter. Der Eingang “F” bestimmt die Cutoff-Frequenz in Hz, der Eingang “Res” bestimmt die Resonanz (im Bereich von 0 bis 1). Der Eingang “B” bestimmt die Stärke der Anhebung an der Mittenfrequenz in dB.

## H.64. EQ > Static Filter > Integrator



Integriert das ankommende Audio-Signal unter Verwendung der Rechtecksummen-Methode. Ein Event am Eingang “Rst” setzt den Ausgang des Moduls auf den Wert dieses Events zurück.

## H.65. Event Processing > Accumulator



Berechnet die Summe der Werte am oberen Eingang. Ein Event am Eingang "Rst" setzt den Ausgang des Moduls auf den Wert dieses Events zurück. Der untere Ausgangs-Wert ist die Summe aller vorangegangenen Events, der obere Ausgangs-Wert ist die Summe aller vorangegangenen Events mit Ausnahme des letzten.

## H.66. Event Processing > Clk Div



Clock-Frequenzteiler. Die am oberen Eingang ankommenden Clock-Events werden so gefiltert, dass nur die Events (1),  $N+(1)$ ,  $2N+(1)$  und so weiter durchgelassen werden.  $N$  ist der Wert des unteren Eingangs und bestimmt das Teilungsverhältnis.

## H.67. Event Processing > Clk Gen



Erzeugt Clock-Events mit der vom Eingang (in Hz) festgelegten Rate. Dieses Modul arbeitet nur innerhalb von Audio-Core-Cells.

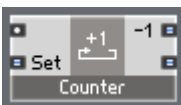
## H.68. Event Processing > Clk Rate



Schätzt die Rate und die Dauer ankommender Clock-Events. Der Ausgang "F" sendet die Rate in Hz, der Ausgang "T" die Dauer in Sekunden. Dieses Modul arbeitet nur innerhalb von Audio-Core-Cells.

Der anfängliche Wert für die Dauer ist Null, die Rate hat einen extrem großen Wert. Ein brauchbares Ergebnis erhalten Sie erst nach dem zweiten Clock-Event.

## H.69. Event Processing > Counter



Zählt die Anzahl der Events am oberen Eingang. Ein Event am Eingang "Rst"

setzt den Ausgang auf den Wert dieses Events zurück. Der untere Ausgangswert ist das Zählungsergebnis aller vorangegangenen Events, der untere Ausgangswert ist das Zählungsergebnis aller vorangegangenen Events mit Ausnahme des letzten.

## H.70. Event Processing > Ctl2Gate



Konvertiert ein Kontroll-Signal (oder Audio-Signal) am oberen Eingang in ein Gate-Signal mit der Amplitude, die der untere Eingang bestimmt. Positive Nulldurchgänge öffnen das Gate, negative Nulldurchgänge schließen das Gate.

## H.71. Event Processing > Dup Flt / IDup Flt



Filtert Events mit doppelten Werten (Duplikate) aus (nur Events mit Werten, die sich von denen ihrer Vorgänger unterscheiden, werden durchgelassen).

## H.72. Event Processing > Impulse



Erzeugt einen ein Sample langen Impuls mit der Amplitude 1 als Reaktion auf ein ankommendes Event. Dieses Modul arbeitet nur innerhalb von Audio-Core-Cells.

## H.73. Event Processing > Random



Erzeugt zufällige Zahlen als Reaktion auf ankommende Clock-Events. Der Wertebereich der Ausgabe reicht von -1 bis 1. Ein Event am Eingang "Seed" wird den Zufallsgenerator mit dem Wert dieses Events neu "besäen".

## H.74. Event Processing > Separator / ISeparator



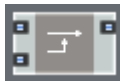
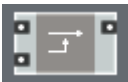
Am oberen Eingang ankommende Events mit Werten oberhalb des über den Eingang “Thld” bestimmten Werts werden an den Ausgang “Hi” geleitet. Alle anderen Events werden an den Ausgang “Lo” geleitet.

## H.75. Event Processing > Thld Crossing



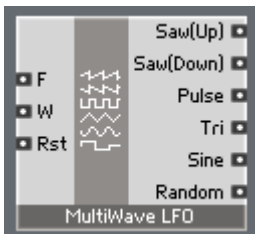
Jedes Mal, wenn ein ansteigendes Signal am oberen Eingang die vom Eingang “Thld” definierte Schwelle überschreitet, wird ein Event vom Ausgang “Up” gesendet. Wenn ein fallendes Signal die Schwelle überschreitet, wird ein Event vom Ausgang “Dn” gesendet.

## H.76. Event Processing > Value / IValue



Ändert den Wert eines am oberen Eingang ankommenden Events auf den zu diesem Zeitpunkt am unteren Eingang anliegenden Wert.

## H.77. LFO > MultiWave LFO



Erzeugt verschiedene phasenstarre Niederfrequenz-Wellenformen gleichzeitig. Der Eingang “F” bestimmt die Rate in Hz, der Eingang “W” kontrolliert die Pulsweite (im Bereich von -1 bis 1, betrifft nur den Ausgang “Pulse”). Am Eingang “Rst” ankommende Events starten den LFO in der durch den Event-Wert (im Bereich von 0 bis 1) bestimmten Phase neu.

## H.78. LFO > Par LFO



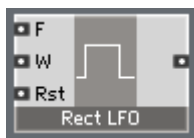
Erzeugt ein parabolisches Niederfrequenz-Kontroll-Signal. Der Eingang “F” bestimmt die Rate in Hz. Am Eingang “Rst” ankommende Events starten den LFO in der durch den Event-Wert (im Bereich von 0 bis 1) bestimmten Phase neu.

## H.79. LFO > Random LFO



Erzeugt ein gestuftes zufälliges Niederfrequenz-Kontroll-Signal. (“zufälliges Sample-and-Hold”). Der Eingang “F” bestimmt die Rate in Hz. Am Eingang “Rst” ankommende Events starten den LFO in der durch den Event-Wert (im Bereich von 0 bis 1) bestimmten Phase neu.

## H.80. LFO > Rect LFO



Erzeugt ein rechteckiges Niederfrequenz-Kontroll-Signal. Der Eingang “F” bestimmt die Rate in Hz, der Eingang “W” die Pulsweite (im Bereich von –1 bis 1). Am Eingang “Rst” ankommende Events starten den LFO in der durch den Event-Wert (im Bereich von 0 bis 1) bestimmten Phase neu.

## H.81. LFO > Saw(down) LFO



Erzeugt ein Niederfrequenz-Kontroll-Signal in Form einer fallenden Sägezahn-Welle. Der Eingang “F” bestimmt die Rate in Hz. Am Eingang “Rst” ankommende Events starten den LFO in der durch den Event-Wert (im Bereich von 0 bis 1) bestimmten Phase neu.

## H.82. LFO > Saw(up) LFO



Erzeugt ein Niederfrequenz-Kontroll-Signal in Form einer steigenden Sägezahn-Welle. Der Eingang “F” bestimmt die Rate in Hz. Am Eingang “Rst” ankommende Events starten den LFO in der durch den Event-Wert (im Bereich von 0 bis 1) bestimmten Phase neu.

## H.83. LFO > Sine LFO



Erzeugt ein sinusförmiges Niederfrequenz-Kontroll-Signal. Der Eingang “F” bestimmt die Rate in Hz. Am Eingang “Rst” ankommende Events starten den LFO in der durch den Event-Wert (im Bereich von 0 bis 1) bestimmten Phase neu.

## H.84. LFO > Tri LFO



Erzeugt ein Niederfrequenz-Kontroll-Signal in Form einer Dreieck-Welle. Der Eingang “F” bestimmt die Rate in Hz. Am Eingang “Rst” ankommende Events starten den LFO in der durch den Event-Wert (im Bereich von 0 bis 1) bestimmten Phase neu.

## H.85. Logic > AND



Verknüpft zwei logische Signale durch eine Konjunktion. Die Ausgabe ist nur dann 1, wenn beide Eingänge den Wert 1 haben; für andere Eingangs-Werte als 0 und 1 ist das Ergebnis undefiniert.

## H.86. Logic > Flip Flop



Der Ausgangs-Wert springt bei jedem am Clock-Eingang empfangenen Event zwischen den Werten 0 und 1 hin und her (auf den jeweils anderen Wert).

## H.87. Logic > Gate2L



Konvertiert ein Gate-Signal in ein logisches Signal. Ein offenes Gate erzeugt den Ausgangs-Wert 1, ein geschlossenes Gate erzeugt den Ausgangs-Wert 0.

## H.88. Logic > GT / IGT



Vergleicht die beiden ankommenden Werte (Version GT: Fließkomma; Version IGT: Integer) und gibt den Wert 1 aus, wenn der Wert am oberen Eingang größer als der Wert am unteren Eingang ist; andernfalls wird 0 ausgegeben.

## H.89. Logic > EQ



Vergleicht die beiden an den Eingängen ankommenden Integer-Werte und gibt den Wert 1 aus, wenn beide Eingangs-Werte gleich sind; andernfalls lautet die Ausgabe 0.

## H.90. Logic > GE



Vergleicht die beiden an den Eingängen ankommenden Integer-Werte und gibt den Wert 1 aus, wenn der obere Wert im Verhältnis zum unteren Wert größer oder gleich ist; andernfalls lautet die Ausgabe 0.



## H.91. Logic > L2Clock



Konvertiert ein Logik-Signal in ein Clock-Signal. Das Umschalten des Eingangssignals von 0 auf 1 sendet das Clock-Event. Für andere Eingangs-Werte als 0 und 1 ist die Funktion undefiniert.

## H.92. Logic > L2Gate



Konvertiert ein Logik-Signal in ein Gate-Signal. Das Umschalten des Eingangssignals von 0 auf 1 öffnet das Gate, beim Umschalten von 1 auf 0 wird das Gate geschlossen. Der Pegel für das offene Gate wird von dem Wert am unteren Eingang bestimmt (Default = 1). Für andere Eingangs-Werte als 0 und 1 ist die Funktion undefiniert.

## H.93. Logic > NOT



Konvertiert 1 in 0 und vice versa. Für andere Eingangs-Werte als 0 und 1 ist das Ergebnis undefiniert.

## H.94. Logic > OR



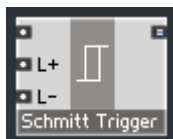
Führt eine Trennung (Disjunktion) zweier logischer Signale durch. Der Ausgang nimmt den Wert 1 an, wenn mindestens einer der Eingänge den Wert 1 hat. Für andere Eingangs-Werte als 0 und 1 ist das Ergebnis undefiniert.

## H.95. Logic > XOR



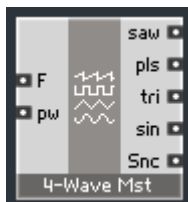
Führt eine exklusive Trennung (exklusive Disjunktion) zweier logischer Signale durch. Der Ausgang nimmt den Wert 1 an, wenn eins der beiden Eingangs-Signale den Wert 1 und das andere den Wert 0 hat. Für andere Eingangs-Werte als 0 und 1 ist das Ergebnis undefiniert.

## H.96. Logic > Schmitt Trigger



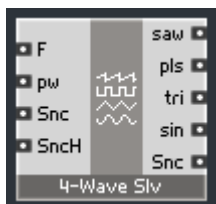
Schaltet den Ausgang auf 1, wenn der Eingangs-Wert größer wird als L+ (Default = 0.67); schaltet den Ausgang auf 0, wenn der Eingangs-Wert weniger als L- (Default = 0.33) wird.

## H.97. Oscillators > 4-Wave Mst



Erzeugt 4 phasenstarre Audio-Wellenformen. Der Eingang “F” bestimmt die Frequenz in Hz. Der Eingang “PW” bestimmt die Pulsweite (im Bereich von -1 bis 1; betrifft nur den Ausgang “pls”). Dieser Oszillator kann mit negativen Frequenzen schwingen und bietet zusätzlich einen Synchronisations-Ausgang für das Oszillator-Modul 4 Wave Slv.

## H.98. Oscillators > 4-Wave Slv



Erzeugt 4 phasenstarre Audio-Wellenformen. Der Eingang “F” bestimmt die Frequenz in Hz. Der Eingang “PW” bestimmt die Pulsweite (im Bereich von -1 bis 1; betrifft nur den Ausgang “pls”).

## H.99. Oscillators > Binary Noise



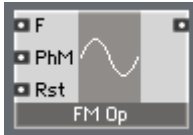
Binärer Rauschgenerator, der Weißes Rauschen erzeugt. Gibt zufällig alternierend die Werte 1 und -1 aus. Ein am Eingang "Seed" ankommendes Event initialisiert den internen Zufallsgenerator mit einem bestimmten Seed-Wert (neu).

## H.100. Oscillators > Digital Noise



Digitaler Rauschgenerator, der Weißes Rauschen erzeugt. Gibt zufällige Werte im Bereich zwischen -1 und 1 aus. Ein am Eingang "Seed" ankommendes Event initialisiert den internen Zufallsgenerator mit einem bestimmten Seed-Wert (neu).

## H.101. Oscillators > FM Op



Klassischer FM-Operator. Gibt eine Sinuswelle aus, deren Frequenz (in Hz) vom Eingang "F" bestimmt wird. Der Sinus kann über den Eingang "PhM" phasenmoduliert werden (in Radianten). Ein am Eingang "Rst" ankommendes Event startet den Oszillator an der durch den Wert des Events (im Bereich von 0 bis 1) spezifizierten Stelle neu.

## H.102. Oscillators > Formant Osc



Erzeugt eine Wellenform mit einer Grundfrequenz, die über den Eingang "F" festgelegt wird (in Hz); die Frequenz der Oberschwingungen (Formanten) wird über den Eingang "Fmt" bestimmt.

## H.103. Oscillators > MultiWave Osc



Erzeugt 4 phasenstarre Audio-Wellenformen. Der Eingang “F” bestimmt die Frequenz in Hz. Der Eingang “PW” bestimmt die Pulsweite (im Bereich von -1 bis 1; betrifft nur den Ausgang “pls”).  
Dieser Oszillator kann nicht mit negativen Frequenzen schwingen.

## H.104. Oscillators > Par Osc



Erzeugt eine parabolische Audio-Wellenform. Der Eingang “F” bestimmt die Frequenz in Hz.

## H.105. Oscillators > Quad Osc



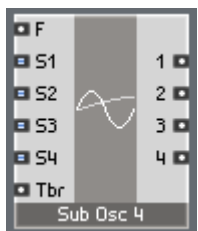
Erzeugt ein Paar phasenstarrer Sinus-Wellenformen mit einem Phasenversatz von 90 Grad. Der Eingang “F” bestimmt die Frequenz in Hz.

## H.106. Oscillators > Sin Osc



Erzeugt eine Sinuswelle. Der Eingang “F” bestimmt die Frequenz in Hz.

## H.107. Oscillators > Sub Osc 4



Erzeugt 4 phasenstarre Subharmonische. Die “grundlegende” Frequenz wird vom Eingang “F” bestimmt (in Hz). Die Nummern der Subharmonischen werden von den Eingängen S1 bis S4 spezifiziert (im Bereich zwischen 1 und 120). Der Eingang “Tbr” kontrolliert den harmonischen Gehalt der Ausgangs-Wellenform (im Bereich zwischen 0 und 1).

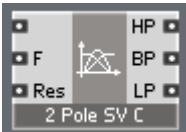
## H.108. VCF > 2 Pole SV



2-poliges Zustandsvariablenfilter (State Variable Filter). Der Eingang “F” bestimmt die Cutoff-Frequenz in Hz, der Eingang “Res” legt die Resonanz fest (im Bereich von 0 bis 0,98).

Die Ausgänge “HP”, “BP” und “LP” erzeugen Hochpass-, Bandpass- respektive Tiefpass-Signale.

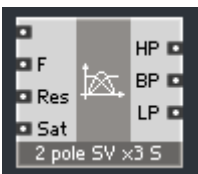
## H.109. VCF > 2 Pole SV C



2-poliges Zustandsvariablenfilter (kompensierte Version). Bietet ein verbessertes Verhalten bei hohen Cutoff-Einstellungen. Der Eingang “F” bestimmt die Cutoff-Frequenz in Hz, der Eingang “Res” legt die Resonanz fest (im Bereich von 0 bis 0,98). Sie können auch negative Resonanz-Werte verwenden, die das Abfallen des Signals weiter verwischen.

Die Ausgänge “HP”, “BP” und “LP” erzeugen Hochpass-, Bandpass- respektive Tiefpass-Signale.

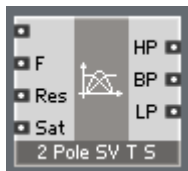
## H.110. VCF > 2 Pole SV (x3) S



2-poliges Zustandsvariablenfilter mit optionalem Oversampling (Version x3) und Sättigung. Der Eingang “F” bestimmt die Cutoff-Frequenz in Hz, der Eingang “Res” legt die Resonanz fest (im Bereich von 0 bis 1). Der Eingang “Sat” bestimmt den Grad der Sättigung (typischer Bereich: 8 bis 32).

Die Ausgänge “HP”, “BP” und “LP” erzeugen Hochpass-, Bandpass- respektive Tiefpass-Signale.

## H.111. VCF > 2 Pole SV T (S)



2-poliges Zustandsvariablenfilter mit Tabellen-Kompensation und optionaler Sättigung (Version S). Bietet ebenfalls ein verbessertes Verhalten bei hohen Cutoff-Einstellungen, klingt aber etwas anders als das Modul *2 Pole SV C*. Der Eingang “F” bestimmt die Cutoff-Frequenz in Hz, der Eingang “Res” legt die Resonanz fest (im Bereich von 0 bis 1). Der Eingang “Sat” bestimmt den Grad der Sättigung (typischer Bereich: 8 bis 32).

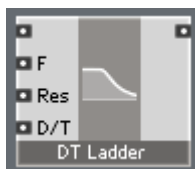
Die Ausgänge “HP”, “BP” und “LP” erzeugen Hochpass-, Bandpass- respektive Tiefpass-Signale.

## H.112. VCF > Diode Ladder



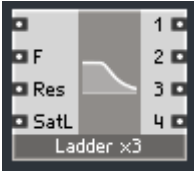
Lineare Nachbildung eines Dioden-Leiter-Filters. Der Eingang “F” bestimmt die Cutoff-Frequenz in Hz, der Eingang “Res” legt die Resonanz fest (im Bereich von 0 bis 0,98).

## H.113. VCF > D/T Ladder



Lineare Leiter-Filter-Nachbildung, die stufenlos zwischen Dioden- und Transistor-Leiter-Verhalten überblendet werden kann. Der Eingang “F” bestimmt die Cutoff-Frequenz in Hz, der Eingang “Res” legt die Resonanz fest (im Bereich von 0 bis 0,98). Der Eingang “D/T” blendet zwischen Diode und Transistor über (0 = Diode, 1 = Transistor)

## H.114. VCF > Ladder x3



Nachbildung eines Transistor-Leiter-Filters mit dreifachem Oversampling und Sättigung. Der Eingang “F” bestimmt die Cutoff-Frequenz in Hz, der Eingang “Res” legt die Resonanz fest (im Bereich von 0 bis 1). Der Eingang “SatL” bestimmt den Grad der Sättigung (typischer Bereich: 1 bis 32).

Die Ausgänge 1 bis 4 führen die Signale der entsprechenden Stufen der emulierten “Leiter”. Verwenden Sie die 4. Stufe für den klassischen Leiter-Filter-Sound.

# Anhang I. Core cell library

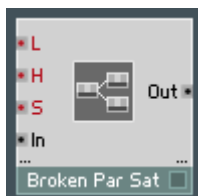
## I.1. Audio Shaper > 3-1-2 Shaper



Audio-Signal-Shaper mit veränderlichem Anteilen von Verzerrungen 2. und 3. Ordnung. Der Anteil und die Art der Verzerrung wird über den Eingang "Shp" kontrolliert:

Shp = 0	kein Shaping
Shp > 0	Shaping 3. Ordnung
Shp < 0	Shaping 2. Ordnung

## I.2. Audio Shaper > Broken Par Sat



Kaputter parabolischer Sättiger (Saturator). Hat ein lineares Segment um den Nullpegel herum.

Der Eingang "L" bestimmt den Ausgangspegel für die "volle Sättigung" (Default = 1).

Der Eingang "H" bestimmt die Härte (Bereich 0...1). Größere Werte bedingen ein größeres lineares Segment um den Nullpegel herum.

Der Eingang "S" kontrolliert die Symmetrie der Shaping-Kurve (Bereich -1...1). Am Wert 0 ist die Kurve symmetrisch.

## I.3. Audio Shaper > Hyperbol Sat



Einfacher hyperbolischer Sättiger. Der Eingang "L" bestimmt den Ausgangspegel für die "volle Sättigung" (Default = 1). Allerdings erreicht dieser Typ Sättiger niemals die "volle Sättigung".



## I.4. Audio Shaper > Parabol Sat



Einfacher parabolischer Sättiger. Der Eingang “L” bestimmt den Ausgangspegel für die “volle Sättigung” (Default = 1). Anmerkung: Die volle Sättigung wird bei einem Eingangspegel erreicht, der doppelt so hoch ist wie der Wert am Eingang L.

## I.5. Audio Shaper > Sine Shaper 4/8



Sinus-Shaper 4./8. Ordnung. Der Shaper 8. Ordnung hat eine bessere Sinus-Approximation, braucht aber mehr CPU-Leistung.

## I.6. Control > ADSR



Erzeugt eine ADSR-Hüllkurve.

- A, D, R bestimmen die Zeiten für Attack, Decay und Release in Sekunden
- S bestimmt den Sustain-Pegel (im Bereich von 0 bis 1; bei 1 ist der Sustain-Pegel gleich dem Spitzenpegel)
- G Gate-Eingang. Positive ankommende Events starten die Hüllkurve (neu). Events mit dem Wert 0 oder mit negativen Wertschließen die Hüllkurve.
- GS Gate-Empfindlichkeit. Bei einer Empfindlichkeit von 0 hat der Spitzenpegel der Hüllkurve immer eine Amplitude von 1. Bei einer Empfindlichkeit von 1 ist der Spitzenpegel gleich dem positiven Gate-Pegel.

**RM** Retrigger-Modus. Wählt zwischen analogem und digitalem Modus und zwischen Retrigger- und Legato-Modus. Im “digitalen” Modus startet die Hüllkurve immer bei Null neu, während sie im “analogen” Modus immer bei ihrem aktuellen Ausgangspegel startet. Im “Retrigger”-Modus starten nachfolgende positive Gate-Events die Hüllkurve neu, während die Hüllkurve im “Legato”-Modus nur dann neu startet, wenn das Gate seinen Wert von “negativ” oder “Null” in “positiv” ändert. Die erlaubten Werte für den Parameter “RM” sind:

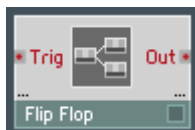
RM = 0	Analog-Retrigger (Default)
RM = 1	Analog-Legato
RM = 2	Digital-Retrigger
RM = 3	Digital-Legato

## I.7. Control > Env Follower



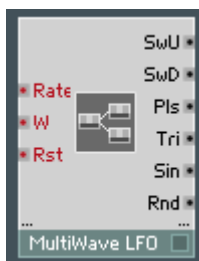
Erzeugt ein Kontroll-Signal, das der Hüllkurve des ankommenden Audio-Signals “folgt”. Die Eingänge “A” und “D” bestimmen die Attack- und Decay-Zeiten des “Verfolger-Signals” in Sekunden..

## I.8. Control > Flip Flop



Der Ausgangs-Wert springt bei jedem am Clock-Eingang empfangenen Event zwischen den Werten 0 und 1 hin und her (auf den jeweils anderen Wert).

## I.9. Control > MultiWave LFO



Erzeugt verschiedene phasenstarre Niederfrequenz-Wellenformen gleichzeitig. Der Eingang "F" bestimmt die Rate in Hz, der Eingang "W" kontrolliert die Pulsweite (im Bereich von -1 bis 1, betrifft nur den Ausgang "Pulse"). Am Eingang "Rst" ankommende Events starten den LFO in der durch den Event-Wert (im Bereich von 0 bis 1) bestimmten Phase neu.

## I.10. Control > Par Ctl Shaper



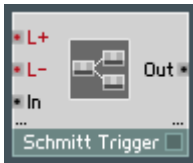
Wendet eine doppelte parabolische Kurve auf ein Kontroll-Signal an. Das Eingangssignal muss im Bereich zwischen -1 und 1 liegen. Das Ausgangssignal liegt ebenfalls im Bereich zwischen -1 und 1. Der Grad der Beugung des Signals wird über den Eingang "b" kontrolliert (dessen Bereich ebenfalls von -1 bis 1 reicht).

- b = 0                      keine Beugung (linearer Kurvenverlauf)
- b = -1                    maximal mögliche Beugung auf die x-Achse
- b = 1                      maximal mögliche Beugung auf die y-Achse zu

Sie können diesen Shaper auch für Signale verwenden, deren Wertebereich von 0 bis 1 reicht; in diesem Fall wird nur eine Hälfte der Kurve genutzt.

Typische Verwendung: Shaping von Velocity-Werten und anderen Kontroll-Signalen

## I.11. Control > Schmitt Trigger



Schaltet den Ausgang auf 1, wenn der Eingangs-Wert größer wird als L+ (Default = 0.67); schaltet den Ausgang auf 0, wenn der Eingangs-Wert weniger als L- (Default = 0.33) wird.

## I.12. Control > Sine LFO



Erzeugt ein sinusförmiges Niederfrequenz-Kontroll-Signal. Der Eingang “F” bestimmt die Rate in Hz. Am Eingang “Rst” ankommende Events starten den LFO in der durch den Event-Wert (im Bereich von 0 bis 1) bestimmten Phase neu.

### I.13. Delay > 2/4 Tap Delay 4p



2/4-stufiges Delay mit 4-Punkt-Interpolation. Die Eingänge T1 bis T4 bestimmen die Delay-Zeit in Sekunden für jede der Stufen.

Die maximale Delay-Zeit liegt standardmäßig bei 44100 Samples, beträgt also 1s bei einer Sampling-Rate von 44,1kHz. Um die Delay-Zeit einzustellen, ändern Sie die Größe des Arrays im Delay-Macro.

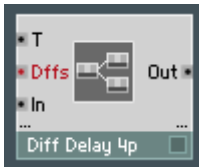
### I.14. Delay > Delay 4p



Interpoliertes 4-Punkt-Delay. Der Eingang “T” bestimmt die Delay-Zeit in Millisekunden.

Die maximale Delay-Zeit liegt standardmäßig bei 44100 Samples, beträgt also 1s bei einer Sampling-Rate von 44,1kHz. Um die Delay-Zeit einzustellen, ändern Sie die Größe des Arrays im Delay-Macro.

### I.15. Delay > Diff Delay 4p



Interpoliertes 4-Punkt-Diffusions-Delay. Der Eingang “T” bestimmt die Delay-Zeit in Millisekunden. Der Eingang “Dffs” bestimmt den Diffusionsfaktor.

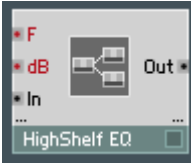
Die maximale Delay-Zeit liegt standardmäßig bei 44100 Samples, beträgt also 1s bei einer Sampling-Rate von 44,1kHz. Um die Delay-Zeit einzustellen, ändern Sie die Größe des Arrays im Delay-Macro.

## I.16. EQ > 6dB LP/HP EQ



1-Pol-Tiefpass-/ Hochpass-EQ (6 dB/Oktave). Der Eingang “F” bestimmt die Cutoff-Frequenz (in Hz).

## I.17. EQ > HighShelf EQ



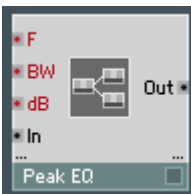
1-Pol-High-Shelving -EQ. Der Eingang “dB” bestimmt die Stärke der Anhebung hoher Frequenzen in dB (negative Werte beschneiden die hohen Frequenzen), der Eingang “F” bestimmt die Mitten-Übergangsfrequenz in Hz.

## I.18. EQ > LowShelf EQ



1-Pol-Low-Shelving-EQ. Der Eingang “dB” bestimmt die Stärke der Anhebung tiefer Frequenzen in dB (negative Werte beschneiden die tiefen Frequenzen), der Eingang “F” bestimmt die Mitten-Übergangsfrequenz in Hz.

## I.19. EQ > Peak EQ



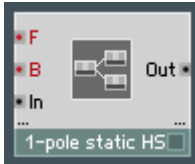
2-Pol-Peak- bzw. Notch-EQ. Der Eingang “F” bestimmt die Mittenfrequenz in Hz, der Eingang “BW” bestimmt die Bandbreite des Equalizers in Oktaven und der Eingang “dB” bestimmt die Höhe des Spitzenpegels (Peak). Negative Werte am Eingang “dB” erzeugen eine Kerbe (Notch).

## I.20. EQ > Static Filter > 1-pole static HP



1-poliges statisches Hochpassfilter. Der Eingang “F” bestimmt die Cutoff-Frequenz in Hz.

## I.21. EQ > Static Filter > 1-pole static HS



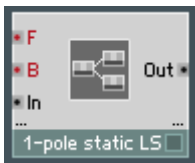
1-poliges statisches High-Shelving-Filter. Der Eingang “F” bestimmt die Cut-off-Frequenz in Hz, der Eingang “B” bestimmt die Stärke der Anhebung der hohen Frequenzen in dB.

## I.22. EQ > Static Filter > 1-pole static LP



1-poliges statisches Tiefpassfilter. Der Eingang “F” bestimmt die Cutoff-Frequenz in Hz.

## I.23. EQ > Static Filter > 1-pole static LS



1-poliges statisches Low-Shelving-Filter. Der Eingang “F” bestimmt die Cut-off-Frequenz in Hz, der Eingang “B” bestimmt die Stärke der Anhebung der tiefen Frequenzen in dB.

## I.24. EQ > Static Filter > 2-pole static AP



2-poliges statisches Allpassfilter. Der Eingang “F” bestimmt die Cutoff-Frequenz in Hz, der Eingang “Res” bestimmt die Resonanz (im Bereich von 0 bis 1).

## I.25. EQ > Static Filter > 2-pole static BP



2-poliges statisches Bandpassfilter. Der Eingang “F” bestimmt die Cutoff-Frequenz in Hz, der Eingang “Res” bestimmt die Resonanz (im Bereich von 0 bis 1).

## I.26. EQ > Static Filter > 2-pole static BP1



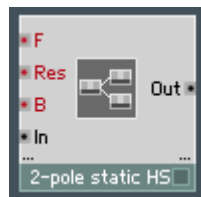
2-poliges statisches Bandpassfilter. Der Eingang “F” bestimmt die Cutoff-Frequenz in Hz, der Eingang “Res” bestimmt die Resonanz (im Bereich von 0 bis 1). Die Verstärkung an der Cutoff-Frequenz ist immer 1, ungeachtet der Resonanz.

## I.27. EQ > Static Filter > 2-pole static HP



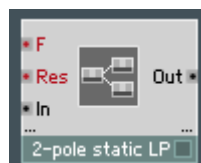
2-poliges statisches Hochpassfilter. Der Eingang “F” bestimmt die Cutoff-Frequenz in Hz, der Eingang “Res” bestimmt die Resonanz (im Bereich von 0 bis 1).

## I.28. EQ > Static Filter > 2-pole static HS



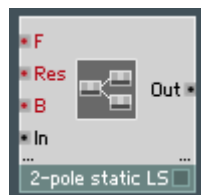
2-poliges statisches High-Shelving-Filter. Der Eingang "F" bestimmt die Cutoff-Frequenz in Hz, der Eingang "Res" bestimmt die Resonanz (im Bereich von 0 bis 1). Der Eingang "B" bestimmt die Stärke der Anhebung der hohen Frequenzen in dB.

## I.29. EQ > Static Filter > 2-pole static LP



2-poliges statisches Tiefpassfilter. Der Eingang "F" bestimmt die Cutoff-Frequenz in Hz, der Eingang "Res" bestimmt die Resonanz (im Bereich von 0 bis 1).

## I.30. EQ > Static Filter > 2-pole static LS



2-poliges statisches Low-Shelving-Filter. Der Eingang "F" bestimmt die Cutoff-Frequenz in Hz, der Eingang "Res" bestimmt die Resonanz (im Bereich von 0 bis 1). Der Eingang "B" bestimmt die Stärke der Anhebung der tiefen Frequenzen in dB.

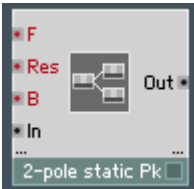


### I.31. EQ > Static Filter > 2-pole static N



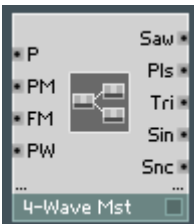
2-poliges statisches Notch-Filter. Der Eingang “F” bestimmt die Cutoff-Frequenz in Hz, der Eingang “Res” bestimmt die Resonanz (im Bereich von 0 bis 1).

### I.32. EQ > Static Filter > 2-pole static Pk



2-poliges statisches Peak-Filter. Der Eingang “F” bestimmt die Cutoff-Frequenz in Hz, der Eingang “Res” bestimmt die Resonanz (im Bereich von 0 bis 1). Der Eingang “B” bestimmt die Stärke der Anhebung an der Mittenfrequenz in dB.

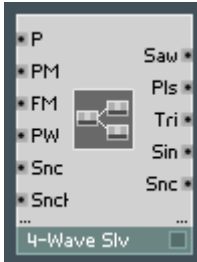
### I.33. Oscillator > 4-Wave Mst



Erzeugt 4 phasenstarke Audio-Wellenformen. Die Tonhöhe des Oszillators wird über den Eingang “P” durch eine MIDI-Noten-Nummer spezifiziert. Der Eingang “F” bestimmt die Frequenz in Hz. Der Eingang “PW” bestimmt die Pulsweite (im Bereich von -1 bis 1; betrifft nur den Ausgang “pls”). Kann über den Eingang “PM” (in Halbtonschritten, exponentiell) und über den Eingang “FM” (in Hz, linear) moduliert werden. Der Eingang “pw” bestimmt die Pulsweite der Puls-Wellenform (im Bereich zwischen - 1 und 1).

Dieser Oszillator kann mit negativen Frequenzen schwingen und bietet zusätzlich einen Synchronisations-Ausgang für das Oszillator-Modul 4 Wave Slv.

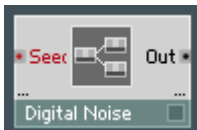
### I.34. Oscillator > 4-Wave Slv



Erzeugt 4 phasenstarre Audio-Wellenformen. Die Tonhöhe des Oszillators wird über den Eingang “P” durch eine MIDI-Noten-Nummer spezifiziert. Der Eingang “PW” bestimmt die Pulsweite (im Bereich von –1 bis 1; betrifft nur den Ausgang “pls”). Kann über den Eingang “PM” (in Halbtonschritten, exponentiell) und über den Eingang “FM” (in Hz, linear) moduliert werden. Der Eingang “pw” bestimmt die Pulsweite der Puls-Wellenform (im Bereich zwischen – 1 und 1).

Dieser Oszillator kann mit negativen Frequenzen schwingen und kann mit einem anderen Oszillator-Modul der Typen 4 Wave Mst oder 4 Wave Slv synchronisiert werden. Der Eingang “SncH” kontrolliert die Strenge der Synchronisation (0 = kein Sync, 1 = Hard Sync, 0...1 = verschiedene Abstufungen der Synchronisations-Strenge). Zum Synchronisieren eines weiteren Moduls des Typs 4 Wave Slv steht der Ausgang “Snc” zur Verfügung.

### I.35. Oscillator > Digital Noise



Digitaler Rauschgenerator, der Weißes Rauschen erzeugt. Gibt zufällige Werte im Bereich zwischen –1 und 1 aus. Ein am Eingang “Seed” ankommendes Event initialisiert den internen Zufallsgenerator mit einem bestimmten Seed-Wert (neu).

### I.36. Oscillator > FM Op



Klassischer FM-Operator. Gibt eine Sinuswelle aus, deren Tonhöhe über den Eingang “P” bestimmt wird (als MIDI-Noten-Nummer). Der Sinus kann über den Eingang “PhM” phasenmoduliert werden (in Radianen). Ein am Eingang “Rst” ankommendes Event startet den Oszillator in der durch den Wert des Events (im Bereich von 0 bis 1) spezifizierten Phase neu.

### I.37. Oscillator > Formant Osc



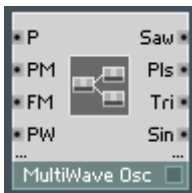
Erzeugt eine Wellenform mit einer Grundfrequenz, die über den Eingang “P” bestimmt wird (als MIDI-Noten-Nummer); die Frequenz der Oberschwingungen (Formanten) wird über den Eingang “Fmt” bestimmt.

### I.38. Oscillator > Impulse



Erzeugt einen ein Sample langen Impuls mit der Amplitude 1 als Reaktion auf ein ankommendes Event.

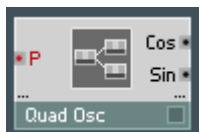
### I.39. Oscillator > MultiWave Osc



Erzeugt 4 phasenstarre Audio-Wellenformen. Die Tonhöhe des Oszillators wird über den Eingang “P” durch eine MIDI-Noten-Nummer spezifiziert. Kann über den Eingang “PM” (in Halbtonschritten, exponentiell) und über den Eingang “FM” (in Hz, linear) moduliert werden. Der Eingang “pw” bestimmt die Pulsweite der Puls-Wellenform (im Bereich zwischen – 1 und 1).

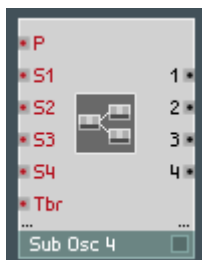
Dieser Oszillator kann nicht mit negativen Frequenzen schwingen.

## I.40. Oscillator > Quad Osc



Erzeugt ein Paar phasenstarrer Sinus-Wellenformen mit einem Phasenversatz von 90 Grad. Die Tonhöhe des Oszillators wird über den Eingang "P" durch eine MIDI-Noten-Nummer spezifiziert.

## I.41. Oscillator > Sub Osc



Erzeugt 4 phasenstarre Subharmonische. Die "grundlegende" Frequenz wird über den Eingang "P" durch eine MIDI-Noten-Nummer spezifiziert. Die Nummern der Subharmonischen werden von den Eingängen S1 bis S4 spezifiziert (im Bereich zwischen 1 und 120). Der Eingang "Tbr" kontrolliert den harmonischen Gehalt der Ausgangs-Wellenform (im Bereich zwischen 0 und 1).

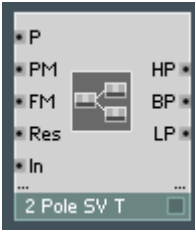
## I.42. VCF > 2 Pole SV C



2-poliges Zustandsvariablenfilter (kompensierte Version). Bietet ein verbessertes Verhalten bei hohen Cutoff-Einstellungen. Die Cutoff-Frequenz wird über den Eingang "P" durch eine MIDI-Noten-Nummer spezifiziert und kann über den Eingang "PM" (in Halbtonschritten, exponentiell) und über den Eingang "FM" (in Hz, linear) moduliert werden. Der Eingang "Res" legt die Resonanz fest (im Bereich von 0 bis 0,98). Sie können auch negative Resonanz-Werte verwenden, die das Abfallen des Signals weiter verwischen.

Die Ausgänge "HP", "BP" und "LP" erzeugen Hochpass-, Bandpass- respektive Tiefpass-Signale.

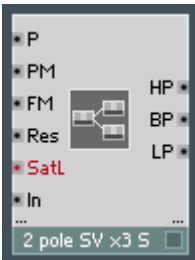
### I.43. VCF > 2 Pole SV T



2-poliges Zustandsvariablenfilter mit Tabellen-Kompensation und optionaler Sättigung (Version S). Bietet ebenfalls ein verbessertes Verhalten bei hohen Cutoff-Einstellungen, klingt aber etwas anders als das Modul 2 Pole SV C. Die Cutoff-Frequenz wird über den Eingang “P” durch eine MIDI-Noten-Nummer spezifiziert und kann über den Eingang “PM” (in Halbtonschritten, exponentiell) und über den Eingang “FM” (in Hz, linear) moduliert werden. Der Eingang “Res” legt die Resonanz fest (im Bereich von 0 bis 1).

Die Ausgänge “HP”, “BP” und “LP” erzeugen Hochpass-, Bandpass- respektive Tiefpass-Signale.

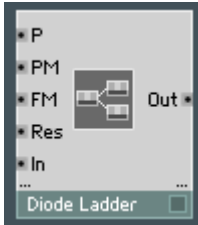
### I.44. VCF > 2 Pole SV x3 S



2-poliges Zustandsvariablenfilter mit optionalem Oversampling (Version x3) und Sättigung. Die Cutoff-Frequenz wird über den Eingang “P” durch eine MIDI-Noten-Nummer spezifiziert und kann über den Eingang “PM” (in Halbtonschritten, exponentiell) und über den Eingang “FM” (in Hz, linear) moduliert werden. Der Eingang “Res” legt die Resonanz fest (im Bereich von 0 bis 1). Der Eingang “Sat” bestimmt den Grad der Sättigung (typischer Bereich: 8 bis 32).

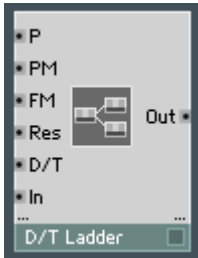
Die Ausgänge “HP”, “BP” und “LP” erzeugen Hochpass-, Bandpass- respektive Tiefpass-Signale.

## I.45. VCF > Diode Ladder



Nachbildung eines Dioden-Leiter-Filters. Die Cutoff-Frequenz wird über den Eingang "P" durch eine MIDI-Noten-Nummer spezifiziert und kann über den Eingang "PM" (in Halbtonschritten, exponentiell) und über den Eingang "FM" (in Hz, linear) moduliert werden. Der Eingang "Res" legt die Resonanz fest (im Bereich von 0 bis 0,98).

## I.46. VCF > D/T Ladder



Lineare Leiter-Filter-Nachbildung, die stufenlos zwischen Dioden- und Transistor-Leiter-Verhalten überblendet werden kann. Die Cutoff-Frequenz wird über den Eingang "P" durch eine MIDI-Noten-Nummer spezifiziert und kann über den Eingang "PM" (in Halbtonschritten, exponentiell) und über den Eingang "FM" (in Hz, linear) moduliert werden. Der Eingang "Res" legt die Resonanz fest (im Bereich von 0 bis 0,98). Der Eingang "D/T" blendet zwischen Diode und Transistor über (0 = Diode, 1 = Transistor).

## I.47. VCF > Ladder x3



Nachbildung eines Transistor-Leiter-Filters mit dreifachem Oversampling und Sättigung. Die Cutoff-Frequenz wird über den Eingang “P” durch eine MIDI-Noten-Nummer spezifiziert und kann über den Eingang “PM” (in Halbtonschritten, exponentiell) und über den Eingang “FM” (in Hz, linear) moduliert werden. Der Eingang “Res” legt die Resonanz fest (im Bereich von 0 bis 1). Der Eingang “SatL” bestimmt den Grad der Sättigung (typischer Bereich: 1 bis 32).

Die Ausgänge 1 bis 4 führen die Signale der entsprechenden Stufen der emulierten “Leiter”. Verwenden Sie die 4. Stufe für den klassischen Leiter-Filter-Sound.

# Index

## Symbols

1/96 Clock.....	49, 55
1 / Square Root .....	64
12-Step .....	116
16-Step .....	116
4-Ramp .....	89, 132
4-Step .....	88
5-Ramp .....	90, 133
5-Step .....	89
6-Ramp .....	90, 134
6-Step .....	89, 116
8-Ramp .....	90
8-Step .....	89, 116

## A

a * b+ c.....	59
Accumulator.....	169
AD-Env.....	125
ADBDR-Env.....	128
ADBDSR-Env.....	129
Add.....	57
ADR-Env.....	126
ADSR-Env.....	127
AHDBDR - Env.....	131
AHDSR - Env.....	130
Allpass 1-Pole.....	137
Amp/Mixer .....	70
AR-Env .....	126
Arccos.....	66
Arcsin.....	66
Arctan .....	66
Array-Module.....	326
A to E.....	187
A to E (Perm) .....	187
A to E (Trig) .....	187
A to Gate .....	188

Audio	
Inputs.....	290
Outputs.....	290
Audio Smoother .....	190
Audio Table .....	167
Audio Voice Combiner.....	185
Ausgänge. ....	<i>See</i> Ports

## B

Beat Loop .....	111
Bi-Pulse.....	86
Bi-Saw.....	74
Boolean-Control- Verbindungen (BoolCtl) .....	310
Button .....	24

## C

Cells. ....	<i>See</i> Core Cells
Ch. Aftertouch .....	48, 53
Chopper.....	162
Chopper-Modulator.....	162
Clipper.....	160
Clock-Signale.....	269
Clock Oscillator.....	91
Compare .....	61
Compare/Equal .....	61
Constant .....	57
Controller .....	47, 53
Core-Events. ....	<i>See</i> Events
Core-Struktur.....	211
Core Cells .....	205
(um)benennen.....	227
Audio.....	290
Event.....	263



Ports .....	223
User-Library .....	207
Verwendung.....	206, 209
Core Macros	
(um)benennen.....	241
erstellen.....	239
Parameter Solid .....	241, 299
Ports .....	240
User-Library .....	285
Core Module	
einsetzen .....	221
Integer-Modus.....	319, 321
Verarbeitungsreihenfolge	261, 271
Counter.....	170
Crossfade.....	68
Ctrl. Shaper 1 BP.....	171
Ctrl. Shaper 2 BP.....	172
Ctrl. Shaper 3 BP.....	172

## D

D-Env .....	122
DBDR-Env.....	123
DBDSR-Env.....	124
Debug-Modus .....	282
Default-Signale	
von Eingängen. ....	<i>See</i> Eingänge
Denormal Cancel Modul.....	303
Denormale Werte .....	301, 305
Differentiator .....	150
Diffuser Delay.....	154
Distributor.....	69
Divide x/y.....	59
DR-Env .....	122
DSR-Env .....	123
Dynamische Port-Verwaltung .....	20

## E

Eingänge.....	<i>See</i> Ports
Default-Signale von .....	242
ES Ctl Modul .....	324, 349
Events .....	256
gleichzeitige .....	259
Routing.....	310
Event Smoother .....	190
Event Table .....	179
Event V.C. All.....	186
Event V.C. Max .....	186
Event V.C. Min .....	186
Expon. (A).....	62
Expon. (F).....	63

## F

Fader.....	21
Feedback .....	293
Anzeige von .....	237, 294
Auflösung von .....	295, 321
um Macros herum .....	297
Fließkomma-	
Genauigkeit .....	241, 316, 345
Frequency Divider .....	166, 171
Frequenzteiler.....	166
From Voice .....	189

## G

Gate .....	46
Geiger.....	93
Genauigkeit	
Fließkomma.....	
.....	<i>See</i> Fließkomma-Genauigkeit
Grain Cloud .....	109
Grain Cloud Delay .....	156
Grain Delay .....	155
Grain Pitch Former .....	105
Grain Resynth.....	101

## H

H-Env.....	121
High Shelf EQ.....	148
High Shelf EQ FM.....	149
Hold.....	179
HP/LP 1-Pole.....	136
HP/LP 1-Pole FM.....	137
HR-Env.....	121
Hybrid-Module.....	19

## I

IC Receive.....	202
IC Send.....	202
Impulse.....	86
Impulse FM.....	87
Impulse Sync.....	87
INFs.....	305
Initialisierung.....	273, 323
Initialisierungs-Event.....	274, 281, 286, 291, 306, 323
In Port.....	200
Integrator.....	151
Invert, -X.....	58
Iteration.....	175

## K

Knob.....	24
-----------	----

## L

Ladder Filter.....	145
Ladder Filter FM.....	146
Lamp.....	29
Latch-Modul.....	269, 270, 280, 308, 312, 321, 348
Level Lamp.....	30
Level Meter.....	32
LFO.....	119

List.....	25
Log (A).....	63
Log (F).....	63
Logic-Signale.....	254
Logic AND.....	173
Logic EXOR.....	173
Logic NOT.....	174
Logic OR.....	173
Low Shelf EQ.....	149
Low Shelf EQ FM.....	150

## M

Macros.....	<i>See</i> Core-Macros
Master Tune/Level.....	191
Merge.....	176
Merge-Modul... ..	278, 279, 312, 349
Meter.....	31
MIDI Channel Info.....	194
Mirror 1 Level.....	161
Mirror 2 Levels.....	161
Mod. Clipper.....	160
Modulations Macros.....	287, 291, 321, 348
Module.....	<i>See</i> Core-Module
Modulo.....	60
Mouse Area.....	41
Multi-Ramp.....	89
Multi-Sine.....	81
Multi-Step.....	88
Multi-Tap Delay.....	153
Multi/HP 4-Pole.....	143
Multi/HP 4-Pole FM.....	144
Multi/LP 4-Pole.....	141
Multi/LP 4-Pole FM.....	142
Multi/Notch 2-Pole.....	139
Multi/Notch 2-Pole FM.....	140
Multi 2-Pole.....	138
Multi 2-Pole FM.....	138
Multi Display.....	38

Multi Picture .....	33
Multiplex16 .....	117
Multiply .....	58
Multi Text.....	34

## N

NaNs.....	305
Noise.....	92
Note Pitch.....	45
Note Pitch/Gate .....	52
Note Range Info.....	193

## O

Object Bus	
Connections (OBC) ..	270, 317, 327
Off Velocity .....	47
On Velocity.....	47
Order.....	174
OSC Receive .....	203
OSC Send .....	203
Out Port.....	200

## P

Panner.....	69
Parabol.....	77
Par FM .....	77
Par PWM .....	79
Par Sync .....	78
Peak Detector.....	166
Peak EQ.....	147
Peak EQ FM .....	147
Picture.....	33
Pitchbend .....	45, 52
Poly Aftertouch .....	48, 53
Poly Display.....	38
Ports	
Audio.....	290

Audio/Event .....	213
erzeugen .....	223
Event .....	263
Event-Sendereihenfolge von...	263
realtive Reihenfolge .....	211
Power x y .....	64
Precision	
floating point. ....	<i>See</i> FP Precision
Pro-52 Filter .....	145
Program Change .....	49, 54
Pulse.....	82
Pulse 1-ramp.....	84
Pulse 2-ramp.....	85
Pulse FM .....	83
Pulse Sync .....	83

## Q

QNaNs.....	305
Quantize .....	62
QuickBusses.....	229
QuickConst.....	242, 265

## R

R/W Order-Modul .....	331
Ramp .....	90
Random .....	92
Randomizer .....	170
Read-Modul .....	270, 276, 328
Initialisierung von .....	274
Read [] Macro .....	334
Receive.....	200
Reciprocal 1/x .....	59
Rect./Sign .....	60
Rectifier.....	60
Relay 1,2 .....	68
RGB Lamp .....	30
Router-Modul .....	310, 349
Router 1,2 .....	177

Router 1->M .....	178	Sin .....	65
Router M->1 .....	177	Sin/Cos .....	65
<b>S</b>		Sine .....	79
Sample & Hold .....	166	Sine FM .....	80
Sample Lookup .....	114	Sine Sync .....	80
Sampler .....	96	Single Delay .....	152
Sampler FM .....	97	Single Trig. Gate .....	46
Sampler Loop .....	98	Slew Limiter .....	165
Sampling-Rate-Clock .....	269,	Slow Random .....	120
.....292, 294, 298, 308, 314, 334		Snapshot .....	194
Saturator .....	159	Snap Value .....	197
Saturator 2 .....	159	Song Pos .....	50, 55
Saw FM .....	72	Square Root .....	64
Saw Pulse .....	74	SŠgezahn/Nadelpuls-Oszillator ...	74
Saw Sync .....	73	Stacked Macro .....	44
Sawtooth .....	72	Start/Stop .....	49, 54
Scanner .....	67	Step Filter .....	176
Schlechte Werte .....	301, 305	Stereo Amp .....	71
Scope .....	37	Stereo Pan .....	69
Sel. Note Gate .....	46	Subtract .....	58
Sel. Poly AT .....	48, 54	Switch .....	26
Selector .....	67	Sync Clock .....	50
Send .....	200	System Info .....	193
Separator .....	175	<b>T</b>	
Sequencer .....	115	Tabellen-Modul .....	342
Set Random .....	196	Tapedeck 1-Ch .....	182
Shaper 1 BP .....	162	Tapedeck 2-Ch .....	185
Shaper 2 BP .....	163	Tempo Info .....	191
Shaper 3 BP .....	163	Text .....	34
Shaper Cubic .....	165	Timer .....	178
Shaper Parabolic .....	164	To Voice .....	188
Signale		Tri/Par Symm .....	76
Audio .....	232, 290	Triangle .....	75
Clock .....	<i>See</i> Clock-Signale	Tri FM .....	75
Event .....	249	Tri Sync .....	76
Fließkomma (Float) .....	315, 351	Tuning Info .....	192
Integer .....	317, 351		
Kontroll- .....	232		

## **U**

Unison Spread.....	196
Unit Delay.....	158

## **V**

Value.....	176
Voice Info.....	192
Voice Shift .....	189
Vorzeichen-Vergleich.....	322

## **W**

Write Modul.....	270, 276, 323, 328
Write [] macro .....	330

## **X**

XY.....	35
---------	----

## **Z**

Z1 Modul .....	272, 294,
.....	295, 308, 321