



REAKTOR 5

REFERENCIA AL NÚCLEO Y MÓDULO

La información contenida en este documento está sujeta a cambios sin previo aviso y no representa compromiso alguno por parte de NATIVE INSTRUMENTS GmbH. El software descrito en este documento está sujeto a un acuerdo de licencia y no puede ser copiado a otros medios. Ninguna parte de esta publicación puede ser copiada, reproducida, almacenada o transmitida de manera alguna ni por ningún medio y para ningún propósito sin el permiso escrito previo de NATIVE INSTRUMENTS GmbH, de aquí en más mencionado como NATIVE INSTRUMENTS. Todos los productos y nombres de compañías son marcas registradas de sus respectivos propietarios.

Por lo demás, el hecho de que estés leyendo este texto significa que eres el propietario de una versión legal y no de una copia ilegal. NATIVE INSTRUMENTS GmbH puede seguir creando y desarrollando software de audio innovador sólo gracias a gente honesta y legal como tú. Muchas gracias en nombre de toda la empresa.

Esta guía del usuario fue escrita por: NATIVE INSTRUMENTS, Len Sasso

Un agradecimiento especial par el Beta Test Team, cuya valiosa colaboración no solo estuvo en rastrear errores, sino en hacer de éste un mejor producto.



NATIVE INSTRUMENTS

© NATIVE INSTRUMENTS GmbH, 2007. Todos los derechos reservados.

Alemania

NATIVE INSTRUMENTS GmbH

Schlesische Str. 28

D-10997 Berlin

Germany

info@native-instruments.de

www.native-instruments.de

Estados Unidos de América

NATIVE INSTRUMENTS North America, Inc.

5631 Hollywood Boulevard

Los Angeles, CA 90028

USA

sales@native-instruments.com

www.native-instruments.com

Contenido

Referencia de los Módulos	19
Panel	21
Fader	21
Knob.....	24
Button	24
List.....	25
Switch.....	26
Lamp	28
Level Lamp.....	29
RGB Lamp.....	29
Meter	30
Level Meter.....	31
Picture	31
Multi Picture.....	32
Text	33
Multi Text	33
XY	33
Scope	35
Multi Display y Poly Display	36
Mouse Area	38
Stacked Macro	40
MIDI In.....	41
Note Pitch	41
Pitchbend.....	41
Gate	42
Single Trig. Gate.....	42
Sel. Note Gate	43
On Velocity	43
Off Velocity.....	43
Controller	43
Ch. Aftertouch	44
Poly Aftertouch	44
Sel. Poly AT	45
Program Change.....	45
Start/Stop	45
1/96 Clock.....	46
Sync Clock	46
Song Pos.....	47
Channel Message	47

MIDI Out.....	48
Note Pitch/Gate.....	48
Pitchbend.....	48
Controller	49
Ch. Aftertouch	49
Poly Aftertouch	49
Sel. Poly AT	50
Program Change.....	50
Start/Stop	50
1/96 Clock.....	51
Song Pos.....	51
Channel Message	51
Math.....	52
Constant	53
Add	53
Subtract.....	53
Invert, -X.....	53
Multiply.....	54
$a * b + c$	54
Reciprocal $1/x$	54
Divide x/y	55
Modulo $x \% y$	55
Rectifier	56
Rect./Sign	56
Compare	56
Compare/Equal	57
Quantize.....	57
Expon. (A)	58
Expon. (F)	58
Log (A).....	58
Log (F).....	59
Power x^y	59
Square Root.....	59
$1 / \text{Square Root}$	60
Sine.....	60
Sine/Cos.....	60
Arcsin	61
Arccos.....	61
Arctan.....	61

Signal Path	62
Selector/Scanner	62
Relay 1,2.....	63
Crossfade	63
Distributor/Panner	64
Stereo Pan.....	64
Amp/Mixer.....	65
Stereo Amp/Mixer	65
Oscillator	66
Sawtooth.....	66
Saw FM.....	67
Saw Sync	67
Saw Pulse	68
Bi-Saw	68
Triangle	69
Tri FM.....	69
Tri Sync.....	70
Tri/Par Symm	70
Parabol	71
Par FM.....	72
Par Sync	72
Par PWM.....	73
Sine.....	74
Sine FM	74
Sine Sync	75
Multi-Sine	76
Pulse	77
Pulse FM.....	78
Pulse Sync	79
Pulse 1-ramp	80
Pulse 2-ramp	80
Bi-Pulse	81
Impulse.....	82
Impulse FM	82
Impulse Sync	82
Multi-Step	83
4-Step	83
5-Step	84
6-Step	84
8-Step	84

Multi-Ramp	84
4-Ramp.....	84
5-Ramp.....	85
6-Ramp.....	85
8-Ramp.....	85
Ramp.....	85
Clock Oscillator	86
Noise	87
Random	87
Geiger	87
Samplers.....	88
Sampler	90
Sampler FM.....	91
Sampler Loop	92
Grain Resynth	94
Grain Pitch Former	98
Grain Cloud	102
Beat Loop	104
Sample Lookup	106
Sequencer.....	108
Sequencer	108
6-Step	108
8-Step	109
12-Step.....	109
16-Step.....	109
Multiplex 16	110
LFO, Envelope.....	111
LFO	111
Slow Random.....	112
H - Env	113
HR - Env	113
D - Env.....	114
DR - Env	114
DSR - Env	115
DBDR - Env	116
DBDSR-Env	117
AD - Env.....	118
AR - Env.....	118
ADR-Env	119
ADSR - Env	120

ADBDR - Env	121
ADBDSR-Env	122
AHDSR - Env	123
AHDBDR - Env	124
4-Ramp.....	125
5-Ramp.....	126
6-Ramp.....	128
Filter.....	130
HP/LP 1-Pole	130
HP/LP 1-Pole FM	131
Allpass 1-Pole	131
Multi 2-Pole.....	132
Multi 2-Pole FM	132
Multi/Notch 2-Pole	133
Multi/Notch 2-Pole FM.....	134
Multi/LP 4-Pole.....	135
Multi/LP 4-Pole FM	136
Multi/HP 4-Pole	137
Multi/HP 4-Pole FM.....	138
Pro-52 Filter	139
Ladder Filter	139
Ladder Filter FM	140
Peak EQ.....	141
Peak EQ FM	141
High Shelf EQ.....	142
High Shelf EQ FM	143
Low Shelf EQ	143
Low Shelf EQ FM	144
Differentiator	144
Integrator	145
Delay	145
Single Delay.....	145
Multi-Tap Delay	147
Diffuser Delay	147
Delay Granular	148
Grain Cloud Delay.....	150
Unit Delay	152
Audio Modifier.....	152
Saturator	152

Saturator 2	153
Clipper	153
Mod. Clipper	154
Mirror 1 Level	154
Mirror 2 Levels	155
Chopper	155
Shaper 1 BP	156
Shaper 2 BP	156
Shaper 3 BP	157
Shaper Parabolic	158
Shaper Cubic	158
Slew Limiter	159
Peak Detector	159
Sample & Hold	160
Frequency Divider	160
Audio Table	161
Event Processing	163
Accumulator	163
Counter	163
Randomizer	164
Frequency Divider	164
Ctrl. Shaper 1 BP	165
Ctrl. Shaper 2 BP	165
Ctrl. Shaper 3 BP	166
Logic AND	166
Logic OR	167
Logic EXOR	167
Logic NOT	167
Order	168
Iteration	168
Separator	169
Value	169
Merge	170
Step Filter	170
Router M->1	170
Router 1,2	171
Router 1->M	171
Timer	172
Hold	172
Event Table	173

Auxiliary	175
Tapedeck 1-Ch	175
Tapedeck 2-Ch	178
Audio Voice Combiner	179
Event V.C. All	179
Event V.C. Max	179
Event V.C. Min	180
A to E	180
A to E (Trig)	180
A to E (Perm)	181
A to Gate	181
To Voice	182
From Voice	182
Voice Shift	183
Audio Smoother	183
Event Smoother	184
Master Tune/Level	185
Tempo Info	186
Voice Info	186
Tuning Info	186
System Info	187
Note Range Info	187
MIDI Channel Info	188
Snapshot	188
Set Random	191
Unison Spread	191
Snap Value	191
Snap Value Array	192
Terminal	194
In Port	194
Out Port	194
Send	194
Receive	194
IC Send	196
IC Receive	196
OSC Send	197
OSC Receive	197
Apéndice	198

Primeros pasos en Reaktor Core	199
Qué es Reaktor Core	199
Usar células core.....	200
Usar células core en un ejemplo real	203
Edición básica de células core	205
Entra en Reaktor Core	211
Células core de evento y de audio.....	211
Crea tu primera célula core.....	212
Señales de audio y de control	224
Construye tus primeras macros de Reaktor Core.....	231
El audio como señal de control	238
Event signals.....	239
Señales lógicas	244
Fundamentos de Reaktor Core: modelo de señal corel	246
Valores.....	246
Eventos	246
Simultaneous events	249
Orden de procesamiento	250
Revisión de las células core de evento.....	252
Estructuras con categoría interna.....	258
Señales de reloj.....	258
Conexiones de Bus de objeto	259
4.3. Inicialización	262
Construye un acumulador de eventos	265
Event merging.....	266
Event accumulator with reset and initialization	268
Arregla el modelador de eventos.....	274
Audio processing at its core	277
Audio signals	277
Bus de reloj de frecuencia de muestreo.....	279
Connection feedback	280
Feedback en torno a macros	283
Valores anormales	287
Otras cifras peligrosas.....	291
Construir un filtro paso-bajo de 1-pole	292
Procesamiento condicional	295
Rutas de evento	295
Construir una macro de recorte de señales	297
Construir un oscilador de diente de sierra simple	299

Más tipos de señales.....	301
Señales flotantes.....	301
Señales enteras.....	303
Construir un contador de evento.....	306
Construir una macro como contador de tramos ascendentes.....	307
Series.....	310
Introducción a las series.....	310
Construir un selector de señales de audio.....	313
Construye un delay.....	319
Tablas.....	326
Construir estructuras óptimas.....	330
Macros.....	330
Rutear y asociar.....	331
Operaciones numéricas.....	332
Conversión entre flotantes y enteros.....	333
Apéndice A. El interfaz de usuario de Reaktor Core.....	335
A.1. Células core.....	335
A.2. Módulos/macros core.....	335
A.3. Puertos Core.....	336
A.4. Edición de estructuras core.....	336
Apéndice B. Conceptos de Reaktor Core.....	338
B.1. Señales y eventos.....	338
B.2. Inicialización.....	338
B.3. Conexiones OBC.....	338
B.4. Routing.....	339
B.5. El cerrojo.....	339
B.6. Clocking.....	339
Apéndice C. Puertos macro core.....	340
C.1. Entrada.....	340
C.2. Salida.....	340
C.3. Cerrojo (entrada).....	340
C.4. Cerrojo (salida).....	340
C.5. Bool C (entrada).....	340
C.6. Bool C (salida).....	340
Apéndice D. Apéndice D. Puertos de células core.....	341
D.1. Entrada (modo audio).....	341
D.2. Salida (modo audio).....	341
D.3. Entrada (modo evento).....	341
D.4. Salida (modo evento).....	341

Apéndice E. Buses incorporados.....	342
E.1. SR.C	342
E.2. SR.R	342
Apéndice F. Módulos internos	342
F.1. Const.....	342
F.2. Math > +.....	342
F.3. Math > -	342
F.4. Math > *	343
F.5. Math > /.....	343
F.6. Math > x	343
F.7. Math > -x	343
F.8. Math > DN Cancel	343
F.9. Math > ~log.....	344
F.10. Math > ~exp	344
F.11. Bit > Bit AND.....	344
F.12. Bit > Bit OR.....	344
F.13. Bit > Bit XOR	345
F.14. Bit > Bit NOT.....	345
F.15. Bit > Bit <<	345
F.16. Bit > Bit >>	345
F.17. Flow > Router.....	346
F.18. Flow > Compare	346
F.19. Flow > Compare Sign	346
F.20. Flow > ES Ctl	347
F.21. Flow > ~BoolCtl	347
F.22. Flow > Merge.....	347
F.23. Flow > EvtMerge	347
F.24. Memory > Read	348
F.25. Memory > Write	348
F.26. Memory > R/W Order.....	348
F.27. Memory > Array	349
F.28. Memory > Size []	349
F.29. Memory > Index.....	349
F.30. Memory > Table	350
F.31. Macro	350
Apéndice G. Macros expertas	351
G.1. Clipping > Clip Max / IClip Max	351
G.2. Clipping > Clip Min / IClip Min	351
G.3. Clipping > Clip MinMax / IClipMinMax.....	351

G.4. Math > 1 div x	351
G.5. Math > 1 wrap	351
G.6. Math > lmod	352
G.7. Math > Max / IMax	352
G.8. Math > Min / IMin	352
G.9. Math > round	352
G.10. Math > sign +-	352
G.11. Math > sqrt (>0)	353
G.12. Math > sqrt	353
G.13. Math > x(>0)^y	353
G.14. Math > x^2 / x^3 / x^4	353
G.15. Math > Chain Add / Chain Mult	353
G.16. Math > Trig-Hyp > 2 pi wrap	353
G.17. Math > Trig-Hyp > arcsin / arccos / arctan	354
G.18. Math > Trig-Hyp > sin / cos / tan	354
G.19. Math > Trig-Hyp > sin -pi..pi / cos -pi..pi / tan -pi..pi	354
G.20. Math > Trig-Hyp > tan -pi4..pi4	354
G.21. Math > Trig-Hyp > sinh / cosh / tanh	354
G.22. Memory > Latch / lLatch	354
G.23. Memory > z^-1 / z^-1 ndc	354
G.24. Memory > Read []	355
G.25. Memory > Write []	355
G.26. Modulation > x + a / Integer > lx + a	355
G.27. Modulation > x * a / Integer > lx * a	356
G.28. Modulation > x - a / Integer > lx - a	356
G.29. Modulation > a - x / Integer > la - x	356
G.30. Modulation > x / a	356
G.31. Modulation > a / x	356
G.32. Modulation > xa + y	357
Apéndice H. Macros estándar	357
H.1. Audio Mix-Amp > Amount	357
H.2. Audio Mix-Amp > Amp Mod	357
H.3. Audio Mix-Amp > Audio Mix	358
H.4. Audio Mix-Amp > Audio Relay	358
H.5. Audio Mix-Amp > Chain (amount)	358
H.6. Audio Mix-Amp > Chain (dB)	358
H.7. Audio Mix-Amp > Gain (dB)	359
H.8. Audio Mix-Amp > Invert	359
H.9. Audio Mix-Amp > Mixer 2 ... 4	359
H.10. Audio Mix-Amp > Pan	359

H.11. Audio Mix-Amp > Ring-Amp Mod.....	360
H.12. Audio Mix-Amp > Stereo Amp	360
H.13. Audio Mix-Amp > Stereo Mixer 2 ... 4.....	360
H.14. Audio Mix-Amp > VCA.....	361
H.15. Audio Mix-Amp > XFade (lin).....	361
H.16. Audio Mix-Amp > XFade (par)	361
H.17. Audio Shaper > 1+2+3 Shaper.....	362
H.18. Audio Shaper > 3-1-2 Shaper.....	362
H.19. Audio Shaper > Broken Par Sat	362
H.20. Audio Shaper > Hyperbol Sat.....	363
H.21. Audio Shaper > Parabol Sat.....	363
H.22. Audio Shaper > Sine Shaper 4 / 8	363
H.23. Control > Ctl Amount.....	363
H.24. Control > Ctl Amp Mod	364
H.25. Control > Ctl Bi2Uni	364
H.26. Control > Ctl Chain.....	364
H.27. Control > Ctl Invert.....	364
H.28. Control > Ctl Mix.....	365
H.29. Control > Ctl Mixer 2	365
H.30. Control > Ctl Pan	365
H.31. Control > Ctl Relay	365
H.32. Control > Ctl XFade	366
H.33. Control > Par Ctl Shaper	366
H.34. Convert > dB2AF	366
H.35. Convert > dP2FF.....	366
H.36. Convert > logT2sec.....	367
H.37. Convert > ms2Hz.....	367
H.38. Convert > ms2sec	367
H.39. Convert > P2F	367
H.40. Convert > sec2Hz.....	367
H.41. Delay > 2 / 4 Tap Delay 4p	368
H.42. Delay > Delay 1p / 2p / 4p	368
H.43. Delay > Diff Delay 1p / 2p / 4p	368
H.44. Envelope > ADSR.....	369
H.45. Envelope > Env Follower	369
H.46. Envelope > Peak Detector	370
H.47. EQ > 6dB LP/HP EQ	370
H.48. EQ > 6dB LowShelf EQ.....	370
H.49. EQ > 6dB HighShelf EQ.....	370
H.50. EQ > Peak EQ.....	371
H.51. EQ > Static Filter > 1-pole static HP.....	371

H.52. EQ > Static Filter > 1-pole static HS.....	371
H.53. EQ > Static Filter > 1-pole static LP	371
H.54. EQ > Static Filter > 1-pole static LS	372
H.55. EQ > Static Filter > 2-pole static AP	372
H.56. EQ > Static Filter > 2-pole static BP	372
H.57. EQ > Static Filter > 2-pole static BP1	372
H.58. EQ > Static Filter > 2-pole static HP	373
H.59. EQ > Static Filter > 2-pole static HS	373
H.60. EQ > Static Filter > 2-pole static LP	373
H.61. EQ > Static Filter > 2-pole static LS	373
H.62. EQ > Static Filter > 2-pole static N	374
H.63. EQ > Static Filter > 2-pole static Pk	374
H.64. EQ > Static Filter > Integrator.....	374
H.65. Event Processing > Accumulator	374
H.66. Event Processing > Clk Div	375
H.67. Event Processing > Clk Gen	375
H.68. Event Processing > Clk Rate	375
H.69. Event Processing > Counter	375
H.70. Event Processing > Ctl2Gate.....	376
H.71. Event Processing > Dup Flt / IDup Flt	376
H.72. Event Processing > Impulse	376
H.73. Event Processing > Random	376
H.74. Event Processing > Separator / ISeparator	376
H.75. Event Processing > Thld Crossing	377
H.76. Event Processing > Value / IValue	377
H.77. LFO > MultiWave LFO	377
H.78. LFO > Par LFO.....	377
H.79. LFO > Random LFO.....	378
H.80. LFO > Rect LFO.....	378
H.81. LFO > Saw(down) LFO	378
H.82. LFO > Saw(up) LFO	378
H.83. LFO > Sine LFO	379
H.84. LFO > Tri LFO.....	379
H.85. Logic > AND.....	379
H.86. Logic > Flip Flop.....	379
H.87. Logic > Gate2L	379
H.88. Logic > GT / IGT	380
H.89. Logic > EQ	380
H.90. Logic > GE	380
H.91. Logic > L2Clock	380
H.92. Logic > L2Gate	380

H.93. Logic > NOT	381
H.94. Logic > OR	381
H.95. Logic > XOR	381
H.96. Logic > Schmitt Trigger.....	381
H.97. Oscillators > 4-Wave Mst.....	382
H.98. Oscillators > 4-Wave Slv.....	382
H.99. Oscillators > Binary Noise	382
H.100. Oscillators > Digital Noise.....	383
H.101. Oscillators > FM Op	383
H.102. Oscillators > Formant Osc.....	383
H.103. Oscillators > MultiWave Osc.....	383
H.104. Oscillators > Par Osc	384
H.105. Oscillators > Quad Osc.....	384
H.106. Oscillators > Sin Osc.....	384
H.107. Oscillators > Sub Osc 4	384
H.108. VCF > 2 Pole SV.....	385
H.109. VCF > 2 Pole SV C.....	385
H.110. VCF > 2 Pole SV (x3) S.....	385
H.111. VCF > 2 Pole SV T (S)	386
H.112. VCF > Diode Ladder.....	386
H.113. VCF > D/T Ladder.....	386
H.114. VCF > Ladder x3.....	387
Apéndice I. Core cell library	388
I.1. Audio Shaper > 3-1-2 Shaper	388
I.2. Audio Shaper > Broken Par Sat.....	388
I.3. Audio Shaper > Hyperbol Sat.....	388
I.4. Audio Shaper > Parabol Sat.....	389
I.5. Audio Shaper > Sine Shaper 4/8	389
I.6. Control > ADSR.....	389
I.7. Control > Env Follower.....	390
I.8. Control > Flip Flop.....	390
I.9. Control > MultiWave LFO	390
I.10. Control > Par Ctl Shaper.....	391
I.11. Control > Schmitt Trigger.....	391
I.12. Control > Sine LFO	391
I.13. Delay > 2/4 Tap Delay 4p	392
I.14. Delay > Delay 4p	392
I.15. Delay > Diff Delay 4p.....	392
I.16. EQ > 6dB LP/HP EQ	393
I.17. EQ > HighShelf EQ.....	393

I.18. EQ > LowShelf EQ	393
I.19. EQ > Peak EQ	393
I.20. EQ > Static Filter > 1-pole static HP.....	394
I.21. EQ > Static Filter > 1-pole static HS.....	394
I.22. EQ > Static Filter > 1-pole static LP	394
I.23. EQ > Static Filter > 1-pole static LS	394
I.24. EQ > Static Filter > 2-pole static AP.....	395
I.25. EQ > Static Filter > 2-pole static BP.....	395
I.26. EQ > Static Filter > 2-pole static BP1	395
I.27. EQ > Static Filter > 2-pole static HP.....	395
I.28. EQ > Static Filter > 2-pole static HS.....	396
I.29. EQ > Static Filter > 2-pole static LP	396
I.30. EQ > Static Filter > 2-pole static LS	396
I.31. EQ > Static Filter > 2-pole static N.....	396
I.32. EQ > Static Filter > 2-pole static Pk	397
I.33. Oscillator > 4-Wave Mst	397
I.34. Oscillator > 4-Wave Slv	397
I.35. Oscillator > Digital Noise	398
I.36. Oscillator > FM Op	398
I.37. Oscillator > Formant Osc	398
I.38. Oscillator > Impulse.....	399
I.39. Oscillator > MultiWave Osc	399
I.40. Oscillator > Quad Osc	399
I.41. Oscillator > Sub Osc	400
I.42. VCF > 2 Pole SV C.....	400
I.43. VCF > 2 Pole SV T.....	401
I.44. VCF > 2 Pole SV x3 S	401
I.45. VCF > Diode Ladder.....	402
I.46. VCF > D/T Ladder.....	402
I.47. VCF > Ladder x3.....	403
Glosario	404

Referencia de los Módulos

Los módulos son los componentes más elementales en un ensemble de REAKTOR. Esta sección está principalmente dirigida a aquellos usuarios que quieren construir sus propios ensembles desde la base. Si tú sientes que todavía no tienes la experiencia necesaria como para poder hacerlo, REAKTOR te ofrece algunas alternativas para poder acceder a su potencial:

- Si no te interesa en absoluto construir tus propios ensembles, trabaja en el nivel de ensembles.
- Si quieres unir tus instrumentos favoritos en un ensemble y usarlos a la vez, trabaja en el nivel de instrumentos.
- Si quieres construir tus propios Instrumentos haciendo uso de los bloques preconstruidos, trabaja en el nivel macro.
- Si quieres disponer de control total sobre cada parámetro de tu ensemble, trabaja en el nivel de módulos.

Esta referencia enumera todos los módulos disponibles en REAKTOR y te proporciona una descripción básica de cada uno de ellos. Esta descripción incluye los ajustes de propiedades, si están disponibles, y una descripción de todos los puertos de entrada y de salida.

Todos los objetos de REAKTOR (Módulos, Instrumentos, Macros y Ensembles) tienen un recuadro para la etiqueta en sus Propiedades. Esta etiqueta aparece en el icono del objeto en la estructura. Por defecto, la etiqueta de los módulos describe la función del módulo. Renombra con cuidado los módulos, sobre todo si quieres que otros usuarios de REAKTOR comprendan tu estructura.

Módulos Híbridos

Muchos módulos en REAKTOR son módulos **híbridos**. Después de insertar un módulo como estos, por defecto aparece como un módulo de procesamiento de eventos indicado por etiquetas y puertos rojos. Un punto verde en un puerto indica que puedes conectar cables de audio o de evento en dicho puerto. En cuanto conectes un cable de audio a una de sus entradas, el módulo se convertirá en un módulo de procesamiento de audio indicado por puntos y puertos negros. En cambio, si conectas un cable de evento, en el puerto aparecerá un punto rojo, indicando que es de evento. Un módulo híbrido tiene el siguiente comportamiento:

- Si mezclas cables de evento y de audio en sus entradas o si sólo conectas cables de audio, el módulo siempre trabajará con tasa de audio.
- Si sólo conectas cables de evento a sus entradas, funcionará con tasa de evento.
- Si la salida del módulo está conectada a una entrada de evento de otro módulo, se convertirá automáticamente en un módulo de procesamiento de evento, y ya no podrás añadir cables de audio bajo ningún concepto en las entradas de este módulo.
- Si la salida del módulo está conectada a una entrada de audio, el módulo puede funcionar como módulo de evento o de audio, dependiendo del tipo de cables que conectes en sus entradas.
- Si el módulo ya funciona con tasa de audio debido a sus conexiones de entrada, no puedes conectar su salida a una entrada de evento en otro módulo.

Administración Dinámica de Puertos

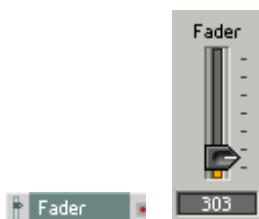
En los casos en los que tiene sentido, el módulo ofrece una posibilidad de administración **dinámica** de los puertos de entrada y/o salida. Puedes añadir entradas al módulo arrastrando cables adicionales sobre el módulo en el área de un puerto existente mientras mantienes la tecla **Ctrl**. Podrás ver dónde se coloca el nuevo puerto al mantener el cursor del ratón sobre el módulo de destino. La posición vertical del cursor del ratón determina dónde se va a colocar el puerto, de modo que podrás generar nuevos puertos entre ya existentes.

Panel

Los módulos de panel proporcionan controles de pantalla para distintos procesamiento de REAKTOR. Se pueden ajustar independientemente para que aparezcan en los Paneles de Control A o B y así ordenarlos de diferentes formas en los paneles. Algunos módulos de panel sólo tienen valores y se incluyen luces, medidores, e indicadores de los procesos que se llevan a cabo en Reaktor así como módulos gráficos y de texto de ornamentación. El resto de los módulos generan o encaminan datos para el procesamiento de Reaktor, y se incluyen knobs, botones, interruptores, menús y controladores XY (que se pueden usar como displays o como controles).

Fader

Panel



Con el control slider del panel puedes ajustar el valor que va a salir (como evento y audio) a través del módulo correspondiente en la estructura. La señal es mono – cuando está conectado a una entrada polifónica todas las voces reciben el mismo valor.

Propiedades - Página Function

Range (Rango): El valor de salida corresponde a la posición de la manilla, cuyo Rango está entre los límites **Min** y **Max** ajustados en la ventana del diálogo de propiedades. Con el parámetro **Stepsize** puedes cuantizar los valores indicados y de salida para ajustar los pasos, por ejemplo, para que aparezcan menos dígitos decimales. Si no, también puedes usar el recuadro **Num Steps** para ajustar la resolución del fader. Los valores de ambos recuadros, **Stepsize** y **Num Steps**, son dependientes uno del otro. Si cambias el valor de uno de ellos, se cambiará automáticamente en el otro. En el recuadro **Stepsize** introducirías el Rango de valores por paso, mientras que en **Num Steps** se representa el número total de pasos sobre el Rango total de valores del fader.

La mayor resolución posible para un fader es 127.000 pasos, que conseguirás introduciendo 0 en el recuadro **Stepsize** o 127.000 en el recuadro **Num Steps**.

Nota: Puedes introducir una resolución de pasos en **Num Steps** mayor que la resolución MIDI de 128 estandarizada. Pero ten cuidado porque, aunque Reaktor puede usar una mayor resolución interna, sólo puede intercambiar datos MIDI con hardware o software externo dentro de la resolución MIDI de 128. Puede que en algunos casos, los distintos entornos de producción afecten a la funcionalidad del sonido de un ensemble. Normalmente no tendría por qué ser un problema puesto que la resolución MIDI es suficiente como para controlar los parámetros. No obstante, en ciertas situaciones (por ejemplo, usando un filtro con alta resonancia mientras se mueve el corte de frecuencia) es posible que quieras disponer de una resolución mayor, de la que también podrás disponer dentro de Reaktor.

Mouse Res especifica la distancia en pixels que el indicador del ratón ha de cubrir desde el valor más bajo hasta el más alto. Si introduces un valor en el recuadro **Mouse Res** el doble de alto que el que hay en el recuadro **Pixel in Y** dentro de **Appearance**, el recorrido entre el valor mínimo y el máximo con el ratón será el doble de largo.

Si **Num Steps** es mayor que **Mouse Res**, no alcanzarás todos los pasos moviendo el ratón. Por otro lado, si **Mouse Res** es mayor que **Num Steps**, el número de pixels por paso se puede ajustar a un valor mayor que 1.

El valor **Default (Por defecto)** se usa cada vez que ocurre una inicialización del control. El valor se usará en las siguientes situaciones:

- Presionando el botón **Default** en la ventana de instantáneas, reiniciarás todos los controles del instrumento/ensemble.
- Si aparece “default” en uno de los destinos de morphing dentro de la ventana de instantáneas, puedes hacer morphing entre una instantánea y los valores por defecto de todos los controles.
- Si añades un control a un instrumento que ya contiene una lista de instantáneas, el valor por defecto se usa para el nuevo control hasta que vuelvas a guardar la instantánea con el nuevo valor para ese control o actives **Snap Isolate**.

Si el interruptor **Snap Isolate** está activado, evitarás que el fader reaccione ante activaciones de instantáneas

Activando **Random Isolate** evitarás que el fader responda a la función random de la instantánea.

ID for Snapshot Files: El número ID se usa para la administración de instantáneas en REAKTOR. Los números se introducen automáticamente cada vez que insertas un controlador de panel (fader, knob, botón, interruptor...) Si cambias este número ID (si importas instantáneas desde otro instrumento con controles similares, por ejemplo) las instantáneas ya existentes no reconocerán el elemento de panel y lo ignorarán. Modifica este número sólo si sabes exactamente lo que estás haciendo.

Propiedades – Página de Información

Dentro del recuadro **Info** en las Propiedades, puedes introducir un texto de explicación de las funciones del fader. El texto aparecerá cuando dejes el ratón unos segundos sobre el fader en el panel, siempre y cuando esté encendida la opción Show Hints.

Propiedades – Página Appearance

La etiqueta que introduzcas aquí sólo aparecerá por encima del fader en el panel si **Label** está activado en la sección **Visible** de las Propiedades.

Asimismo, el valor actualmente ajustado sólo aparecerá como un número por debajo del fader si **Value** está activado. También puedes ocultar el propio bitmap del fader, de forma que sólo veas la caja de valores. En este caso, todavía podrías usar el fader del panel haciendo clic y arrastrando arriba y abajo sobre la caja de valores.

Los faders pueden aparecer **vertical** u **horizontalmente** en el panel. Escoge el botón apropiado en la sección **Form** para conseguir la apariencia deseada.

El ancho del fader se puede ajustar libremente en el recuadro **Pixel in Y** dentro de la sección **Size**. También puedes definir la anchura del fader con tres botones **Big**, **Medium** y **Small**.

Propiedades – Página de Conexión

La página de conexión de las propiedades para los Controles de Panel está descrita en la sección *Propiedades de Conexión de los Controles de panel en el capítulo 18.5*.

Knob



Panel

Igual que el fader pero con una diferencia de apariencia en el panel.

Button



Panel

Con el control de botón del panel seleccionas el valor que va a salir (como evento o audio) a través del módulo correspondiente en la estructura. Los valores de salida en los estados On y Off se ajustan con **On Value** y **Off Value** en las propiedades de los botones. La señal es mono – si está conectado a una entrada poli, todas las voces reciben el mismo valor.

Propiedades – Página Function

Range (Rango): Los valores que se envían a través del Botón al presionarlo y liberarlo se pueden definir en los recuadros **On Value** y **Off Value**.

Mode: Hay tres posibilidades en el modo de operación: **Trigger**, **Gate** y **Toggle**. En el modo Trigger, los eventos sólo se generan cuando se presiona el botón, y no ocurre nada cuando se libera. En el modo Gate, cuando se libera el botón se envía un evento con valor 0. En el modo Toggle, el botón cambia su estado cada vez que se presiona.

Default = On: Activa el valor por defecto que se usa en diferentes acciones dentro de REAKTOR.

Activando **Snap Isolate** evitarás que el botón reaccione ante activaciones de instantáneas.

Activando **Random Isolate** evitarás que el botón responda a la función random de las instantáneas.

ID for Snapshot Files: Mira la descripción en Fader.

Propiedades – Página de Información

Dentro del recuadro **Info** en las Propiedades, puedes introducir un texto de explicación de las funciones del botón. El texto aparecerá cuando dejes el ratón unos segundos sobre el botón en el panel, siempre y cuando esté encendida la opción Show Hints.

Properties - Página Appearance

La etiqueta que introduzcas aquí sólo aparecerá por encima del botón en el panel si **Label** está activado en la sección **Visible** de las Propiedades. Asimismo, el valor actualmente ajustado sólo aparecerá como un número por debajo del botón si **Value** está activado.

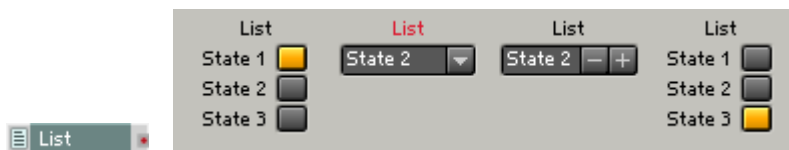
El botón puede tener tres tamaños: **Small**, **Medium** y **Big**.

Propiedades – Página de Conexión.

La página de conexión de las propiedades de los Controles de Panel está descrita en la sección *Propiedades de Conexión de los Controles de Panel en el capítulo 18.5*.

List

Panel



El módulo List se usa para construir listas de pantalla, menús desplegables, selectores de botones y displays de textos de desplazamiento.

Propiedades – Página Function

Las propiedades de **Función** contienen una lista dentro de la cual puedes **Agregar**, **Insertar** y **Borrar** ítems individuales. También puedes especificar el número de ítems usando directamente entradas numéricas (**NUM Entries**). Los ítems de la lista aparecerán automáticamente en el panel de control en el formato de display seleccionado. Para cada nuevo ítem, puedes introducir un valor que se enviará a la salida del módulo cuando se seleccione ese ítem.

La página de función también contiene un generador de valores. Con esta herramienta puedes generar valores para múltiples entradas en la lista. Ejemplo: quieres tener un incremento de 4 en lugar de 1 para cada próxima entrada. Por lo tanto, tendrás que introducir “4” como **Stepsize** en el Generador de Valores y presionar el botón **Apply**. Ahora, los valores de la columna **Value** de la lista de entradas se modificarán de acuerdo con los ajustes del Generador de Valores.

Mouse Resolution sólo se aplica al control de panel si escoges el estilo **Spin** en la página de **Apariencia**, donde podrás hacer clic sobre la entrada de control y arrastrar arriba o abajo para cambiar la entrada.

Propiedades – Página Appearance

La representación del panel de este módulo cambia dependiendo del estilo escogido en las propiedades de la página de **Apariencia**. Encontrarás los siguientes estilos disponibles:

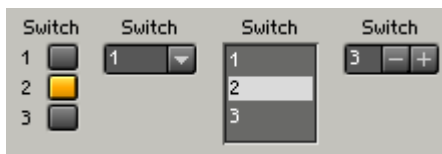
- **Button:** Cada puerto de entrada del módulo crea un botón. Todos los botones aparecerán verticalmente en el panel del instrumento. El botón seleccionado aparecerá con el color indicador del instrumento.
- **Menu:** Cada puerto de entrada del módulo crea una nueva entrada en una lista desplegable.
- **Text Panel:** Cada puerto de entrada del módulo crea una nueva entrada en una lista que muestra múltiples entradas a la vez. Si has creado más entradas de las que caben en el display del panel especificado en los recuadros **Size X** y **Size Y** de la página de Apariencia, aparecerán barras de desplazamiento en el panel.
- **Spin:** Cada puerto de entrada del módulo crea una nueva entrada en la lista. Puedes navegar por la lista con los botones * y – que hay a mano derecha del display del panel de entradas de la lista.

Los recuadros **Size X** y **Size Y** controlan el tamaño del display del elemento de control en el panel.

Puertos

- **Out:** Salida de evento para el valor asociado con el ítem seleccionado.

Switch



Panel

Los interruptores se usan para cambiar la corriente de la señal. No contienen ningún procesamiento de señal en sí mismos, pero establecen una conexión

entre sus módulos. La salida se conecta a la entrada seleccionada presionando el botón correspondiente o seleccionando la entrada de la lista. El resto de entradas sin conectar quedarán desconectadas.

Los módulos cuyas salidas estén desconectadas, a través de la acción del interruptor, se desactivarán automáticamente para reducir la carga innecesaria de CPU. La luz de estado de un módulo permanecerá apagada mientras el módulo esté desactivado.

Este módulo es un módulo híbrido, de modo que puede procesar señales de evento y de audio, dependiendo de sus conexiones, y contiene una administración dinámica de sus puertos de entrada.

Propiedades – Página Function

Enable Switch Off: Si esta opción está habilitada, el interruptor puede tener un estado en el que no hay entradas seleccionadas. Si tu display está ajustado al estilo de botón en la página de **Apariencia** de las propiedades, es posible desconectar todos los botones haciendo clic una segunda vez sobre el botón actualmente seleccionado. Si ajustas cualquiera de los otros estilos de display, dispondrás de una entrada adicional llamada “Off”.

Min Num Port Groups: Puedes establecer el número mínimo de puertos del módulo al margen del número de cables que hayas conectado al módulo.

Mouse Resolution sólo se aplica al control de panel si has elegido el estilo **Spin** en la página de **Apariencia**, donde podrás hacer clic sobre la entrada de control y arrastrar arriba y abajo para cambiar la entrada.

Propiedades – Página Appearance

La etiqueta sólo se mostrará por encima del botón en el panel si está activado **Label** en la página de **Apariencia** de las propiedades.

El control de panel puede aparecer en estos tres tamaños: **Small**, **Medium** y **Big**.

Si usas un interruptor con dos puertos de entrada, sólo aparecerá un botón (el superior) cuando se seleccione **1 Toggle Button:** Cuando el botón está encendido, se conecta la entrada 1, cuando el botón está apagado, se conecta la entrada 2.

La representación del panel de este módulo cambia dependiendo del estilo seleccionado en la página de **Apariencia** de las propiedades. Encontrarás los siguientes estilos disponibles:

- **Button:** Cada puerto de entrada del módulo crea un botón. Todos los botones aparecerán verticalmente en el panel del instrumento. El botón actualmente seleccionado aparecerá en el color indicador del instrumento.
- **Menu:** Cada puerto de entrada del módulo crea una nueva entrada en una lista desplegable.
- **Text Panel:** Cada puerto de entrada del módulo crea una nueva entrada en una lista que muestra múltiples entradas a la vez. Si has creado más entradas de las que caben en el display del panel especificado en los recuadros **Size X** y **Size Y** en la página de Apariencia, aparecerá una barra de desplazamiento en el panel.
- **Spin:** Cada puerto de entrada del módulo crea una nueva entrada en una lista. Puedes navegar por la lista con los botones + y – que hay a la derecha del display del panel de la lista de entradas.

Lamp



Panel

Luz indicadora para una señal monofónica

La luz en el panel se ilumina siempre que la señal de entrada (sampleada a 25Hz) esté dentro del Rango ajustado en **Min** y **Max** dentro de la página de Función en las propiedades. Es decir, la luz se enciende cuando el valor de la señal es mayor que **Min** y menor o igual que **Max**.

Si conectas el modo **Continuous** en las propiedades, el color de la luz se desdibujará hacia dentro y hacia fuera entre los valores **Min** y **Max**.

El color de la luz (rojo, verde, azul, amarillo o indicador) se puede elegir en las propiedades. Si conectas **Indicator Color**, la luz usará el color indicador elegido en las propiedades del instrumento.

Finalmente, los botones **Set On Color** y **Set Off Color** permiten definir colores personalizados para las posiciones de la luz.

Si no haces uso de la opción **Has Frame**, las luces en el panel se pueden poner una al lado de otra sin espacios intermedios.

La etiqueta sólo aparecerá sobre la luz si **Label** está activado en las propiedades.

Level Lamp



Panel

Luz indicadora para una señal monofónica con ajustes logarítmicos.

La luz en el panel se ilumina siempre que el nivel de entrada esté dentro del Rango ajustado en **Min** y **Max** (en dB) en las propiedades, es decir, la luz estará encendida cuando el valor de la señal sea mayor que **Min** y menor o igual a **Max**.

Si conectas el modo **Continuous** en las propiedades, el color de la luz se desdibujará hacia adelante y hacia atrás entre los valores de **Min** y **Max**.

El color de la luz (rojo, verde, azul, amarillo o indicador) se puede elegir en las propiedades. Si usas **Indicador Color**, la luz utilizará el color del indicador elegido en las propiedades del instrumento.

Finalmente, los botones **Set On Color** y **Set Off Color** permiten definir colores personalizados para las posiciones de la luz..

Si conectas la opción **Has Frame**, las luces en el panel se pueden poner una al lado de otra sin espacios intermedios.

La etiqueta sólo se verá por encima de la luz si se activa **Label** en las propiedades.

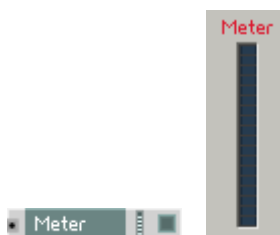
RGB Lamp



Panel

El módulo RGB Lamp es un display coloreado que puede cambiar de tamaño. Su color se controla con los valores que aparecen en sus tres entradas de color. Su tamaño horizontal y vertical se ajusta en pixels dentro de las propiedades de **Apariencia**.

- **R**: Entrada de audio para la intensidad del componente rojo. Rango 0 a 1.
- **G**: Entrada de audio para la intensidad del componente verde. Rango 0 a 1.
- **B**: Entrada de audio para la intensidad del componente azul. Rango 0 a 1.



Indicador de valor para una señal monofónica.

El valor de la señal entrante es sampleada (a 25Hz) y se muestra en una escala lineal. El Rango mostrado se ajusta con **Min** y **Max** en las propiedades.

El color del medidor (rojo, verde, azul, amarillo o indicador) se puede elegir en las propiedades. Si conectas **Indicator Color**, la luz tendrá el color indicador elegido en las propiedades del instrumento.

Finalmente, los botones **Set On Color** y **Set Off Color** permiten definir colores personalizados para las secciones superior e inferior del medidor.

Number Of Segments define el número de elementos singulares de los que se compone el metro.

Con **Size X (Segment)** y **Size Y (Segment)** puedes definir el tamaño de uno de los elementos del metro. El tamaño general del metro será el resultado de **Size Y (Segment)** multiplicado por el **Number of Segments**.

La etiqueta sólo aparecerá por encima del metro en el panel si **Label** está activado en las propiedades.

Level Meter

Panel



Indicador de nivel para una señal de audio monofónica.

La amplitud de la señal de audio conectada se mostrará en una escala logarítmica. El Rango mostrado se ajusta en dB con **Min** y **Max** en las propiedades.

El color del medidor (rojo, verde, azul, amarillo o indicador) se puede elegir en las propiedades. Si conectas **Indicador Color**, la luz usará el color indicador elegido en las propiedades del instrumento.

Finalmente, los botones **Set On Color** y **Set Off Color** permiten definir colores personalizados para las secciones superior e inferior del medidor.

Number Of Segments define el número de elementos singulares de que se compone el medidor.

Con **Size X (Segment)** y **Size Y (Segment)** podrás definir el tamaño de los elementos del metro. El tamaño general del metro será el resultado de **Size Y (Segment)** multiplicado por el **Number of Segments**.

La etiqueta sólo aparecerá sobre el medidor en el panel si **Label** está activado en las propiedades.

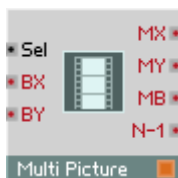
Picture

Panel



Permite cargar un bitmap decorativo en el panel desde un archivo de imagen en formato TGA (extensión del archivo *.tga). El módulo bitmap no tiene entradas ni salidas. El tamaño del display se ajustará al tamaño de la imagen. Puedes modificar los frames visibles en las propiedades. La imagen no se puede cambiar de tamaño dentro de Reaktor; para ello tendrás que usar un software de edición de imágenes externo.

Selecciona la opción **Save bitmap with ensemble** en las propiedades para almacenar los datos de imagen como parte del archivo de Reaktor.



El módulo Multi Picture es un controlador bidimensional (como el módulo XY) que informa de la posición del ratón y del estado del botón del ratón en sus salidas. Soporta animación multi-frames cuando las imágenes están ajustadas así en la ventana de Propiedades de la imagen indicando el número de frames (animaciones) y su orientación (horizontal y vertical).

Los formatos de imagen 24 bit BMP y 32 bit Targa (no comprimidos) se pueden utilizar en varios lugares de Reaktor: Como fondo del panel de controles de instrumentos y macros, como iconos de estructuras de macros e instrumentos, y en módulos de panel de control Picture y Multi Picture. La ventaja de Targa es que soporta un canal alfa, que se puede usar como máscara en las partes visibles del gráfico – las partes sin máscara serán transparentes. Esto resulta útil en knobs redondos sobre un fondo rectangular, por ejemplo.

Puedes cargar imágenes usando el menú desplegable Select Picture en las propiedades de cualquier objeto que pueda proyectar imágenes. Si abres una imagen, automáticamente aparecerá la ventana de Propiedades de la imagen, donde puedes hacer los ajustes relevantes. Todas las imágenes se comparten automáticamente y están disponibles para todos los módulos apropiados.

- **Sel:** Entrada de audio para seleccionar el frame por números. Rango desde 0 hasta el número del último frame.
- **MX:** Salida de evento para la posición horizontal (X) del ratón cuando el ratón está dentro de la imagen.
- **MY:** Salida de evento para la posición vertical (Y) del ratón cuando el ratón está dentro de la imagen.
- **MB:** Salida de evento para el estado del botón del ratón (0 cuando está pulsado, 1 cuando no está pulsado).
- **N - 1:** Salida de evento para el número de animaciones que has introducido en las propiedades de la imagen -1.

Text

Panel

The icon for the Text module, showing a green rectangular button with the word "Text" in white.

El módulo Text no procesa ninguna señal, su propósito es sólo añadir texto a la estructura. Por ejemplo, se puede usar para anotar el autor y los datos de creación de un instrumento y para explicar cómo funciona.

Multi Text

Panel

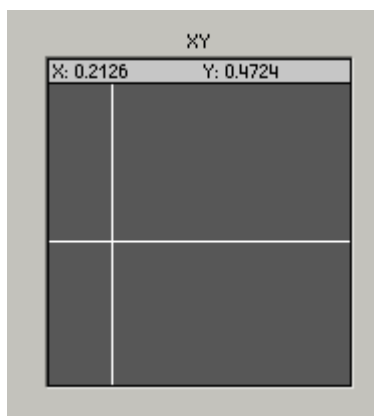
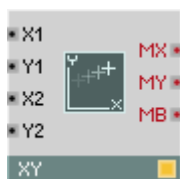
The icon for the Multi Text module, showing a green rectangular button with the text "Multi Text" and a small icon of a document with a plus sign.

El módulo Multi Text proporciona un display variable para los paneles de control. Se puede añadir cualquier número de ítems de texto en las propiedades del módulo, donde los ítems también se pueden editar o borrar.

In: Entrada de audio para el número de ítems de texto que van a mostrarse.

XY

Panel



El recuadro de control XY tiene dos funciones: Muestra las señales de entrada de audio y actúa como controlador bidimensional que se usa con el ratón.

Propiedades – Página Function

La opción **Always Active** permite que el módulo pueda activar una rama de la señal conectada a uno de los puertos de entrada del módulo.

En **Incremental Mouse Mode** el control reacciona de manera similar a un knob. En este modo, puedes hacer clic sobre cualquier parte dentro del dis-

play del panel del control y mover el ratón sin tener que reajustar el valor de la posición del indicador del ratón directamente. En lugar de esto, el valor seguirá el indicador del ratón con un offset fijo.

Propiedades – Página Appearance

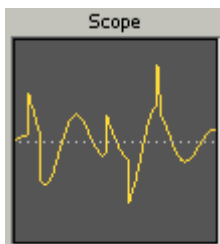
En las propiedades del módulo puedes ajustar el tipo de representación para visualizar las señales de audio. Las entradas **X1** e **Y1** controlan la posición del objeto visual (**Píxel** o **Cross**). Cuando el objeto se ajusta a **Bar** o **Rectangle**, **X1** e **Y1** controlan una esquina y **X2** e **Y2** controlan la esquina contraria del objeto.

Para mostrar los datos de audio que cambian rápidamente selecciona el modo **Scope**. El resto de los modos evalúan la entrada a una tasa menor.

También puedes ajustar el tamaño de la cruz que aparece en la posición del ratón.

Todo lo que dibujes en el display desaparecerá más o menos rápido, dependiendo del valor ajustado en **Fade Time** dentro de las propiedades (el valor máximo es 99).

- **X1**: Entrada de audio para la coordenada X1 del objeto visual.
- **Y1**: Entrada de audio para la coordenada Y1 del objeto visual.
- **X2**: Entrada de audio para la coordenada X2 del objeto visual.
- **Y2**: Entrada de audio para la coordenada Y2 del objeto visual.
- **MX**: Salida de evento para la posición X del cursor del ratón cuando el botón del ratón se pulsa dentro del display.
- **MY**: Salida de evento para la posición Y del cursor del ratón cuando el botón del ratón se ha pulsado dentro del display.
- **MB**: Estado del botón izquierdo del ratón (1 = presionado; 0 = Liberado).



Osciloscopio para mostrar señales de tiempo variante.

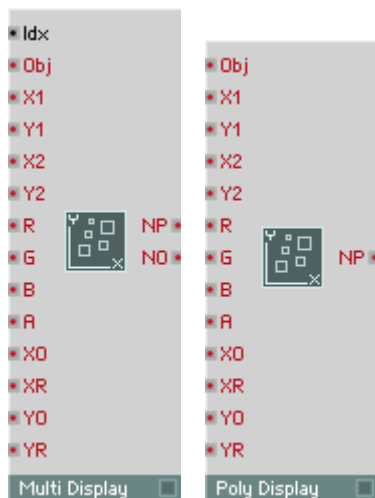
Cada vez que se recibe un Evento en la entrada Trigger (**Trg**), el módulo empieza a grabar la señal de audio de la entrada (**In**) y la muestra en el panel. Cuando no se reciben eventos activadores, el display continúa mostrando la señal guardada y, ajustando las entradas de control relevantes, se puede ver ampliada y en distintas posiciones.

El tamaño del gráfico en el panel se puede ajustar en las propiedades de **Apariencia** del módulo con **Pixel in X** y **Pixel in Y**.

En la página de **Función** puedes ajustar el buffer usado por el scope en ms.

Normalmente se conecta un módulo **A to E Trig** a la entrada **Trg** para activar el Scope desde una señal de audio.

- **Trg**: Entrada de evento mono para la señal activadora que se sincroniza con la curva.
- **TP**: Entrada de evento mono para controlar el offset en tiempo (Time Position) de la curva en milisegundos. La curva comienza en el borde izquierdo del display **TP** ms después del evento activador.
- **TS**: Entrada de evento mono para controlar la escala de tiempo de la curva representada. Desde el borde izquierdo al derecho, el display muestra **TS** ms de la señal.
- **YP**: Entrada de evento mono para controlar la posición de amplitud (Y Position) de la curva. **YP** = -1 corresponde al borde inferior del display, +1 al superior.
- **YS**: Entrada de evento mono para controlar la escala de amplitud (Y Scale) de la curva. La diferencia entre los valores de señal de arriba y de abajo del display muestran valores entre +**YS** and -**YS**.
- **In**: Entrada de audio mono para la señal que se va a representar.



Los módulos Multi Display y Poly Display permiten la representación y manipulación de múltiples objetos gráficos (cruces, barras, imágenes, animaciones, etc.). Puedes definir una serie de parámetros (tipo, posición, tamaño y color) individualmente para cada objeto gráfico.

La principal diferencia entre los dos módulos es que en el módulo Multi Display, el número de objetos gráficos está especificado en el recuadro Number Of Objects en las propiedades, mientras que en el módulo Poly Display, el número de objetos gráficos viene especificado por el número de voces del instrumento.

La ventaja de Multi Display es que puede mostrar cualquier número de objetos gráficos, al margen del número de voces polifónicas. Cada objeto se puede alojar independientemente usando la entrada Idx. Al contrario, el módulo Poly Display podría considerarse que es más fácil de programar, ya que no requiere ningún código de índice de procesamiento, y todos los objetos se pueden alojar simultáneamente a través del procesamiento polifónico paralelo. No obstante, el número de objetos se limita al número de voces del instrumento.

Las propiedades ofrecen una variedad de opciones para la personalización del display, incluyendo fondo e imagen.

Cuando se usa junto con los módulos Mouse Area y Snap Value Array, el Multi Display y Poly Display permiten construir unos elementos de diseño muy sofisticados (secuenciadores, por ejemplo).

Todas las entradas del Multi Display aceptan señales de evento monofónicas. Idx también acepta señales de audio monofónicas.

- **Idx:** Índice de alojamiento del elemento gráfico. El índice se basa en 1; es decir, el ID del primer elemento es 1 (no 0). Los valores fraccionales se redondean al entero más cercano. Cuando los valores Idx están fuera del Rango de 1 a N, el comportamiento depende de la opción Index Behaviour en las propiedades. El orden de almacenamiento de los objetos gráficos está determinado por sus valores Idx. Los objetos con menores valores de Idxs aparecerán encima de los objetos con mayores valores de Idxs. Es esencial ajustar el valor Idx antes de transmitir los eventos a los otros puertos.
- **Obj:** Tipo de objeto gráfico, o selección de frame de la imagen.
- 4: Cruz.
- 3: Línea desde X1, Y1 a X1, Y1 del siguiente objeto del mismo tipo.
- -2: Línea desde X1, Y1 to X2, Y2.
- -1: Barra.
- 0: Rectángulo.
- 1 ... NP: índice de imagen que especifica una imagen sencilla (1) o una serie de imágenes en una animación (1, 2, 3, ..., NP), donde NP es el número total de imágenes en la animación.

Los valores fraccionales se redondean al número entero más cercano.

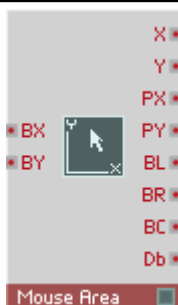
Si la opción Ignore Index Obj (en propiedades) está activada, todos los objetos gráficos se ajustan al mismo tipo.

- **X1, Y1:** Coordenadas de la primera esquina del área del objeto gráfico, o si la opción 'Center to X1, Y1' está seleccionada en las propiedades, esos valores definen las coordenadas del centro del objeto.
- **X2, Y2:** Coordenadas de la segunda esquina del área del objeto gráfico.
- **R, G, B:** Estas entradas definen la cantidad de componentes Rojo (Red), Verde (Green) y Azul (Blue) del color del objeto (Rango 0 a 1). Si la opción Ignore Index RGB (en las propiedades) está activada, todos los objetos se ajustan al mismo color.
- **A:** Transparencia del objeto (0 = totalmente transparente; 1 = totalmente opaco).

- **XO, YO:** Valores menores visibles de X e Y (para el desplazamiento). Las entradas de estos puertos sobrescriben las entradas de los recuadros X Origin e Y Origin en las propiedades.
- **XR, YR:** Rango de la representación vertical/horizontal, (para zoom). Las entradas en estos puertos sobrescriben las entradas de los recuadros X Range e Y Range en las propiedades.
- **NP:** En la salida se refleja el número de imágenes en la animación.
- **NO:** En la salida se refleja el número de objetos gráficos.

Las entradas y salidas del Poly Display son las mismas que para Multi Display, excepto que Poly Display no tiene entrada Idx ni No Output. Todas las entradas de Poly Display aceptan señales de evento polifónicas, excepto para XO, XR, YO e YR, que aceptan señales de evento monofónicas.

Mouse Area



Panel

El módulo Mouse Area detecta y transmite acciones del ratón y cambios de posición. El Mouse Area puede tener su propio color de contorno y de relleno, y por lo tanto, se puede usar como un elemento de panel en sí mismo. Pero sobre todo se usa como cubierta invisible (transparente) encima de otros módulos, especialmente sobre los módulos Multi Display y Poly Display, para construir elementos de elevada customización.

Las salidas X e Y del Mouse Area pueden operar en modo absoluto o incremental (como lo seleccionen en las propiedades). Cuando el modo Incremental está activado, las posiciones de X e Y se ajustan incrementalmente según los múltiples arrastres del ratón, del mismo modo que funcionan los knobs y faders prefabricados en Reaktor. Cuando el usuario comienza un nuevo arrastre, las salidas X e Y transmiten valores relativos al lugar en el que terminó el arrastre previo, al contrario que la posición absoluta del cursor del ratón. Las entradas BX y BY sólo son efectivas en el modo incremental, y se usan para ajustar la “base” incremental de los siguientes arrastres (sobrescribi-

endo el último movimiento del ratón). Normalmente se conectan a módulos Snap Value para asegurar que la base incremental se ajusta de acuerdo con el último movimiento de ratón guardado en la instantánea.

En las propiedades, Outline Style especifica la apariencia del contorno de la caja Mouse Area. Si seleccionas 'Rectangle', se dibuja un contorno de 1 pixel alrededor de la caja Mouse Area, mientras que si seleccionas 'Bar', el área se rellenará con un color sólido.

La opción Active State especifica la acción que hace que el contorno de la caja cambie del estado 'inactivo' a 'activo'. Escoge desde 'Selección' (el estado activo ocurre cada vez que la caja Mouse Area se selecciona), o desde las opciones de botones Izquierda/Derecha/Centro (el estado activo ocurre cada vez que el botón del ratón relevante se presiona). Las propiedades de Color del Contorno permiten definir los ajustes de color y transparencia para los estados activo e inactivo.

También en las propiedades están los valores para la posición X e Y, que permiten colocar el Mouse Area en el panel del instrumento en la cuadrícula 4x4 de Reaktor.

BX: Entrada para ajustar el valor base para los cambios incrementales en la salida X. Los valores válidos incluyen aquellos dentro del Rango X, como viene definido en las propiedades.

BY: Entrada para ajustar el valor base para los cambios incrementales en la entrada Y. Los valores válidos incluyen aquellos dentro del Rango Y, como viene definido en las propiedades.

Observa que las entradas BX y BY sólo influyen sobre las salidas X e Y cuando la opción Incremental Mode está activada en las propiedades.

X: Salida para la posición horizontal del ratón, aumentada y limitada a los valores del Rango X (en las propiedades). La salida X sólo informa de las posiciones del ratón que están dentro de la caja Mouse Area (como viene definido en Size X y Size Y en las propiedades).

Y: Salida para la posición horizontal del ratón, ampliada y limitada a los valores del Rango Y (en las propiedades). La salida Y sólo informa de las posiciones del ratón que están dentro de la caja Mouse Area (como viene definido por Size X y Size Y en las propiedades).

PX: Posición horizontal en pixels del ratón relativa a la línea de origen X (borde izquierdo de la caja Mouse Area). Al mover el ratón a la izquierda de la línea de origen X transmite valores de píxel negativos, y al moverla hacia la derecha transmite valores positivos. Al contrario

que la salida X, que está limitada a los movimientos dentro de la caja Mouse Area, PX informa de las posiciones del ratón en todas partes, a izquierda y derecha de la pantalla.

PY: Posición vertical en pixels relativa a la línea de origen Y (borde izquierdo de la caja Mouse Area). Al mover el ratón por encima de la línea de origen Y transmite valores de píxel negativos, y al moverla hacia abajo trasmite valores positivos. Al contrario que la salida Y, que está limitada a los movimientos dentro de la caja Mouse Area, PY informa de las posiciones del ratón en todas partes, a izquierda y derecha de la pantalla.

BL: Estado del botón izquierdo del ratón: 1 cuando está presionado; 0 cuando no lo está.

BR: Estado del botón derecho del ratón: 1 cuando está presionado; 0 cuando no lo está.

BC: Estado del botón central del ratón: 1 cuando está presionado; 0 cuando no lo está.

Db: Transmite un evento sencillo con valor 1 cada vez que ocurre un doble clic. El valor Db permanece en 1.0 después del doble clic, así que tendrás que probar con eventos dB, no con valores estáticos.

Stacked Macro



Panel

Stacked Macros permite que múltiples objetos gráficos compartan el mismo área del panel del instrumento. Los objetos que aparezcan dentro del área de una vez se pueden controlar usando el módulo Panel Index.

Después de insertar un Stacked Macro en tu estructura, colócalo en la posición correcta del panel, y ajusta el tamaño deseado en las propiedades. Luego inserta 2 o más macros (macros normales, no Stacked Macros) dentro de Stacked Macro, junto con el módulo Panel Index. Sólo será visible una de las macros 'normales' (junto con los elementos de panel dentro de ella) a la vez. Cuál es la macro visible, depende del valor de entrada en el módulo Panel Index. (Para descubrir el número de índice de una macro, haz clic con el botón derecho del ratón sobre la macro en el panel del instrumento – el número de índice aparecerá en el menú contextual).

MIDI In

Los módulos MIDI In encaminan mensajes MIDI a Reaktor desde dispositivos MIDI externos. Hay distintos módulos para los distintos tipos de datos MIDI incluyendo notas, velocidad, Pitchbend, aftertouch poly y mono, controladores, cambios de programa y reloj MIDI. Reaktor también genera mensajes de puerta separados para eventos de nota MIDI y hay módulos para ello. Algunos módulos MIDI In también pueden usar una conexión interna u OSC.

Note Pitch

MIDI In



Fuente de evento polifónica para la tonalidad de eventos MIDI Note On.

Un evento Note On ajusta la salida a un valor determinado por el número de la tecla (Note Pitch). El Rango de los valores de salida se ajusta con **Min** y **Max** en la ventana del diálogo de propiedades. La resolución es de 128 pasos. Un evento Note Off no tiene efecto.

Al usar el Rango por defecto desde **Min** = 0 a **Max** = 127 y controlar la entrada **P** de un oscilador (para controlar el tono logarítmico) se puede tocar de forma moderada. Una unidad corresponde a un semitono y el Do central está a 60. Ajustando **Min** y **Max** a diferentes valores, la tonalidad se puede desplazar y escalar, es decir, para tocar en cuartos de tono.

Pitchbend

MIDI In



Fuente de eventos monofónica para eventos MIDI Pitchbend (rueda de cambios de tonalidad). La resolución es de 16384 pasos. Cuando el controlador de Pitchbend está en su posición neutral, el valor de salida es siempre 0. El Rango del valor de salida para Pitchbend hacia arriba o hacia abajo se puede ajustar independientemente en la ventana del diálogo de propiedades. Para el Pitchbend hacia abajo, el Rango se ajusta con **Min** y hacia arriba con **Max**.

Al controlar la entrada **P** de un oscilador (para controlar el tono logarítmico), es decir, después de añadir un valor de tono de la nota, y con un Rango de **Min** = -1 a **Max** = 1 puedes cambiar el tono hacia arriba o hacia abajo un semitono. OL Un rango de -12 a +12 significa Pitchbend dentro de ± 12 semitonos, es decir, ± 1 octava.



Fuente de evento polifónica para eventos MIDI Note On y Note Off.

Un evento Note On ajusta la salida a un valor determinado por la intensidad de pulsación (Velocity). El Rango del valor de salida se ajusta con **Min** y **Max** en la ventana del diálogo de propiedades. La resolución es de 128 pasos. Un evento Note Off ajusta la salida a cero. Para desactivar la dinámica de pulsación tendrás que ajustar **Min** y **Max** al mismo valor.

Single Trig. Gate



Fuente de evento monofónica para eventos MIDI Note On y Note Off con característica de activación simple.

Un evento Note On ajusta la salida a un valor determinado por la intensidad de pulsación (Velocity). El Rango del valor de salida se ajusta con **Min** y **Max** en la ventana del diálogo de propiedades. La resolución es de 128 pasos. Un evento Note Off ajusta la salida a cero. Para desactivar la dinámica de pulsación tendrás que ajustar **Min** y **Max** al mismo valor.

Sólo produce un evento la primera nota, y por lo tanto, activa (o re-activa) una envolvente conectada. Una nueva nota que empieza mientras la anterior todavía se mantiene (legato) no genera un evento y no activa la envolvente.

Sel. Note Gate

MIDI In



Fuente de evento monofónica para eventos Note On y Note Off seleccionados.

Un evento Note On que tiene el número de nota configurado ajusta la salida a un valor determinado por la intensidad de pulsación (Velocity). El Rango del valor de salida se ajusta con **Min** y **Max** en la ventana del diálogo de propiedades. La resolución es de 128 pasos. Un evento Note Off con el número de tecla configurado ajusta la salida a cero. Para desactivar la dinámica de pulsación tendrás que ajustar **Min** y **Max** al mismo valor.

On Velocity

MIDI In



Fuente de evento polifónica para eventos MIDI Note On Velocity. Un evento Note On ajusta la salida a un valor determinado por la intensidad de pulsación. El Rango del valor de salida se ajusta con **Min** y **Max** en la ventana del diálogo de propiedades. La resolución es de 128 pasos.

Off Velocity

MIDI In



Fuente de evento polifónica para eventos MIDI Note Off Velocity. Un evento Note Off ajusta la salida a un valor determinado por la velocidad al soltar la tecla. El Rango del valor de salida se ajusta con **Min** y **Max** en la ventana del diálogo de propiedades. La resolución es de 128 pasos.

Hay pocos teclados que generen un valor que no sea cero para este evento.

Controller

MIDI In



Fuente de evento monofónica para eventos MIDI Controller. Un evento ajusta la salida a un valor determinado por la posición del controlador (rueda de modulación). El número del controlador se ajusta en la ventana del diálogo de propiedades con **Controller No** y el Rango del valor de salida se ajusta con **Min** y **Max**. La resolución es de 128 pasos.

La posición actual de todos los controladores MIDI (y faders) de un instrumento se pueden guardar en una instantánea y volver a cargar en otro momento. Si **Snap Isolate** está activado, el valor de salida del módulo controlador no responderá al cargar una instantánea.

La salida de un controlador es monofónica y se puede usar como señal de evento o de audio.

Los números de algunos controladores MIDI estándar son:

- 1 Rueda de modulación
- 2 Controlador de Breath
- 7 Volumen
- 10 Panoramización
- 64 Sustain
- 65 Portamento
- 66 Hold (Sostenuto)

Mira la implementación de tu teclado MIDI para comprobar cuáles son los controles que se pueden transmitir.

Ch. Aftertouch

MIDI In



Fuente de evento monofónica para eventos MIDI Channel Aftertouch. Un evento ajusta la salida a un valor determinado por la presión de las teclas. El Rango del valor de salida se ajusta con **Min** y **Max** en la ventana del diálogo de propiedades. La resolución es de 128 pasos.

Poly Aftertouch

MIDI In



Fuente de eventos polifónica para eventos MIDI Poly Aftertouch. Un evento ajusta la salida a un valor determinado por la presión de las teclas. El valor sólo se ajusta para la voz en particular del instrumento Reaktor que toca la nota asociada a la tecla. El Rango del valor de salida se ajusta con **Min** y **Max** en la ventana del diálogo de propiedades. La resolución es de 128 pasos.

Hay pocos teclados que generen eventos Poly Aftertouch.



Fuente de evento monofónica para eventos MIDI Poly Aftertouch.

Un evento que tiene el número de nota seleccionado ajusta la salida a un valor determinado por la presión de las teclas. El número de la tecla (Note Number) se ajusta en el diálogo de Propiedades con **Controller No** y el Rango del valor de salida se ajusta con **Min** y **Max**. La resolución es de 128 pasos.

Hay pocos teclados que generen eventos Poly Aftertouch.

El valor actual se puede guardar (junto con los faders y controladores MIDI) en una instantánea desde donde puedes volver a cargarlos en cualquier momento. Si el interruptor **Snap Isolate** está activado, el valor de salida del módulo no cambiará cuando cargues la instantánea.

Program Change

MIDI In



Fuente de eventos monofónica para eventos de cambio de programa MIDI. Un evento MIDI ajusta la salida del módulo a un valor determinado por el número de programa transmitido. El Rango del valor de salida se ajusta con **Min** y **Max** en la ventana del diálogo de propiedades. La resolución es de 128 pasos.

Al usar este módulo, es posible que quieras apagar la opción **Prog. Change Enable** en las Propiedades del Instrumento para que los eventos MIDI Program Change no carguen además instantáneas.

Start/Stop

MIDI In



Fuente para eventos de Start y Stop para la sincronización con dispositivos externos o con el reloj maestro interno.

La salida del módulo **Start/Stop** es una señal de puerta monofónica. Su valor se puede ajustar con **Output Value** en las propiedades. La señal salta al valor ajustado cuando se presiona el botón de inicio en la barra de herramientas, o cuando se recibe un evento MIDI Start. La señal vuelve a saltar a cero cuando se presiona el botón Stop o cuando se recibe un evento MID Stop.

El módulo normalmente se conecta a la entrada de reinicio de los secuenciadores y divisores de eventos sincronizados con el reloj MIDI para provocar así un inicio sincronizado.

1/96 Clock

MIDI In



Fuente de una señal de reloj que corresponde al reloj MIDI externo o al reloj maestro interno.

Para cada nota 1/96 se envía un evento a la salida. El valor se puede ajustar como **Output Value** en las Propiedades.

Al contrario que con la fuente **Sync Clock**, los eventos de Reloj **1/96** se han de procesar a través de un **Event Freq. Divider**. Tiene la ventaja de que permite experimentar con factores de división arbitrarios.

Sync Clock

MIDI In



Fuente para una señal de reloj que se deriva de un reloj MIDI externo o del reloj maestro interno.

La salida del módulo **Sync Clock** es una señal de puerta monofónica. El valor se puede ajustar con **Output Value** en las Propiedades. A cada beat, la puerta salta a su valor respectivo y regresa a cero después de cierto intervalo de tiempo. El módulo se activa y desactiva a través de eventos Start/Stop

La velocidad de la señal de reloj (negras, corcheas, semicorcheas, etc.) se puede ajustar en **Rate** dentro de las propiedades.

La duración de la señal de puerta (corchea, semicorchea, etc.) se puede ajustar en **Duration** dentro de las propiedades.

Song Pos

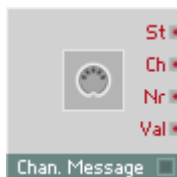


MIDI In

Fuente para Song Position, que cuenta en notas 1/96 desde el inicio de la canción. Normalmente se conecta a la entrada (A) de **Modulo**, y una constante de 6 a la entrada (B) para conseguir una cuenta de semicorcheas en la salida (Div).

- **96**: Salida de evento para la cuenta entera de 1/96 (24 por negra). Usa **Modulo** para conseguir cuentas con otras unidades de notas.
- **96a**: Salida de audio para la posición de la canción fraccional medida en notas 1/96 (24 por negra).

Channel Message



MIDI In

Fuente monofónica para recibir mensajes de canal MIDI desde un dispositivo MIDI externo (teclado, secuenciador, etc.) o internamente desde otro instrumento dentro del ensemble. El orden de salida de los puertos (mira abajo) está estrictamente definido de arriba abajo. Así asegurarás que el tipo (por ejemplo, Control Change) y la fuente (por ejemplo, controlador 7 en el canal 1) del mensaje se transmita siempre antes del valor actual.

St: Puerto de salida que informa del tipo de mensaje recibido.

- 0 = Note Off
- 1 = Note On
- 2 = Poly Aftertouch
- 3 = Control Change
- 4 = Program Change
- 5 = Channel Aftertouch
- 6 = Pitchbend

Ch: Número del canal MIDI (1-16).

Nr: Número de una Nota, Control Change, o Program Change (0-127).

Val: Velocidad de una Nota, presión de Aftertouch, o valor de Control Change y Pitchbend. Con MIDI externo, los valores son 7-Bit cuantizados (14-Bit para Pitchbend). Con MIDI interno, los valores son puntos flotantes ilimitados 32 –Bit. Todos los valores se ponen en escala de acuerdo con los ajustes de Min y Max en las propiedades, que es desde 0 a 1 por defecto. Otro ajuste común es de 0 a 127, para ayudar en la interpretación de valores en ciertas situaciones (por ejemplo, el cambio de programa).

MIDI Out

Reaktor puede generar (además de procesar) mensajes MIDI. Los módulos MIDI Out se usan para enviarlos a dispositivos externos. Hay módulos que se corresponden con los MIDI In. Algunos módulos MIDI Out también se pueden usar en una conexión interna u OSC.

Note Pitch/Gate



MIDI Out

Convierte una señal de evento monofónica o polifónica en eventos MIDI Note. Cada evento en la entrada **G** de la puerta genera un mensaje MIDI en el puerto MIDI que Reaktor usa como salida. El valor del evento especifica la intensidad de pulsación (Velocity). Un valor de evento de 1 produce una intensidad de pulsación de 127. Un evento cuyo valor es cero, produce un evento Note Off.

La tonalidad de los eventos Note On y Note Off es el valor actual de la entrada de tonalidad **P** en el Rango establecido por **Min** y **Max**. Un evento con valor igual a **Min** ajusta el valor de MIDI Note Pitch a 0; un evento con un valor igual a **Max** ajusta el valor de MIDI Note Pitch a 127.

Pitchbend



MIDI Out

Convierte una señal de evento monofónica en eventos MIDI Pitchbend. Cada evento genera un mensaje MIDI en el puerto MIDI que Reaktor usa como salida. El Rango del valor de entrada se ajusta con **Min** y **Max**. La resolución de salida es 16384 pasos. Un evento con un valor igual a **Min** ajusta el valor de MIDI Pitchbend a -8192; un evento con valor igual a **Max** ajusta el valor de MIDI Pitchbend a +8191

Controller

MIDI Out



Convierte una señal de evento monofónica en eventos de controlador MIDI. Cada evento genera un mensaje MIDI en el puerto MIDI que Reaktor usa como salida. El número del controlador se ajusta en el diálogo de propiedades con **Controller No** y el Rango del valor de entrada se ajusta con **Min** y **Max**. La resolución de salida es 128 pasos. Un evento con valor igual a **Min** ajusta el valor del controlador MIDI a 0; un evento con valor igual a **Max** ajusta el valor de controlador MIDI a 127.

Ch. Aftertouch

MIDI Out



Convierte una señal de evento monofónica en eventos MIDI Chanel Aftertouch. Cada evento genera un mensaje MIDI en el puerto MIDI que Reaktor usa como salida. El Rango del valor de entrada se ajusta con **Min** y **Max**. La resolución de salida es de 128 pasos. Un evento con valor igual a **Min** ajusta el valor del MIDI Aftertouch a 0; un evento con valor igual a **Max** ajusta el valor del MIDI Aftertouch a 127.

Poly Aftertouch

MIDI Out



Los eventos Aftertouch (**AT**) se convierten en eventos MIDI Poly Aftertouch. El valor momentáneo en la entrada Pitch (**P**) se convierte en el número de nota. Los valores entre **Min** y **Max** ofrecen como resultado números de nota

entre 0 y 127. Los valores Aftertouch entre 0 y 1 se convierten en valores entre 0 y 127.

- **P**: Entrada para la tonalidad (Pitch) convertida en número de nota MIDI. Los valores entre **Min** y **Max** ofrecen como resultado números de nota entre 0 y 127.
- **AT**: Entrada para la señal Aftertouch. Los valores entre 0 y 1 se convierten en valores MIDI Aftertouch entre 0 y 127.

Sel. Poly AT

MIDI Out



Convierte una señal de evento monofónica en eventos MIDI Poly Aftertouch. Cada evento genera un mensaje MIDI en el puerto MIDI que Reaktor usa como salida. El número de la tecla para la que se ajusta la presión (Note Number) se establece en el diálogo de propiedades con **Note No** y el rango del valor de entrada se ajusta con **Min** y **Max**. La resolución de salida es de 128 pasos. Un evento con valor igual a **Min** ajusta el valor de Aftertouch a 0; un evento con valor igual a **Max** ajusta el valor de Aftertouch a 127.

Hay pocos dispositivos MIDI que procesen eventos Poly Aftertouch.

Program Change

MIDI Out



Convierte una señal de evento monofónica en eventos MIDI Program Change. Cada evento genera un mensaje MIDI en el puerto MIDI que Reaktor usa como salida. El Rango del valor de entrada se ajusta con **Min** y **Max**. La resolución de salida es de 128 pasos. Un evento con valor igual a **Min** ajusta el valor de MIDI Program Change a 0; un evento con valor igual a **Max** ajusta el valor de MIDI Program Change a 127.

Start/Stop

MIDI Out



Los eventos de entrada crean eventos Start/Continue/Stop en la salida MIDI.

- **G:** Un evento positivo envía un mensaje Start o Continue. Un evento negativo o cero envía un mensaje Stop.
- **Rst:** Después de un evento positivo en la entrada Reset, el siguiente evento positivo en Gate envía un mensaje Start (a cero).

1/96 Clock

MIDI Out

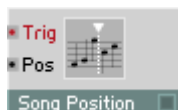


Los eventos de entrada crean eventos de reloj (1/96) en la salida MIDI.

- **In:** Un evento con valor positivo crea un evento de reloj MIDI.

Song Pos

MIDI Out



El valor de la entrada **Pos** se envía con un evento en la entrada **Trig** como Song Position.

- **Trg:** Cada evento con un valor positivo crea un evento MIDI Song Position.
- **Pos:** El valor en esta entrada (como múltiplo de notas 1/96)) junto con un evento Trigger se envían como MIDI Song Position.

Channel Message

MIDI Out



Fuente monofónica para enviar mensajes de canal MIDI a un dispositivo MIDI externo (secuenciador, etc.), o internamente a otro instrumento dentro del ensemble. Los mensajes de canal se transmiten cada vez que un evento llega al puerto **St**. Por lo tanto, el valor del mensaje deseado y el destino han de definirse correctamente con valores apropiados en las otras entradas antes de que lleguen los eventos a la entrada **St**. Todas las entradas aceptan solamente señales monofónicas.

- **St:** Entrada de evento que define el tipo de mensaje de canal que se va a enviar. Se envía un mensaje de canal cada vez que llega un evento a esta entrada.
 - 0 = Note Off
 - 1 = Note On
 - 2 = Poly Aftertouch
 - 3 = Control Change
 - 4 = Program Change
 - 5 = Channel Aftertouch
 - 6 = Pitchbend
- Ch:** Entrada de audio para el número de canal MIDI (1-16). Los valores fraccionales que llegan a Ch se redondean hacia arriba al entero más cercano; por ejemplo, un valor de 0.5 se redondearía a 1.
- Nr:** Entrada de audio para el número de Nota, Control Change o Program Change (0-127).
- Val:** Entrada de audio para Velocity de la nota, presión de Aftertouch, o valor de Control Change y Pitchbend.

Math

Tú calculas la ecuación, y Reaktor la lleva a cabo. Aquí encontrarás módulos para realizar las operaciones matemáticas más comunes como suma, resta, multiplicación y división, y otro tipo de funciones matemáticas menos familiares como valor absoluto, arco-tangente o valor inverso de la raíz. También hay módulos de conversión exponencial y logarítmica para conseguir entradas de módulos en Reaktor de diferentes escalas. Por ejemplo, algunos módulos de reaktor tienen entradas de frecuencia lineal (medidas en Hz) mientras que otros tienen entradas de frecuencia exponencial (medidas en semitonos y ajustadas a través de una numeración de notas MIDI). Hay módulos para convertir entre estos dos formatos.

Todos los módulos de esta sección son módulos híbridos, lo que quiere decir que se pueden usar como módulos de evento o de audio dependiendo de sus entradas. Muchos de los módulos (**Add** y **Mult**, por ejemplo) también tienen entradas dinámicas indicadas por tres pequeños puntos abajo a la izquierda del icono del módulo.

Si se arrastra un cable hasta una parte vacía en la sección de puertos de entrada (el borde izquierdo del icono del módulo) de un módulo con entradas dinámicas mientras se sujeta la tecla **Ctrl**, se creará una nueva entrada automáticamente.

Constant

Math



Fuente monofónica para un valor constante. El valor se ajusta con **Constant Value** en la ventana del diálogo de propiedades. El módulo sólo envía un evento una vez, para la inicialización cuando se activa.

Add

Math



Suma dos o más señales de evento o de audio. La señal de salida es la suma de las señales de entrada (**Out = In1 + In2 + In3** etc.).

También se puede usar como simple mezclador multi-canal donde el nivel de todos los canales se fija en 0 dB

Subtract

Math



Sustractor para dos señales de evento o de audio. La señal de salida es el resultado de restar la segunda señal de entrada a la primera. (**Out = In1 - In2**).

Invert, -X

Math



Invierte una señal de audio o de evento. La señal de salida es la inversa de la señal de entrada (**Out = -In**).

Si simplemente se invierte un sonido, no se notará ningún efecto, excepto cuando se combine de alguna forma con el sonido sin invertir. Pero si se invierten señales de control, normalmente tiene efectos muy audibles.

Multiply



Math

Multiplicador para dos más señales de evento o de audio. La señal de salida es el producto de las señales de entrada (**Out = In1 * In2 * In3 etc**).

La aplicación más común es un amplificador controlado por una señal (correspondiente a VCA en los sintetizadores analógicos) cuando se aplica una señal de audio en una entrada, y un factor de amplificación en la otra.

Si se conecta un valor cero en alguna de sus entradas, su valor de salida siempre será cero.

También se puede usar para calcular el cuadrado de una señal cuando la misma señal se conecta a dos entradas (**Out = In * In = In²**).

Cuando se conectan dos sonidos diferentes a las entradas, la salida será la modulación anular de los dos sonidos.

$a * b + c$



Math

Combinado de multiplicador-sumador:

La salida es el resultado de $(A * B) + C$.

Reciprocal $1/x$



Math

La salida es uno dividido por la entrada.

Hay que tener cuidado con los valores de entrada bajos (cerca de cero) ya que los valores de salida serán muy altos. Si la señal de entrada es exactamente cero, la salida también será cero.

Normalmente, procesar señales de sonido no resulta muy útil. El módulo se usa más bien para procesar señales de control (que no estén cerca de cero).

Divide x/y



Math

La salida es la entrada superior dividida entre la entrada inferior.

Ten cuidado con los valores bajos (cerca de cero) de la entrada inferior ya que los valores de salida serán muy altos. Si la señal de entrada es exactamente cero, la salida también será cero.

Dividir entre señales de sonido no resulta muy útil.

Siempre que puedas, usa un multiplicador en lugar de un divisor para señales de audio, ya que la carga de CPU se reducirá considerablemente. Por ejemplo, en lugar de dividir entre una constante o una señal de evento, invierte la señal de evento ($1/x$) y multiplica el audio por el resultado.

Modulo x % y



Math

Calcula la división entera de dos valores de entrada además del resto.

- **A:** entrada híbrida para que una señal **A** se divida entre **B**. Rango típico: [0 ... 100].
- **B:** Entrada híbrida **B** que se usa para dividir **A**. **B** normalmente es un número entero, aunque no tiene por qué serlo. Rango: [1 ... 100].
- **Div:** Salida híbrida para la división entera de **A** y **B**. Es el número entero mayor, menor que o igual a A/B . Rango típico: [0 ... 100].
- **Mod:** Salida híbrida para el Modulo de **A** y **B**. Es el resto de la división entera entre **A** y **B**. Rango: [0 ... B].

Rectifier



Math

La señal de entrada se rectifica, es decir, los valores negativos se invierten y se hacen positivos.

- **In:** Entrada híbrida para la señal que se va a rectificar. Los valores negativos se hacen positivos con igual magnitud.
- **Out:** Salida híbrida para la señal rectificada.

Rect./Sign



Math

La señal de entrada se rectifica, es decir, los valores negativos se invierten y se hacen positivos. El signo de la señal de entrada queda disponible en la salida **Sign**.

- **In:** Entrada híbrida para la señal que se quiere rectificar y cuyo signo se quiere analizar.
- **Sign:** Salida híbrida para el signo de la señal de entrada. 1 para señales positivas; -1 para señales negativas.
- **Out:** Salida híbrida para la señal rectificada. Siempre es positiva.

Compare



Math

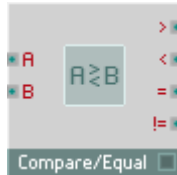
Función de Comparación Lógica. Compara los dos valores de entrada y ajusta las dos salidas de acuerdo con el resultado de la comparación.

- **A:** Entrada híbrida para el primero de los dos valores que se van a comparar.
- **B:** Entrada híbrida para el segundo de los valores que se van a comparar.
- **>:** Salida híbrida para el resultado de la comparación “**A** mayor que **B**”. (0 = FALSO, 1 = VERDADERO)

- **<:** Salida híbrida para el resultado de la comparación “**A** menor que **B**”. (0 = FALSO, 1 = VERDADERO)

Compare/Equal

Math

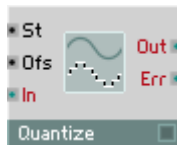


Función de Comparación Lógica. Compara los dos valores de entrada y ajusta las cuatro salidas de acuerdo con el resultado de la comparación.

- **A:** Entrada híbrida para el primero de los dos valores que se van a comparar.
- **B:** Entrada híbrida para el segundo de los valores que se van a comparar.
- **>:** Salida híbrida para el resultado de la comparación “**A** mayor que **B**”. (0 = FALSO, 1 = VERDADERO)
- **<:** Salida híbrida para el resultado de la comparación “**A** menor que **B**”. (0 = FALSO, 1 = VERDADERO)
- **=:** Salida híbrida para el resultado de la comparación “**A** igual que **B**”. (0 = FALSO, 1 = VERDADERO)
- **!=:** Salida híbrida para el resultado de la comparación “**A** no igual que **B**”. (0 = FALSO, 1 = VERDADERO)

Quantize

Math



Cuantizador para señales de evento y de audio, con ancho de paso (Step Size) ajustable.

La señal de entrada se redondea al paso de cuantización más cercano antes de salir.

- **St:** Entrada híbrida para controlar el ancho del paso de cuantización. Cuando está a cero, la señal no se cuantiza.

- **In:** Entrada híbrida para la señal que se va a cuantizar.
- **Out:** Salida híbrida para la señal cuantizada.
- **Err:** Salida híbrida para el error de cuantización que se genera con el redondeo: $\text{Err} = \text{Out} - \text{In}$.

Expon. (A)



Math

Función exponencial para convertir valores de niveles logarítmicos en dB en valores de amplitud lineales.

- **Lvl:** Entrada híbrida para los valores de nivel logarítmicos en dB que se van a convertir en valores de amplitud lineales. Rango típico: [-50 ... 10].
- **A:** Salida híbrida para la señal de control de amplitud lineal.

Expon. (F)



Math

Función exponencial para convertir valores de tonalidad logarítmicos en semitonos en valores de frecuencia lineal en Hz.

- **P:** Entrada híbrida para valores de tonalidad logarítmicos en semitonos que se van a convertir en valores de frecuencia lineal en Hz. Rango típico: [0 ... 127].
- **F:** Salida híbrida para una señal de control de frecuencia en Hz.

Log (A)



Math

Función logarítmica para convertir valores de amplitud lineales en valores de nivel logarítmicos en dB. También sirve para conducir entradas de tiempo logarítmicas de envolventes, etc.

- **A:** Entrada híbrida para el valor de amplitud lineal que se va a convertir en valor de nivel logarítmico en dB. Rango típico: [0 ... 1000].
- **Lvl:** Salida híbrida para el nivel en dB. Rango típico: [-60 ... 0].

Log (F)



Math

Función logarítmica para convertir valores de frecuencia lineal en Hz en valores de tonalidad logarítmicos en semitonos.

- **F:** Entrada híbrida para el valor de frecuencia lineal en Hz que se va a convertir en valor de tonalidad logarítmico en semitonos. Rango típico: [0 ... 5000].
- **P:** Salida híbrida para la tonalidad en semitonos. Rango típico: [0 ... 100].

Power x y



Math

Función potencial. La salida entrega el resultado de X a la potencia de Y (normalmente representada como X^Y o X^Y).

- **X:** Entrada híbrida para la base.
- **Y:** Entrada híbrida para el exponente.
- **X^Y :** Salida híbrida para el resultado del cálculo.

Square Root



Math

Calcula la raíz cuadrada de los valores de entrada.

- **In:** Entrada híbrida para el argumento de la función de raíz cuadrada. Para valores de entrada negativos, la salida es cero. Rango típico: [0 ... 100].
- **Out:** Salida híbrida para la raíz cuadrada del valor de entrada. Rango típico: [0 ... 10].

1 / Square Root



Math

Este módulo calcula el inverso de la raíz cuadrada de la entrada. Es más interesante usar este módulo en lugar del Square Root seguido del Reciprocal. Para valores de entrada negativos, la salida será cero.

- **In:** Entrada híbrida para el argumento.
- **Out:** Salida híbrida para el valor.

Sine



Math

Este módulo calcula la función senoidal trigonométrica. Tanto la entrada como la salida tienen un rango de -1 a 1. Para calcular una entrada basada en grados, primero tendrás que dividir entre 360. Para calcular una entrada basada en radianes, divide entre 2π (aproximadamente 6.283). El rango de salida es de 1 (para entrada 0.25) a -1 (entrada 0.75).

- **In:** Entrada híbrida para el argumento.
- **Out:** Salida híbrida para el valor.

Sine/Cos



Math

Para cada evento de entrada, calcula el seno y el coseno. Un valor de entrada de 1.0 corresponde a un período completo de las funciones de seno y coseno (es decir, 360 grados).

- **In:** Entrada híbrida para el argumento de la función de seno. Un valor de 1.0 corresponde a un período de la función de seno (360 grados). Rango típico: [-1 ... 1].
- **Sin:** Salida híbrida para el seno del valor de entrada. Rango: [-1 ... 1].
- **Cos:** Salida híbrida para el coseno del valor de entrada. Rango: [-1 ... 1].

Arcsin

Math



Este módulo calcula la función de seno inversa. (El arco seno de x es el número entre 0 y 1 cuyo seno es x). Puesto que el Rango de salida de la función de seno es -1 a 1, la función de arco seno sólo es válida para entradas dentro del Rango. Para argumentos < -1 el módulo entrega -0.25 y para argumentos > 1 , 0.25.

- **In:** Entrada híbrida para el argumento.
- **Out:** Salida híbrida para el arco seno de la entrada.

Arccos

Math



Este módulo calcula la función inversa del coseno. (El arco coseno de x es el número entre 0 y 1 cuyo coseno es x). Puesto que el Rango de salida de la función de coseno está entre -1 a 1, la función de arco coseno sólo es válida para entradas dentro de ese Rango. Para argumentos < -1 el módulo entrega 0.5 y para argumentos > 1 , entrega 0.

- **In:** Entrada híbrida para el argumento.
- **Out:** Salida híbrida para el arco coseno de la entrada.

Arctan

Math



Este módulo calcula la función de la tangente inversa. (La tangente es el seno dividido por el coseno). El Rango de salida es desde -0.25 a 0.25, que se corresponde con -90 a 90 grados.

- **In:** entrada híbrida para el argumento.
- **Out:** Salida híbrida para el arco tangente de la entrada.

Signal Path

Los módulos de ruta de la señal permiten encaminar flexiblemente tanto señales de audio como de control en las estructuras de Reaktor. Se incluyen mezcladores, selectores de entrada y salida, interruptores de control remoto (se llaman Relays), fundidos cruzados, y panoramizadores.

Selector/Scanner



Signal Path

Selector/Scanner. Las entradas se leen a través del valor en la entrada **Pos**. Cuando **Pos** es un número entero, obtendrás sólo una señal de entrada, si no, obtendrás una mezcla de de dos entradas. La señal de salida se obtiene haciendo un fundido cruzado entre las dos señales cuyos índices están más cerca del valor de la entrada **Pos**. El fundido cruzado se hace de acuerdo con el modo de fundido especificado en las Propiedades.

Cuando se ha seleccionado el modo **Wrap** en las Propiedades, **Pos** trabaja como bucle, y así $\text{Max}+1$ es lo mismo que 0, $\text{Max}+2$ es 1, etc.

El Selector/Scanner crea interesantes efectos cuando las entradas se conectan desde el Multitap Delay y la posición de lectura se controla con un Ramp Oscillator.

El módulo tiene una administración dinámica de los puertos. El número de los puertos de entrada se puede definir con **Min Num Port Groups** en la página **Function** de las Propiedades.

- **Pos**: Entrada híbrida para seleccionar la entrada(s) que se van a escanear. **Pos** = 0 selecciona **In0**, **Pos** = 1 selecciona **In1**, **Pos** = 0.5 da una mezcla de **In0** y **In1**. Rango típico: [0 ... Max]
- **In 0...Max**: Entrada híbrida para las señales que se van a escanear.
- **Out**: Salida para la señal escaneada.

Relay 1,2



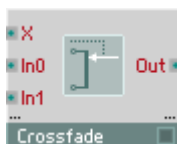
Signal Path

La entrada superior está conectada a la salida si $Ctl > 0$. Si no, será la entrada inferior la que esté conectada a la salida. Si la señal inferior no existe, se producirá un señal cero en la salida.

El módulo puede tener uno o dos puertos de entrada. El número de puertos de entrada se puede definir en **Min Num Port Groups** en la página **Function** de las Propiedades.

- **Ctl**: Entrada de control para seleccionar una señal de entrada.
- **In**: Entrada(s) de señal.
- **Out**: Salida para la señal seleccionada.

Crossfade



Signal Path

Módulo de fundido cruzado para dos señales. La señal de salida mezcla las dos señales de entrada **In0** y **In1**. La proporción de mezcla se controla a través de la entrada **X**.

El módulo tiene una administración dinámica de los puertos. El número de puertos pares de entradas (**In0** y **In1**) y de salidas se puede definir con **Min Num Port Groups** en la página **Function** de las Propiedades.

- **X**: Entrada híbrida de control para la cantidad de mezcla. Valor entre 0 ($Out = In1$) y 1 ($Out = In2$).
- **In1, In2**: Entradas híbridas para las dos señales que se van a mezclar.
- **Out**: Salida híbrida para la señal mezclada. ($Out = (1 - X) In1 + X In2$).



Distribuidor/panoramizador. La señal de la entrada se conectará/panoramizará a la salida(s) seleccionada en la entrada Pos. Cuando Pos es un número entero, la señal se conectará a una sola señal de entrada; si no, tendrás una panoramización entre las dos entradas. La panoramización se hace de acuerdo con el modo establecido en las Propiedades. Si el modo Wrap está seleccionado en las Propiedades, Pos trabajará como bucle, y así Max+1 será lo mismo que 0, Max+2 será 1, etc.

El módulo tiene una administración dinámica de los puertos. El número de puertos de salida se puede especificar en **Min Num Port Groups** en la página **Function** de las Propiedades.

- **Pos:** Entrada para seleccionar la salida.
- **In:** Entrada de la señal.
- **Out:** Salida para la señal de entrada.



Panoramizador izquierda/derecha. Cambiando el nivel de la señal de las dos salidas, la señal de entrada se posiciona en un campo estéreo. La suma de valores de salida **L** y **R** es siempre el doble del valor de entrada, es decir, la entrada se distribuye en las dos salidas en una tasa variable.

El módulo tiene una administración dinámica de los puertos. El número de puertos pares de entradas y salidas (**L** y **R**) se puede definir en **Min Num Port Groups** de la página **Function** en las Propiedades.

- **Pan:** Entrada de control para la posición izquierda/derecha. Valor -1 = izquierda; 0 = centro; +1 = derecha.
- **In:** Entrada híbrida de la señal para la señal que se quiere panoramizar.
- **L:** Salida híbrida de la señal para la señal del canal izquierdo.
- **R:** Salida híbrida de la señal para la señal del canal derecho.



Amplificador/Mezclador para un número ajustable de señales de entrada. Las señales de entrada se amplifican (o atenúan) en las cantidades respectivas de las entradas **Lvl** (en dB) y luego se suman para formar la salida.

El módulo tiene una administración dinámica de los puertos. El número de puertos pares de entrada (**Lvl** y **In**) se puede definir con **Min Num Port Groups** en la página **Function** de las Propiedades.

- **Lvl**: Entrada de control logarítmica para controlar la ganancia, valor en dB.
 - **In**: Entrada de audio para la señal que se va a amplificar.
 - **Out**: Salida para la señal mezclada
- $$(\text{Out} = \text{In1} \cdot 10^{\text{Lvl1}/20} + \text{In2} \cdot 10^{\text{Lvl2}/20} \text{ etc.}).$$



Amplificador para un número ajustable de señales de audio con panoramizador izquierda/derecha incorporado. Las señales de entrada se amplifican (o atenúan) en la cantidad ajustada en las entradas **Lvl** (en dB). Cambiando el nivel de la señal en las dos salidas, las señales de entrada se posicionan en el campo estéreo.

El módulo tiene una administración dinámica de los puertos. El número de grupos de puertos de entrada (**Lvl**, **Pan** y **In**) se pueden definir en **Min Num Port Groups** en la página **Function** de las Propiedades.

- **Lvl**: Entrada logarítmica para controlar la ganancia, valor en dB. Cuando el valor es negativo, la señal de salida es menor que la señal de entrada.
- **Pan**: Entrada de control para la posición izquierda/derecha. Valor -1 = izquierda, 0 = centro, 1 = derecha.

- **In:** Entrada de señal de audio para la señal que se quiere amplificar y panoramizar.
- **L:** Salida de la señal de audio para la señal amplificada del canal izquierdo.
- **R:** Salida de la señal de audio para la señal amplificada del canal derecho.

Oscillator

Reaktor usa el término oscilador para una gran cantidad de generadores de señal. De hecho, cualquier proceso de generación de señales que no implique usar samples se llama oscilador. Aquí se incluyen los usuales generadores de forma de onda de ciclo sencillo, generadores de pulso, de paso, de ruido, de tabla, etc.

Todos los osciladores de Reaktor pueden funcionar a cualquier frecuencia, desde 0Hz (parado), a través del espectro total de audio, hasta los límites establecidos por la frecuencia de muestreo. De esta forma, todos se pueden usar como LFOs o para producir ondas de sonido. Si se usa un oscilador como LFO para modular la entrada de otro módulo que sólo acepta eventos (por ejemplo, P en lugar de F) tendrás que insertar un módulo A to E para la conversión.

Sawtooth



Oscillator

Oscilador para forma de onda de diente de sierra con control de tonalidad logarítmico y modulación de amplitud lineal.

- **P:** Entrada de evento logarítmica para controlar la tonalidad (frecuencia del oscilador). Valor en semitonos (69 = 440 Hz).
- **A:** Entrada de audio para controlar la amplitud. La señal de salida se mueve entre +A y -A.
- **Out:** Salida de la señal de audio para la forma de onda de diente de sierra.

Saw FM

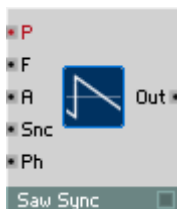


Oscillator

Oscilador para forma de onda de diente de sierra con control de tonalidad logarítmico, modulación de frecuencia lineal (FM) y modulación de amplitud lineal.

- **P:** Entrada de evento logarítmica para controlar la tonalidad (frecuencia del oscilador). Valor en semitonos (69 = 440 Hz).
- **F:** Entrada de audio para la modulación de frecuencia lineal. Valor en Hz. **P** y **F** juntas determinan la frecuencia del oscilador.
- **A:** Entrada de audio para controlar la amplitud. La señal de salida se mueve entre +A y -A.
- **Out:** Salida de la señal de audio para la forma de onda de sierra.

Saw Sync



Oscillator

Oscilador para formas de onda de sierra con sincronización, control de tonalidad logarítmico, modulación de frecuencia lineal (FM) y modulación de amplitud lineal.

Cada vez que la señal de sincronización vaya desde cero hasta valores positivos (borde ascendente) la fase del oscilador se reiniciará a la posición especificada en la entrada de fase.

- **P:** Entrada de evento logarítmica para controlar la tonalidad (frecuencia del oscilador). Valor en semitonos (69 = 440 Hz).
- **F:** Entrada de audio para la modulación de frecuencia lineal. Valor en Hz. **P** y **F** juntas determinan la frecuencia del oscilador.
- **A:** Entrada de audio para controlar la amplitud. La señal de salida se mueve entre +A y -A.

- **Snc:** Entrada de audio para controlar la sincronización de la forma de onda. Un borde ascendente reinicia el oscilador.
- **Ph:** Entrada para especificar la fase (posición de la forma de onda) a la que el oscilador se reinicia cuando ocurre la sincronización.
- **Out:** Salida de la señal de audio para la forma de onda de diente de sierra.

Saw Pulse

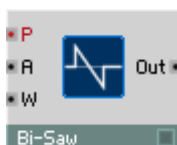


Oscillator

Oscilador para forma variable de diente de sierra/pulso de aguja, con control de tonalidad logarítmico y modulación de amplitud lineal. La pendiente de la rampa es variable y con ella se puede cambiar la forma de la onda desde un diente de sierra normal a un pequeño pulso triangular.

- **P:** Entrada de evento logarítmico para controlar la tonalidad (frecuencia del oscilador). Valor en semitonos ($69 = 440$ Hz).
- **A:** Entrada de audio para controlar la amplitud. La señal de salida se mueve entre $+A$ y $-A$.
- **Slp:** Entrada de audio para controlar la pendiente de la forma de onda. Un valor 0 produce un diente de sierra normal; un valor mayor acorta la rampa hasta dejar una punta pequeña (aguja).
- **Out:** Salida de la señal de audio para la forma de onda de diente de sierra/pulso.

Bi-Saw



Oscillator

Oscilador bipolar para forma de onda de diente de sierra con fase cero. Lleva control de tonalidad logarítmico, modulación del ancho de pulso (PWM) y modulación de amplitud lineal.

- **P:** Entrada de evento logarítmico para controlar la tonalidad (frecuencia del oscilador). Valor en semitonos. ($69 = 440$ Hz).

- **A:** Entrada de audio para controlar la amplitud. La señal de salida se mueve entre $+A$ y $-A$.
- **W:** Entrada de audio para controlar el ancho de pulso (PWM). El Rango de los valores va desde 0 hasta 6 aproximadamente. Con **W** = 0 se obtiene un diente de sierra normal; con **W** mayor que 0, se obtiene pulsos más cortos y fases cero más largas.
- **Out:** Salida de la señal de audio para la forma de onda bipolar de diente de sierra con fase cero.

Triangle



Oscillator

Oscilador con forma de onda triangular simétrica con control de tonalidad logarítmico y modulación de amplitud lineal.

- **P:** Entrada de evento logarítmica para controlar la tonalidad (frecuencia del oscilador). Valor en semitonos ($69 = 440$ Hz).
- **A:** Entrada de audio para controlar la amplitud. La señal de salida se mueve entre $+A$ y $-A$.
- **Out:** Salida de la señal para la forma de onda triangular.

Tri FM

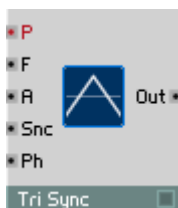


Oscillator

Oscilador para forma de onda triangular simétrica con control de tonalidad logarítmico, modulación de frecuencia lineal (FM) y modulación de amplitud lineal.

- **P:** Entrada de evento logarítmica para controlar la tonalidad (frecuencia del oscilador). Valor en semitonos ($69 = 440$ Hz).
- **F:** Entrada de audio para la modulación de frecuencia lineal. Valor en Hz. **P** y **F** juntas determinan la frecuencia del oscilador.
- **A:** Entrada de audio para controlar la amplitud. La señal de salida se mueve entre $+A$ y $-A$.
- **Out:** Salida de la señal de audio para la forma de onda triangular.

Tri Sync



Oscillator

Oscilador para forma de onda triangular simétrica con sincronización, control de tonalidad logarítmico, modulación de frecuencia lineal (FM) y modulación de amplitud lineal.

Cada vez que una señal de sintonización vaya desde 0 hasta valores positivos (borde ascendente) la fase del oscilador se reiniciará a la posición especificada en la entrada de fase.

- **P:** Entrada de evento logarítmica para controlar la tonalidad (frecuencia del oscilador). Valor en semitonos ($69 = 440$ Hz).
- **F:** Entrada de audio para la modulación de frecuencia lineal. Valor en Hz. **P** y **F** juntas determinan la frecuencia del oscilador.
- **A:** Entrada de audio para controlar la amplitud. La señal de salida se mueve entre $+A$ y $-A$.
- **Snc:** Entrada de audio para controlar la sincronización de la forma de onda. Un borde ascendente reinicia el oscilador.
- **Ph:** Entrada para especificar la fase (posición de la forma de onda) a la que el oscilador se reiniciará cuando ocurra la sincronización.
- **Out:** Salida de la señal de audio para la forma de onda triangular.

Tri/Par Symm



Oscillator

Oscilador para señales triangulares y parabólicas (casi sinusoidales) variables, con control de tonalidad logarítmico (P) y modulación de amplitud lineal (A). La simetría se puede ajustar con (W). Al ajustar la simetría (con W) permite una serie de formas de onda desde simétricas con pocos armónicos hasta asimétricas con un gran espectro.

- **P:** Entrada de evento logarítmica para controlar la tonalidad (frecuencia del oscilador). Valor en semitonos ($69 = 440 \text{ Hz}$).
- **A:** Entrada de audio para controlar la amplitud. La señal de salida se mueve entre $+A$ y $-A$.
- **W:** Entrada de evento para controlar la simetría de la forma de onda. Un valor de -1 produce un diente de sierra de borde ascendente; 0 produce un triángulo simétrico; y $+1$ produce un diente de sierra descendente.
- **Tri:** Salida de la señal de audio para la señal triangular.
- **Par:** Salida de la señal de audio para la señal parabólica.

Parabol



Oscillator

Oscilador para formas de onda parabólicas con control logarítmico de tonalidad y modulación de amplitud lineal. La forma de onda consiste en dos mitades que constituyen cada sección de la parábola. El oscilador suena como una onda sinusoidal pero con armónicos añadidos a un nivel muy bajo. En muchos casos se puede usar en lugar del generador sinusoidal, ya que implica menos carga de cálculos.

- **P:** Entrada de evento logarítmico para controlar la tonalidad (frecuencia del oscilador). Valor en semitonos ($69 = 440 \text{ Hz}$).
- **A:** Entrada de audio para controlar la amplitud. La señal de salida se mueve entre $+A$ y $-A$.
- **Out:** Salida de la señal de audio para la forma de onda parabólica.

Par FM

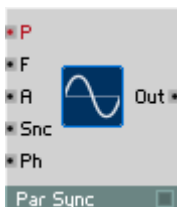


Oscillator

Oscilador para formas de onda parabólicas, con control logarítmico de tonalidad, modulación de frecuencia lineal (FM) y modulación de amplitud lineal. El oscilador suena como una onda sinusoidal con armónicos añadidos a un nivel muy bajo. En muchos casos se puede usar en lugar del generador sinusoidal, ya que implica menos carga de cálculos.

- Entrada de evento logarítmica para controlar la tonalidad (frecuencia del oscilador). Valor en semitonos ($69 = 440$ Hz).
- **F**: Entrada de audio para la modulación de frecuencia lineal. Valor en Hz. **P** y **F** juntas determinan la frecuencia del oscilador.
- Entrada de audio para controlar la amplitud. La señal de salida se mueve entre +A y -A.
- **Out**: Salida de la señal de audio para la forma de onda parabólica.

Par Sync



Oscillator

Oscilador para formas de onda parabólicas con sincronización, control logarítmico de tonalidad, modulación de frecuencia lineal (FM) y modulación de amplitud lineal. El oscilador suena como una onda sinusoidal

Cada vez que la señal de sincronización va desde cero hasta valores positivos (borde ascendente) la fase del oscilador se reinicia a la posición especificada por la entrada de fase.

- **P**: Entrada de evento logarítmica para controlar la tonalidad (frecuencia del oscilador). Valor en semitonos ($69 = 440$ Hz).
- **F**: Entrada de audio para la modulación de frecuencia lineal. Valor en Hz. **P** y **F** juntas determinan la frecuencia del oscilador.

- **A:** Entrada de audio para controlar la amplitud. La señal de salida se mueve entre $+A$ y $-A$.
- **Snc:** Entrada de audio para controlar la sincronización de la forma de onda. Un borde ascendente reinicia el oscilador.
- **Ph:** Entrada para especificar la fase (posición en la forma de onda) a la que el oscilador se reinicia cuando tiene lugar la señal de sincronización.
- **Out:** Salida de la señal de audio para la forma de onda parabólica.

Par PWM



Oscillator

Oscilador para formas de onda parabólicas variables con control logarítmico de tonalidad y modulación de amplitud lineal. Se puede controlar la relación entre la longitud de la parte superior y la inferior de la curva, para conseguir de esta forma cambiar desde una onda parabólica simétrica normal, a una parábola simple.

Esta forma de onda variable es particularmente efectiva cuando se usa como LFO.

- **P:** Entrada de evento logarítmica para controlar la tonalidad (frecuencia del oscilador). Valor en semitonos ($69 = 440$ Hz).
- **A:** Entrada de audio para controlar la amplitud. La señal de salida se mueve entre $+A$ y $-A$.
- **W:** Entrada de audio para controlar la simetría de la forma de onda. Un valor de -1 produce parábolas abiertas hacia abajo; 0 una onda parabólica simétrica normal; y $+1$ parábolas abiertas hacia arriba.
- **Out:** Salida de la señal de audio para la forma de onda parabólica variable.

Sine



Oscillator

Oscilador para formas de onda sinusoidales puras con control logarítmico de tonalidad y modulación de amplitud lineal.

Si la tonalidad de la onda sinusoidal no ha de ser completamente pura, se puede sustituir por un oscilador parabólico ya que implica menos carga de cálculos.

- **P:** Entrada de evento logarítmica para controlar la tonalidad (frecuencia del oscilador). Valor en semitonos ($69 = 440$ Hz).
- **A:** Entrada de audio para controlar la amplitud. La señal de salida se mueve entre $+A$ y $-A$.
- **Out:** Salida de la señal de audio para la forma de onda sinusoidal.

Sine FM

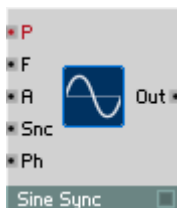


Oscillator

Oscilador para formas de onda sinusoidales puras con control logarítmico de tonalidad, modulación de frecuencia lineal (FM) y modulación de amplitud lineal.

Si la onda sinusoidal no ha de ser completamente pura, se puede usar el oscilador parabólico, ya que implica menos carga de cálculos.

- **P:** Entrada de evento logarítmica para controlar la tonalidad (frecuencia del oscilador). Valor en semitonos ($69 = 440$ Hz).
- **F:** Entrada de audio para la modulación de frecuencia lineal. Valor en Hz. **P** y **F** juntas determinan la frecuencia del oscilador.
- **A:** Entrada de audio para controlar la amplitud. La señal de salida se mueve entre $+A$ y $-A$.
- **Out:** Salida de la señal de audio para la forma de onda sinusoidal.



Oscilador para formas de onda sinusoidales puras con sincronización, control logarítmico de tonalidad, modulación de frecuencia lineal (FM) y modulación de amplitud lineal.

Cada vez que la señal de sincronización va desde cero hasta valores positivos (borde ascendente), la fase del oscilador se reinicia a la posición especificada por la entrada de fase.

Si la onda sinusoidal no ha de ser completamente pura, se puede usar el oscilador parabólico, ya que implica menos carga de cálculos.

- **P:** Entrada de evento logarítmica para controlar la tonalidad (frecuencia del oscilador). Valor en semitonos ($69 = 440$ Hz).
- **F:** Entrada de audio para la modulación de frecuencia lineal. Valor en Hz. **P** y **F** juntas determinan la frecuencia del oscilador.
- **A:** Entrada de audio para controlar la amplitud. La señal de salida se mueve entre $+A$ y $-A$.
- **Snc:** Entrada de audio para controlar la sincronización de la forma de onda. Un borde ascendente reinicia el oscilador.
- **Ph:** Entrada para especificar la fase (posición en la forma de onda) a la que el oscilador se va a reiniciar cuando ocurra la sincronización. **Ph** = 0: Fase = 0 grados (mitad de la pendiente ascendente), **Ph** = 0.5: fase = 180 grados (mitad de la pendiente descendente), **Ph** = 1: Phase = 360 grados (igual que 0 grados).
- **Out:** Salida de la señal de audio para la forma de onda sinusoidal.



Oscilador para Síntesis Aditiva. Una forma de onda se construye sobre sus armónicos superponiendo ondas sinusoidales individuales. Para cada componente, la amplitud **A** y el múltiplo de frecuencia **F** se pueden ajustar.

- **P:** Entrada de evento logarítmica para controlar la tonalidad (frecuencia del oscilador). Valor en semitonos (69 = 440 Hz).
- **F1:** Entrada para el factor de frecuencia (número armónico) de la primera forma de onda sinusoidal. Escala: múltiplos de la frecuencia fundamental. Rango típico: [0 ... 20].
- **A1:** Entrada para controlar la amplitud lineal de la primera onda sinusoidal. También se puede usar para tremolo y modulación anular. Rango típico: [0 ... 1].
- **F2:** Entrada para el factor de frecuencia (número armónico) de la segunda onda sinusoidal. Escala: múltiplos de la frecuencia fundamental. Rango típico: [0 ... 20].
- **A2:** Entrada para controlar la amplitud lineal de la segunda onda sinusoidal. También se puede usar para tremolo y modulación anular. Rango típico: [0 ... 1].
- **F3:** Entrada para el factor de frecuencia (número armónico) de la tercera onda sinusoidal. Escala: múltiplos de la frecuencia fundamental. Rango típico: [0 ... 20].
- **A3:** Entrada para controlar la amplitud lineal de la tercera onda sinusoidal. También se puede usar para tremolo y modulación anular. Rango típico: [0 ... 1].
- **F4:** Entrada para el factor de frecuencia (número armónico) de la cuarta onda sinusoidal. Escala: múltiplos de la frecuencia fundamental. Rango típico: [0 ... 20].

- **A4:** Entrada para controlar la amplitud lineal de la cuarta onda sinusoidal. También se puede usar para tremolo y modulación anular. Rango típico: [0 ... 1].
- **S1:** Salida para el primer componente de la onda sinusoidal.
- **S2:** Salida para el segundo componente de la onda sinusoidal.
- **S3:** Salida para el tercer componente de la onda sinusoidal.
- **S4:** Salida para el cuarto componente de la onda sinusoidal.
- **Out:** Salida para la señal generada por el oscilador añadiendo todas las ondas sinusoidales.

Pulse



Oscillator

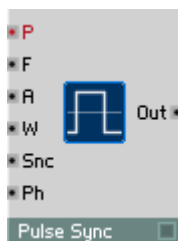
Oscilador para formas de onda rectangulares con control logarítmico de tonalidad, modulación de ancho de pulso (PWM) y modulación de amplitud lineal.

- **P:** Entrada de evento logarítmica para controlar la tonalidad (frecuencia del oscilador). Valor en semitonos (69 = 440 Hz).
- **A:** Entrada de audio para controlar la amplitud. La señal de salida se mueve entre +A y -A.
- **W:** Entrada de audio para controlar el ancho de pulso (PWM). El Rango de valores va desde -1 a 1. Forma de onda simétrica (onda cuadrada) 50:50 con **W** = 0, relación de pulsos 33:66 con -0.33, 66:33 con 0.33, 75:25 con 0.5, 90:10 con 0.8. Lo:Hi = $(1 + \mathbf{W}) / (1 - \mathbf{W})$.
- **Out:** Salida para la señal de audio con la forma de onda rectangular.



Oscilador para formas de onda rectangulares con control logarítmico de tonalidad, modulación de frecuencia lineal (FM), modulación de ancho de pulso (PWM) y modulación de amplitud lineal.

- **P**: Entrada de evento logarítmica para controlar la tonalidad (frecuencia del oscilador). Valor en semitonos (69 = 440 Hz).
- **F**: Entrada de audio para la modulación de frecuencia lineal. Valor en Hz. **P** y **F** juntas determinan la frecuencia del oscilador.
- **A**: Entrada de audio para controlar la amplitud. La señal de salida se mueve entre +A y -A.
- **W**: Entrada de audio para controlar el ancho de pulso (PWM). El Rango de valores va desde -1 a 1. Forma de onda simétrica (onda cuadrada) 50:50 con **W** = 0, relación de pulsos 33:66 con -0.33, 66:33 con 0.33, 75:25 con 0.5, 90:10 con 0.8. Lo:Hi = $(1 + \mathbf{W}) / (1 - \mathbf{W})$.
- **Out**: Salida de la señal de audio para la forma de onda rectangular.

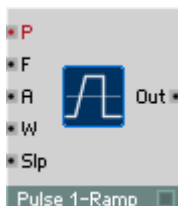


Oscilador para formas de onda rectangulares con sincronización, control logarítmico de tonalidad, modulación de frecuencia lineal (FM), modulación de ancho de pulso (PWM) y modulación de amplitud lineal.

Cada vez que la señal de sincronización vaya desde cero hasta valores positivos (borde ascendente) la fase del oscilador se reinicia a la posición especificada en la entrada de fase.

- **P:** Entrada de evento logarítmica para controlar la tonalidad (frecuencia del oscilador). Valor en semitonos (69 = 440 Hz).
- **F:** Entrada de audio para la modulación de frecuencia lineal. Valor en Hz. **P** y **F** juntas determinan la frecuencia del oscilador.
- **A:** Entrada de audio para controlar la amplitud. La señal de salida se mueve entre +A y -A.
- **W:** Entrada de audio para controlar el ancho de pulso (PWM). El Rango de valores va desde -1 a 1. Forma de onda simétrica (onda cuadrada) 50:50 con **W** = 0, relación de pulsos 33:66 con -0.33, 66:33 con 0.33, 75:25 con 0.5, 90:10 con 0.8. Lo:Hi = $(1 + \mathbf{W}) / (1 - \mathbf{W})$.
- **Snc:** Entrada de audio para controlar la sincronización de la forma de onda. Un borde ascendente reinicia el oscilador.
- **Ph:** Entrada para especificar la fase (posición en la forma de onda) a la que el oscilador se reinicia cuando ocurre la sincronización.
- **Out:** Salida de la señal de audio para la forma de onda rectangular.

Pulse 1-ramp

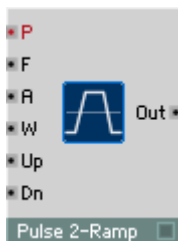


Oscillator

Oscilador para formas de onda trapezoidales rectangulares con control logarítmico de tonalidad, modulación de frecuencia lineal (FM), modulación de ancho de pulso (PWM) y modulación de amplitud lineal. La forma de onda es una mezcla de rectángulo y diente de sierra. El borde descendente es vertical.

- **P:** Entrada de evento logarítmica para controlar la tonalidad (frecuencia del oscilador). Valor en semitonos (69 = 440 Hz).
- **F:** Entrada de audio para la modulación de frecuencia lineal. Valor en Hz. **P** y **F** juntas determinan la frecuencia del oscilador.
- **A:** Entrada de audio para controlar la amplitud. La señal de salida se mueve entre +A y -A.
- **W:** Entrada de audio para controlar el ancho de pulso (PWM). El Rango de valores va desde -1 a 1.
- **Slp:** Entrada de audio para controlar la pendiente del borde ascendente de la forma de onda. Cuando **Slp** es cero (o cuando la entrada no está conectada) la forma de onda no se eleva y la salida es siempre cero; es decir, no hay sonido. Rango típico de valores: 1 . . . 20.
- **Out:** Salida de la señal de audio para la forma de onda trapezoidal.

Pulse 2-ramp



Oscillator

Oscilador para formas de onda trapezoidales con control logarítmico de tonalidad, modulación de frecuencia lineal (FM), modulación de ancho de pulso (PWM) y modulación de amplitud lineal. La forma de onda es una mezcla de rectángulo, diente de sierra y triángulo: Los dos bordes de una onda rectangular se pueden achatar, y la rampa se puede ajustar.

- **P:** Entrada de evento logarítmica para controlar la tonalidad (frecuencia del oscilador). Valor en semitonos ($69 = 440 \text{ Hz}$).
- **F:** Entrada de audio para la modulación de frecuencia lineal. Valor en Hz. **P** y **F** juntas determinan la frecuencia del oscilador.
- **A:** Entrada de audio para controlar la amplitud. La señal de salida se mueve entre $+A$ y $-A$.
- **W:** Entrada de audio para controlar el ancho de pulso (PWM). El rango de valores va desde -1 a 1 .
- **Up:** Entrada de audio para controlar la pendiente ascendente de la forma de onda.
- **Dn:** Entrada de audio para controlar la pendiente descendente de la forma de onda.
- **Out:** Salida de la señal de audio para la forma de onda trapezoidal.

Bi-Pulse



Oscillator

Oscilador para formas de onda rectangulares bipolares con fase cero. Con control logarítmico de tonalidad, modulación de ancho de pulso (PWM) y modulación de amplitud lineal.

- **P:** Entrada de evento logarítmica para controlar la tonalidad (frecuencia del oscilador). Valor en semitonos ($69 = 440 \text{ Hz}$).
- **A:** Entrada de audio para controlar la amplitud. La señal de salida se mueve entre $+A$ y $-A$.
- **W:** Entrada de audio para controlar el ancho de pulso (PWM). El rango de valores va desde -1 a 1 . Forma de onda simétrica (onda cuadrada) 50:50 con **W** = 0, con valores mayores de **W** se consiguen pulsos más cortos y fases cero más largas.
- **Out:** Salida de la señal de audio para la forma de onda rectangular bipolar.

Impulse



Oscillator

Oscilador para formas de onda de pulso con entrada logarítmica de control de tonalidad (P) y modulación de amplitud lineal (A).

- **P:** Entrada de evento logarítmica para controlar la tonalidad (frecuencia del oscilador). Valor en semitonos (69 = 440 Hz).
- **A:** Entrada de audio para controlar la amplitud.
- **Out:** Salida de la señal de audio para la forma de onda de pulso.

Impulse FM

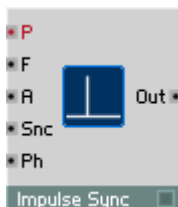


Oscillator

Oscilador para forma de onda de pulso con entrada logarítmica de control de tonalidad (P), modulación de frecuencia lineal (F) y modulación de amplitud lineal (A).

- **P:** Entrada de evento logarítmica para controlar la tonalidad (frecuencia del oscilador). Valor en semitonos (69 = 440 Hz).
- **F:** Entrada de audio para la modulación de frecuencia lineal. Valor en Hz. **P** y **F** juntas determinan la frecuencia del oscilador.
- **A:** Entrada de audio para controlar la amplitud.
- **Out:** Salida de la señal de audio para la forma de onda de pulso.

Impulse Sync



Oscillator

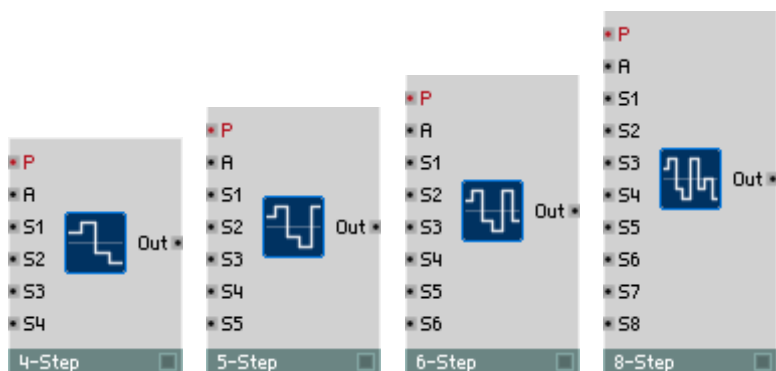
Oscilador para formas de onda de pulso sincronizable con entrada logarítmica de control de tonalidad (P), modulación de frecuencia lineal (F) modulación de amplitud lineal (A) y entrada Sync (Snc).

Cada vez que la señal de sincronización vaya desde cero hasta valores positivos (borde ascendente) la fase del oscilador se reinicia a la posición especificada en la entrada de fase.

- **P:** Entrada de evento logarítmica para controlar la tonalidad (frecuencia del oscilador). Valor en semitonos ($69 = 440$ Hz).
- **F:** Entrada de audio para la modulación de frecuencia lineal. Valor en Hz. **P** y **F** juntas determinan la frecuencia del oscilador.
- **A:** Entrada de audio para controlar la amplitud.
- **Snc:** Entrada de audio para la sincronización de la forma de onda. Un borde ascendente reinicia el oscilador.
- **Ph:** Entrada para especificar la fase (posición en la forma de onda) a la que el oscilador se reinicia cuando ocurre la sincronización.
- **Out:** Salida de la señal de audio para la forma de onda de pulso.

Multi-Step

Oscillator



4-Step

Oscillator

Oscilador para formas de onda de 4 pasos con control logarítmico de tonalidad y modulación lineal de amplitud. Cada nivel de paso es independiente de los otros.

- **P:** Entrada de evento logarítmica para controlar la tonalidad (frecuencia del oscilador). Valor en semitonos ($69 = 440$ Hz).
- **A:** Entrada de audio para controlar la amplitud. La señal de salida se mueve entre **+A** y **-A**.

- **S1:** Entrada de audio para controlar el nivel del primer paso.
- **S2:** Entrada de audio para controlar el nivel del segundo paso.
- **S3:** Entrada de audio para controlar el nivel del tercer paso.
- **S4:** Entrada de audio para controlar el nivel del cuarto paso.
- **Out:** Salida de la señal de la forma de onda de 4 pasos.

5-Step

Oscillator

Como el de 4 pasos, pero de 5.

6-Step

Oscillator

Como el de 4 pasos, pero de 6.

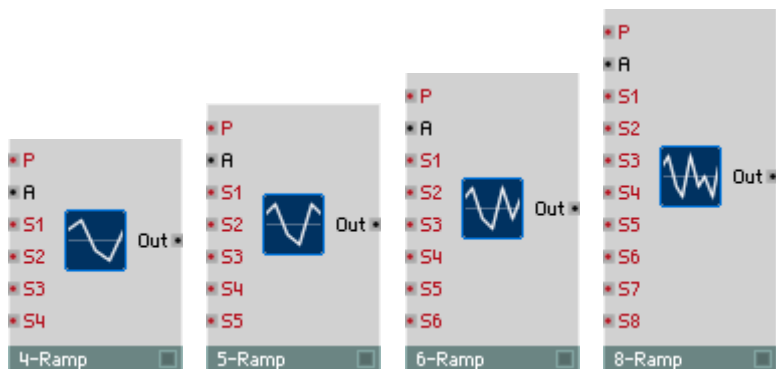
8-Step

Oscillator

Como el de 4 pasos, pero de 8.

Multi-Ramp

Oscillator



4-Ramp

Oscillator

Oscilador para formas de onda de 4 pendientes con control logarítmico de tonalidad y modulación lineal de amplitud. El nivel de cada uno de los puntos unidos por las pendientes se puede ajustar de forma independiente.

- **P:** Entrada de evento logarítmica para controlar la tonalidad (frecuencia del oscilador). Valor en semitonos ($69 = 440 \text{ Hz}$).
- **A:** Entrada de audio para controlar la amplitud. La señal de salida se mueve entre $+A$ y $-A$.
- **S1:** Entrada de evento para controlar el nivel del primer punto.
- **S2:** Entrada de evento para controlar el nivel del segundo punto.
- **S3:** Entrada de evento para controlar el nivel del tercer punto.
- **S4:** Entrada de evento para controlar el nivel del cuarto punto.
- **Out:** Salida para la señal de audio de la forma de onda de pendiente.

5-Ramp

Oscillator

Como el de 4 pendientes, pero con 5.

6-Ramp

Oscillator

Como el de 4 pendientes, pero con 6.

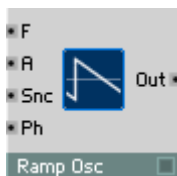
8-Ramp

Oscillator

Como el de 4 pendientes, pero con 8.

Ramp

Oscillator

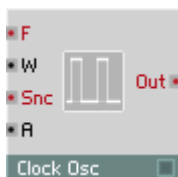


Oscilador para producir una forma de onda de pendiente normalmente usada como señal de control, por ejemplo, para usar un módulo Audio Table como oscilador de forma de onda. La señal asciende desde 0 a A y luego se reinicia instantáneamente.

- **F:** Entrada de audio para controlar la frecuencia del oscilador en Hz. Para controlar la tonalidad en semitonos, usa un evento Event Expon (F).
- **A:** Entrada para controlar la amplitud de la señal. Valor típico: 1.

- **Snc:** Entrada de audio para controlar la sincronización de la forma de onda. Un borde ascendente reinicia el oscilador a **Ph**.
- **Ph:** Entrada de audio para la sincronización de la fase. Cuando el oscilador se ha sincronizado, su salida vuelve a este valor multiplicado por A. Rango típico: 0...1.
- **Out:** Salida de audio para la señal de la pendiente. Rango: 0...A.

Clock Oscillator

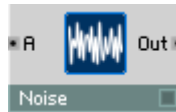


Oscillator

Fuente libre de señal de reloj. Un oscilador interno (monofónico) produce pulsos regulares de reloj On/Off en forma de eventos, por ejemplo, para dirigir un secuenciador. El valor de los eventos en On se puede ajustar en las propiedades del módulo.

- **F:** Entrada de evento para controlar la frecuencia del reloj en Hz. Para controlar el Tempo en BPM, calcula el valor en Hz así: *relojes-por-beat* x BPM/60. Así, para relojes en semicorcheas de compases en 4/4 (es decir, cuatro semicorcheas por beat), tendrás $F = 4/60 \times \text{BPM}$ o $0.0667 \times \text{BPM}$.
- **W:** Entrada de evento para controlar el ancho de pulso, es decir, la relación de duración de la fase On a la fase Off. Rango de valores: -1 a 1. Forma de onda simétrica (misma duración en On y Off) con $W = 0$; Lo:Hi = $(1 + W) / (1 - W)$.
- **Snc:** Entrada de evento para la sincronización de la forma de onda. Un evento positivo sincroniza la fuente de reloj.
- **A:** Entrada para controlar la amplitud de la señal de reloj. Tendrás que conectar algo, si no, sólo se producirán eventos de valor cero. Valor típico: 1.
- **Out:** Salida de señal de evento para la señal de reloj que alterna entre valores On y cero.

Noise



Oscillator

Generador de ruido. Produce ruido blanco, es decir, una señal randomizada que contiene las mismas cantidades de todas las frecuencias. La señal sólo consta de dos valores, $A/2$ y $-A/2$, que aparecen de forma randomizada.

- **A:** Entrada de audio para controlar la amplitud de la señal de salida.
- **Out:** Salida de señal de audio para la forma de onda de ruido.

Random



Oscillator

Generador aleatorio de valores. Produce formas de onda de pasos donde el nivel de cada paso se randomiza en el intervalo interno dado con igual distribución. Funciona como un generador de ruido con distribución uniforme seguida de un circuito sample&hold sujeto a una frecuencia regular.

- **P:** Entrada de evento logarítmico para controlar la tasa de pasos (frecuencia sample&hold). Valores en semitonos ($69 = 440$ Hz).
- **A:** Entrada de audio para controlar la amplitud, La señal de salida es un valor randomizado entre $+A$ and $-A$.
- **Out:** Salida de audio para la señal aleatoria.

Geiger



Oscillator

Genera eventos en intervalos aleatorios, igual que un detector de partículas de radiación Geiger Counter. El promedio de tasa de eventos se puede controlar con la entrada **P**. **Rnd** controla la randomización de los eventos.

- **P:** Entrada de control para controlar logarítmicamente el promedio de tasa o densidad de los eventos de salida randomizados. Rango típico: [-50 ... 50]

- **Rnd:** Entrada de control para la distribución randomizada de eventos: 0 = completamente randomizado; 1 = completamente regular. Rango típico: [0 ... 1]
- **Out:** Salida de audio para los clics de tiempo randomizados.
- **Out:** Salida para los eventos de tiempo randomizados. Para la activación, usa por ejemplo una envolvente (**G**).

Samplers

Si algo genera audio, y no es un oscilador, entonces se trata de un sampler. Los samplers de Reaktor incluyen reproductores de samples básicos así como sofisticados procesadores de samples para síntesis granular, Pitch y Time Shifting, y división de beats. Incluso hay uno que sirve para seleccionar samples individuales a través de un número.

Dependiendo del módulo sampler seleccionado, encontrarás distintos ajustes en las propiedades:

Properties - Página Function

- **Waveform-Button:** Si haces clic en el botón que lleva el icono de la forma de onda, se abrirá el Editor de Mapas de Samples.
- **Embed Samples In Ensemble** permite almacenar tus samples con el Ensemble guardado.
- **No Stereo** detiene la reproducción estéreo (sólo se usa el canal izquierdo). Su activación reduce la carga de procesamiento.
- El menú desplegable **Quality** ajusta la calidad de reproducción del sample en tres opciones (**pobre, buena y excelente**). La calidad mayor incrementa la carga de procesamiento. El término “calidad” se refiere a la ausencia de ruido. Naturalmente, el ruido a veces puede mejorar la “calidad” musical de un sample.
- **Oscil. Mode** pone el módulo en el modo oscilador.

Los samplers en modo oscilador dan por sentado que los samples usados contienen formas de onda o WaveSets. Una forma de onda es la representación de una vibración simple, A través de la repetición de la reproducción se puede recrear una vibración periódica. Es entonces cuando el módulo se

convierte en un oscilador digital. Los módulos samplers en modo de oscilador interpretan los valores de la entrada **P** igual que los osciladores en Reaktor – como números de nota MIDI – y producen vibraciones periódicas en la tonalidad correspondiente.

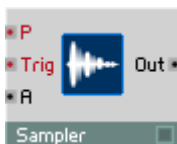
En el modo de oscilador, los módulos **Sampler** y **Sampler FM** interpretan el sample completo como una vibración. Por ejemplo, para producir un sonido a 440Hz, el sample completo se toca 440 veces por segundo, independientemente de la duración del sample.

En el modo de forma de onda, el **Sampler Loop** interpreta la duración del loop como vibraciones periódicas. La duración del loop se lee desde el archivo de sonido, pero se puede cambiar a través de la entrada **LL** del módulo (mira en Referencia de Módulos). Por lo tanto, un sample puede constar de numerosas formas de onda, también conocido como WaveSet. Dando por supuesto que las formas de onda contienen 100 ciclos, entonces, 100 formas de onda encajarían en el sample la cifra de 10,000 ciclos.

La primera forma de onda cubre los ciclos 0-99, la segunda 100-199, etc. Si la duración del loop determina la vibración de una forma de onda, la posición del loop establece digitalmente la selección de la forma de onda del WaveSet. El **Sampler Loop** usa la entrada **LS** para controlar el punto de inicio del loop. En el modo de forma de onda, los valores de esta entrada están cuantizados, y así conseguimos una reproducción entre las formas de onda sin ruido. La posición en un WaveSet se puede controlar fácilmente a través de la dinámica de pulsación, etc. Obviamente, estos samples WaveSet se tienen que crear o generar desde un sample. La librería contiene muchos ejemplos de WaveSets. **Sampler Loop** integra Síntesis WaveSet a través de las capacidades de síntesis existentes en Reaktor. De esta forma, por primera vez la Síntesis WaveSet se puede combinar con la síntesis FM.

Properties - Página Appearance

- **Picture:** Activa esta opción para visualizar un display de forma de onda en el panel para el módulo sampler.
- **Size X/Size Y:** Permite ajustar el tamaño del display de la forma de onda para el módulo sampler en pixels.
- **Scroll Bar:** Activa esta opción para visualizar una barra de desplazamiento bajo el display de la forma de onda y poder así navegar. La barra de desplazamiento te permite moverte (arrastra la barra desde el centro) y hacer zoom (arrastra la barra en una de sus esquinas hacia la izquierda o hacia la derecha).



Es un reproductor que se usa para la reproducción polifónica y transportada de un sample o mapa de samples.

La administración de los samples se lleva a cabo en el **Editor de Mapas de Samples**.

Si **Loop** está activado, el sample completo se repite continuamente y se reinicia a través de un evento positivo en la entrada Trigger. Si **Loop** está desactivado, y llega un evento positivo a la entrada Trigger, el sample funcionará desde el principio hasta el fin o, si el parámetro de dirección está ajustado a **Backward**, funcionará desde el final hasta el principio.

Si **Oscillator Mode** está activado, la tasa de reproducción se ajustará para adaptar la duración del sample actual y producir así el tono correcto cuando opere como un oscilador de forma de onda.

- **P**: Entrada de control logarítmica para la tasa de reproducción (tonalidad) y para seleccionar un sample desde el mapa de samples cargado. Si **P** = **Root Key** del sample actual, el sample se reproducirá en su tonalidad original.
- **Trig**: Un evento positivo en esta entrada activará la nueva reproducción del sample otra vez desde el principio.
- **A**: Entrada de control para la amplitud de salida.
- **Out**: Salida del reproductor de samples.

Sampler FMSampler



Es un reproductor que se usa para la reproducción polifónica y transportada de un sample o mapa de samples. La entrada **F** y la entrada de control de punto de inicio (**St**) te permiten manipular la reproducción del sample.

La administración de los samples se lleva a cabo en el **Editor de Mapas de Samples**.

Si **Loop** está activado, el sample completo se repite continuamente y se reinicia a través de un evento positivo en la entrada Trigger, que se ha ajustado en la entrada **St**. Si **Loop** está desactivado y hay un evento positivo en la entrada Trigger, el sample funcionará desde el principio hasta el fin o, si el parámetro de dirección está ajustado a **Backward**, funcionará desde el final hasta el principio.

Si **Oscillator Mode** está activado, la tasa de reproducción se ajustará para adaptar la duración del sample actual y producir así el tono correcto cuando opere como un oscilador de forma de onda.

- **P**: Entrada de control logarítmica para la tasa de reproducción (tonalidad) y para seleccionar un sample desde el mapa de samples cargado. Si **P** = **Root Key** del sample actual, el sample se reproducirá en su tonalidad original.
- **F**: Entrada de control lineal para la modulación de la tasa de reproducción. El efecto que se consigue es la modulación de la frecuencia – lo mismo que para los módulos osciladores en REAKTOR. Con valores negativos grandes, el sample se reproduce hacia atrás.
- **St**: Entrada de control para que el punto de inicio se use cuando llegue el siguiente activador. La posición se ajusta en milisegundos desde el inicio del sample.
- **Trig**: Un evento positivo en esta entrada activa el sample para que se reproduzca otra vez desde el principio.
- **A**: Entrada de control para la amplitud de salida.
- **Out**: Salida del reproductor de samples.
- **Lng**: Salida de evento polifónica para la duración del sample actual en milisegundos.



Es un reproductor universal para la reproducción polifónica y transportada de samples estéreo y mono, mapas de samples y WaveSets.

La administración de samples se lleva a cabo con el **Editor de Mapas de Samples**.

Después de un evento **Gate** positivo, la reproducción del sample empieza desde el punto de inicio (configurable en la entrada **St**). Si **Loop** se desactiva, el sample funcionará una vez desde el punto de inicio hacia el final o, si está activada la opción **Backward**, funcionará desde el punto de inicio hasta el principio. Si **Loop** está activado, el sample se repetirá continuamente dentro del rango del loop en cuanto la reproducción introduzca este rango. El rango del loop se programa usando datos desde el archivo de sonido desde el cual se cargó el sample. Si el archivo de sonido no contiene ningún dato, se cogerá el principio del sample como principio del loop, y la duración del sample como duración del loop. Los datos del loop que se extraen del archivo de sonido son ajustes por defecto. Estos ajustes siempre se usan si las entradas Loop Start (**LS**) o Loop Length (**LL**) no están conectadas a otros módulos.

Si la opción **Loop in Release** está activada, a la reproducción del loop se le aplicará un fundido o fundido de salida con un evento **Gate**. Dependiendo de la dirección que se ajuste, el sample funcionará hacia el principio o hacia el final y luego se le aplicará fundido de salida. Si la opción **Loop in Release** se activa o si la reproducción de loop está desactivada, los eventos **Gate** tendrán poco efecto o ninguno.

Si la opción **No Stereo** está activada (puedes ajustarlo en las propiedades), los samples estéreo se tratarán como samples mono, es decir, se envía la misma señal a la salida **L** y a la **R**. Si este es el caso, **Sampler Loop** sólo usará el canal izquierdo de los samples estéreo. Esta opción está disponible para disponer de menos carga de procesamiento.

- **G:** Un evento positivo en esta entrada activa la salida desde la posición ajustada en la entrada **St**. Si hay valores negativos en la entrada, la

reproducción del loop se interrumpirá a menos que esté activada la opción **loop in Release**.

- **P**: Entrada logarítmica de control para la tasa de reproducción (tonalidad). Si **P** = **Root Key** del sample actual, el sample se reproducirá en su tono original. Además, si la entrada **Sel** no está conectada, **P** determina la selección de samples desde el mapa de samples. En **Oscillator Mode**, el **Sampler Loop** funciona como oscilador digital. Del mismo modo que los módulos osciladores de REAKTOR, **P** determina la tonalidad fundamental de la oscilación generada.
- **Sel**: Entrada de control para seleccionar un sample desde el mapa de samples. Si esta entrada no está conectada, los valores presentes en la entrada **P** se usan en su lugar.
- **F**: Entrada de control lineal para modular la tasa de reproducción. El efecto que se consigue es la modulación de frecuencia – igual que los módulos osciladores de REAKTOR.
- **St**: Entrada de control para el punto de inicio que se va a usar cuando ocurra el siguiente evento. La posición se ajusta en milisegundos desde el principio del sample.
- **LS**: Entrada de control para el punto de inicio en milisegundos desde el principio del sample. Si esta entrada no está conectada, se usarán los datos del loop guardados en el archivo de sonido. Si no hay almacenado ningún dato de loop en el archivo de sonido, se usará por defecto **LS** = 0. Los cambios en esta entrada tienen efecto cuando se activan los loops y cuando se alcanza el límite de un loop.
- **LL**: Entrada de control para la duración del loop en milisegundos. Si esta entrada está desconectada, se usarán los datos de loop almacenados en el archivo de sonido. Si no hay datos del loop almacenados en el archivo, se usará por defecto **LL** = duración del sample. Los cambios en esta entrada tienen efecto cuando se activan los samples y cuando se alcanza el límite de un loop.
- **A**: Entrada de control para la amplitud de salida.
- **L**: Salida de audio para el canal izquierdo del reproductor de samples. Cuando se han procesado samples mono o cuando la opción **No Stereo** se ha activado, la señal será la misma que la de la salida **R**.
- **R**: Salida de audio para el canal derecho del reproductor de samples. Cuando se han procesado señales mono o cuando la opción **No Stereo** se ha conectado, la señal será la misma que la de la salida **L**.

- **Lng:** Salida polifónica de evento para la duración del sample actual en milisegundos.

Grain Resynth

Sampler



Es un sintetizador en tiempo real para la reproducción polifónica, transportada de samples estéreo o mono y mapas de samples. **Sample Resynth** te permite controlar la tonalidad y la velocidad de la reproducción independientemente y en tiempo real, y también te deja manipular los samples extensamente.

Los samples “estándar” como los familiares samplers hardware, o los módulos **Sampler**, **Sampler FM** y **Sampler Loop** de REAKTOR, mantienen un indicador para cada voz. Este indicador muestra la posición actual del sample. La amplitud de salida del sampler siempre es la misma que la amplitud del sample en la posición del indicador. El indicador se mueve con más o menos rapidez a través del sample y por lo tanto, genera una amplitud en la salida que varía con el tiempo – es decir, una oscilación.

La rapidez de movimiento del indicador determina la velocidad de reproducción del sample (es decir, la velocidad de un beat loop). Al mismo tiempo, también determina la tonalidad: cuanto más lento sea el sampleado de la señal (que se ha grabado en el sample), más largos serán los períodos de los osciladores en la salida – y por lo tanto el tono decrece. Si el indicador deja de moverse, la amplitud de salida deja de cambiar y no se producirá ninguna señal audible.

Al igual que en un sampler convencional, **Sample Resynth** usa un indicador en la posición actual del sample. No obstante, la señal de la salida no es simplemente la amplitud del sample en la posición del indicador. Lo que ocurre

en realidad, es que la señal de salida se genera por un sintetizador dentro del módulo. El sintetizador *resintetiza* la señal en la posición del indicador. La tonalidad de la señal generada por este sintetizador es independiente de la velocidad del indicador. Incluso si el indicador no se mueve en absoluto, el sintetizador continúa para producir un sonido. La tonalidad de salida se determina, como es usual, sobre la entrada **P**, pero la entrada **Sp** determina la velocidad del indicador.

Por supuesto, la señal ralentizada o “congelada” no siempre corresponde a lo que habías imaginado. ¿Cómo suena un martillo sobre un clavo si lo ralentizamos hasta el infinito? El algoritmo de resíntesis que usa **Sample Resynth** está diseñado de forma que se pueden procesar una gran cantidad de sonidos de manera sensible (musicalmente hablando), sutil o dramática. El algoritmo se puede ajustar para configurar los parámetros **G** (granularidad) y **Sm** (suavidad). Esto significa que puedes ajustarlo manualmente para conseguir efectos drásticos. En el diálogo de propiedades puedes activar la opción **Signal-Informed Granulation** de forma separada para cada sample en un mapa de samples. Si esta opción está encendida, el algoritmo de resíntesis en **Resynth** coge la información del sample y la tiene en cuenta. Esta información se llevó a cabo al cargar el sample por primera vez. Esto significa que el algoritmo reacciona a las características del material de audio. Si esta opción está desactivada, los resultados son, generalmente, bastante “electrónicos”.

La administración de samples se lleva a cabo con el **Editor de Mapas de Samples**.

Después de un evento **Gate** positivo, la reproducción del sample empieza desde el punto de inicio (configurable en la entrada **St**). Si **Loop** se desactiva, el sample funcionará una vez desde el punto de inicio hacia el final o, si está activada la opción **Backward**, funcionará desde el punto de inicio hasta el principio. Si **Loop** está activado, el sample se repetirá continuamente dentro del Rango del loop en cuanto la reproducción introduzca este Rango. El Rango del loop se pre-programa usando datos desde el archivo de sonido desde el cual se cargó el sample. Si el archivo de sonido no contiene ningún dato, se cogerá el principio del sample como principio del loop, y la duración del sample como duración del loop. Los datos del loop que se extraen del archivo de sonido son ajustes por defecto. Estos ajustes siempre se usan si las entradas Loop Start (**LS**) o Loop Length (**LL**) no están conectadas a otros módulos.

Si la opción **Loop in Release** está activada, a la reproducción del loop se le aplicará un fundido o fundido de salida con un evento **Gate**. Dependiendo de la dirección que se ajuste, el sample funcionará hacia el principio o hacia el

final y luego se le aplicará un fundido de salida. Si la opción **Loop in Release** se activa o si la reproducción de loop está desactivada, los eventos **Gate** tendrán poco efecto o ninguno.

Si la opción **No Stereo** está activada (puedes ajustarlo en las propiedades), los samples estéreo se tratarán como samples mono, es decir, se envía la misma señal a la salida **L** y a la **R**. Si éste es el caso, **Resynth** sólo usará el canal izquierdo de los samples estéreo. Esta opción está disponible para poder disponer de menos carga de procesamiento.

Todas las entradas de **Resynth**, excepto **Gate**, están diseñadas como entradas de audio. Los valores de estas entradas se aplican cuando los samples se activan (es decir, durante los eventos **Gate** positivos). Los cambios efectuados durante la reproducción del sample tendrán efecto sobre el intervalo de **Granularidad** (configurado en la entrada **Gr**).

- **G**: Un evento positivo en esta entrada activa la salida desde la posición ajustada en la entrada **St**. Si hay valores negativos en la entrada, la reproducción del loop se interrumpirá a menos que esté activada la opción **loop in Release**.
- **P**: Entrada de audio logarítmica de control para la tasa de reproducción (tonalidad). Si **P** = **Root Key** del sample actual, el sample se reproducirá en su tono original. Además, si la entrada **Sel** no está conectada, **P** también determina la selección de samples desde el mapa de samples.
- **Sel**: Entrada de audio de control para seleccionar un sample desde el mapa de samples. Si esta entrada no está conectada, los valores presentes en la entrada **P** se usan en su lugar.
- **St**: Entrada de audio de control para el punto de inicio que se va a usar cuando ocurra el siguiente evento. La posición se ajusta en milisegundos desde el principio del sample.
- **LS**: Entrada de audio de control para el punto de inicio en milisegundos desde el principio del sample. Si esta entrada no está conectada, se usarán los datos del loop guardados en el archivo de sonido. Si no hay almacenado ningún dato de loop en el archivo de sonido, se usará por defecto **LS** = 0.
- **LL**: Entrada de audio de control para la duración del loop en milisegundos. Si esta entrada está desconectada, se usarán los datos de loop almacenados en el archivo de sonido. Si no hay datos del loop almacenados en el archivo, se usará por defecto **LL** = duración del

sample. Si **LL** = 0, el movimiento dentro del sample se parará cuando se Rango el punto de inicio del loop; el sonido se congelará en este punto.

- **Sp**: Entrada de audio de control para la velocidad de salida de tonalidad independiente. Los valores presentes en esta entrada se interpretan como factor: Cuando **Sp** = 1, la reproducción tiene lugar en la velocidad original; **Sp** = 2, corresponde al doble de velocidad; y **Sp** = 0, significa stop. Si esta entrada no está conectada, la velocidad original transportada se toma por defecto – como en los samplers convencionales – y así, la velocidad se reduce según decrece la tonalidad.
- **Gr**: Entrada de audio de control para la granularidad del proceso de resíntesis en milisegundos. Este parámetro determina el tamaño de las partículas de sonido usadas en la resíntesis. Si la opción **Signal-Informed Granulation** está activa, los valores en la entrada se usarán por defecto; los valores que en la actualidad se están usando, se adaptarán al material.
- **SO**: Entrada de audio de control para la modulación de la posición del sample (Sample Offset) en milisegundos. Esta entrada se usa para modular la posición del sample independientemente del tono, es decir, a través de un generador de ruido.
- **Sm**: Entrada de audio de control que *suaviza* el proceso de resíntesis. Las partículas de sonido se ajustan usando esta entrada. Los valores pequeños ofrecen como resultado un sonido más áspero.
- **Pan**: Entrada de audio de control para la posición en el campo estéreo (-1 = Izquierda, 0 = Centro, 1 = Derecha).
- **A**: Entrada de audio de control para la amplitud de salida.
- **L**: Salida de audio para el canal izquierdo del resintetizador.
- **R**: Salida de audio para el canal derecho del resintetizador.
- **Lng**: Salida de evento polifónica para la actual posición en el sample en milisegundos.
- **Pos**: Salida de evento polifónica para la posición actual en el sample en milisegundos. Los eventos se generan en intervalos de tiempo que se corresponden con la granularidad ajustada (**Gr**).



Es un resintetizador en tiempo real para reproducción mono o estéreo polifónica de samples y mapas de samples o WaveSets. **Sample Pitch Former** es un sintetizador de WaveSet en el que no sólo se pueden cargar WaveSets, sino cualquier sample que quieras. **Sample Pitch Former** elimina el desarrollo de tonalidad de un sample y le da el tono que tú quieras. Además del control independiente en tiempo real sobre la velocidad de reproducción, **Sample Pitch Former** también te permite llevar a cabo una transposición espectral, es decir, una transposición del timbre independientemente del tono.

Los samples “estándar” como los familiares samplers hardware, o los módulos **Sampler**, **Sampler FM** y **Sampler Loop** de REAKTOR, mantienen un indicador para cada voz. Este indicador muestra la posición actual del sample. La amplitud de salida del sampler siempre es la misma que la amplitud del sample en la posición del indicador. El indicador se mueve con más o menos rapidez a través del sample y por lo tanto, genera una amplitud en la salida que varía con el tiempo – es decir, una oscilación.

La rapidez de movimiento del indicador determina la velocidad de reproducción del sample. Al mismo tiempo, también determina la tonalidad: cuanto más lento sea el muestreo de la señal (que se ha grabado en el sample), más largos serán los períodos de los osciladores en la salida – y por lo tanto el tono decrece. Si el indicador deja de moverse, la amplitud de salida deja de cambiar y no se producirá ninguna señal audible.

Igual que un sampler convencional, **Sample Pitch Former** usa un indicador en la posición actual del sample. No obstante, la señal de la salida no es simplemente la amplitud del sample en la posición del indicador. Lo que ocurre en realidad, es que la señal de salida se genera por un sintetizador dentro

del módulo. El sintetizador *resintetiza* la señal en la posición del indicador. La tonalidad de la señal generada por este sintetizador es independiente de la velocidad del indicador. Incluso si el indicador no se mueve en absoluto, el sintetizador continúa para producir un sonido. La tonalidad de salida se determina, como es usual, sobre la entrada **P**, pero la entrada **Sp** determina la velocidad del indicador.

Los samplers convencionales y el módulo **Sample Resynth** de REAKTOR consiguen un cambio de tonalidad *relativo* a través de la *transposición* de un sample, **Sample Pitch Former** impone al sample cualquier *tono absoluto y definido* que se te ocurra. Por lo tanto, **Sample Pitch Former** se puede usar como oscilador de REAKTOR. Dependiendo del tipo de material procesado y de los ajustes que se usen, el resultado puede sonar “electrónicamente” alterado en mayor o menor grado. **Sample Pitch Former** también genera señales con cierto tono si el sample procesado no tiene un único tono en sí mismo (por ejemplo, unos platos o un bombo). **Sample Pitch Former** produce menos alteraciones del sonido cuanto más restringido sea el tono fundamental del material procesado que se va a definir, y cuanto menos desviado esté el tono que se va a generar, del tono original.

En samplers convencionales y en el módulo the **Sample Resynth** de REAKTOR, la transposición del tono original se hace mano a mano con la transposición de *todos* los componentes espectrales. Esto normalmente se considera una limitación, puesto que la transposición también afecta a ciertos componentes espectrales que quien escucha no espera que sean cambiados. El “efecto Mickie Mouse” que ocurre en la desafinación de las grabaciones de voces, es la consecuencia del transporte de los formantes, cuya posición en una voz humana “real” sería independiente del tono fundamental.

Con ciertos límites, **Sample Pitch Former** desacopla la tonalidad y la posición de los formantes, unos de otros. La posición de los formantes se puede afinar independientemente a través de la entrada (**FS**). Puesto que el oído humano usa todos los componentes espectrales para identificar el tono fundamental, puedes manipular este parámetro para crear sustituciones muy interesantes del tono y el timbre fundamentales, sobre todo en tonos muy bajos. Los expertos de la síntesis consiguen efectos muy similares usando la *sincronización de oscilador*.

La administración de los samples se lleva a cabo con el **Editor de Mapas de Samples**.

Después de un evento **Gate** positivo, la reproducción del sample empieza desde el punto de inicio (configurable en la entrada **St**). Si **Loop** se desactiva, el sample funcionará una vez desde el punto de inicio hacia el final o, si está

activada la opción **Backward**, funcionará desde el punto de inicio hasta el principio. Si **Loop** está activado, el sample se repetirá continuamente dentro del alcance del loop en cuanto la reproducción introduzca este rango. El rango del loop se programa usando datos desde el archivo de sonido desde el cual se cargó el sample. Si el archivo de sonido no contiene ningún dato, se cogerá el principio del sample como principio del loop, y la duración del sample como duración del loop. Los datos del loop que se extraen del archivo de sonido son ajustes por defecto. Estos ajustes siempre se usan si las entradas **Loop Start (LS)** o **Loop Length (LL)** no están conectadas a otros módulos.

Si la opción **Loop in Release** está activada, a la reproducción del loop se le aplicará un fundido o fundido de salida con un evento **Gate**. Dependiendo de la dirección que se ajuste, el sample funcionará hacia el principio o hacia el final y luego se le aplicará fundido de salida. Si la opción **Loop in Release** se activa o si la reproducción de loop está desactivada, los eventos **Gate** menores o iguales que cero no tendrán ningún efecto.

Si la opción **No Stereo** está activada (puedes ajustarlo en las propiedades), los samples estéreo se tratarán como samples mono, es decir, se envía la misma señal a la salida **L** y a la **R**. Si éste es el caso, **Sample Pitch Former** sólo usará el canal izquierdo de los samples estéreo. Esta opción está disponible para disponer de menos carga de procesamiento.

Todas las entradas de **Sample Pitch Former**, excepto **Gate**, están diseñadas como entradas de audio. Los valores de estas entradas se aplican cuando los samples se (re)activan (es decir, durante los eventos **Gate** positivos). Los cambios efectuados durante la reproducción del sample tendrán efecto sobre los intervalos del período de tonalidad fundamental (que podrás ajustar en la entrada **P**).

- **G:** Un evento positivo en esta entrada activa la salida desde la posición ajustada en la entrada **St**. Si hay valores negativos en la entrada, la reproducción del loop se interrumpirá a menos que esté activada la opción **loop in Release**.
- **P:** Entrada de audio logarítmica de control para la tasa de reproducción (tonalidad). Además, si la entrada **Sel** no está conectada, **P** también determina la selección de samples desde el mapa de samples.
- **Sel:** Entrada de audio de control para seleccionar un sample desde el mapa de samples. Si esta entrada no está conectada, los valores presentes en la entrada **P** se usan en su lugar.
- **St:** Entrada de audio de control para el punto de inicio que se va a usar cuando ocurra el siguiente evento activador. La posición se ajusta en milisegundos desde el principio del sample.

- **LS:** Entrada de audio de control para el punto de inicio en milisegundos desde el principio del sample. Si esta entrada no está conectada, se usarán los datos del loop guardados en el archivo de sonido. Si no hay almacenado ningún dato de loop en el archivo de sonido, se usará por defecto **LS** = 0.
- **LL:** Entrada de audio de control para la duración del loop en milisegundos. Si esta entrada está desconectada, se usarán los datos de loop almacenados en el archivo de sonido. Si no hay datos del loop almacenados en el archivo, se usará por defecto **LL** = duración del sample. Si **LL** = 0, el movimiento dentro del sample se parará cuando se alcance el punto de inicio del loop; el sonido se congelará en este punto.
- **Sp:** Entrada de audio de control para la velocidad de salida de tonalidad independiente. Los valores presentes en esta entrada se interpretan como factor: Cuando **Sp** = 1, la reproducción ocurre en la velocidad original; **Sp** = 2, corresponde al doble de velocidad; y **Sp** = 0, significa stop. Si esta entrada no está conectada, se toma el valor 1 por defecto.
- **FS:** Entrada de audio de control para la transposición independiente del tono de la posición de formantes en semitonos.
- **SO:** Entrada de audio de control para la modulación de la posición del sample (Sample Offset) en milisegundos. Esta entrada se usa para modular la posición en el sample independientemente del tono, por ejemplo, a través de un generador de ruido.
- **Sm:** Entrada de audio de control que *suaviza* el proceso de resíntesis. Las partículas de sonido se ajustan usando esta entrada. Los valores pequeños ofrecen como resultado un sonido más áspero.
- **Pan:** Entrada de audio de control para la posición en el campo estéreo (-1 = Izquierda, 0 = Centro, 1 = Derecha).
- **A:** Entrada de audio de control para la amplitud de salida.
- **L:** Salida de audio para el canal izquierdo del resintetizador.
- **R:** Salida de audio para el canal derecho del resintetizador.
- **Lng:** Salida de evento polifónica para la actual posición en el sample en milisegundos.
- **Pos:** Salida de evento polifónica para la posición actual en el sample en milisegundos. Los eventos se generan en intervalos de tiempo que se corresponden con el período de tonalidad fundamental actual.



Sintetizador granular estéreo multi-sample con control independiente de tonalidad **P**, de portamento **PS** (Pitch-Slide), selección de samples **Sel**, posición de samples **Pos** y duración **Len** de cada . La envolvente de cada se puede controlar con **Att** (Attack) y **Dec** (Decay).

Para cada se puede ajustar el delta tiempo con **dt** hasta el inicio del próximo gránulo. El número máximo de gránulos simultáneos se puede ajustar en las Propiedades.

Hay varias entradas fluctuantes (Jitter) que ajustan un alcance para la entrada respectiva.

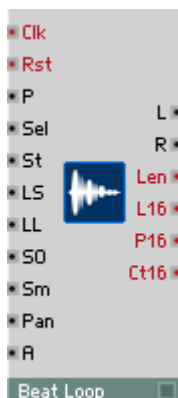
La administración de samples se lleva a cabo con el **Editor de Mapas de Samples**.

- **Trig:** Entrada de evento para la señal de puerta (Gate). (G) > 0 inicia el siguiente gránulo inmediatamente.
- **P:** Entrada de audio para el control logarítmico de la tonalidad (en semitonos). El tono es independiente de la velocidad transversal. El sample suena en el tono original cuando P = Root Key. Rango típico: [0...127], Por defecto: 60.

- **D/F:** Entrada de audio para el control de dirección si P está conectada. Si no, es un control de frecuencia. El sample suena en su tono original cuando $F = 1$; en Reverse cuando $F = -1$. Rango típico: $[-4...4]$, Por defecto: 1.
- **PJ:** Entrada de audio para el control Jitter (en semitonos). Alcance típico: $[0...3]$.
- **PS:** Entrada de audio para control Pitch-Shift logarítmico del gránulo actual en semitonos. Alcance típico: $[-3...3]$.
- **Sel:** Entrada de audio para la selección multi-sample (en semitonos). Cuando está desconectada, se usan los valores en la entrada (P). Alcance típico: $[0...127]$.
- **Pos:** Entrada de audio para seleccionar la posición del archivo de sonido en ms. Alcance: $[0...<\text{duración del sample en ms}>]$.
- **PsJ:** Entrada de audio para el control Jitter de la posición del archivo de sonido en ms. Alcance típico: $[0...<\text{duración del sample en ms}>]$.
- **Len:** Entrada de audio para ajustar la duración del gránulo en ms. Alcance típico: $[10...100]$. Por defecto: 20 ms.
- **LnJ:** Entrada de audio para el control Jitter de la duración del gránulo en ms. Rango típico: $[10...100]$. Por defecto: 0 ms.
- **Att:** Entrada de audio para ajustar el tiempo de attack. Rango: $[0...1]$. Por defecto: 0.2.
- **Dec:** Alcance para ajustar el tiempo de decay. Rango: $[0...1]$. Por defecto: 0.2.
- **Dist:** Entrada de audio para ajustar el delta tiempo antes del comienzo del siguiente gránulo en ms. Rango típico: $[5...100]$. Por defecto: 20.
- **DisJ:** Entrada de audio para el control Jitter del delta tiempo en ms. Rango típico: $[10...100]$. Por defecto: 20 ms.
- **Pan:** Entrada de audio para ajustar la posición en el campo estéreo. Rango: $[-1(\text{Izquierda})...1(\text{Derecha})]$.
- **PnJ:** Entrada de audio para el control Jitter de panoramización. Rango: $[0...1]$.
- **A:** Entrada de audio para el control de amplitud. Rango típico: $[0...1]$. Por defecto: 1.
- **L:** Salida de audio polifónica para el canal izquierdo. Rango típico: $[-1...1]$.

- **R:** Salida de audio polifónica para el canal derecho. Rango típico: [-1...1].
- **Lng:** Salida de evento para la duración de los samples en ms. Rango típico: [0...<duración de los samples en ms>].
- **GTr:** Salida de evento para la activación de gránulos. Entrega 1 cuando inicia un nuevo gránulo; 0 cuando para.

Beat Loop



Sampler

Resintetizador de tiempo real para la reproducción sincronizada de samples beat loop. La transposición de beat-loops se puede llevar a cabo, independientemente de la velocidad de reproducción, a través de la entrada **P**. **Beat Loop** se sincroniza a una fuente de reloj 1/96 que está conectada en la entrada **C** o, si la entrada **C** está sin conectar, se puede sincronizar con el reloj global de REAKTOR. Así, por defecto, todos los **Beat Loops** de REAKTOR funcionan en sincronía con la velocidad interna de otro sample independiente. Además, **Beat Loop** también te permite conectar fácilmente módulos secuenciadores internos de REAKTOR y relojes MIDI para samples rítmicos.

La administración de samples se lleva a cabo con el **Editor de Mapas de Samples**.

Beat Loop requiere samples que se hayan cortado con exactitud, y que contengan 2, 4, 8, 16, o 32, etc. beats. La velocidad de los beat loops usados tendría que estar entre 87 y 174 BPM. Si los samples reúnen estos requisitos, **Beat Loop** puede transmitirlos en una buena calidad incluso aunque se cambie la velocidad de reproducción. La opción **Pitched Sound** relativa a los samples (accesible desde el diálogo de propiedades), puede evitar falsificaciones de la tonalidad de bajos y similares. No obstante, al hacer esto, tendrás que conformarte con un deterioro en la precisión rítmica.

El rango del loop del sample se configura usando las entradas Loop Start (**LS**) y Loop Length (**LL**). Si **LS** y **LL** no están conectadas, el sample completo se reproducirá como loop. Usando eventos **Rst** positivos, la reproducción continuará desde el punto de inicio (puedes ajustarlo en la entrada **St**).

Si la opción **No Stereo** está activada (puedes ajustarlo en el diálogo de propiedades), los samples estéreo se tratarán como mono, es decir, se enviará la misma señal a las dos salidas **L** y **R**. En este caso, **Beat Loop** usa sólo el canal izquierdo de los samples estéreo. Esta opción está disponible para poder disponer de menos carga de procesamiento.

Todas las entradas de **Beat Loop**, excepto **C** y **Rst**, están diseñadas como entradas de audio. Durante la reproducción del sample, los cambios aplicados a los valores tendrán efecto en el intervalo de semicorcheas.

- **C**: Un evento positivo en esta entrada hace avanzar el módulo **Beat Loop** en un valor de 1/96. Si esta entrada no está conectada, el módulo se sincroniza con el reloj global de REAKTOR.
- **Rst**: Un evento positivo en esta entrada reinicia la reproducción en la posición ajustada en la entrada **St**.
- **P**: Entrada de audio de control logarítmica para la tasa de reproducción (tonalidad). Además, si la entrada **Sel** no está conectada, **P** determina también la selección de samples desde el mapa de samples.
- **Sel**: Entrada de audio de control para seleccionar un sample desde el mapa de samples. Si esta entrada no está conectada, se usarán los valores presentes en la entrada **P** en su lugar.
- **St**: Entrada de audio de control para el punto de inicio que se va a usar cuando ocurra el siguiente evento activador. La posición se ajusta en semicorcheas desde el principio del sample.
- **LS**: Entrada de audio de control para el punto de inicio del loop en semicorcheas desde el principio del sample. Si esta entrada no está conectada, por defecto: **LS** = 0.
- **LL**: Entrada de audio de control para la duración del loop en semicorcheas. Si esta entrada está desconectada, se usará por defecto **LL** = duración del sample.
- **SO**: Entrada de audio de control para la modulación de la posición del sample (Sample Offset) en semicorcheas. Esta entrada se usa para conseguir saltos en el Sample que, por ejemplo, se puedan controlar con un secuenciador.

- **Sm:** Entrada de audio de control para la suavidad (*Smoothness*) del proceso de resíntesis. Las partículas de sonido se ajustan usando esta entrada. Los valores muy pequeños generalmente suelen crear ruidos en cada intervalo de semicorchea.
- **Pan:** Entrada de audio de control para la posición del campo estéreo (-1 = Izquierda, 0 = Centro, 1 = Derecha).
- **A:** Entrada de audio de control para la amplitud de salida.
- **L:** Salida de audio para el canal izquierdo del resintetizador.
- **R:** Salida de audio para el canal derecho del resintetizador.
- **Len:** Salida de evento polifónica para la duración del actual sample en milisegundos.
- **L16:** Salida de evento polifónica para la duración del sample actual en semicorcheas.
- **P16:** Salida de evento polifónica para la posición actual en el sample en semicorcheas.
- **Ct16:** Salida de evento polifónica para contar las semicorcheas que han pasado desde el inicio/reinicio.

Sample Lookup



Sampler

Este módulo hace los samples disponibles como tabla de valores. A través de la entrada **Pos** se puede ajustar una posición dentro del sample en milisegundos. El valor del sample en este punto se envía a las salidas.

Puedes cargar archivos de sonido usando **Load Sound...** en el menú contextual.

La administración de los samples se lleva a cabo con el **Editor de Mapas de Samples**.

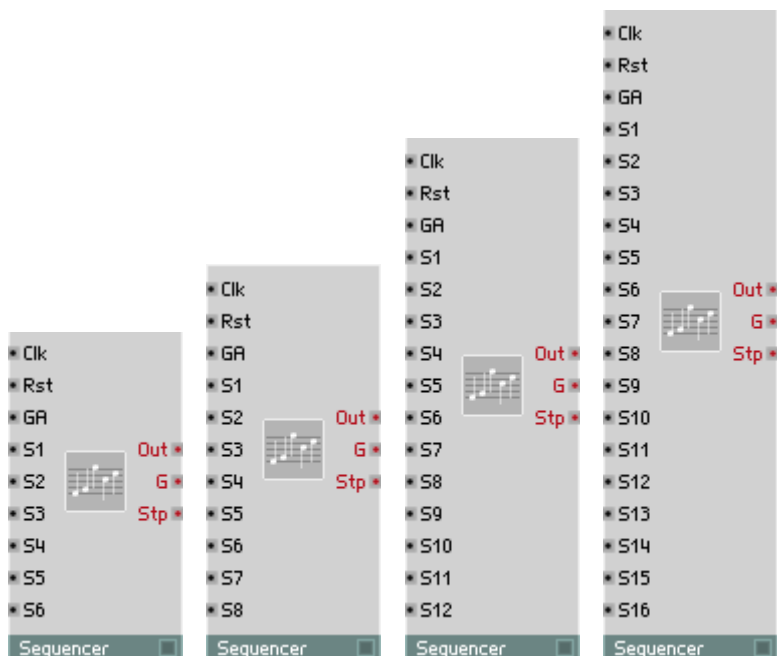
El diálogo de Propiedades te permite seleccionar uno de los tres niveles de calidad que se usan en la interpolación durante la reproducción transportada de samples (**Pobre**, **Buena** y **Excelente**). Si se ajusta a **Pobre**, los valores de los samples no se interpolarán durante la salida. Pero con una calidad de reproducción mayor, hay que pagar el precio de capacidad de procesamiento.

- **Pos:** Entrada de audio para la posición del sample en milisegundos.
- **A:** Entrada de audio para la modulación de amplitud.
- **L:** Salida de audio para el canal izquierdo del sample. Al procesar samples mono, se enviará misma señal que en la salida **R**.
- **R:** Salida de audio para el canal derecho del sample. Al procesar samples mono, se enviará la misma señal que en la salida **L**.
- **Lng:** Salida de evento polifónica para la duración del sample actual en milisegundos.

Sequencer

Reaktor incluye secuenciadores por pasos con puerta en cuatro tamaños, además de un secuenciador que mediante un valor de posición, selecciona una entrada para el procesamiento de señales.

Sequencer



Sequencer

6-Step

Sequencer

Secuenciador con 6 pasos. El valor de salida de cada paso (por ejemplo, para controlar la tonalidad del oscilador) se puede ajustar de manera independiente. Además, se transmitirá una señal de puerta con la amplitud dada por el valor actual de la entrada de amplitud de puerta según avancen los pasos.

- **C:** Entrada de audio para el control de reloj. Una entrada positiva en cero se conecta al siguiente paso. Normalmente aquí se conecta un Pulse Oscillator o un MIDI Sync.

- **Rst:** Entrada de audio para una señal de re-inicio. Una pasada positiva por cero resetea el secuenciador al primer paso. Normalmente, aquí se conecta un botón o el módulo MIDI Start.
- **GA:** Entrada de audio para controlar la amplitud de la salida de puerta. Cuando el valor es cero o la entrada no está conectada, no aparecerá ninguna señal en la salida de puerta.
- **S1:** Entrada de audio para controlar el valor del primer paso.
- **S2:** Entrada de audio para controlar el valor del segundo paso.
- **S3:** Entrada de audio para controlar el valor del tercer paso.
- **S4:** Entrada de audio para controlar el valor del cuarto paso.
- **S5:** Entrada de audio para controlar el valor del quinto paso.
- **S6:** Entrada de audio para controlar el valor del sexto paso.
- **Out:** Salida de evento para la señal de los pasos del secuenciador.
- **G:** Salida de evento para la señal de puerta.
- **St:** Salida de evento para el número de paso actual.

8-Step

Sequencer

Como el secuenciador de 6 pasos, pero con 8 pasos.

12-Step

Sequencer

Como el secuenciador de 6 pasos, pero con 12 pasos.

16-Step

Sequencer

Como el secuenciador de 6 pasos, pero con 16 pasos.



Selector de valores, que resulta muy útil como secuenciador. Un evento en la entrada **Pos** dirige, a través su valor, una de las 16 entradas y el valor actual en esta entrada es el que se envía a la salida **Out**.

Si la entrada **Pos** se alimenta por una señal que crece por pasos, la salida **Out** proporciona una secuencia de valores en las entradas **0...15**. Los valores en la entrada **Pos** se agrupan dentro del rango numérico [0 ... (Len - 1)] para permitir secuencias de duración **Len** variables.

La entrada **Pos** también puede estar alimentada por un elemento de control, o un módulo **Beat Loop**, una fuente de evento random o por el reloj maestro.

- **Pos**: Entrada para controlar la selección. Cada evento en esta entrada supone un evento en la salida. Para que Select 16 funcione como secuenciador, la posición ha de ajustarse al número de semicorcheas que hayan pasado desde el inicio o reinicio.
- **Len**: Entrada para ajustar la duración de la secuencia. Rango: [1 ... 16].
- **0-15**: Entrada de audio para controlar el valor del paso apropiado.
- **Stp**: Número de paso actual en la secuencia. El número de esta salida se calcula como "**Pos** modulo **Len**". Rango: [0 ... <duración de la secuencia>].

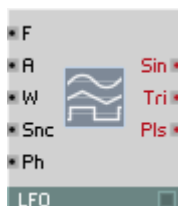
- **Bar:** Número del compás actual. El número de esta salida se calcula como entero(**Pos** / **Len**).
- **Out:** Valor de salida del paso actual de la secuencia.

LFO, Envelope

Esta categoría posee un LFO completamente modulable con diferentes formas de onda, un generador de control random, y generadores de envolventes de todo tipo y descripción, incluyendo generadores de rampas de tiempo/nivel en tres tamaños.

Todas las Envolventes pueden, de forma opcional, representar gráficamente sus curvas en el gráfico del panel. Puedes hacerlo con la opción **Visible** en la página **Appearance** en las propiedades del módulo. El tamaño del display se puede ajustar en las propiedades usando **Size X** y **Size Y**. La línea de tiempo de la curva se puede mostrar, pero no está en escala.

LFO



LFO, Envelope

LFO con salidas de forma de onda rectangulares, triangulares y sinusoidales. Normalmente, se usa como fuente para modular señales (para vibrato, tremolo, etc.). La señal de salida es una corriente de eventos al **Control Rate** (Frecuencia de Muestreo) seleccionado en el menú **Settings**.

Puesto que el LFO opera en esa frecuencia de muestreo, es más eficiente en cuanto a carga de CPU que un oscilador de audio, y hace el mismo trabajo.

- **F:** Entrada de audio para el control de la frecuencia del oscilador en Hz. Para controlar con Tempo en BPM, puedes calcular el valor necesario en Hz así: $F = \text{oscilaciones-por-beat} \times \text{BPM}/60$. Así, por ejemplo, para tres oscilaciones por compás en 4/4 (es decir, $\frac{3}{4}$ oscilaciones por beat), obtendrás $F = (\frac{3}{4})/60 \times \text{BPM}$ ó $0.0125 \times \text{BPM}$.

- **A:** Entrada para controlar la amplitud. La señal de salida se mueve entre $+A$ y $-A$.
- **W:** Entrada de evento para controlar el ancho de pulso, es decir, la relación de duración del período desde ON hasta Off de la onda rectangular, y hacer que las otras formas de onda se desplacen correctamente. Rango de valores: -1 a 1. Formas de onda simétricas con **W** = 0.
- **Snc:** Entrada de evento para la sincronización de la onda LFO. Un evento positivo sincroniza el oscilador, reiniciándolo a la fase dada en la entrada **Ph**.
- **Ph:** Entrada para especificar la fase (posición de la forma de onda) a la que el oscilador se reinicia cuando ocurre la sincronización. **Ph** = 0: Fase = 0 grados (mitad de la rampa ascendente), **Ph** = 0.5: Fase = 180 grados (mitad de la rampa descendente), **Ph** = 1: Fase = 360 grados (igual que 0 grados).
- **Sin:** Salida de audio para la señal de forma de onda sinusoidal.
- **Tri:** Salida de audio para la señal de forma de onda triangular.
- **Pls:** Salida de audio para la señal de forma de onda rectangular.

Slow Random



LFO, Envelope

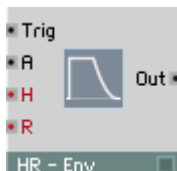
LFO que produce una forma de onda aleatoria por pasos.

- **F:** Entrada de control para la frecuencia en Hz.
- **A:** Entrada de control para la amplitud. El valor de salida está entre $-A$ y $+A$.
- **Out:** Salida de evento para la señal random.



Generador de envolvente con característica Hold. Cuando se activa la envolvente, el valor de salida salta al valor actual de la entrada de amplitud y permanece hasta que ha pasado el tiempo de retención, después de lo cual, la salida vuelve a cero. La envolvente se puede activar en cualquier momento, incluso durante el tiempo de retención.

- **T:** Entrada de audio para la activación de la envolvente en un borde ascendente (el valor va desde cero en dirección positiva).
- **A:** Entrada de audio para el valor de salida durante el tiempo de retención. La salida se samplea en el instante de activación, después de lo cual el valor se retiene en la salida.
- **H:** Entrada de evento logarítmica para controlar el tiempo de retención. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (es decir, en $\text{dB}_{1\text{ms}}$).
- **Out:** Salida de audio para la señal de la envolvente.

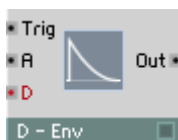


Generador de envolvente con característica Hold-Release. Cuando la envolvente se activa, el valor de salida salta al valor actual de la entrada de amplitud y permanece hasta que ha pasado el tiempo de retención. Después la salida decae exponencialmente según el tiempo de release hasta cero.

- **T:** Entrada de audio para activar la envolvente en un borde ascendente (el valor va desde cero en dirección positiva).
- **A:** Entrada de audio para el valor de salida del tiempo de hold. La entrada se samplea en el instante de activación. Después el valor se retiene en la salida.
- **H:** Entrada de evento logarítmica para controlar el tiempo de hold. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (es decir, en $\text{dB}_{1\text{ms}}$).

- **R:** Entrada de evento logarítmica para controlar el tiempo de release.
0 = 1 ms, 20 = 10 ms, 40 = 100 ms (es decir, en $\text{dB}_{1\text{ms}}$).
- **Out:** Salida de audio para la señal de la envolvente.

D - Env

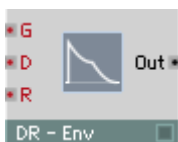


LFO, Envelope

Generador de envolvente con característica Decay. Cuando la envolvente se activa, el valor de salida salta al valor actual de la entrada de amplitud. Luego la salida decae exponencialmente con el tiempo de decay hasta cero.

- **T:** Entrada de audio para activar la envolvente en un borde ascendente (el valor va desde cero en dirección positiva).
- **A:** Entrada de audio para el valor de salida inicial. La entrada se samplea en el instante de activación.
- **D:** Entrada de evento logarítmica para controlar el tiempo de decay.
0 = 1 ms, 20 = 10 ms, 40 = 100 ms (es decir, en $\text{dB}_{1\text{ms}}$).
- **Out:** Salida de audio para la señal de la envolvente.

DR - Env



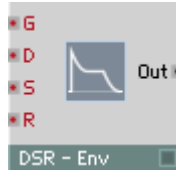
LFO, Envelope

Generador de envolvente con característica de Decay-Release. Cuando la envolvente se activa con un evento de puerta, el valor de salida salta al valor de amplitud de la señal de la puerta. Luego, la salida decae exponencialmente con el tiempo de decay hasta cero. Un evento de puerta con amplitud cero (Note Off) ajusta el tiempo de decay al valor del tiempo de release, que normalmente se ajusta para ser menor.

- **G:** Entrada de evento para la señal de la puerta que activará la envolvente. La amplitud de la señal de la puerta determina el valor de salida inicial.
- **D:** Entrada de evento logarítmica para controlar el tiempo de decay.
. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (es decir, en $\text{dB}_{1\text{ms}}$).

- **R:** Entrada de evento logarítmica para controlar el tiempo de release. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (es decir, en $\text{dB}_{1\text{ms}}$).
- **Out:** Salida de audio para la señal de la envolvente.

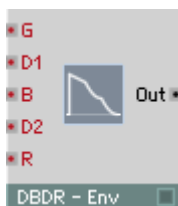
DSR - Env



LFO, Envelope

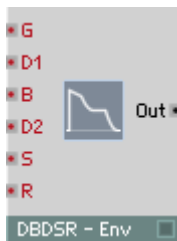
Generador de envolvente con característica de Decay-Sustain–Release. Cuando la envolvente se activa con un evento de puerta, el valor de salida salta al valor de la amplitud de la señal de la puerta. Luego la salida decae exponencialmente con el tiempo de decay hasta el nivel de sustain (multiplicado por la amplitud). Tras un evento de puerta con amplitud cero (con Note Off) la salida decae exponencialmente con el tiempo de release hasta cero.

- **G:** Entrada de evento para la señal de la puerta que activa la envolvente. La amplitud de la señal de la puerta determina el valor de salida inicial.
- **D:** Entrada de evento logarítmica para controlar el tiempo de decay. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (es decir, en $\text{dB}_{1\text{ms}}$).
- **S:** Entrada de evento para controlar el nivel de sustain. Rango típico de valores: 0 (decay a cero) a 1 (hold en el nivel inicial), pero sustain puede ser mayor que 1 o incluso menor que 0.
- **R:** Entrada de evento logarítmica para controlar el tiempo de release. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (es decir, en $\text{dB}_{1\text{ms}}$).
- **Out:** Salida de audio para la señal de la envolvente.



Generador de envolvente con característica Decay–Breakpoint–Release. Cuando la envolvente se activa con un evento de puerta, el valor de salida salta al valor de la amplitud en la señal de la puerta. Luego la salida decae exponencialmente con el tiempo decay-1 hasta que alcanza el nivel del punto de inflexión (multiplicado por la amplitud). Después continúa decayendo exponencialmente con el tiempo decay-2 hasta cero. Un evento de puerta con amplitud cero (Note Off) ajusta el tiempo de decay al tiempo de release, que normalmente se ajusta para ser menor.

- **G:** Entrada de evento para la señal de la puerta que activa la envolvente. La amplitud de la señal de la puerta determina el valor de salida inicial.
- **D1:** Entrada de evento logarítmica para controlar el primer tiempo de decay. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (es decir, en $\text{dB}_{1\text{ms}}$).
- **B:** Entrada de evento logarítmica para controlar el nivel del punto de inflexión. Rango de valores: 0 (nunca usar tiempo decay-2) a 1 (usa inmediatamente tiempo decay-2).
- **D2:** Entrada de evento logarítmica para controlar el segundo tiempo de decay. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (es decir, en $\text{dB}_{1\text{ms}}$).
- **R:** Entrada de evento logarítmica para controlar el tiempo de release. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (es decir, en $\text{dB}_{1\text{ms}}$).
- **Out:** Salida de audio para la señal de la envolvente.

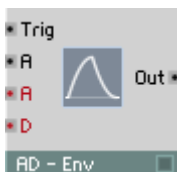


Generador de envolvente con característica Decay-Sustain-Release. Cuando la envolvente se activa con un evento de puerta, el valor de salida salta al valor de la amplitud en la señal de la puerta. Luego, la salida decae exponencialmente con el tiempo decay-1 hasta que alcanza el nivel del punto de inflexión (multiplicado por la amplitud). Luego continúa decayendo con el tiempo decay-2 hasta el nivel de sustain (multiplicado por la amplitud). La salida se retiene en el nivel de sustain hasta que llega un evento de puerta con amplitud cero (Note Off), luego la salida decae exponencialmente con el tiempo de release hasta cero.

- **G:** Entrada de evento para la señal de la puerta que activa la envolvente. La amplitud de la señal de la puerta determina el valor de salida inicial.
- **D1:** Entrada de evento logarítmica para controlar el primer tiempo de decay. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (es decir, en $\text{dB}_{1\text{ms}}$).
- **B:** Entrada de evento logarítmica para controlar el nivel del punto de inflexión. Rango de valores: 0 (nunca usar tiempo decay-2) a 1 (usa inmediatamente tiempo decay-2).
- **D2:** Entrada de evento logarítmica para controlar el segundo tiempo de decay. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (es decir, en $\text{dB}_{1\text{ms}}$).
- **S:** Entrada de evento para controlar el nivel de sustain. Típico Rango: 0 (decay a cero) a 1 (retiene al final del ataque), pero sustain puede ser mayor que 1 o incluso menor que cero.
- **R:** Entrada de evento logarítmica para controlar el tiempo de release. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (es decir, en $\text{dB}_{1\text{ms}}$).
- **Out:** Salida de audio para la señal de la envolvente.

AD - Env

LFO, Envelope



Envolvente Attack-Decay, activada por una rampa positiva de la señal T. El decay comienza inmediatamente cuando ha terminado el tiempo de ataque.

- **T:** Entrada para la señal activadora. Una rampa positiva inicia la envolvente.
- **A:** Entrada de control para la amplitud de salida.
- **A:** Entrada de control para el tiempo de ataque de la envolvente. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (es decir, en $\text{dB}_{1\text{ms}}$).
- **D:** Entrada de control para el tiempo de decay de la envolvente. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (es decir, en $\text{dB}_{1\text{ms}}$).
- **Out:** Salida para la señal de la envolvente.

AR - Env

LFO, Envelope



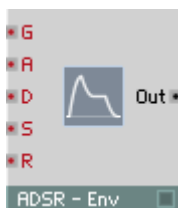
Generador de envolvente con característica Attack-Release. Cuando la envolvente se activa con un evento de puerta, el valor de salida asciende durante el tiempo de ataque con una rampa ascendente hasta el valor de amplitud de la señal de la puerta. Luego la salida se retiene al nivel máximo hasta que llega un evento de puerta con amplitud cero (Note Off), y la salida decae exponencialmente con el tiempo de release hasta cero.

- **G:** Entrada de evento para la señal de la puerta que activa la envolvente. La amplitud de la señal de puerta determina el valor máximo de salida.
- **A:** Entrada de evento logarítmica para controlar el tiempo de ataque. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (es decir, en $\text{dB}_{1\text{ms}}$).
- **R:** Entrada de evento logarítmica para controlar el tiempo de release. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (es decir, en $\text{dB}_{1\text{ms}}$).
- **Out:** Salida de audio para la señal de la envolvente.



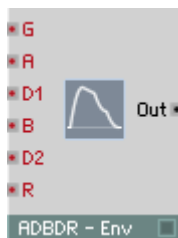
Generador de envolvente con característica Attack-Decay-Release. Cuando la envolvente se activa con un evento de puerta, el valor de salida asciende durante el tiempo de ataque con una rampa lineal hasta el valor de amplitud en la señal de la puerta. Luego decae con el tiempo de decay hasta cero. Cuando llega un evento de puerta con amplitud cero (Note Off), la salida decae exponencialmente con el tiempo de release hasta cero.

- **G:** Entrada de evento para la señal de puerta que activa la envolvente. La amplitud de la señal de puerta determina el valor máximo de salida.
- **A:** Entrada de evento logarítmica para controlar el tiempo de ataque. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (es decir, en $\text{dB}_{1\text{ms}}$).
- **D:** Entrada de evento para controlar el tiempo de decay. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (es decir, en $\text{dB}_{1\text{ms}}$).
- **R:** Entrada de evento logarítmica para controlar el tiempo de release. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (es decir, en $\text{dB}_{1\text{ms}}$).
- **Out:** Salida de audio para la señal de la envolvente.



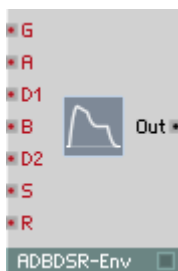
Generador de envolvente con característica clásica Attack-Decay-Sustain-Release. Cuando la envolvente se activa con un evento de puerta, el valor de salida asciende durante el tiempo de ataque con una rampa lineal hasta el valor de amplitud en la señal de la puerta. Luego, decae exponencialmente con el tiempo de decay hasta el nivel de sustain (multiplicado por la amplitud). La salida se retiene en el nivel de sustain hasta que llega un evento de puerta con amplitud cero (Note Off). Después la salida decae exponencialmente con el tiempo de release hasta cero.

- **G:** Entrada de evento para la señal de puerta que activa la envolvente. La amplitud de la señal de puerta determina el valor máximo de salida.
- **A:** Entrada de evento logarítmica para controlar el tiempo de ataque. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (es decir, en $\text{dB}_{1\text{ms}}$).
- **D:** Entrada de evento para controlar el tiempo de decay. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (es decir, en $\text{dB}_{1\text{ms}}$).
- **S:** Entrada de evento para controlar el nivel de sustain. Típico Rango: 0 (decay a cero) a 1 (retiene al final del ataque), pero sustain puede ser mayor que 1 o incluso menor que cero.
- **R:** Entrada de evento logarítmica para controlar el tiempo de release. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (es decir, en $\text{dB}_{1\text{ms}}$).
- **Out:** Salida de audio para la señal de la envolvente.



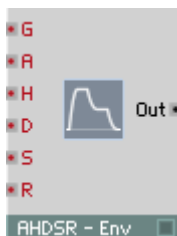
Generador de envolvente con característica Attack-Decay-breakpoint-Decay-Release. Cuando la envolvente se activa con un evento de puerta, el valor de salida asciende durante el tiempo de ataque con una rampa lineal hasta el valor de amplitud de la señal de la puerta. Luego, decae exponencialmente con el tiempo de decay-1 hasta que alcanza el nivel del punto de inflexión (multiplicado por la amplitud). Después, continúa decayendo exponencialmente con el tiempo decay-2 hasta cero. Un evento de puerta con amplitud cero (Note Off) ajusta el tiempo de decay al tiempo de release que normalmente se ajusta para ser menor.

- **G:** Entrada de evento para la señal de puerta que activa la envolvente. La amplitud de la señal de puerta determina el valor máximo de salida.
- **A:** Entrada de evento logarítmica para controlar el tiempo de ataque. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (es decir, en $\text{dB}_{1\text{ms}}$).
- **D1:** Entrada de evento para controlar el primer tiempo de decay. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (es decir, en $\text{dB}_{1\text{ms}}$).
- **B:** Entrada de evento para controlar el nivel del punto de inflexión. Rango de valores: 0 (nunca usar tiempo decay-2) a 1 (usar inmediatamente tiempo decay-2).
- **D2:** Entrada de evento para controlar el segundo tiempo de decay. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (es decir, en $\text{dB}_{1\text{ms}}$).
- **R:** Entrada de evento logarítmica para controlar el tiempo de release. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (es decir, en $\text{dB}_{1\text{ms}}$).
- **Out:** Salida de audio para la señal de la envolvente.



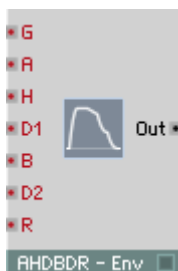
Generador de envolvente con característica Attack-Decay-Breakpoint-Decay-Sustain-Release. Cuando la envolvente se activa con un evento de puerta, el valor de salida asciende durante el tiempo de ataque con una rampa lineal hasta el valor de amplitud de la señal de la puerta. Luego, decae de acuerdo con el primer tiempo de decay con una rampa lineal hasta el nivel del punto de inflexión. Desde aquí, continúa con el segundo tiempo de decay hasta el nivel de sustain (los niveles se multiplican por la amplitud). La salida se retiene al nivel de sustain hasta que llega un evento de puerta con amplitud cero (Note Off). Después, la salida decae exponencialmente con el tiempo de release hasta cero.

- **G:** Entrada de evento para la señal de puerta que activa la envolvente. La amplitud de la señal de puerta determina el valor máximo de salida.
- **A:** Entrada de evento logarítmica para controlar el tiempo de ataque. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (es decir, en $\text{dB}_{1\text{ms}}$).
- **D1:** Entrada de evento para controlar el primer tiempo de decay. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (es decir, en $\text{dB}_{1\text{ms}}$).
- **B:** Entrada de evento para controlar el nivel del punto de inflexión. Rango de valores: 0 a 1 .
- **D2:** Entrada de evento para controlar el segundo tiempo de decay. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (es decir, en $\text{dB}_{1\text{ms}}$).
- **S:** Entrada de evento para controlar el nivel de sustain. Típico Rango: 0 a 1.
- **R:** Entrada de evento logarítmica para controlar el tiempo de release. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (es decir, en $\text{dB}_{1\text{ms}}$).
- **Out:** Salida de audio para la señal de la envolvente.



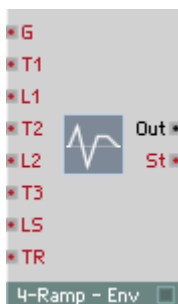
Generador de envolvente con característica Attack-Hold-Decay-Sustain-Release. Cuando la envolvente se activa con un evento de puerta, el valor de salida asciende durante el tiempo de ataque con una rampa lineal hasta el valor de amplitud de la señal de la puerta y permanece hasta que pasa el tiempo de Hold. Luego, decae de acuerdo con el tiempo Decay con una rampa lineal hasta el nivel de sustain. La salida se retiene en el nivel de sustain hasta que llega un evento de puerta con amplitud cero (Note Off) y después la salida decae exponencialmente con el tiempo de release hasta cero.

- **G:** Entrada de evento para la señal de puerta que activa la envolvente. La amplitud de la señal de puerta determina el valor máximo de salida.
- **A:** Entrada de evento logarítmica para controlar el tiempo de ataque. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (es decir, en $\text{dB}_{1\text{ms}}$).
- **H:** Entrada de evento logarítmica para controlar el tiempo de hold. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (es decir, en $\text{dB}_{1\text{ms}}$).
- **D:** Entrada de evento para controlar el tiempo de decay. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (es decir, en $\text{dB}_{1\text{ms}}$).
- **S:** Entrada de evento para controlar el nivel de sustain. Típico Rango: 0 (decay a cero) a 1 (retiene al final del ataque), pero sustain puede ser mayor que 1 o incluso menor que cero.
- **R:** Entrada de evento logarítmica para controlar el tiempo de release. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (es decir, en $\text{dB}_{1\text{ms}}$).
- **Out:** Salida de audio para la señal de la envolvente.



Generador de envolvente con característica Attack-Hold-Decay-Breakpoint-Decay-Release. Cuando la envolvente se activa con un evento de puerta, el valor de salida asciende durante el tiempo de ataque hasta el valor de amplitud de la señal de la puerta y permanece hasta que pasa el tiempo de hold. Luego decae exponencialmente con el tiempo decay-1 hasta que alcanza el nivel del punto de inflexión (multiplicado por la amplitud). Después continúa decayendo exponencialmente con el tiempo decay-2 hasta cero. Un evento de puerta con amplitud cero (Note Off) ajusta el tiempo de decay al tiempo de release, que normalmente se ajusta para ser menor.

- **G:** Entrada de evento para la señal de puerta que activa la envolvente. La amplitud de la señal de puerta determina el valor máximo de salida.
- **A:** Entrada de evento logarítmica para controlar el tiempo de ataque. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (es decir, en $\text{dB}_{1\text{ms}}$).
- **H:** Entrada de evento logarítmica para controlar el tiempo de hold. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (es decir, en $\text{dB}_{1\text{ms}}$).
- **D1:** Entrada de evento para controlar el primer tiempo de decay. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (es decir, en $\text{dB}_{1\text{ms}}$).
- **B:** Entrada de evento para controlar el nivel del punto de inflexión. Rango de valores: 0 (nunca usar tiempo decay-2) a 1 (usar inmediatamente tiempo decay-2).
- **D2:** Entrada de evento para controlar el segundo tiempo de decay. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (es decir, en $\text{dB}_{1\text{ms}}$).
- **R:** Entrada de evento logarítmica para controlar el tiempo de release. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (es decir, en $\text{dB}_{1\text{ms}}$).
- **Out:** Salida de audio para la señal de la envolvente.

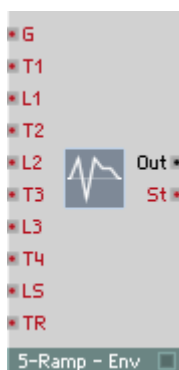


Generador de envolvente de 4 fases con rampas lineales. Cuando la envolvente se activa con un evento de puerta, el valor de salida asciende hasta el primer nivel, alcanzándolo dentro del tiempo establecido para la primera fase. Luego continúa hasta el segundo nivel dentro del tiempo establecido para la segunda fase, y así. El tercer nivel es el nivel de sustain, en el que se retiene la salida hasta que llega un evento de puerta con amplitud cero (Note Off). Luego la salida regresa a cero dentro del último período de tiempo.

- **G:** Entrada de evento para la señal de la puerta que activa la envolvente. La amplitud de la señal de la puerta se combina con todos los valores de niveles para determinar los niveles actuales alcanzados.
- **T1:** Entrada de evento logarítmico para controlar el tiempo de la primera fase. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (es decir, en $\text{dB}_{1\text{ms}}$).
- **L1:** Entrada para controlar el nivel que se alcanza al final de la primera fase.
- **T2:** Entrada de evento logarítmico para controlar el tiempo de la segunda fase. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (es decir, en $\text{dB}_{1\text{ms}}$).
- **L2:** Entrada para controlar el nivel que se alcanza al final de la segunda fase.
- **T3:** Entrada de evento logarítmico para controlar el tiempo de la tercera fase. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (es decir, en $\text{dB}_{1\text{ms}}$).
- **L3:** Entrada para controlar el nivel que se alcanza al final de la tercera fase. Este es el nivel de sustain.
- **TR:** Entrada de evento logarítmico para controlar el tiempo de la última fase, que es release. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (es decir, en $\text{dB}_{1\text{ms}}$).

- **St:** Salida de evento para la fase actual de la envolvente (1,2 ...). Después del final de la fase de release y antes de un nuevo activador, el valor es 0. Con un procesamiento apropiado, este valor se puede usar para encadenar otras envolventes, o para que la envolvente se active a sí misma.
- **Out:** Salida de audio para la señal de la envolvente.

5-Ramp



LFO, Envelope

Generador de envolvente de 5 fases con rampas lineales. Cuando la envolvente se activa con un evento de puerta, el valor de salida asciende hasta el primer nivel, alcanzándolo dentro del tiempo establecido para la primera fase. Luego continúa hacia el segundo nivel dentro del tiempo establecido para la segunda fase, y así. El cuarto nivel es el de sustain, en el que la salida se retiene hasta que llega un evento de puerta con amplitud cero (Note Off). Luego la salida regresa a cero dentro del último período de tiempo.

- **G:** Entrada de evento para la señal de la puerta que activa la envolvente. La amplitud de la señal de la puerta se combina con todos los valores de niveles para determinar los niveles actuales alcanzados.
- **T1:** Entrada de evento logarítmico para controlar el tiempo de la primera fase. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (es decir, en $\text{dB}_{1\text{ms}}$).
- **L1:** Entrada para controlar el nivel que se alcanza al final de la primera fase.
- **T2:** Entrada de evento logarítmico para controlar el tiempo de la segunda fase. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (es decir, en $\text{dB}_{1\text{ms}}$).

- **L2:** Entrada para controlar el nivel que se alcanza al final de la segunda fase.
- **T3:** Entrada de evento logarítmico para controlar el tiempo de la tercera fase. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (es decir, en $\text{dB}_{1\text{ms}}$).
- **L3:** Entrada para controlar el nivel que se alcanza al final de la tercera fase.
- **T4:** Entrada de evento logarítmico para controlar el tiempo de la cuarta fase. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (es decir, en $\text{dB}_{1\text{ms}}$).
- **LS:** Entrada para controlar el nivel que se alcanza al final de la cuarta fase. Este es el nivel de sustain.
- **TR:** Entrada de evento logarítmica para controlar el tiempo para la última fase, que es release. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (es decir, en $\text{dB}_{1\text{ms}}$).
- **St:** Salida de evento para la fase actual de la envolvente (1,2 ...). Después del final de la fase de release y antes de un nuevo activador, el valor es 0. Con un procesamiento apropiado, este valor se puede usar para encadenar otras envolventes, o para que la envolvente se active a sí misma.
- **Out:** Salida de audio para la señal de la envolvente.



Generador de envolvente de 6 fases con rampas lineales. Cuando la envolvente se activa con un evento de puerta, el valor de salida asciende hacia el primer nivel, alcanzándolo dentro del tiempo establecido para la primera fase. Luego continúa hacia el segundo nivel dentro del tiempo establecido para la segunda fase, y así. El quinto nivel es el nivel de sustain, en el que se retiene la salida hasta que llega un evento de puerta de amplitud cero (Note Off). Luego la salida regresa a cero dentro del tiempo del último período.

- **G:** Entrada de evento para la señal de la puerta que activa la envolvente. La amplitud de la señal de la puerta se combina con todos los valores de niveles para determinar los niveles actuales alcanzados.
- **T1:** Entrada de evento logarítmico para controlar el tiempo de la primera fase. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (es decir, en $\text{dB}_{1\text{ms}}$).
- **L1 :** Entrada para controlar el nivel que se alcanza al final de la primera fase.
- **T2:** Entrada de evento logarítmico para controlar el tiempo de la segunda fase. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (es decir, en $\text{dB}_{1\text{ms}}$).
- **L2:** Entrada para controlar el nivel que se alcanza al final de la segunda fase.
- **T3:** Entrada de evento logarítmico para controlar el tiempo de la tercera fase. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (es decir, en $\text{dB}_{1\text{ms}}$).

- **L3:** Entrada para controlar el nivel que se alcanza al final de la tercera fase.
- **T4:** Entrada de evento logarítmico para controlar el tiempo de la cuarta fase. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (es decir, en $\text{dB}_{1\text{ms}}$).
- **L4:** Entrada para controlar el nivel que se alcanza al final de la cuarta fase.
- **T5:** Entrada de evento logarítmico para controlar el tiempo de la quinta fase. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (es decir, en $\text{dB}_{1\text{ms}}$).
- **LS:** Entrada para controlar el nivel que se alcanza al final de la quinta fase. Este es el nivel de sustain.
- **TR:** Entrada de evento logarítmica para controlar el tiempo para la última fase, que es release. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (es decir, en $\text{dB}_{1\text{ms}}$).
- **St:** Salida de evento para la fase actual de la envolvente (1,2 ...). Después del final de la fase de release y antes de un nuevo activador, el valor es 0. Con un procesamiento apropiado, este valor se puede usar para encadenar otras envolventes, o para que la envolvente se active a sí misma.
- **Out:** Salida de audio para la señal de la envolvente.

Filter

Los filtros forman la mayor colección dentro de los procesadores de señal de Reaktor. Encontrarás 22 de ellos cubriendo todas tus necesidades, desde los filtros estándar de paso-alto, paso-bajo y paso-banda, hasta emulaciones de los clásicos filtros sintetizadores, desde Sequential a Moog. Hay también filtros de allpass para reverb y circuitos de dispersión, además de filtros de integración y diferenciación.

Todos los filtros de REAKTOR pueden operar en cualquier frecuencia, desde 0Hz (señal constante) hasta un campo completo del audio en el límite ajustado por la frecuencia de muestreo. Esto significa que todos son igualmente apropiados para el procesamiento de audio o para controles suavizantes de la señal (por ejemplo, portamento). Al usar un filtro para procesar la entrada en un puerto que sólo acepta eventos (por ejemplo, P en lugar de F) tendrás que insertar un módulo **A to E** para convertir.

Todos los filtros y EQs pueden mostrar opcionalmente el desarrollo de sus frecuencias en un panel gráfico. Puedes hacerlo a través de la opción **Visible** en la página **Appearance** dentro de las Propiedades. El tamaño del display se puede ajustar con las opciones **Size X** y **Size Y** de las Propiedades. El eje de frecuencia de la curva es logarítmico y cambia desde 10Hz a 20Hz.

HP/LP 1-Pole



Filter

Filtro de 1 polo con salidas de paso-bajo y paso-alto (pendiente 6dB/octava) y control logarítmico para la frecuencia de corte.

- **P:** Entrada de evento logarítmica para controlar la frecuencia de corte. Valor en semitonos. (69 = 440 Hz).
- **In:** Entrada de señal de audio para la señal que se va a filtrar.
- **HP:** Salida de audio para la señal del filtro de paso-alto.
- **LP:** Salida de audio para el filtro de paso-bajo.

HP/LP 1-Pole FM

Filter



Filtro de 1 polo con salidas de paso-alto, paso-bajo (pendiente 6dB/octava), y control logarítmico y lineal de la frecuencia de corte.

- **P:** Entrada de evento logarítmica para controlar la frecuencia de corte. Valor en semitonos. (69 = 440 Hz).
- **F:** Entrada de audio para el control lineal de la frecuencia de corte. Valor en Hz. **P** y **F** juntas determinan la frecuencia del oscilador.
- **In:** Entrada de audio para la señal que se va a filtrar.
- **HP:** Salida de audio para la señal del filtro de paso-alto.
- **LP:** Salida de audio para el filtro de paso-bajo.

Allpass 1-Pole

Filter

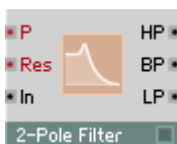


Filtro allpass de primer orden. Este tipo de filtro tiene poca pendiente, pero el desfase entre la entrada y la salida crece desde 0 grados en frecuencias bajas, y hasta -180 grados en frecuencias altas. En la frecuencia de corte controlada a través de la entrada P, se produce un desfase de -90 grados.

- **P:** Entrada de control logarítmica para la frecuencia de corte, donde el desfase es de -90 grados. Escala: 1 semitono por unidad, 43 = G1 = 98Hz, 84 = C5 = 1047Hz.
- **In:** Entrada para la señal que se va a filtrar (desfasar).
- **Out:** Salida para la señal filtrada (señal con desfase) con el allpass.

Multi 2-Pole

Filter



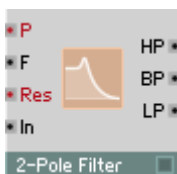
Filtro de 2 polos con salidas de paso-alto y paso-bajo (pendiente 12dB/octava), paso-banda (arriba y abajo 6dB/octava), resonancia variable, y control logarítmico de frecuencia de corte.

La ganancia del paso-banda siempre es 1 (0dB), mientras que la ganancia en la frecuencia de corte incrementa con la resonancia. Precaución: cuando **Res** es casi 1 se generan amplitudes muy altas.

- **P**: Entrada de evento logarítmica para controlar la frecuencia de corte. Valores en semitonos. (69 = 440 Hz).
- **Res**: Entrada de evento para controlar la resonancia/atenuación del filtro. Rango de valores desde 0 (máxima atenuación, no hay resonancia) a 1 (no hay atenuación, máxima resonancia, auto-oscilación). Con valores altos de resonancia el factor Q del filtro = frecuencia [Hz] / ancho de banda [Hz] $\cong 1 / (2 - 2 \text{ Res})$.
- **In**: Entrada de audio para la señal que se va a filtrar.
- **HP**: Salida de audio para la señal del filtro de paso-alto.
- **BP**: Salida de audio para la señal del filtro de paso-banda.
- **LP**: Salida de audio para la señal del filtro de paso-bajo.

Multi 2-Pole FM

Filter

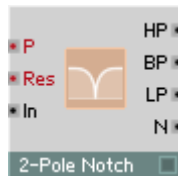


Filtro de 2 polos con salida de paso-alto y paso-bajo (pendiente 12dB/octava), paso-banda (arriba y abajo 6dB) resonancia variable, y control logarítmico y lineal de la frecuencia de corte.

La ganancia del paso-banda es siempre 1 (0 dB), mientras que la ganancia de la frecuencia de corte incrementa con la resonancia. Precaución: cuando **Res** es casi 1 se generan amplitudes muy altas.

- **P:** Entrada de evento logarítmica para controlar la frecuencia de corte. Valores en semitonos. (69 = 440 Hz).
- **F:** Entrada de audio para el control lineal de la frecuencia de corte. Valor en Hz. **P** y **F** juntas determinan la frecuencia del oscilador.
- **Res:** Entrada de evento para controlar la resonancia/atenuación del filtro. Rango de valores desde 0 (máxima atenuación, no hay resonancia) a 1 (no hay atenuación, máxima resonancia, auto-oscilación). Con valores altos de resonancia el factor Q del filtro = frecuencia [Hz] / ancho de banda [Hz] $\cong 1 / (2 - 2 \text{ Res})$.
- **In:** Entrada de audio para la señal que se va a filtrar.
- **HP:** Salida de audio para la señal del filtro de paso-alto.
- **BP:** Salida de audio para la señal del filtro de paso-banda.
- **LP:** Salida de audio para la señal del filtro de paso-bajo.

Multi/Notch 2-Pole



Filter

Filtro de 2 polos con salida de bloqueo de banda (Notch), salida de paso-alto y paso-bajo (pendiente 12dB/octava), paso-banda (arriba y abajo 6dB/octava), resonancia variable, y control logarítmico de la frecuencia de corte.

La ganancia de la frecuencia de corte es siempre 1 (0dB), mientras que la ganancia del paso-banda decrece cuando crece la resonancia.

En la salida de filtro notch, la frecuencia seleccionada se elimina completamente.

- **P:** Entrada de evento logarítmica para controlar la frecuencia de corte. Valores en semitonos. (69 = 440 Hz).
- **Res:** Entrada de evento para controlar la resonancia/atenuación del filtro. Rango de valores desde 0 (máxima atenuación, no hay resonancia) a 1 (no hay atenuación, máxima resonancia, auto-oscilación). Con valores altos de resonancia el factor Q del filtro = frecuencia [Hz] / ancho de banda [Hz] $\cong 1 / (2 - 2 \text{ Res})$.
- **In:** Entrada de audio para la señal que se va a filtrar.
- **HP:** Salida de audio para la señal del filtro de paso-alto.

- **BP:** Salida de audio para la señal del filtro de paso-banda.
- **LP:** Salida de audio para la señal del filtro de paso-bajo.
- **N:** Salida de audio para la señal del filtro de bloqueo de banda (notch).

Multi/Notch 2-Pole FM



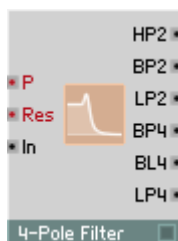
Filter

Filtro de 2 polos con salida para bloqueo de banda (notch), salida de paso-alto y paso-bajo (pendiente 12dB/octava), paso-banda (arriba y abajo 6dB/octava), resonancia variable y control logarítmico de la frecuencia de corte.

La ganancia de la frecuencia de corte siempre es 1 (0dB), mientras que la ganancia del paso-banda decrece cuando crece la resonancia.

En la salida de filtro notch, la frecuencia seleccionada se elimina completamente.

- **P:** Entrada de evento logarítmica para controlar la frecuencia de corte. Valores en semitonos. ($69 = 440 \text{ Hz}$).
- **F:** Entrada de audio para el control lineal de la frecuencia de corte. Valor en Hz. **P** y **F** juntas determinan la frecuencia del oscilador.
- **Res:** Entrada de evento para controlar la resonancia/atenuación del filtro. Rango de valores desde 0 (máxima atenuación, no hay resonancia) a 1 (no hay atenuación, máxima resonancia, auto-oscilación). Con valores altos de resonancia el factor Q del filtro = $\text{frecuencia [Hz]} / \text{ancho de banda [Hz]} \cong 1 / (2 - 2 \text{ Res})$.
- **In:** Entrada de audio para la señal que se va a filtrar.
- **HP:** Salida de audio para la señal del filtro de paso-alto.
- **BP:** Salida de audio para la señal del filtro de paso-banda.
- **LP:** Salida de audio para la señal del filtro de paso-bajo.
- **N:** Salida de audio para la señal del filtro de bloqueo de banda (notch).



Filtro de 4 polos con salidas de paso-bajo (pendiente 24dB/octava), paso-banda (12/12 dB/octava) y paso-bajo/banda (6/8 dB/octava); salidas de 2 polos de paso-alto y bajo (12dB/octava) y paso-banda (6/6 dB/octava); resonancia variable y control logarítmico de frecuencia de corte.

La ganancia del paso-banda siempre es 1 (0dB), mientras que la ganancia de la frecuencia de corte incrementa con la resonancia. Precaución: cuando **Res** es casi 1, se generan amplitudes muy altas.

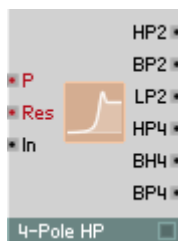
- **P**: Entrada de evento logarítmica para controlar la frecuencia de corte. Valores en semitonos. (69 = 440 Hz).
- **Res**: Entrada de evento para controlar la resonancia/atenuación del filtro. Rango de valores desde 0 (máxima atenuación, no hay resonancia) a 1 (no hay atenuación, máxima resonancia, auto-oscilación).
- **In**: Entrada de audio para la señal que se va filtrar.
- **HP2**: Salida de audio para la señal del filtro de paso-alto de 2 polos.
- **BP2**: Salida de audio para la señal del filtro de paso-banda de 2 polos.
- **LP2**: Salida de audio para la señal del filtro de paso-bajo de 2 polos.
- **BP4**: Salida de audio para la señal del filtro de baso-banda de 4 polos.
- **BL4**: Salida de audio para la señal del filtro de paso-banda/bajo de 4 polos.
- **LP4**: Salida de audio para la señal del filtro de paso-bajo de 4 polos.



Filtro de 4 polos con salidas de paso-bajo (pendiente 24dB/octava), paso-banda (12/12dB/octava) y paso-banda/bajo (6/18dB/octava); salidas de 2 polos de paso-bajo y alto (12dB/octava) y paso-banda (6/6dB/octava), resonancia variable, y control logarítmico y lineal de la frecuencia de corte.

La ganancia del paso-banda es siempre 1 (0dB), mientras que la ganancia de la frecuencia de corte incrementa con la resonancia. Precaución: cuando **Res** es casi 1, se generan amplitudes muy altas.

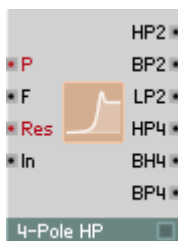
- **P**: Entrada de evento logarítmica para controlar la frecuencia de corte. Valores en semitonos. (69 = 440 Hz).
- **F**: Entrada de audio para el control lineal de la frecuencia de corte. Valor en Hz. **P** y **F** juntas determinan la frecuencia del oscilador.
- **Res**: Entrada de evento para controlar la resonancia/atenuación del filtro. Rango de valores desde 0 (máxima atenuación, no hay resonancia) a 1 (no hay atenuación, máxima resonancia, auto-oscilación).
- **In**: Entrada de audio para la señal que se va filtrar.
- **HP2**: Salida de audio para la señal del filtro de paso-alto de 2 polos.
- **BP2**: Salida de audio para la señal del filtro de paso-banda de 2 polos.
- **LP2**: Salida de audio para la señal del filtro de paso-bajo de 2 polos.
- **BP4**: Salida de audio para la señal del filtro de baso-banda de 4 polos.
- **BL4**: Salida de audio para la señal del filtro de paso-banda/bajo de 4 polos.
- **LP4**: Salida de audio para la señal del filtro de paso-bajo de 4 polos.



Filtro de 4 polos con salidas paso-alto (24dB/octava), paso-banda (12/12dB/octava) y paso-banda/alto (18/6dB/octava); salidas de 2 polos de paso-alto y bajo (12dB/octava) y paso-banda (6/6dB/octava); resonancia variable y control logarítmico de la frecuencia de corte.

La ganancia del paso-banda es siempre 1 (0dB), mientras que la frecuencia de corte incrementa con la resonancia. Precaución: Cuando **Res** es casi 1, se generan amplitudes muy altas.

- **P**: Entrada de evento logarítmica para controlar la frecuencia de corte. Valores en semitonos. (69 = 440 Hz).
- **Res**: Entrada de evento para controlar la resonancia/atenuación del filtro. Rango de valores desde 0 (máxima atenuación, no hay resonancia) a 1 (no hay atenuación, máxima resonancia, auto-oscilación).
- **In**: Entrada de audio para la señal que se va filtrar.
- **HP2**: Salida de audio para la señal del filtro de paso-alto de 2 polos.
- **BP2**: Salida de audio para la señal del filtro de 2 polos paso-banda.
- **LP2**: Salida de audio para la señal del filtro de paso-bajo de 2 polos.
- **HP4**: Salida de audio para la señal del filtro de paso-alto de 4 polos.
- **BH4**: Salida de audio para la señal del filtro de paso-banda/alto de 4 polos.
- **BP4**: Salida de audio para la señal del filtro de paso-banda de 4 polos.



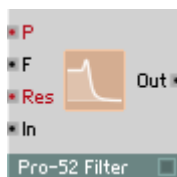
Filtro de 4 polos con salidas de paso-alto (pendiente 24dB/octava), paso-banda (12/12dB/octava) y paso-banda/alto (18/6dB/octava); salidas de 2 polos paso-alto y bajo (12dB/octava) y paso-banda (6/6dB/octava); resonancia variable, y control lineal y logarítmico de la frecuencia de corte.

La ganancia del paso-banda es siempre 1 (0dB), mientras que la ganancia de la frecuencia de corte incrementa con la resonancia. Precaución: Cuando **Res** es casi 1, se generan amplitudes muy altas.

- **P**: Entrada de evento logarítmica para controlar la frecuencia de corte. Valores en semitonos. (69 = 440 Hz).
- **F**: Entrada de audio para el control lineal de la frecuencia de corte. Valor en Hz. **P** y **F** juntas determinan la frecuencia del oscilador.
- **Res**: Entrada de evento para controlar la resonancia/atenuación del filtro. Rango de valores desde 0 (máxima atenuación, no hay resonancia) a 1 (no hay atenuación, máxima resonancia, auto-oscilación).
- **In**: Entrada de audio para la señal que se va filtrar.
- **HP2**: Salida de audio para la señal del filtro de paso-alto de 2 polos.
- **BP2**: Salida de audio para la señal del filtro de paso-banda de 2 polos.
- **LP2**: Salida de audio para la señal del filtro de paso-bajo de 2 polos.
- **HP4**: Salida de audio para la señal del filtro de paso-alto de 4 polos.
- **BH4**: Salida de audio para la señal del filtro de paso-banda/alto de 4 polos.
- **BP4**: Salida de audio para la señal del filtro de paso-banda de 4 polos.

Pro-52 Filter

Filter



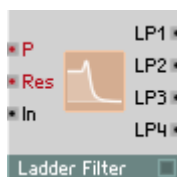
Filtro basado en el sintetizador virtual analógico Pro 52. Es un filtro de paso-bajo (24dB/octava) de 4 polos con resonancia variable y control logarítmico y lineal de la frecuencia de corte.

El filtro llega a la auto-oscilación cuando **Res** se aproxima a 1. La amplitud de la auto-oscilación es aproximadamente 1.

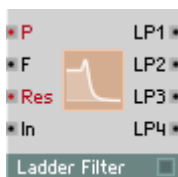
- **P**: Entrada de evento logarítmica para controlar la frecuencia de corte. Valores en semitonos. (69 = 440 Hz).
- **F**: Entrada de audio para el control lineal de la frecuencia de corte. Valor en Hz. **P** y **F** juntas determinan la frecuencia del oscilador.
- **Res**: Entrada de evento para controlar la resonancia/atenuación del filtro. Rango de valores desde 0 (máxima atenuación, no hay resonancia) a 1 (no hay atenuación, máxima resonancia, auto-oscilación).
- **In**: Entrada de audio para la señal que se va filtrar.
- **Out**: Salida de audio para la señal del filtro de paso-paso de 4 polos.

Ladder Filter

Filter



Igual que el **Ladder Filter FM** pero sin la entrada **F** de modulación de frecuencia. Mira la siguiente descripción de módulo.



Filtro basado en el clásico circuito Ladder patentado por Bob Moog. Es un filtro de 4 polos con diferentes salidas de paso-bajo: pendiente 24dB/octava, 18dB/octava, 12dB/octava y 6dB/octava. También tiene resonancia variable y control logarítmico y lineal de la frecuencia de corte.

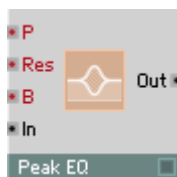
La característica de saturación del circuito analógico se puede simular opcionalmente con **Distortion** en las Propiedades. Cuando esta opción está habilitada, el filtro llega a la auto-oscilación cuando el valor de **Res** es 1 o más. La amplitud de la auto-oscilación es aproximadamente 1 cuando **Res** es 1, pero puede ser mucho mayor con valores mucho más altos de Res.

El filtro también tiene las opciones para seleccionar la calidad de la simulación: **Standard**, **High** y **Excellent**. El efecto es particularmente notable cuando Distortion está habilitado. Por supuesto, cuanta más calidad, más carga de procesamiento en la CPU.

- **P**: Entrada de evento logarítmica para controlar la frecuencia de corte. Valores en semitonos. ($69 = 440$ Hz).
- **F**: Entrada de audio para el control lineal de la frecuencia de corte. Valor en Hz. **P** y **F** juntas determinan la frecuencia del oscilador.
- **Res**: Entrada de evento para controlar la resonancia/atenuación del filtro. Rango de valores desde 0 (máxima atenuación, no hay resonancia) a 1 (no hay atenuación, máxima resonancia, auto-oscilación). Los valores por encima de 1 son posibles en el modo Distortion.
- **In**: Entrada de audio para la señal que se va a filtrar.
- **LP1**: Salida de audio para la señal del filtro de paso-bajo 6dB/octava.
- **LP2**: Salida de audio para la señal del filtro de paso-bajo 12dB/octava.
- **LP3**: Salida de audio para la señal del filtro de paso-bajo 18dB/octava.
- **LP4**: Salida de audio para la señal del filtro de paso-bajo 24dB/octava.

Peak EQ

Filter

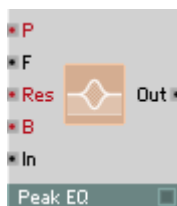


Ecualizador paramétrico con aumento/reducción ajustable, ancho de banda y control logarítmico de frecuencia. Con Peak EQ se puede amplificar o atenuar una frecuencia específica de la señal y una banda más o menos estrecha o ancha de frecuencias alrededor de aquella. Las frecuencias más distantes no se verán afectadas.

- **P:** Entrada de evento logarítmica para controlar la frecuencia de corte. Valores en semitonos. (69 = 440 Hz).
- **Res:** Entrada de evento para controlar la resonancia del filtro (factor Q). Rango de valores desde 0 (resonancia mínima, máximo ancho de banda) hasta 1 (máxima resonancia, mínimo ancho de banda). Si la resonancia está alta, el factor Q del filtro = frecuencia [Hz] / ancho de banda [Hz] $\cong 1 / (2 - 2 \text{ Res})$.
- **B:** Entrada de evento para controlar la ampliación/recorte en dB. Con **B** = 0 la señal queda inalterada.
- **In:** Entrada de audio para la señal que se va a ecualizar.
- **Out:** Salida de audio para la señal ecualizada.

Peak EQ FM

Filter



Ecualizador paramétrico con ampliación/recorte ajustable, ancho de banda y control logarítmico y lineal de la frecuencia. Con Peak EQ se puede amplificar o atenuar una frecuencia específica de la señal y una banda más o menos estrecha o ancha de frecuencias alrededor de aquella. Las frecuencias más distantes no se verán afectadas.

- **P:** Entrada de evento logarítmica para controlar la frecuencia de corte. Valores en semitonos. (69 = 440 Hz).

- **F:** Entrada de audio para el control lineal de la frecuencia de corte. Valor en Hz. **P** y **F** juntas determinan la frecuencia del oscilador.
- **Res:** Entrada de evento para controlar la resonancia del filtro (factor Q). Rango de valores desde 0 (resonancia mínima, máximo ancho de banda) hasta 1 (máxima resonancia, mínimo ancho de banda). Si la resonancia está alta, el factor Q del filtro = frecuencia [Hz] / ancho de banda [Hz] $\cong 1 / (2 - 2 \text{ Res})$.
- **B:** Entrada de evento para controlar la ampliación/recorte en dB. Con **B** = 0 la señal queda inalterada.
- **In:** Entrada de audio para la señal que se va a ecualizar.
- **Out:** Salida de audio para la señal ecualizada.

High Shelf EQ



Filter

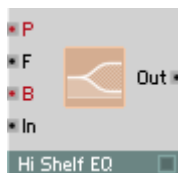
Ecualizador paramétrico con característica High Shelving, ampliación/recorte ajustable, ancho de banda y control logarítmico de la frecuencia.

El nivel de frecuencias por encima de la frecuencia de corte se amplía o reduce a través de la cantidad ajustada. Las frecuencias bajas no se ven afectadas.

- **P:** Entrada de evento logarítmica para controlar la frecuencia de corte. Valores en semitonos. (69 = 440 Hz).
- **B:** Entrada de evento para controlar la ampliación/recorte en dB. Con **B** = 0 la señal queda inalterada.
- **In:** Entrada de audio para la señal que se va a ecualizar.
- **Out:** Salida de audio para la señal ecualizada.

High Shelf EQ FM

Filter



Ecualizador paramétrico con característica High Shelving, con ampliación/recorte ajustable, ancho de banda y control logarítmico y lineal de la frecuencia.

El nivel de frecuencias por encima de la frecuencia de corte se ensalza o reduce a través de la cantidad ajustada. Las frecuencias bajas no se ven afectadas.

- **P:** Entrada de evento logarítmica para controlar la frecuencia de corte. Valores en semitonos. (69 = 440 Hz).
- **F:** Entrada de audio para el control lineal de la frecuencia de corte. Valor en Hz. **P** y **F** juntas determinan la frecuencia del oscilador.
- **B:** Entrada de evento para controlar la ampliación/recorte en dB. Con **B** = 0 la señal queda inalterada.
- **In:** Entrada de audio para la señal que se va a ecualizar.
- **Out:** Salida de audio para la señal ecualizada.

Low Shelf EQ

Filter



Ecualizador paramétrico con característica Low Shelving, ampliación/recorte ajustable, ancho de banda y control logarítmico de la frecuencia.

El nivel de frecuencias por encima de la frecuencia de corte se amplía o reduce a través de la cantidad ajustada. Las frecuencias bajas no se ven afectadas.

- **P:** Entrada de evento logarítmica para controlar la frecuencia de corte. Valores en semitonos. (69 = 440 Hz).
- **B:** Entrada de evento para controlar la ampliación/recorte en dB. Con **B** = 0 la señal queda inalterada.

- **In:** Entrada de audio para la señal que se va a ecualizar.
- **Out:** Salida de audio para la señal ecualizada.

Low Shelf EQ FM

Filter



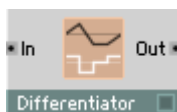
Ecualizador paramétrico con característica Low Shelving, ampliación/recorte ajustable, ancho de banda y control logarítmico y lineal de la frecuencia.

El nivel de frecuencias por encima de la frecuencia de corte se amplía o reduce a través de la cantidad ajustada. Las frecuencias bajas no se ven afectadas.

- **P:** Entrada de evento logarítmica para controlar la frecuencia de corte. Valores en semitonos. ($69 = 440$ Hz).
- **F:** Entrada de audio para el control lineal de la frecuencia de corte. Valor en Hz. **P** y **F** juntas determinan la frecuencia del oscilador.
- **B:** Entrada de evento para controlar la ampliación/recorte en dB. Con **B** = 0 la señal queda inalterada.
- **In:** Entrada de audio para la señal que se va a ecualizar.
- **Out:** Salida de audio para la señal ecualizada.

Differentiator

Filter



El diferenciador entrega la pendiente que hay en la señal de entrada en unidades por milisegundo. El efecto es parecido al filtro de paso-alto donde la amplificación es proporcional a la frecuencia. Ganancia de unidad a 159 Hz.

- **In:** Entrada de audio para la señal que se va a diferenciar.
- **Out:** Salida de audio para la señal diferenciada.



Integrador con entrada Reset. El valor de salida cambia a través de una pendiente dada en la entrada en unidades por milisegundo. El efecto es parecido al filtro de paso-bajo donde la amplificación es proporcional a $1/\text{frecuencia}$. Ganancia de unidad a 159 Hz.

- **In:** Entrada de audio para la señal que se va a integrar.
- **Set:** Entrada de evento para el reajuste. Cuando se recibe un evento, la señal de salida se reajusta al valor del evento.
- **Out:** Salida de audio para a señal integrada.

Delay

REAKTOR posee seis tipos de delay para realizar efectos de retardo. Existen delays Two Tap, dos delays granulares, un difusor (para efectos de reverberación) y el Unit Delay que se puede usar para el procesamiento de loops y el modelado físico.

Single Delay



Delay

Retardo polifónico para señales de evento y audio. La señal de entrada aparece en la salida con un retardo según el tiempo ajustado. El retardo se controla en la entrada **Dly**.

El límite superior para el tiempo de delay se puede configurar en el diálogo de propiedades del módulo en el campo **Max Delay Buffer** (por defecto 1 segundo), y afecta el uso de memoria. El valor que se puede configurar depende del RAM disponible. Para una frecuencia de muestreo de 44,1kHz se necesitan 172 kB de RAM por segundo de buffer y por voz; para un minuto se requieren 10MB. Cuando el módulo se usa como Event-Delay (el puerto In

está rojo indicando que el módulo está en modo de procesamiento de eventos), puedes ajustar **Max Count Of Buffered Events** en el mismo sitio.

Este módulo reemplaza varios módulos de versiones anteriores de REAKTOR:

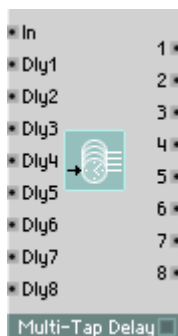
- Se comporta como un **Static Delay** de REAKTOR 3, cuando se conecta una señal de audio a la entrada **In**, y una señal de evento a la entrada **Dly**. Cuando el tiempo de retardo no corresponde a un número entero de samples, la señal de salida se determina mediante interpolación. Esto puede alterar ligeramente el sonido. El método de interpolación se puede seleccionar en el diálogo de propiedades. Una interpolación lineal puede reducir ligeramente el contenido de altas frecuencias.
- Se comporta como un **Modulation Delay** de REAKTOR 3, cuando se conecta una señal de audio a la entrada **In**, y otra a la entrada **Dly**. El retardo se puede modular continuamente mediante una señal de audio en la entrada **Dly**. Cuando el tiempo de retardo no corresponde a un número entero de samples, la señal de salida se determina mediante interpolación. El método de interpolación se puede seleccionar en el diálogo de propiedades. Una interpolación lineal puede reducir ligeramente el contenido de altas frecuencias.
- Se comporta como un **Event Delay** de REAKTOR 3 cuando se conecta una señal de evento a la entrada **In**.

Puertos

- **Dly**: Entrada híbrida para el tiempo de delay en milisegundos.
- **In**: Entrada híbrida para la señal de salida que queremos retardar.
- **Out**: Salida híbrida para la señal que queremos retardar.

Multi-Tap Delay

Delay



Cadena Multi-Tap-Delay para señales de audio. Cuando el tiempo de retardo corresponde a un número no entero de samples, se interpola. El método de interpolación se puede seleccionar en las propiedades.

La salida normalmente se conecta a un mezclador o escáner.

- **In:** Entrada de audio para la señal que queremos retardar.
- **Dly1...8:** Entradas de audio para el control del tiempo de delay en milisegundos. Rango típico: [0...1000].
- **1...8:** Salidas de audio para la señal de entrada retardada. 1 se retarda con el tiempo Dly1, y 2 con Dly2 etc.

Diffuser Delay

Delay



El difusor es un filtro all pass que contiene un elemento de retardo con feed-back. El efecto es una difusión de la señal de entrada sin enfatizar determinadas frecuencias. La aplicación típica es la de efectos de reverberación. A este efecto se ponen en serie varios módulos de este tipo y se les asignan distintos tiempos de delay en milisegundos.

El retardo se puede modular continuamente mediante una señal de audio en la entrada **Dly**. Cuando el tiempo de retardo no corresponde a un número entero de samples, la señal de salida se determina mediante interpolación. Esto puede alterar ligeramente el sonido. El método de interpolación se puede seleccionar en el diálogo de propiedades. La interpolación lineal podría reducir

componentes en altas frecuencias del sonido. El grado de acople interno se ajusta con la entrada **Dffs**. Ajustando a cero en el tiempo de delay, el Difusor funciona como filtro all pass de 1 polo. Así se puede construir un Phaser, poniendo en serie varios difusores, y modulando el parámetro **Dffs**. El límite superior para el tiempo de delay se puede configurar en el diálogo de propiedades del módulo (normalmente 200 ms), y afecta el uso de memoria.

- **Dly**: Entrada híbrida para el tiempo de delay en milisegundos.
- **Dffs**: Entrada de evento para controlar el coeficiente de difusión. Rango de valores:-1...1. Dffs=0: el módulo es delay puro, Dffs = 1: salida = entrada, Dffs= -1:salida = -entrada. Los valores más útiles para Dffs están cerca de 0.5.
- **In**: Entrada para la señal de audio que queremos someter a difusión.
- **Out**: Salida para la señal de audio de difusión..

Delay Granular



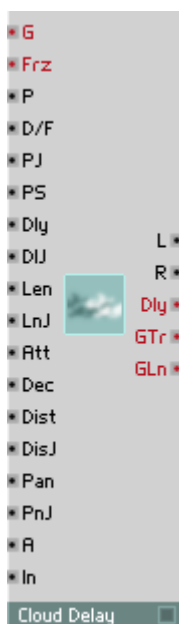
Delay

Elemento de retardo y Pitch-Shifter para señales de audio. La señal de entrada aparece en las salidas con un retardo según el tiempo ajustado en la entrada **Dly**, transportado por el valor presente en la entrada **P** en semitonos. La señal de entrada se divide en partículas de sonido cuyo tamaño se controla mediante la entrada *Granularity* (**Gr**). La entrada *Smoothness* (**Sm**) permite controlar la aspereza del sonido. La posición de las partículas de sonido en el espacio estéreo se define mediante la entrada **Pan**.

El tiempo de delay de **Grain Delay** se puede variar sin afectar la tonalidad. En combinación con generadores aleatorios y de ruido se pueden lograr efectos interesantes.

En el diálogo propiedades se puede configurar la calidad de reproducción y también el límite superior del tiempo de delay. El tiempo de delay real máximo disponible puede diferir del valor configurado en hasta 50%.

- **P:** Entrada de audio logarítmica para controlar el transporte en semitonos (pitch).
- **Dly:** Entrada de audio para controlar el tiempo de delay en milisegundos.
- **Gr:** Entrada de audio para controlar la granularidad del proceso de resíntesis, en milisegundos. Este parámetro define el tamaño de las partículas de sonido usadas para la resíntesis.
- **Sm:** E Entrada de audio para controlar la **Smoothness** (suavidad) del proceso de resíntesis. Se ejerce influencia sobre la formulación de las partículas de sonido. Valores pequeños suelen dar un resultado sonoro más áspero.
- **Pan:** Entrada de audio para controlar la posición en el espacio estéreo (-1=izquierda, 0=centro, 1=derecha).
- **A:** Entrada de audio que controla la amplitud de salida.
- **In:** Entrada para la señal que queremos retardar..
- **L:** Salida de audio para el canal izquierdo del delay..
- **R:** Salida de audio para el canal derecho del delay..
- **Dly:** Salida de evento polifónica en la que está presente el tiempo de delay actual. Esta salida produce un evento siempre cuando se genera una nueva partícula de sonido.



- El módulo Grain Cloud Delay es muy similar al módulo Grain Cloud de la sección del sampler. La diferencia reside en que el módulo Grain Cloud trabaja con samples en el RAM, mientras que el módulo delay correspondiente procesa el buffer de audio en constante variación que es alimentado por el puerto de entrada del módulo (etiqueta In).
- **Trig:** Entrada de evento para disparar el próximo . Valores > 0 inician de inmediato el próximo . El valor = -1 suprime el próximo (ver entrada Dist).
- **Frz:** Entrada de evento para congelar el buffer de audio. Valores > 0 congelan el buffer
- **P:** Entrada de audio para el control logarítmico de la tonalidad (en semitonos). La tonalidad es independiente de la velocidad de reproducción. Rango típico: [-20...20].
- **D/F:** Entrada de audio para el control de la dirección de reproducción, cuando está conectado P. Cuando no, sirve a los efectos del control de frecuencia. El buffer de audio se reproduce en su tonalidad original, cuando F=1, y al revés, cuando F=-1. Rango típico: [-4...4], por defecto: 1.

- **PJ:** Entrada de audio para desviaciones aleatorias de la tonalidad (pitch jitter) en semitonos. Rango típico: [0...3].
- **PS:** Entrada de audio para el control logarítmico del offset de la tonalidad (pitch shift) del actual en semitonos. Rango típico: [-3...3].
- **Dly:** Entrada de audio para el control del tiempo de delay en ms. Rango permitido: [0...duración del buffer].
- **DIJ:** Entrada de audio para desviaciones aleatorias del tiempo de delay. Rango permitido: [0...duración del buffer].
- **Len:** Entrada de audio para la configuración de la duración del gránulo en ms. Rango típico: 30 ms.
- **LnJ:** Entrada de audio para el control de la desviación aleatoria de la longitud (Length Jitter) en ms. Rango típico: 0 ms.
- **Att:** Entrada de audio para la configuración del tiempo de attack. Rango: 0.2.
- **Dec:** Entrada de audio para la configuración del tiempo de decay. Rango: 0.2.
- **Dist:** Entrada de audio para la configuración del tiempo delta para el tiempo hasta el inicio del próximo gránulo en ms. Rango: 20.
- **DisJ:** Entrada de audio para el control de la desviación aleatoria del delta tiempo (Delta time Jitter). Rango: 20 ms.
- **Pan:** Entrada de audio para colocar la panorámica en el campo estéreo. Rango -1 a 1.
- **PnJ:** Entrada de audio para ajustar la desviación aleatoria. Rango 0 a 1.
- **A:** Entrada de audio para el control de la amplitud. Rango : por defecto: 1.
- **In:** Entrada de audio para la señal de audio a retardar.
- **L:** Salida polifónica para el canal estéreo izquierdo.
- **R:** Salida polifónica para el canal estéreo derecho.
- **Dly:** Salida de evento polifónica para el tiempo de delay en cada inicio de .
- **Gtr:** Salida de evento para disparar un gránulo. Entrega 1 cuando se inicia un y 0 cuando se detiene.



Retarda una señal de audio por el tiempo de un sample de audio (1/frecuencia del sample). Cada estructura que usa cualquier tipo de feedback, siempre tiene integrado un unit delay en algún punto del bucle del feedback. Cuando no se usó explícitamente un módulo de unit delay, el programa agrega por sí sólo un delay en algún punto del bucle.

- **In:** Entrada para la señal a retardar por un sample de audio.
- **Out:** Salida para la señal a retardar.

Audio Modifier

Los módulos de procesamiento de audio de Reaktor generan distintos tipos de distorsión (por ejemplo Shaping, Clipping y Mirroring). Además existen los módulos Slew Limiter, Peak Detector, Sample and Hold, y módulos Frequency Divider.

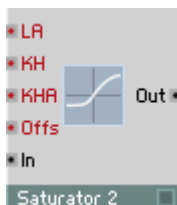


Distorsión con una entrada-salida ajustada para la transición suave hacia la saturación. El valor de salida se limita a ± 2 (para valores entrantes mayores de ± 4). Los valores muy pequeños no varían.

- **In:** Entrada de audio para la señal que será saturada
- **Out:** Salida de audio para la señal saturada.

Saturator 2

Audio Modifier



Saturator 2 es un saturador asimétrico, parabólico de cuarto orden que permite el acceso a la curva de saturación

- **LA:** Asimetría del nivel. Con $LA = 0$, los niveles de saturación para señales positivas y negativas son idénticas. Con $LA > 0$ se reduce el nivel positivo. Con $LA = 1$, es 0. Con $LA < 0$ se reduce el nivel negativo. Con $LA = -1$ es 0.
- **KH:** Knee Hardness. Con $KH = 0$, la saturación aumenta lo más suave posible. El rango completo entre 0 y el nivel de saturación se usa para la curva redondeada. Con valores ascendentes para KH, el rango de la curva se reduce a $(1 - KH)$ del nivel de saturación. Con $KH = 1$, la señal es cortada de golpe en el nivel de saturación.
- **KHA:** Asimetría de Knee Hardness. Con $KHA = 0$, la knee hardness es idéntica para señales positivas y negativas. Con $KHA > 0$, la knee hardness se reduce para señales positivas. Con $KHA = 1$ es 0. Con $KHA < 0$, la knee hardness se reduce para señales negativas. Con $KHA = -1$ es 0.
- **Offs:** Esta entrada le agrega un offset a la señal de entrada y la desplaza en relación a la curva de saturación. Para una señal cero, el offset en la salida es compensado completamente.
- **In:** Entrada para la señal que queremos distorsionar.
- **Out:** Salida para la señal distorsionada.

Clipper

Audio Modifier



Distorsión con limitación dura y ajustable de recorte superior e inferior. Cuando la señal de entrada excede el valor máximo, es limitada a este valor (Clipping).

Cuando cae por debajo del valor mínimo, es limitada a ese valor. Los valores de señales intermedios se mantienen inalterados

- **Max:** Entrada de audio para controlar el valor máximo de la señal
- **Min:** Entrada de audio para controlar el valor mínimo de la señal.
- **In:** Entrada de audio para la señal que queremos limitar
- **Out:** Salida de audio para la señal limitada

Mod. Clipper



Audio Modifier

Distorsión con curva de limitación dura y límite variable. Cuando el valor absoluto de la señal de entrada excede el valor máximo, es limitada a este valor. Los valores de señales menores se mantienen inalterados.

- **M:** Entrada de audio para controlar el valor máximo de la señal.
- **In:** Entrada de audio para la señal que queremos limitar.
- **Out:** Salida de audio para la señal limitada.

Mirror 1 Level



Audio Modifier

Distorsión por reflejo de valor de la señal con nivel ajustable del reflejo. Los valores de señal que están por encima del valor del espejo, son “reflejados” en el mismo, de manera que serán menores. Valores menores se mantienen inalterados

- **Max:** Entrada de audio para controlar el nivel de reflejo
- **In:** Entrada de audio para la señal que vamos a modificar
- **Out:** Salida de audio para la señal modificada

Mirror 2 Levels



Audio Modifier

Distorsión por doble reflejo del valor de la señal con nivel ajustable del reflejo. Los valores de señal que están sobre el valor del espejo, son reflectados hacia abajo. Valores de señal que están por debajo del valor del espejo inferior, son reflectados hacia arriba. Los valores de señales intermedios se mantienen inalterados.

- **Max:** Entrada de audio para controlar el valor superior del espejo.
- **Min:** Entrada de audio para controlar el valor inferior del espejo.
- **In:** Entrada de audio para la señal que vamos a modificar
- **Out:** Salida de audio para la señal modificada.

Chopper



Audio Modifier

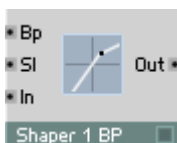
Modulador chopper que alterna la amplificación de la señal de entrada entre un valor variable y 1.

Cuando la señal de modulación es positiva, el valor de entrada es multiplicado por X, y con la señal de modulación negativa, el valor de entrada queda inalterado.

- **M:** Entrada de audio para la señal de modulación (sólo es importante el signo).
- **X:** Entrada de audio para controlar el factor de amplificación con **M** positivo.
- **In:** Entrada de audio para la señal a modular.
- **Out:** Salida de audio para la señal modulada

Shaper 1 BP

Audio Modifier

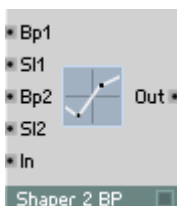


Modelador parcial de señal de curva de entrada/salida con puntos de corte. La pendiente de la curva por encima de los puntos de corte se puede ajustar. Los valores de señales de entrada por debajo de los puntos de corte se mantienen inalterados.

- **Bp:** Entrada de audio para controlar el valor de los puntos de corte
- **Sl:** Entrada de audio para controlar la pendiente de la parte superior de la curva (1= sin alteración de señal).
- **In:** Entrada de audio para la señal a modelar.
- **Out:** Salida de audio de la señal modulada.

Shaper 2 BP

Audio Modifier



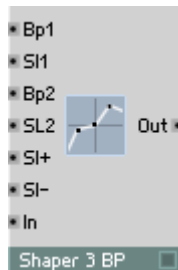
Modelador parcial de señal de curva de entrada/salida con dos puntos de corte.

La pendiente de la curva sobre el punto de inflexión superior y por debajo del inferior se pueden ajustar. Los valores de señales de entrada entre los puntos de corte se mantienen inalterados

- **Bp1:** Entrada de audio para controlar los puntos de corte superiores.
- **Sl1:** Entrada de audio para controlar la pendiente de la parte superior de la curva de entrada/salida (1 = la señal no varía).
- **Bp2:** entrada de audio que controla el nivel de los puntos de corte inferiores.
- **Sl2:** Entrada de audio que controla la pendiente de la parte inferior de la curva de entrada/salida (1 = no varía la señal).

- **In:** Entrada de audio para la señal a modular
- **Out:** Salida de audio para la señal modelada.

Shaper 3 BP



Audio Modifier

Modelador parcial de señal de curva de entrada/salida con tres puntos de corte.

La pendiente de la curva sobre el punto de inflexión superior y por debajo del inferior y el punto cero se pueden ajustar.

Los valores de la señal entre los puntos de corte no varían.

- **Bp1:** Entrada de audio para controlar el nivel superior de los puntos de corte.
- **Sl1:** Entrada de audio para controlar la pendiente de la parte superior de la curva de entrada/salida.
- **Bp2:** Entrada de audio para controlar el nivel inferior de los puntos de corte.
- **SL2:** Entrada de audio para controlar la pendiente de la parte inferior de la curva de entrada/salida. (1 = no varía la señal).
- **SI+:** Entrada de audio para controlar la pendiente de la curva de entrada/salida entre cero y el punto de inflexión superior. (1 = no varía la señal).
- **SI-:** Entrada de audio para controlar la pendiente de la curva de entrada/salida entre el punto de inflexión inferior y cero. (1 = no varía la señal).
- **In:** Entrada de audio para la señal que será modelada.
- **Out:** Salida de audio para la señal modelada.

Shaper Parabolic



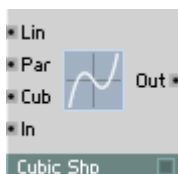
Audio Modifier

Modelador de señal con curva de entrada/salida parabólica (polinomio de segundo orden).

Las partes lineal y cuadrada de la señal se pueden ajustar. ($\text{Out} = \text{Lin In} + \text{Par In}^2$).

- **Lin:** Entrada de audio para controlar el componente lineal, no distorsionado.
- **Par:** Entrada de audio para controlar el componente cuadrado ,distorsionado.
- **In:** Entrada de audio para la señal que será modelada.
- **Out:** Salida de audio para la señal modelada.

Shaper Cubic



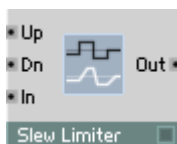
Audio Modifier

Modelador de señal con curva de entrada/salida cúbico-parabólica (polinomio de tercer orden).

Los componentes lineal, cuadrado y cúbico de la señal se pueden ajustar ($\text{Out} = \text{Lin In} + \text{Par In}^2 + \text{Cub In}^3$).

- **Lin:** Entrada de audio para controlar el componente lineal no distorsionado.
- **Par:** Entrada de audio para controlar el componente cuadrado distorsionado
- **Cub:** Entrada de audio para controlar el componente cúbico distorsionado
- **In:** Entrada de audio para la señal que vamos a modelar.
- **Out:** Salida de audio para la señal modelada.

Slew Limiter



Audio Modifier

Limitador con tasa máxima (slew) ajustable por separado para direcciones positivas y negativas. La señal de salida sigue la señal de entrada. En variaciones rápidas de la señal y saltos, la salida sólo sigue con una pendiente (rampa) limitada hasta volver a alcanzar la señal de entrada

- **Up:** Entrada de audio para controlar la tasa máxima (en 1/sec) para señales en ascenso
- **Dn:** Entrada de audio para controlar la tasa máxima para señales en descenso. Se trata de unidades por segundo.
- **In:** Entrada de audio para la señal que vamos a limitar
- **Out:** Salida de audio para la señal limitada.

Peak Detector



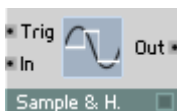
Audio Modifier

Detector de amplitud de picos. La señal entrante es rectificada y suavizada con un tiempo de relajación ajustable. El tiempo de ataque es cero.

El efecto del Peak Detector consiste en que el valor de salida sigue a la envolvente de la amplitud de la señal de entrada. Usa este módulo como rastreador de envolventes .

- **Rel:** Entrada de evento logarítmico para controlar el tiempo de relajación. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (i.e. in $\text{dB}_{1\text{ms}}$).
- **In:** Entrada de audio para la señal que vamos a rectificar
- **Out:** Salida de audio para la señal.

Sample & Hold



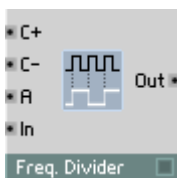
Audio Modifier

Elemento de muestreo y retención con entrada de reloj.

Cuando la señal de reloj aumenta a más de 0, el valor de entrada actual se dirige a la salida y se retiene hasta el próximo pulso de reloj. Así, en la salida resulta una forma de onda escalonada

- **C:** Entrada de audio para la señal de reloj. La entrada es muestreada al llegar a este margen.
- **In:** Entrada de audio para la señal que será muestreada.
- **Out:** Salida de audio para la señal muestreada.

Frequency Divider

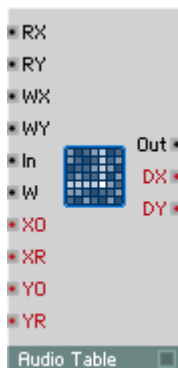


Audio Modifier

Divisor de frecuencias (suboscilador de pulso) con duraciones de nivel altas y bajas ajustables independientemente.

Una onda de pulso es generada al contar el punto de cruce cero de la señal entrante. La frecuencia de la salida de la forma de onda será una fracción de la frecuencia de la entrada de la forma de onda. El ratio de la frecuencia puede ser ajustado ($f_{out} = 2 f_{in} / (C+ + C-)$). Cuando **C+** y **C-** se eligen diferentes, resulta una forma de onda asimétrica.

- **C+:** Entrada de control para la cantidad de cruces por cero de la señal de entrada durante la fase alta de la señal de salida.
- **C-:** Entrada de control para la cantidad de cruces por cero de la señal de entrada durante la fase baja de la señal de salida.
- **A:** Entrada de control para la amplitud. La señal de salida oscila entre **+A** y **-A**.
- **In:** Entrada de audio para la señal que se va a dividir
- **Out:** Salida de audio para la señal de frecuencia dividida.



Contiene una tabla de valores. La tabla puede ser leída como señal de audio; se puede guardar audio en la tabla; y el contenido de la tabla se puede visualizar y editar gráficamente. La tabla se puede usar unidimensionalmente (una serie de valores representados por X), o bidimensionalmente (columnas representadas por X e Y).

El valor entregado está definido por la posición de lectura determinada por las entradas **RX** y **RY**. Una señal presente en la puerto **In** del módulo se guarda en celdas individuales de la tabla acorde a la posición de escritura determinada por las entradas **WX** y **WY**.

X es la posición horizontal de izquierda a derecha, e Y la posición vertical de arriba hacia abajo. El recuento empieza siempre con 0 para el primer elemento.

La visualización en el panel puede mostrar todos los datos, o sólo los de un rango determinado. Muchas opciones en las propiedades permiten al usuario determinar el comportamiento.

Encontrará una descripción completa de las propiedades, menús y atajos de teclas en la sección Los módulos de Tabla a partir de la página - 165

- **RX:** Entrada de audio para la posición X de la celda en la tabla de la que se leen datos.
- **RY:** Entrada de audio para la posición Y de la celda en la tabla de la que se leen datos. Esta se usa en modo 2D o para dirigirse al número de fila cuando hay más de una fila.
- **WX:** Entrada de audio para la posición X de la celda de la tabla a la que se escriben datos.
- **WY:** Entrada de audio para la posición Y de la celda de la Tabla en que los datos son escritos.

- **W:** Entrada de audio para activar el proceso de escritura de la tabla.
- **In:** Entrada de audio para la señal que se escribe en la tabla. Cuando el valor **W** es mayor a 0, el valor presente en In se escribe en la tabla en la posición determinada por **WX** y **WY**
- **XO:** Entrada de evento para el desplazamiento horizontal del rango de datos visualizado. **XO** controla la posición de datos que se visualiza (según la alineación de la vista). El valor de **XO** se especifica en las propiedades en unidades
- **XR:** Entrada de evento para el alcance horizontal de visualización de los datos. **XR** determina cuántas unidades de datos caben en la visualización y permite, por ejemplo, abrir o cerrar el zoom
- **YO:** Entrada de evento para el desplazamiento vertical del rango de datos visualizado. **YO** controla la posición de datos que se visualiza (según la alineación de la vista). El valor de **YO** se especifica en las propiedades en unidades
- **YR:** Entrada de evento para el rango vertical de visualización de los datos en modo 2D. **YR** determina cuántas unidades de datos caben en la visualización y permite, por ejemplo, abrir o cerrar el zoom.
- **Out:** Salida de audio de la lectura de señal desde la Tabla y definida por las entradas **RX**, **RY** y **R**
- **DX:** Salida de evento para la longitud de la fila horizontal de la tabla en unidades
- **DY:** Salida de evento para la altura de la columna vertical de la tabla en unidades

Event Processing

Los módulos de evento se usan como contadores en operaciones lógicas, para dividir y combinar señales y para medir el tiempo. También encontrarás módulos para dar forma y randomizar señales de control. Finalmente, hay un módulo Table completo en cuanto a características equivalente al Audio Table dentro de la sección de osciladores.

Accumulator



Event Processing

Acumulador para valores de eventos (suma). El valor de cada evento en la entrada se añade al valor total almacenado en el módulo. El valor de la suma actualizada se envía como evento a través de la salida.

- **In:** Entrada para los eventos que se van a acumular.
- **Set:** Entrada de evento para (re)ajustar la suma interna. El valor de un nuevo evento en esta entrada determina el nuevo valor de la suma interna.
- **Out:** Salida de evento para el valor total acumulado.

Counter

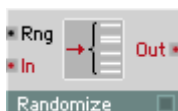


Event Processing

Contador controlado por eventos. Un evento positivo en la entrada adecuada incrementa o reduce el valor de salida en uno.

- **Up:** Entrada para contar en incremento. Sólo los eventos de valor positivo, no cero, tienen efecto incrementando el valor de salida en uno.
- **Dwn:** Entrada para contar en reducción. Sólo los eventos con valor positivo, no cero, tienen efecto reduciendo el valor de salida en uno.
- **Set:** Entrada de evento para (re)ajustar el valor del contador. El valor de un evento en esta entrada determina el nuevo valor del contador.
- **Out:** Salida de evento para el valor del contador.

Randomizer



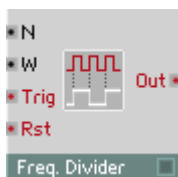
Event Processing

Randomizador de eventos con ancho de difusión ajustable.

Los eventos entrantes se modifican con una aleación positiva o negativa según el alcance (**Out** = **In** + X, donde $-\mathbf{Rng} \leq X \leq \mathbf{Rng}$).

- **Rng**: Entrada de audio para controlar el alcance de la modificación aleatoria de la señal.
- **In**: Entrada de evento para la señal que se va a modificar.
- **Out**: Salida de evento para la señal modificada.

Frequency Divider



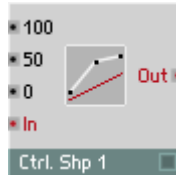
Event Processing

Divisor de frecuencias para señales de evento. Tras una cantidad de eventos de input dependiente del ancho de pulso W, la señal de salida vuelve a cero.

- **N**: Entrada de control para el número de eventos de entrada por período de salida. ($N < 2$: sin división, $2 \dots 2.99$: factor de división 2, $3 \dots 3.99$: factor de división 3, etc.).
- **W**: Entrada de control para el ancho de pulso de la señal de salida. Rango de valores: $0 \dots 1$ (0% ... 100%). **W** = 0 : la señal de puerta en **Out** regresa a cero con el próximo evento en **In**, **W** = 0.5 : la señal de puerta en **Out** regresa a cero tras la mitad del período, **W** = 1 : la señal de puerta en **Out** está en On todo el tiempo.
- **In**: Entrada para la señal de evento (reloj) que se va a dividir en frecuencias (por ejemplo, pulsos MIDI Clock. Sólo los eventos con un valor positivo, no cero, tienen efecto.
- **Rst**: Entrada de evento para re-ajustar el contador interno para forzar un inicio sincronizado (conecta a una señal MIDI Start si usas el **Event Freq. Divider** para sincronización MIDI). Requiere un evento con valor positivo, no cero.
- **Out**: Salida de evento para la señal dividida en frecuencias.

Ctrl. Shaper 1 BP

Event Processing

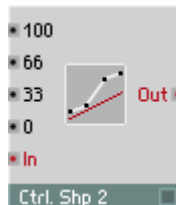


Formador de señal de evento con curva parcialmente lineal de entrada y salida y punto de inflexión fijo. La salida se produce a través de la interpolación lineal entre los puntos dados.

- **100:** Valor de salida en **In** = 1 (100%).
- **50:** Valor de salida del punto de inflexión en **In** = 0.5 (50%).
- **0:** valor de salida en **In** = 0 (0%).
- **In:** entrada de evento para la señal que se va a formar.
- **Out:** Salida de evento para la señal formada.

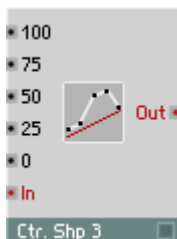
Ctrl. Shaper 2 BP

Event Processing



Formador de señales de evento con curva parcialmente lineal de entrada /salida y dos puntos de inflexión fijos. La salida se produce a través de la interpolación lineal entre los puntos dados.

- **100:** Valor de salida en **In** = 1 (100%).
- **66:** Valor de salida en el punto de inflexión superior en **In** = 0.66 (66%).
- **33:** Valor de salida en el punto de inflexión inferior en **In** = 0.33 (33%).
- **0:** Valor de salida en **In** = 0 (0%).
- **In:** Entrada de evento para la señal que se va a formar.
- **Out:** Salida de evento para la señal formada.



Formador de señales de evento con curva parcialmente lineal de entrada/salida y tres puntos de inflexión fijos. La salida se produce a través de la interpolación lineal entre los puntos dados.

- **100:** Valor de salida en **In** = 1 (100%).
- **75:** Valor de salida para el punto de inflexión superior en **In** = 0.75 (75%).
- **50:** Valor de salida para el punto de inflexión medio en **In** = 0.5 (50%).
- **25:** Valor de salida para el punto de inflexión inferior en **In** = 0.25 (25%).
- **0:** Valor de salida en **In** = 0 (0%).
- **In:** Entrada de evento para la señal que se va a formar.
- **Out:** Salida de evento para la señal formada.

Logic AND

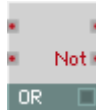
Event Processing



Puerta lógica para señales de evento. La salida es la combinación lógica AND de los dos valores de entrada, es decir, la salida es 1 cuando ambas señales son positivas, si no, la salida es 0.

La entrada trata los valores cero y por debajo como estado lógico Falso; los valores por encima de cero, como estado lógico Verdadero.

Logic OR



Event Processing

Puerta lógica para señales de evento. La salida es la combinación lógica OR de los dos valores de entrada, es decir, la salida es 1 cuando una o ambas señales son positivas, si no, la salida es 0.

La entrada trata los valores cero y por debajo como estado lógico Falso; los valores por encima de cero, como estado lógico Verdadero.

Logic EXOR



Event Processing

Puerta lógica para señales de evento. La salida es la combinación lógica EXOR de los dos valores de entrada, es decir, la salida es 1 cuando una de las señales, pero no ambas, son positivas, si no, la salida es 0. Así, el efecto es que una de las entradas invierte el valor lógico de la otra.

La entrada trata los valores cero y por debajo como estado lógico Falso; los valores por encima de cero, como estado lógico Verdadero.

Logic NOT

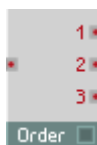


Event Processing

Puerta lógica para señales de evento. La salida es la negación lógica de la señal de entrada, es decir, la salida es 0 cuando la entrada es positiva, y 1 cuando es negativa.

La entrada trata los valores cero y por debajo como estado lógico Falso; los valores por encima de cero, como estado lógico Verdadero.

Order

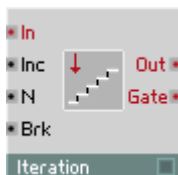


Event Processing

Orden de eventos. Un evento que llega a la entrada se transmite con el mismo valor en todas las salidas, pero en un orden definido: primero va a **1**, luego a **2** y finalmente a **3**. Los eventos recorren TODOS los módulos en la cadena conectada a **1**, antes de ir al primer módulo conectado a **2**, etc.

- **In:** Entrada para eventos que se van a reenviar en un orden definido.
- **1:** El evento se envía primero por esta salida.
- **2:** El evento se envía después por esta segunda salida.
- **3:** El evento se envía después por esta tercera salida.

Iteration



Event Processing

Un evento que llega a la entrada **Trg** pasa a la salida y activa una serie de eventos de salida **N** adicionales. Cada evento subsiguiente tiene el valor de su predecesor incrementado por el valor de la entrada **Inc**. Todos los eventos se envían antes de procesar el siguiente sample de audio. Este módulo se puede usar cuando necesites un evento interactivo de cálculo. Ayuda a evitar la construcción de loops de eventos que pueden hacer inestables las operaciones de Reaktor.

- **Trg:** Entrada activadora. Un evento en esta entrada activa al evento N+1 en la salida.
- **Inc:** Incremento del valor de cada evento subsiguiente.
- **N:** Número de eventos de salida adicionales. El valor ha de ser mayor o igual que el número entero (las décimas se cortarán).

Separator



Event Processing

Separación de eventos. Todos los eventos recibidos se comparan con el valor de threshold y se envían a una u otra salida.

Un buen uso del separador sería convertir una señal de puerta en una señal activadora filtrando los eventos cero (**Thld** = 0, **Hi** = salida de activación).

- **Thld**: Entrada de audio de control para el valor de threshold.
- **In**: Entrada para los eventos que se van a separar y enviar a las dos salidas.
- **Hi**: Salida para aquellos eventos cuyo valor es mayor que el nivel de threshold.
- **Lo**: Salida para aquellos eventos cuyo valor sea menor que el nivel de threshold.

Value



Event Processing

Reemplazador de valores para eventos. Los eventos que llegan a la entrada **In** reemplazan su valor por el valor actual en la entrada **Val**, y luego se envían con el nuevo valor a la salida.

Este módulo se puede usar también como módulo sample&hold controlado por eventos.

- **In**: Entrada para los eventos cuyo valor va a reemplazarse.
- **Val**: Entrada para especificar el nuevo valor de los eventos.
- **Out**: Salida de evento para los eventos cambiados de valor.

Merge



Event Processing

Unificador para señales de evento. Cuando se conecta más de un cable en la entrada, el valor de salida es el del último evento de entrada recibido, independientemente de dónde provenga ese cable.

El módulo tiene administración dinámica de los puertos de entrada. El número de puertos de entrada se puede definir con **Min Num Port Groups** en la página **Function** de las propiedades.

Al contrario que las versiones anteriores de REAKTOR, los eventos subsiguientes con el mismo valor, no se filtran. Para ello, usa el módulo **Step Filter**.

Step Filter



Event Processing

Un evento sólo pasa si el valor de entrada es mayor/menor que el valor de entrada anterior +/- tolerancia.

- **In:** Entrada para los eventos que se van a filtrar.
- **Tol:** Entrada para el nivel de tolerancia.
- **Out:** Salida de evento.

Router M->1



Event Processing

Ruteador de múltiples entradas a una salida. Los eventos de la entrada seleccionada podrán pasar, mientras que otros serán filtrados. Las entradas se seleccionan en la entrada **Pos**. Cuando se ha seleccionado el modo **Wrap** en las propiedades, **Pos** trabaja como bucle, y así, Max+1 es igual que 0, Max+2 es 1, etc.

El módulo tiene administración dinámica de los puertos de entrada. El número de puertos de entrada se puede definir con **Min Num Port Groups** en la página **Function** de las propiedades.

- **Pos:** Entrada para seleccionar la salida por la que pasar la señal de entrada.
- **In:** Entrada de señal.
- **Out:** Salida para la señal de entrada seleccionada.

Router 1,2



Event Processing

Interruptor de On/Off o conmutador para señales de evento, con entrada de control. El último evento que pasa se retiene en la salida incluso aunque esté en Off.

El módulo tiene administración dinámica de los puertos de entrada. El número de puertos de entrada se puede definir con **Min Num Port Groups** en la página **Function** de las propiedades entre 1 y 2.

- **Ctrl:** Entrada híbrida para el control del interruptor. Mientras el valor sea mayor que cero, todos los eventos de entrada pasarán a la salida.
- **In:** Entrada de evento para la señal a la que se va aplicar el interruptor.
- **Out:** Salida de evento para la señal con interruptor. Mientras **Ctrl** sea menor que cero, el último valor se retendrá en la salida.

Router 1->M



Event Processing

Ruteador de múltiples entradas a una de salida. Los eventos de la entrada pasarán a la salida seleccionada. La salida se selecciona con la entrada Pos. Cuando el modo Wrap está seleccionado en la Propiedades, Pos trabajará en bucle, y así, Max +1 es igual a cero; Max+2 es 1, etc.

- El módulo tiene administración dinámica de los puertos de entrada. El número de puertos de entrada se puede definir con **Min Num Port Groups** en la página **Function** de las Propiedades.

- **Pos:** Entrada para seleccionar la salida por la que va a pasar la señal de entrada.
- **In:** Entrada de señal.
- **Out:** Salida para la señal de entrada.

Timer



Event Processing

Se mide el tiempo transcurrido entre los dos últimos eventos recibidos, y sale en forma de evento. También, se calcula y se envía la frecuencia cuyo período de oscilación es igual a la duración medida.

- **In:** Entrada polifónica para los eventos cuya distancia en tiempo se va a medir.
- **F:** Entrada de evento polifónica para la frecuencia de los eventos de entrada en Hz.
- **T:** Salida de evento polifónica para el tiempo entre los eventos en milisegundos.

Hold

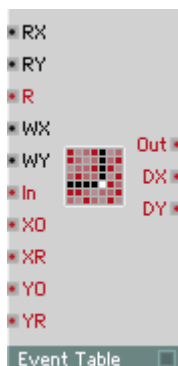


Event Processing

Envolvente de retención.

Cuando el módulo se activa a través de un evento positivo en la entrada **G**, el valor del evento se retiene como valor de salida hasta que ha pasado el tiempo de Hold. Luego la salida salta a cero. El módulo se puede activar en cualquier momento, incluyendo la re-activación durante el tiempo de Hold.

- **G:** Entrada de evento para activar la envolvente. Sólo los eventos con valores positivos (no cero) tendrán efecto.
- **H:** Entrada para controlar el tiempo de Hold en milisegundos.
- **Out:** Salida de evento para la señal de la envolvente.



Retiene una tabla con datos de valores. Los valores de los eventos se pueden leer desde la tabla; los valores de los eventos se pueden guardar en la tabla, y el contenido de la tabla se puede representar y editar gráficamente. La tabla puede ser unidimensional (una fila de valores) representados por X, o bidimensional (una serie de filas y columnas, o un grupo de filas independientes) representadas por X e Y.

El valor de la salida se toma desde la tabla leyendo la posición dada por las entradas **RX** y **RY**. Los valores que llegan a la entrada **W** del módulo se almacenan en celdas individuales de la tabla de acuerdo con la posición del cable dada en las entradas **WX** y **WY**.

X es la posición horizontal de izquierda a derecha, e Y es la posición vertical de arriba abajo. La cuenta siempre empieza en 0 para el primer elemento.

El display del panel del módulo puede mostrar todos los datos o sólo parte de ellos. Hay muchas opciones en las propiedades que te permitirán hacerlo a tu medida.

Para más detalles sobre las propiedades, menús y atajos de teclas en *módulos Table* en el capítulo 22.

- **RX**: Entrada de audio para la posición X de un celda de la tabla desde la que se leen los datos.
- **RY**: Entrada de audio para la posición Y de una celda de la tabla desde la que se leen los datos. Se usa en el modo 2D o para dirigirse al número de fila si hay más de una.
- **R**: Entrada de evento para activar la operación de lectura de la tabla en la posición dada por **RX** y **RY**. Cada evento produce un evento en la salida **Out**.

- **WX:** Entrada de audio para la posición X de una celda de la tabla en la que se van a escribir los datos.
- **WY:** Entrada de audio para la posición Y de una celda de la tabla en la que se van a escribir los datos.
- **In:** Entrada de evento para escribir dentro de la tabla en la posición dada por **WX** y **WY**. El valor de los datos escritos en la celda de la tabla es el valor del evento que llega a la entrada **In**.
- **XO:** Entrada de evento para el desplazamiento horizontal de la región de datos representada. **XO** controla la posición de datos que aparece en el display (de acuerdo con View Alignment). El valor de **XO** está en las unidades especificadas en las propiedades.
- **XR:** Entrada de evento para el alcance horizontal de la región de datos representada. **XR** controla la cantidad de unidades de datos encajados en el display, es decir, te permite hacer zoom dentro del área.
- **YO:** Entrada de evento para el desplazamiento vertical de la región de datos representada. **YO** controla la posición de datos que aparece en el display (de acuerdo con View Alignment). El valor de **YO** está en las unidades especificadas en las propiedades.
- **YR:** Entrada de evento para el alcance vertical de la región de datos representada en el modo 2D. **YR** controla la cantidad de unidades de datos encajados en el display, es decir, te permite hacer zoom dentro de los datos.
- **Out:** Salida de evento para los datos desde la celda controlada por las entradas **RX**, **RY** y **R**.
- **DX:** Salida de evento para el tamaño de las filas horizontales de la tabla en unidades.
- Salida de evento para el tamaño de las columnas verticales de la tabla en unidades.

Auxiliary

Si hay algo que todavía no has encontrado, seguramente estará aquí. Primero encontrarás la plataforma para grabar y reproducir archivos de audio. Luego, hay módulos para la administración de voces polifónicas, para convertir señales de audio en señales de control, y para reportar el estado de varios procesos de Reaktor como la afinación o el ajuste de tempo.

Tapedeck 1-Ch



Auxiliary

Módulo Tapedeck de un canal para grabar y reproducir señales de audio. Los archivos de audio se pueden leer y escribir tanto en la memoria como en el disco duro, dependiendo de los ajustes en las propiedades.

En el modo de disco duro, los archivos se escriben dentro de una carpeta que especifiques en las preferencias de Reaktor. Reaktor realiza la conversión de frecuencia de muestreo al instante y escribe el archivo en la frecuencia de muestreo que se haya usado actualmente en tu sistema de audio.

En el modo de memoria, puedes representar la forma de onda del archivo de audio cargado en el panel a través de **Picture** en la tabla **Appearance** de las propiedades. La forma de onda completa del archivo de audio se encajará automáticamente en el tamaño (**Size**) de la imagen que hayas introducido en la tabla **Appearance**.

Modo Memory

Al seleccionar la casilla **Keep audio only in memory** en las propiedades, el módulo Tapedeck comienza a trabajar en modo de memoria y la sección con los botones **Memory**, **Save**, **Save as...** y **Reload** se hace activa.

El tiempo máximo de grabación se ajusta con **Recording Size (sec)** en segundos. El tamaño de este valor depende de la cantidad de RAM disponible en el ordenador. Con una frecuencia de muestreo de 44.1KHz, necesitarás

86KB de RAM para cada segundo de buffer; para un minuto necesitarás 5MB. Si se configura demasiada memoria para el Tapedeck, el modo de trabajo en bucle de la memoria virtual producirá como consecuencia una actividad considerable en el disco duro durante la grabación, lo que puede perjudicar el cálculo de audio en Reaktor.

Los archivos de audio se pueden importar para la reproducción con el botón **Select File...** en las propiedades del módulo Tapedeck. Un archivo que ya se haya importado se puede cargar de nuevo haciendo clic en el botón **Reload**, es decir, si se ha modificado con el editor de samples.

Durante la importación de un archivo de audio, el buffer de audio se adapta a los datos. Si en un momento posterior se quiere realizar una grabación más larga, primero hay que ingresar un valor mayor en **Max Recording Size** (sec) en el diálogo de propiedades del Tapedeck. El archivo de audio cargado automáticamente se adapta a la frecuencia de muestreo actual de Reaktor.

Nota: Ten presente que en el modo de memoria, Reaktor convierte la frecuencia de muestreo de todos los archivos de audio en el Tapedeck a una nueva frecuencia de muestreo cada vez que Reaktor se conecta a una nueva tasa. No cambies la frecuencia de muestreo si tu ensemble contiene samples de audio de la RAM que está usando Tapedeck. Si el archivo de audio se guarda en el disco duro, puedes usar el botón **Reload** después de cambiar la frecuencia de muestreo para importar el archivo de nuevo.

Se puede exportar una grabación usando **Save** en las propiedades. Si el archivo ya ha sido exportado, se te preguntará si quieres sobrescribir el archivo, es decir, si has hecho una nueva grabación en el Tapedeck. Con **Save As...** puedes guardar el archivo con un nuevo nombre.

Modo Harddisk

Al seleccionar la casilla **Stream audio from/to harddisk** en las propiedades, el módulo Tapedeck comienza a trabajar en el modo de disco duro. Los archivos de audio se leen y escriben directamente desde el disco duro. Puedes seleccionar un archivo de audio para la reproducción con el botón **Select File...** Con el botón **New File** crearás un nuevo archivo llamado “untitled” (si ya existe un archivo con ese nombre en tu carpeta de Audio, automáticamente se le añadirá un número adicional). Puedes editar ese nombre directamente en el cuadro de nombre dentro de las propiedades.

Si **Value** está activado en las propiedades, aparecerá un display en el panel con el nombre del archivo. Abriendo el menú contextual de este cuadro, también podrás importar y exportar archivos.

- **Rec:** Entrada de evento para conectar el modo de grabación con On y Off, es decir, con una señal de puerta. El inicio de la grabación sucederá con un evento de valor positivo. El fin de la grabación sucederá con un evento de valor negativo o cero, o cuando se alcance el tiempo máximo de grabación.
- **Play:** Entrada de evento para conectar el modo de reproducción con On y Off, es decir, con un evento de puerta. El inicio de la reproducción sucederá con un evento de valor positivo; el fin de la reproducción sucederá con un evento de valor negativo o cero.
- **Lp:** Entrada de evento para conectar el modo en loop de la reproducción con On y Off. Cuando esta entrada recibe un valor positivo, la reproducción comienza otra vez desde el principio, una vez que ha llegado al final. El resultado es un loop infinito mientras esté conectado el On.
- **In:** Entrada de audio monofónica para la señal que se va a grabar. Valor máximo±1. Para grabar una señal polifónica tendrás que insertar un Voice Combiner.
- **Pse:** Entrada de control para pausa. Un valor mayor que cero activa el modo pausa, otros valores reanudan la reproducción. Rango típico: [0 ... 1].
- **Pos:** Entrada para ajustar la posición de reproducción en ms. Rango típico: [0 ... 20000].
- **Spd:** Velocidad variable de la reproducción. Los valores han de ser mayores que 0. Un valor de 1 significa velocidad normal. Rango típico: [0.8 ... 1.2].
- **Out:** Salida de audio para la señal de reproducción o para la señal que se ha grabado.
- **Rec:** 1 = Tapedeck está grabando, si no, 0. Rango típico: [0 ... 1].
- **Play:** 1 = Tapedeck está reproduciendo, si no, 0. Rango típico: [0 ... 1].
- **Wrp:** Un evento con valor 1 se envía cada vez que, en la reproducción en loop, se alcanza el punto inicial.
- **Pos:** 1 = Tapedeck está en modo de pausa, si no, 0. Rango típico: [0 ... 1].

- **Time:** Posición actual de grabación/reproducción en ms. Rango típico: [0 ... <duración del sample en ms>].
- **Lng:** Duración de la grabación en ms. Rango típico: [0 ... <duración del sample en ms>].

Tapedeck 2-Ch

Auxiliary



Funciona igual que Tapedeck 1-Ch pero tiene dos canales para grabación y reproducción de las señales de audio. Importa y exporta archivos estéreo.

- **In L:** Entrada de audio monofónica para la señal que se va a grabar en el canal izquierdo. Valor máximo ± 1 . Para grabar una señal polifónica tendrás que insertar un Voice Combiner.
- **In R:** Entrada de audio monofónica para la señal que se va a grabar en el canal derecho. Valor máximo ± 1 . Para grabar una señal polifónica tendrás que insertar un Voice Combiner.
- **L:** Salida de audio para la señal de reproducción o de grabación del canal izquierdo.
- **R:** Salida de audio para la señal de reproducción o de grabación del canal derecho.

Audio Voice Combiner



Auxiliary

Combinador de voces de audio. Convierte una señal de audio polifónica en monofónica sumando todas las voces.

- **In:** Entrada de audio polifónica para la señal que se va a convertir en mono.
- **Out:** Salida de audio monofónica para la señal convertida.

Event V.C. All Auxiliary



Combinador de voces de evento. Convierte una señal de evento polifónica en monofónica enviando los eventos de todas las voces a la misma voz monofónica.

- **In:** Entrada de evento polifónica para la señal que se va a convertir en mono.
- **Out:** Salida de evento monofónica para la señal convertida.

Event V.C. Max



Auxiliary

Combinador de voces de evento con selección de valor máximo. Convierte una señal de evento polifónica en monofónica encontrando la voz con el valor actual mayor y enviándola a la salida mono. Para reconocer las voces activas se necesita una entrada de señal de puerta polifónica.

- **In:** Entrada de evento polifónica para la señal cuyo máximo valor se va a entregar.
- **G:** Entrada de evento polifónica para la señal del instrumento polifónico.
- **Out:** Salida de evento monofónica para la señal con el valor máximo.

Event V.C. Min



Auxiliary

Combinador de voces de evento con selección de valor mínimo. Convierte una señal de evento polifónica en monofónica encontrando la voz con el valor actual menor y enviándola a la salida mono. Para reconocer las voces activas se necesita una entrada de señal de puerta polifónica.

- **In:** Entrada de evento polifónica para la señal cuyo mínimo valor se va a entregar.
- **G:** Entrada de evento polifónica para la señal del instrumento polifónico.
- **Out:** Salida de evento monofónica para la señal con el valor mínimo.

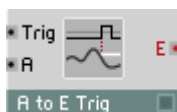
A to E



Auxiliary

Convertidor de audio a evento. La señal de audio se samplea usando la Frecuencia de Muestreo especificada en el menú **Settings** y se envía el valor convertido en corriente de eventos.

A to E (Trig)



Auxiliary

Convertidor de audio a evento con entrada Trigger. Cuando la señal de la entrada activadora va desde cero hasta valores positivos (borde ascendente) la señal de audio se samplea y se entrega en forma de evento.

- **T:** Entrada de audio para activar la conversión, que se produce con un borde ascendente.
- **In:** Entrada de audio para la señal que se va a samplear y entregar en forma de evento.
- **Out:** Salida de audio para la señal sampleada.

A to E (Perm)



Auxiliary

Convertidor de audio a evento con conversión permanente a intervalos regulares y frecuencia de muestreo ajustable. La señal de audio se samplea con la frecuencia dada y se entrega en forma de evento. Cuando se usa este módulo con alta frecuencia (por encima de 1000Hz) la carga de CPU que genera el procesamiento de eventos puede aumentar considerablemente. Normalmente **F** = 200 Hz es suficiente.

- **F:** Entrada de audio para controlar la frecuencia de muestreo. Valor en Hz.
- **In:** Entrada de audio para la señal que va a samplear y entregar en forma de evento.
- **Out:** Salida de evento para la señal sampleada.

A to Gate



Auxiliary

Convertidor de audio a evento de puerta con entrada de amplitud de puerta. Cuando la señal activadora va desde cero hasta valores positivos (borde ascendente) la señal de puerta se activa con la amplitud del valor actual de la entrada de amplitud. Cuando la entrada activadora regresa a cero o valores negativos (borde descendente) la señal de puerta se desactiva.

- **T:** Entrada de audio para activar la señal de puerta.
- **A:** Entrada de audio para controlar la amplitud de los eventos de puerta.
- **Out:** Salida de evento para la señal de evento de puerta.

To Voice



Auxiliary

Las señales de entrada monofónicas se transmiten a una voz de la señal de salida polifónica. El número de la voz se entrega a través del valor actual de la entrada **V**. Las señales se descartan cuando el valor de **V** no es número de voz válido.

Resulta útil para crear secuenciadores polifónicos, por ejemplo.

- **V**: Entrada monofónica para definir el número de voz al que se va a enviar un evento. Rango típico: [1 ... 10].
- **In**: Entrada monofónica híbrida para las señales que se van a enviar a una de las voces polifónicas.
- **Out**: Salida polifónica híbrida. Las señales de entrada se transmiten (con su valor original) en una voz de esta señal polifónica.

From Voice

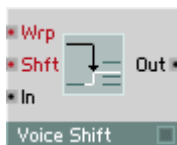


Auxiliary

Se selecciona una voz desde la señal de entrada polifónica a través del valor actual en la entrada **V**. Sólo se transmiten a la salida monofónica los eventos de la voz seleccionada. Todos los eventos de otras voces se descartan.

- **V**: Entrada monofónica para definir el número de la voz cuyos eventos van a pasar. Rango típico: [1 ... 10].
- **In**: Entrada híbrida. Una voz de esta señal polifónica se envía a la salida.
- **Out**: Salida monofónica híbrida. La señal de entrada perteneciente a una voz se transmite (con su valor original) en esta salida monofónica.

Voice Shift



Auxiliary

El módulo Voice Shift se usa para reiniciar los valores de entrada polifónicos, y que los valores de salida se puedan remapear sobre las voces de la forma deseada. Por ejemplo, podrías usar Voice Shift para remapear los valores de las voces 1, 2, 3 y 4 a las voces de salida 2, 3, 4, 1, o a las voces 3, 4, 2, y así. Si hay múltiples voces de entrada que se recolocan en la misma voz de salida, se sumarán. Por ejemplo, si las voces de entrada 1 y 2 se recolocan en la voz 3, la voz 3 consistirá en la señal de voz 1 + la señal de voz 2.

Wrp: Valor de evento monofónico que alterna entre el cambio de voces en bucle On y Off (Off por defecto). Cuando está en Off (es decir, $Wrp \leq 0$), las voces que se han cambiado a números de voces inválidos (es decir, menos que 1 o mayores que el número de voces polifónicas) se descartan. Cuando está en On ($Wrp > 0$), las voces cambiadas a números de voz inválidos, se re-mapean hasta un número válido de voz con un módulo Math. Por ejemplo, un cambio de +1 en un instrumento de 3-vozes, provocará que la voz 3 se remapee a la voz 1.

Sh: Valor de evento polifónico que controla el cambio de voz. Los valores Sh positivos cambian las voces hacia arriba (por ejemplo, $Sh = 1$ cambiaría la voz de entrada 1 a la voz de salida 2), y los valores Sh negativos cambian la voz hacia abajo ($Sh = -2$ cambiaría la voz 3 a la voz 1). Puesto que Sh es una entrada polifónica, cada voz se puede cambiar por su propio Offset. El valor por defecto de Sh es 1.

- **In:** Entrada para la señal polifónica cuyas voces se van a cambiar.
- **Out:** Salida para la señal con las voces cambiadas.

Audio Smoother



Auxiliary

Suavizador para señales de evento monofónicas con salida de audio. Normalmente, un suavizador se conecta después de un fader o botón para conseguir transiciones más suaves.

Los saltos del valor de la señal de evento de entrada se suaviza a través de rampas. El **Tiempo de Transición** (en milisegundos) se puede ajustar en las propiedades. Exactamente después de que éste tiempo haya transcurrido, la salida alcanza el mismo valor que la entrada, suponiendo que no haya más saltos en la entrada. Cuanto más largos sean los tiempos de transición, más fuerte es el efecto del suavizador.

- **In:** Entrada de evento mono para la señal que se quiere suavizar.
- **Out:** Salida de audio mono para la señal suavizada.

Event Smoother



Auxiliary

Suavizador para señales de evento monofónicas. Normalmente, un suavizador se conecta después de un fader o botón para conseguir transiciones más suaves.

Los saltos del valor de la señal de evento de entrada se suaviza a través de rampas. El **Tiempo de Transición** (en milisegundos) se puede ajustar en las propiedades. Exactamente después de que este tiempo haya transcurrido, la salida alcanza el mismo valor que la entrada, suponiendo que no haya más saltos en la entrada. Cuanto más largos sean los tiempos de transición, más fuerte es el efecto del suavizador.

Durante la transición, los eventos se entregan con la tasa seleccionada en el **Control Rate** dentro del menú **Settings**. Cuanto mayor es la tasa, mayor es la resolución de la transición del suavizador.

- **In:** Entrada de evento mono para la señal que se quiere suavizar.
- **Out:** Salida de evento mono para la señal suavizada.



Controla el nivel y afinación general.

- **Tun:** Entrada de control para la afinación maestra. Escala: 1 semitono por unidad. A 0.0 la nota La3 se afina a 440 Hz. Rango: [-1 ... 1].
- **Lvl:** Entrada de control para el nivel maestro en los convertidores de salida. Escala: 1dB por unidad 0.0 = ganancia de unidad = 0.0 dB. Rango [-60 ... 0].
- **Tun:** Salida para la afinación maestra.
- **Lvl:** Salida para el nivel maestro.

Tempo Info

Auxiliary

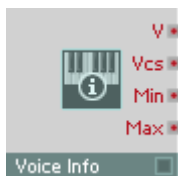


Fuente para el tiempo actual medido en Beats por Segundo. Para conseguir un valor en BPMs, multiplica por 60.

- **Out:** Salida de evento para el tempo actual en Beats por Segundo (Hz).

Voice Info

Auxiliary

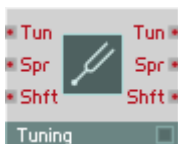


Módulo de Información de Voces.

- **V:** Salida polifónica para el número ID de cada voz [1, 2, 3 ... voces].
- **Vcs:** Salida para el actual Número de Voces en el instrumento.
- **Min:** Salida para el ajuste Min Unison Voices del instrumento. Éste es el número mínimo de voces asignadas a una tecla para tocar al unísono.
- **Max:** Salida para el ajuste Max Unison Voices. Este es el número máximo de voces asignadas a una tecla para tocar al unísono.

Tuning Info

Auxiliary



Control para la información sobre los ajustes de parámetros de afinación en un instrumento.

- **Tun:** Control para la afinación del instrumento en semitonos.
- **Spr:** Control para la cantidad de propagación de unísono en semitonos.
- **Shft:** Control para el transporte de notas MIDI entrantes en semitonos.

- **Tun:** Salida para el tono del instrumento en semitonos.
- **Spr:** Salida para la cantidad de propagación de afinación en semitonos.
- **Shft:** Salida para el transporte de notas MIDI entrantes en semitonos.

System Info



Auxiliary

Fuente para información sobre el sistema: Frecuencia de muestreo (en samples/seg), tasa de control (en Hz) y carga de CPU (en %).

- **SR:** Salida para la frecuencia de muestreo actual en samples por segundo.
- **CR:** Salida para la tasa de control actual en Hz.
- **DCIk:** Salida para la tasa de display actual en frames por segundo. Se envía un evento justo antes de cada actualización del display.
- **CPU:** Salida para la carga de CPU actual en porcentaje.

Note Range Info



Auxiliary

Módulo de Información sobre el Rango de Notas.

- **Upr:** Entrada para seleccionar el límite superior para las notas MIDI recibidas.
- **Lwr:** Entrada para seleccionar el límite inferior para las notas MIDI recibidas.
- **Upr:** Salida para la selección del límite superior de la recepción de notas (upper range).
- **Lwr:** Salida para la selección del límite inferior de la recepción de notas (lower range).

MIDI Channel Info

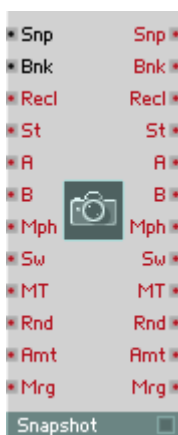


Auxiliary

Módulo de Información de Canal MIDI.

- **ICh:** Entrada para seleccionar el canal MIDI de entrada.
- **OCh:** Entrada para seleccionar el canal MIDI de salida.
- **ICh:** Salida para el canal MIDI de entrada actual del instrumento.
- **OCh:** Salida para el canal MIDI de salida actual del instrumento.

Snapshot



Auxiliary

El módulo Snapshot permite cambiar instantáneas desde la estructura de REAKTOR, y efectuar morphing entre ellas. El módulo tiene un procesador de listas interno que funciona exactamente como el módulo List (mira la sección de Módulos de Panel).

Apariencia

La representación del panel de este módulo cambia dependiendo del estilo escogido en la página **Appearance** de las Propiedades. Están disponibles los siguientes estilos:

- **Button:** Cada puerto de entrada del módulo crea un botón. Todos los botones estarán dispuestos en fila verticalmente en el panel del instrumento. El botón actualmente activado se representará en el color Indicador del instrumento.

- **Menu:** Cada puerto de entrada del módulo crea una nueva entrada en una lista desplegable.
- **Text Panel:** Cada puerto de entrada del módulo crea una nueva entrada en la lista que muestra múltiples entradas al mismo tiempo. Si has creado más entradas de las que caben en el tamaño del panel especificado con **Size X** y **Size Y**, en la tabla de Apariencia, aparecerán barras de desplazamiento en el panel.
- **Spin:** Cada puerto de entrada del módulo crea una nueva entrada en la lista. Puedes moverte por la lista con los botones + y – que hay en la parte derecha del display del panel de entradas de la lista.

Los recuadros **Size X** y **Size Y** controlan el tamaño de display de los elementos de control en el panel.

Ports

- **Snp:** Entrada de audio para la instantánea que se desea activar o guardar. Rango: 1 ... 128.
- **Bnk:** Entrada de audio para seleccionar el banco de instantáneas. Rango: 1 ... 16.
- **Recl:** Un evento positivo carga la instantánea seleccionada a través de las entradas Snp y Bnk.
- **St:** Un evento positivo guarda una instantánea en una posición seleccionada por las entradas Snp y Bnk.
- **A:** Un evento positivo coge los valores presentes en las entradas Snp y Bnk para seleccionar una instantánea para la posición A de Morphing.
- **B:** Un evento positivo coge los valores presentes en las entradas Snp y Bnk para seleccionar una instantánea para la posición B de Morphing.
- **Mrph:** Esta entrada controla la posición de Morphing entre las instantáneas cargadas en A y B. Valor ≤ 0.0 : A, Valor $0.0 - 1.0$: Morphing entre A y B. Valor ≥ 1.0 : B.
- Esta entrada controla la posición de Morphing entre las instantáneas cargadas para los valores A y B.
- **Sw:** Los eventos con valores negativos o cero ajustan los interruptores/botones a sus posiciones en la instantánea A. Los eventos con valores positivos ajustan los interruptores/botones a sus posiciones en la instantánea B.

- **MT:** Entrada de evento para el tiempo de Morph en ms.
- **Rnd:** Un valor positivo activa la función aleatoria para la instantánea seleccionada.
- **Amt:** Entrada para la intensidad de la aleatoriedad (randomization amount). Rango: 0.0 ... 1.0 (1.0 = 100%).
- **Mrg:** Un evento positivo activa la función Random Merge.
- **Snph:** Salida para el índice de la instantánea actual. Se envía un evento cada vez que se carga o guarda una instantánea.
- **Bnk:** Salida para el índice del banco de la instantánea actual. Se envía un evento cada vez que se carga o guarda una instantánea.
- **Recl:** Se envía un evento con valor = 1.0 cuando se carga una instantánea.
- **St:** Se envía un evento con valor = 1.0 cuando se guarda una instantánea.
- **A:** Salida para el índice de la instantánea de la posición A de Morph. Cuando se carga una instantánea o se selecciona para la posición A se envía un evento.
- **B:** Salida para el índice de la instantánea de la posición B de Morph. Cuando se carga una instantánea o se selecciona para la posición A se envía un evento.
- **Mrph:** Salida para la posición de Morph. Valor = 0.0 : A, Valor 0.0 - 1.0: Morphing entre A y B, Valor = 1.0: B.
- **Sw:** Salida para la posición de Interruptor A/B. Valor = 0.0 si los interruptores/botones están ajustados a sus posiciones en la instantánea A. Valor = 1.0 si los interruptores/botones están ajustados a sus posiciones en la instantánea B.
- **MT:** Salida para el tiempo de morph en ms.
- **Rnd:** Un evento (con valor = 1.0) se envía cuando se activa la función random.
- **Amt:** Salida para la cantidad de randomización. Rango: 0.0 ... 1.0 (1.0 = 100%)
- **Mrg:** Un evento (con valor = 1.0) se envía cuando se activa la función Random Merge.

Set Random

Auxiliary



Ajusta el Random-Seed e inicializa así el generador de números aleatorios que se usa en todos los módulos **Event Randomize**, **Slow Random** y **Geiger**. Sólo son afectados los módulos que están en el mismo instrumento. Cada valor genera otra secuencia única de números aleatorios.

Unison Spread

Auxiliary



Fuente de evento polifónica para un valor constante usado para separar los parámetros de voces unísonas.

En el modo unísono, las diferentes voces que tocan la misma nota, necesitan tener sus parámetros un poco separados para poder diferenciarse y conseguir así un sonido con más cuerpo, y no simplemente más alto.

Para la tonalidad, esto sucede automáticamente y se controla con el parámetro **Unison Spread** en las propiedades del instrumento. Otros parámetros se pueden separar añadiendo un valor generado por el Unison Spread.

El valor en la entrada del módulo determina la cantidad de separación. En la salida, tendrás un valor que es diferente para cada una de las voces que tocan la misma nota. El valor sólo cambia una vez que la nota se haya tocado.

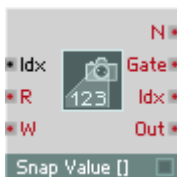
Snap Value

Auxiliary



Este módulo almacena el valor de entrada con una instantánea y entrega el valor cuando se carga la instantánea.

- **In:** Entrada para el valor que se va guardar en una instantánea. La señal de entrada se samplea en el momento en que se guarda la instantánea. Si está conectada a una fuente de evento, los eventos de entrada pasarán a la salida.
- **Out:** Salida para el valor guardado en la instantánea que se ha cargado más recientemente.



Guarda y carga series de valores (fraccionales) de puntos flotantes en/desde el buffer de edición e instantáneas.

Un simple Snap Value Array puede almacenar 1-40 series, y cada serie puede contener un número de elementos, limitado sólo por la capacidad del sistema de memoria. Todas las series contienen el mismo número de elementos. Tanto el número de elementos como el número de elementos por serie se definen en las propiedades.

La memoria para el módulo Snap Value Array funciona de forma dinámica. Es decir, la creación de instantáneas adicionales supone más memoria, y el borrado de instantáneas, requiere de menos memoria. Así podemos conservar los requisitos de memoria en un estado óptimo.

Cuando la opción Snap Isolate está habilitada en las propiedades, sólo se almacenan los valores más recientemente escritos en cada serie de elementos (y cargados durante la instalación del ensemble) y no se guardan o cargan datos en las operaciones de instantáneas.

Una aplicación típica del Snap Value Array es guardar datos de secuenciación en las instantáneas. Para ello, normalmente se usa junto con módulos Event Table o Multi Display.

Todas las entradas del Snap Value Array aceptan sólo señales monofónicas.

- **Idx:** Índice de los elementos de la serie al guardar (para operaciones de lectura y de escritura). Las series se basan en 1, es decir, el índice del primer elemento es 1 (no cero). Los valores fraccionales se redondean al número entero más cercano. Cuando los valores de Idx están fuera del Rango 1 a N, el comportamiento depende de la opción Index Behaviour en las propiedades.
- **R:** Cuando llega un evento (o un valor) a la entrada R (lectura), el elemento de la serie especificado por la entrada Idx se lee, y su valor se transmite al puerto de salida. En otras palabras, cada evento de la entrada R propaga un evento sencillo en el puerto de salida de la serie. Las series múltiples se colocan en paralelo por el mismo índice Idx. Así, los eventos que llegan a la entrada R propagan eventos de

salida en cada puerto de salida de la serie (del valor del elemento seleccionado en Idx). Es esencial ajustar el valor Idx antes de que los eventos lleguen a la entrada R.

- **W:** Cuando un evento llega a la entrada W, su valor se escribe en el elemento de la serie [Idx], sobrescribiendo cualquier dato que hubiese allí. El Snap Value Array proporciona un puerto de entrada W independiente para cada serie que contiene (1 serie por defecto, pero se pueden crear más en las propiedades). Cada puerto se puede renombrar como quieras. Cuando la opción Events Thru está habilitada en las propiedades, los eventos que llegan a las entradas W pasan a las salidas Out correspondientes.
- **N:** La salida N informa del número de elementos de cada serie.
- **Gate:** Gate envía un evento con valor 1 antes de que el puerto de salida de la serie envíe eventos, y luego envía un evento de valor 0.
- **Idx:** Salida que informa del número de elementos actualmente leídos o escritos (en respuesta a los eventos que llegan a los puertos R y W, o desde las operaciones de instantáneas, como cargar o Morph). Con respecto a las operaciones de lectura, el número Idx se reportará antes de que se transmita el evento a las salidas Out.
- **Out:** El valor del elemento de la serie al que se accede (definido por Idx) se transmite en respuesta a los eventos del puerto R y las operaciones de instantáneas (cargar o Morph). Cuando la opción Self-Iteration está habilitada en las propiedades, todos los elementos de la serie se entregan en modo serial (es decir, el primer elemento, luego el segundo, el tercero...) cuando se da cualquiera de las siguientes operaciones: inicialización, activación, carga de instantáneas, randomización, random merge, y morph. Self-Iteration permite la actualización de grupos completos de datos con operaciones de Snap.

Terminal

Las terminales se usan para el encaminamiento de señales de audio y de control dentro y fuera de las estructuras de Instrumentos o Macros de Reaktor. Puedes crear terminales automáticamente cuando arrastres un cable a un Instrumento o Macro dentro de la ventana de estructura mientras mantienes presionada la tecla Ctrl.

In Port

Terminal



Terminal para señales de evento y audio para crear puertos de entrada para Instrumentos y Macros.

Out Port

Terminal



Terminal para señales de evento y de audio para crear puertos de salida para Instrumentos y Macros.

Send

Terminal



Terminal de envío para señales de evento y de audio para crear una conexión sin cables dentro de un Instrumento.

Cada módulo Send insertado crea una entrada en una lista de fuentes de señal disponibles en las Propiedades de un módulo Receive.

Receive

Terminal



Terminal receptora para señales de audio y evento para crear una conexión sin cables dentro de un Instrumento.

Propiedades – Página Function

Para cada módulo Send insertado en el mismo Instrumento, se crea una entrada de lista. El nombre de entrada de la lista se creará de acuerdo con la etiqueta del módulo Send. Los botones **Up** y **Down** que hay encima de la lista sirven para mover una entrada seleccionada hacia arriba o hacia abajo.

La lista contiene las siguientes columnas que se pueden ordenar haciendo clic sobre el encabezado correspondiente:

- **#:** Muestra la posición en la lista del módulo Send. El valor por defecto se refiere a este número.
- **Label:** El nombre del módulo Send. Cuando se cambia el nombre del módulo Send, se actualiza la etiqueta en la lista.
- **State:** Indica si es posible una conexión en este módulo Send. Sólo establece una conexión si el cuadro marca **OK**.
- **Use:** Haz clic en este cuadro para ajustar una conexión en el módulo Send apropiado. Una conexión existente se indica con una cruz.

Mouse Resolution sólo se aplica al control del panel si eliges el estilo **Spin** en la página **Appearance**, donde podrás hacer clic en la entrada del control y arrastrar arriba o abajo para cambiar la entrada.

El valor **Default** se usa cada vez que ocurre una inicialización del control. En este caso la entrada **#** de la lista se seleccionará de acuerdo con el valor **Default** introducido aquí.

Propiedades - página Appearance

La representación del panel de este módulo cambia dependiendo del estilo escogido en la página **Appearance** de las propiedades. Están disponibles los siguientes estilos:

- **Button:** Cada puerto de entrada del módulo crea un botón. Todos los botones estarán dispuestos en fila vertical en el panel del instrumento. El botón actualmente activado se representará en el color Indicador del instrumento.
- **Menu:** Cada puerto de entrada del módulo crea una nueva entrada en una lista desplegable.
- **Text Panel:** Cada puerto de entrada del módulo crea una nueva entrada en la lista que muestra múltiples entradas al mismo tiempo. Si has creado más entradas de las que caben en el interior del panel especificado con **Size X** y **Size Y**, en la tabla de Apariencia, aparecerán barras de desplazamiento en el panel

- **Spin:** Cada puerto de entrada del módulo crea una nueva entrada en la lista. Puedes moverte por la lista con los botones + y – que hay en la parte derecha del display del panel de entradas de la lista.

Los recuadros **Size X** y **Size Y** controlan el tamaño de display de los elementos de control en el panel.

IC Send

Terminal



Transmite señales de evento monofónicas a cualquier módulo capaz de recibir IC (Internal Connection). Estos módulo incluyen IC Receive, pero también varios elementos de panel (como knobs e interruptores). Puesto que las conexiones trabajan globalmente (es decir, a nivel de ensemble) este módulo se puede usar para hacer conexiones sin cables entre los diferentes instrumentos dentro del ensemble.

El módulo IC Send tiene un display del panel que permite configurar las conexiones desde el panel del instrumento. Todos los módulos del ensemble capaces de recibir IC aparecerán aquí, excepto aquello con la opción 'No Entry in IC Menu' seleccionada en las propiedades. Las conexiones también se pueden establecer en el diálogo de propiedades.

IC Receive

Terminal



Recibe y entrega señales de evento monofónicas a módulos conectados a través de IC. Normalmente este módulo se usa con módulos IC Send, pero se puede conectar a cualquier módulo capacitados para una conexión IC (como knobs e interruptores).

Las conexiones IC se pueden establecer en las propiedades, y cuando se conectan módulos IC Send, las conexiones también se pueden establecer usando el interface del IC Send.



Los mensajes OSC se pueden transmitir mediante los módulos **OSC Send** y **OSC Receive**. Ambos módulos poseen una administración dinámica de puertos, es decir, se pueden agregar nuevos puertos de entrada y salida arrastrando un cable a un área vacía al lado de los puertos existentes, manteniendo pulsado **Ctrl** (borde izquierdo si es el Send, y derecho si es el Receive).

Las conexiones múltiples al módulo **OSC Send** dan como resultado mensajes OSC con múltiples argumentos. Por ejemplo, si conectas las salidas **MX** y **MY** del módulo **XY** a un módulo **OSC Send**, cada mensaje OSC tendrá ambos valores; el de X y el de Y. Tendrás que conectar múltiples salidas del módulo **OSC Receive** para recibir múltiples argumentos – una para cada argumento hasta un máximo de 10).

Si has creado más de un puerto de entrada para un módulo **OSC Send**, siempre el primer puerto de entrada del módulo activará el mensaje OSC.

Los mensajes OSC también se pueden enviar y recibir directamente desde algunos Módulos de Reaktor, por ejemplo, controles de panel. Los ajustes para enviar o recibir son los mismos que en los módulos OSC terminal.

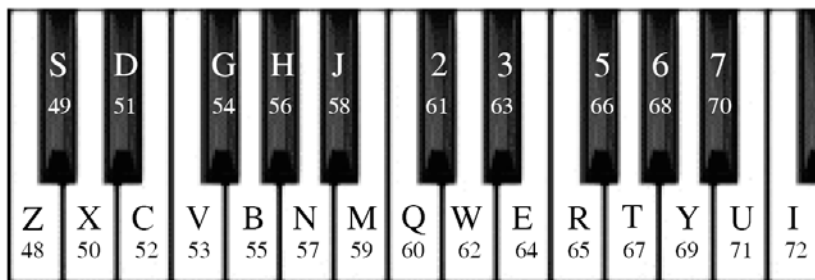
El módulo tiene administración dinámica de los puertos. El número de puertos de entrada se puede definir en **Min Num Port Groups** en la página **Function** de las Propiedades.



Mira el módulo OSC Send.

Apéndice

Transposing Incoming MIDI Notes



- Holding down the **Shift** key raises all incoming MIDI notes by two octaves (+24).
- Holding down XP: **Ctrl + Shift** / OS X: X + Shift lowers all incoming MIDI notes by two octaves (-24).

Primeros pasos en Reaktor Core

Qué es Reaktor Core

Reaktor Core es un nuevo nivel de funcionalidad dentro de Reaktor con nuevas (diferentes) funciones. Puesto que aún permanece el nivel de funcionalidad anterior, nos referiremos a estos dos niveles como *nivel core* y *nivel primario* respectivamente. Del mismo modo, cuando hablemos de “estructura del nivel primario” nos estaremos refiriendo a la estructura interna de un instrumento o una macro, pero no a la de un ensemble. Las características de Reaktor Core no son directamente compatibles con aquellas del nivel primario de Reaktor, de forma que necesitarás una interconexión entre ellas. Esta interconexión son las *células core*. Las células core residen dentro de las estructuras del nivel primario, y son y se comportan de manera similar a los módulos internos del nivel primario. He aquí un ejemplo de estructura, que utiliza una célula core *HighShelf EQ*, que es diferente de la versión del módulo interno del nivel primario en la forma en que responde a los controles de frecuencia y realce:

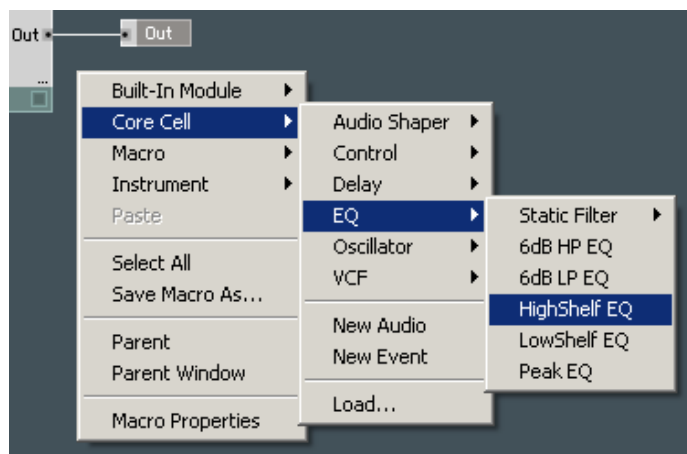


Dentro de las células core están las estructuras de Reaktor Core, que proporcionan un método muy eficaz para implementar una funcionalidad DSP de bajo nivel personalizada y también para construir estructuras de procesamiento de señal a mayor escala usando dicha funcionalidad. Más adelante analizaremos en detalle esta estructura. Aunque una de las especialidades de Reaktor Core es la capacidad de construir estructuras DSP de bajo nivel, no se limita exclusivamente a eso. Incluso aunque no seas un experto en DSP, hemos incluido una librería de módulos pre-construidos que podrás conectar dentro de las estructuras core, al igual que harías dentro del nivel primario. También te proporcionamos una librería de células core pre-construidas que podrás utilizar inmediatamente dentro de las estructuras del nivel primario.

A decir verdad, en Native Instruments no queremos seguir creando montones de nuevos módulos de nivel primario para las nuevas versiones de Reaktor. En lugar de esto, preferimos construirlos usando nuestra nueva tecnología Reaktor Core y entregarlos en forma de células core. También encontrarás un conjunto de nuevos filtros, envolventes, efectos y demás, disponibles en la librería de células core.

Usar células core

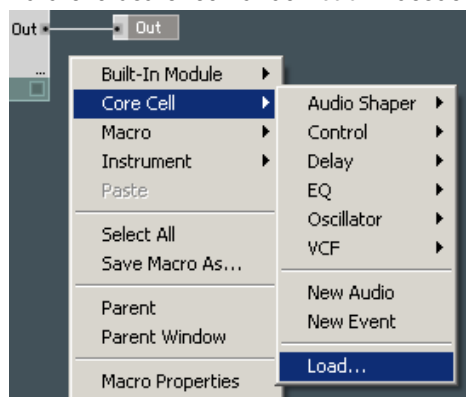
Puedes acceder a la librería de células core desde las estructuras de nivel primario haciendo clic en el botón derecho del ratón sobre el fondo de la pantalla y usando el submenú *Core Cell*:



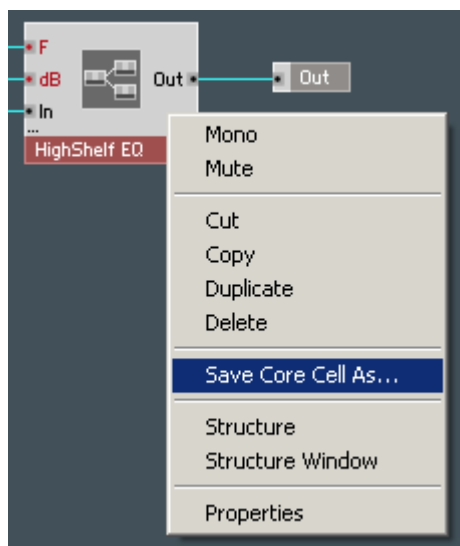
Como puedes ver hay todo tipo de células core que se pueden usar de la misma forma en que lo harías con un módulo interno del nivel primario.

Una limitación importante de las células core es que no podrás usarlas dentro loops de eventos. Cualquier evento de loop que apareciera en la célula core sería bloqueado por Reaktor.

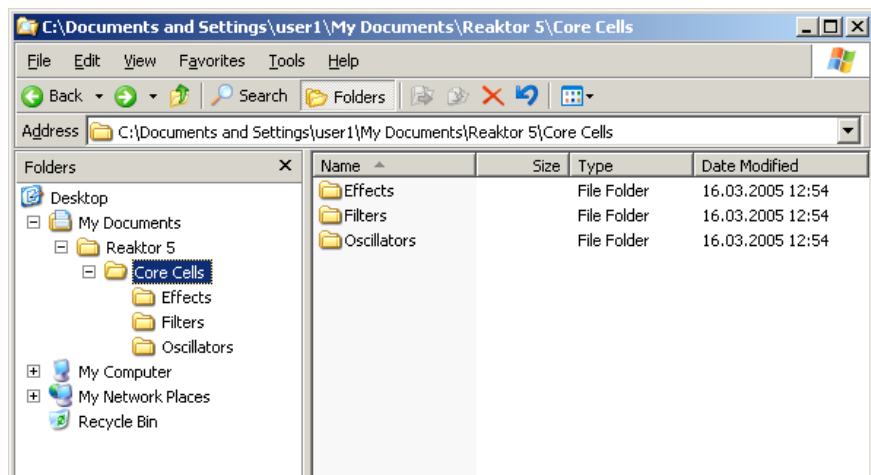
También tienes una opción para insertar células core que no estén en la librería. Para ello usa el comando *Load...* desde el mismo menú *Core Cell*:



Probablemente, antes de tener algunos archivos de células core, necesitarás guardarlos. Para ello haz clic con el botón derecho sobre una célula core y selecciona *Save Core Cell As...*:

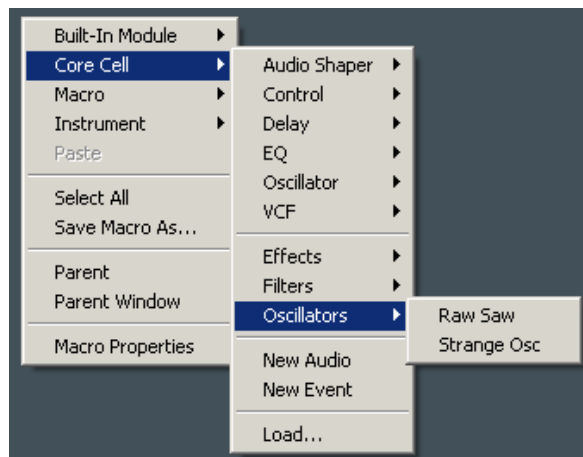


También dispones de las células core en el menú, en lugar de tener que usar el comando Load. Para ello, tienes que ponerlas dentro del sub-directorio “Core Cells” de la carpeta de la librería de usuario. Incluso es mejor que no las pongas directamente en el subdirectorio “Core Cells”, sino que las organices en grupos. He aquí un ejemplo de dicha organización:



“My Documents\Reaktor 5” is the user library folder in this example. On your computer there may be a different path, depending on the choice you’ve made during installation and any changes you’ve made in Reaktor’s preferences. Inside the user library folder there’s a folder named *“Core Cells”*. (Create it manually if it doesn’t exist.)

Dentro de *“Core Cells”* ahora podemos ver la siguiente estructura de carpetas que consiste en las carpetas *“Effects”*, *“Filters”* y *“Oscillators”*. Dentro de esas carpetas hay archivos de células core que aparecerán en la parte del usuario del menú *Core Cell*.



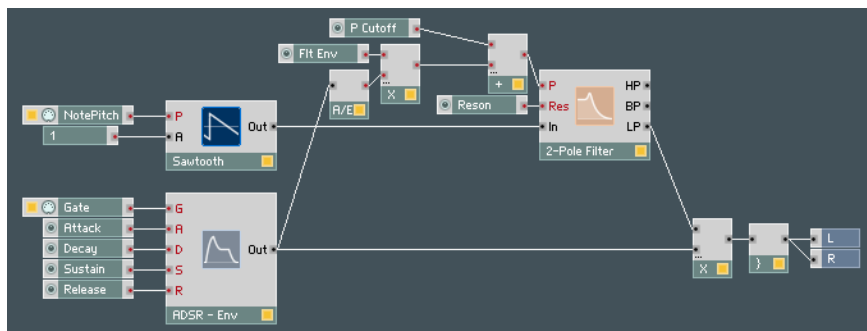
Los contenidos del menú se cargan durante el arranque de Reaktor. De modo que si incluyes nuevos archivos dentro de esta carpeta, tendrás que reiniciar Reaktor.

Las carpetas vacías no aparecen en el menú, han de contener archivos para que se vean.

No incluyas tus propios archivos dentro de la librería del sistema bajo ningún concepto. El directorio de la librería del sistema podría cambiar o desaparecer durante la instalación de actualizaciones, y tus archivos podrían perderse. La librería de usuario es el lugar correcto donde debes colocar cualquier contenido no incluido en el programa.

Usar células core en un ejemplo real

Vamos a coger un instrumento de Reaktor construido exclusivamente con los módulos del nivel primario y a modificarlo incorporando algunas células core. Dentro de la carpeta *Core Tutorial Examples* en tu instalación de Reaktor encontrarás el ensemble *One Osc.ens*: ábrelo. Este ensemble consiste en un solo instrumento con la siguiente estructura interna:



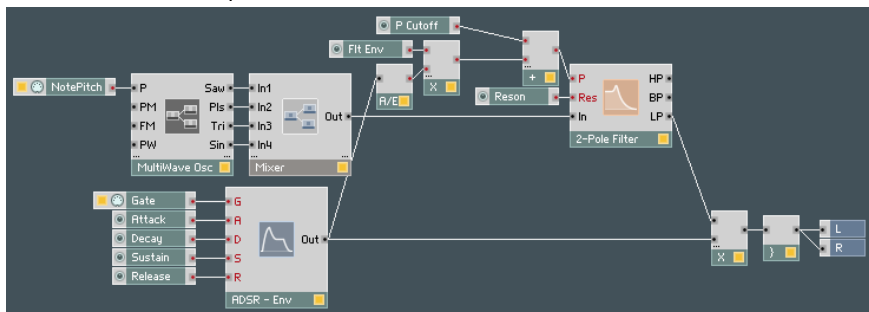
Como puedes ver se trata de un sintetizador sustractivo muy sencillo formado por un oscilador, un filtro y una envolvente. Vamos a reemplazar el oscilador por otro distinto, muy potente. Haz clic con el botón derecho sobre el fondo de la pantalla y selecciona *Core Cell > Oscillator > MultiWave Osc*:



La característica más importante de este oscilador es que proporciona simultáneamente formas de onda analógicas con la fase bloqueada. Vamos a usar su mezcla en lugar de un sencillo oscilador de diente de sierra. Afortunadamente, ya hay un mezclador disponible desde *Insert Macro > Classic Modular > O2 - Mixer Amp > Mixer - Simple - Mono*:

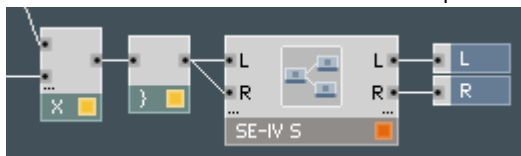


Ahora vamos a conectar el mezclador y el oscilador y a reemplazar el oscilador de diente de sierra por su combinación:

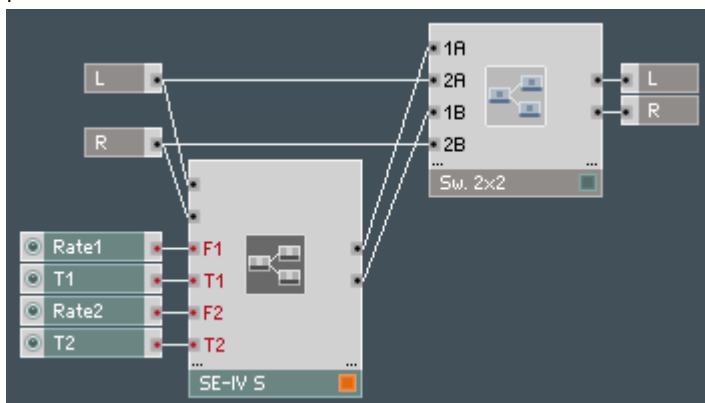


Conecta el panel View. Ahora podrás usar los cuatro faders del mezclador para variar la mezcla de la foma de onda.

Vamos a hacer una modificación más al instrumento – añadimos un efecto de chorus basado en Reaktor Core. Decimos basado en Reaktor Core porque aunque el chorus está construido como célula core, la parte que contiene los controles del panel de este chorus está construida usando las funciones del nivel primario. Es porque actualmente, las estructuras de Reaktor Core no tienen la posibilidad de definir sus propios paneles – los paneles se deben construir en el nivel primario. Selecciona *Insert Macro > Building Blocks > Effects > SE-IV Chorus* e insértalo después del módulo Voice Combiner:



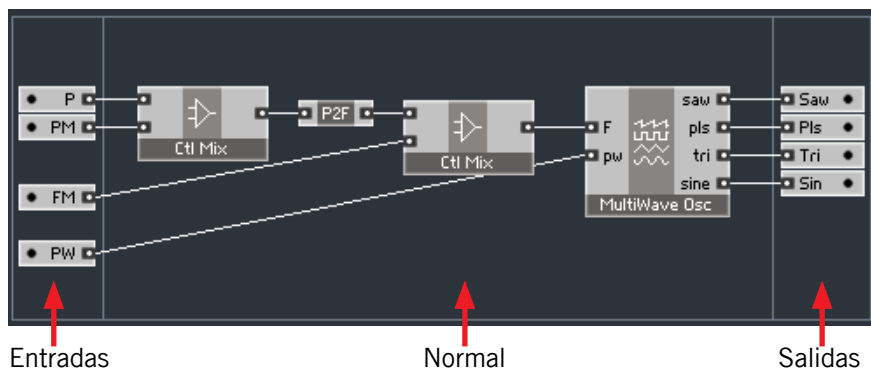
Si miras dentro del chorus verás la célula core de chorus y los controles del panel:



Edición básica de células core

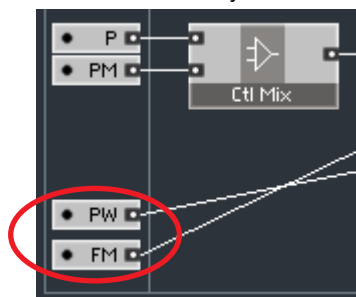
Ahora vamos a aprender algunas cosas sobre la edición de células core. Vamos a empezar con algo sencillo, como reajustar una célula core existente según tus necesidades específicas.

Primero, haz doble clic sobre *MultiWave Osc* para entrar:

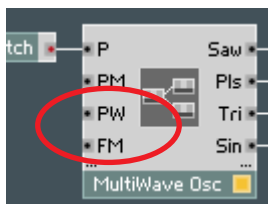


Lo que ves ahora es una estructura de Reaktor Core. Las tres áreas separadas por líneas verticales son para los tres tipos de módulos diferentes: Entradas (a la izquierda), Salidas (a la derecha) y normales (centro).

Las entradas y salidas sólo se pueden mover verticalmente, y su orden relativo encaja con el orden en el que aparecen fuera, mientras que los módulos normales se mueven en todas direcciones. De forma que puedes reorganizar fácilmente su orden externo simplemente moviéndolos. Prueba a mover la entrada *FM* debajo de la *PW*:



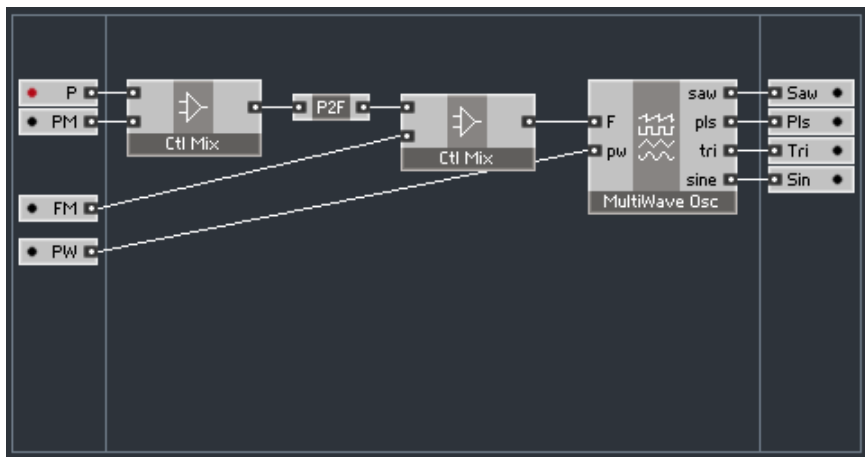
Ahora puedes hacer doble clic sobre el fondo de la pantalla para subir a la estructura primaria externa y comprobar el orden del puerto cambiado:



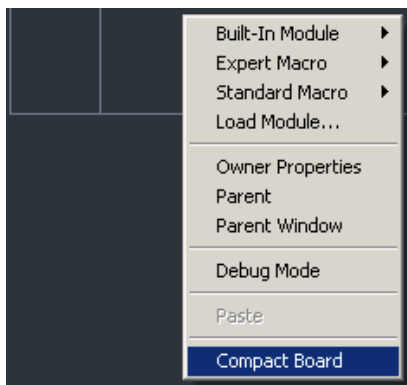
Regresemos al nivel core sin olvidar restablecer el orden original del puerto:



Probablemente habrás notado que si mueves los módulos sobre las tres áreas de la estructura core, seguramente se ampliará para poder acomodar los módulos dentro. No obstante, no se reduce automáticamente, lo que algunas veces hace que las áreas sean demasiado grandes:



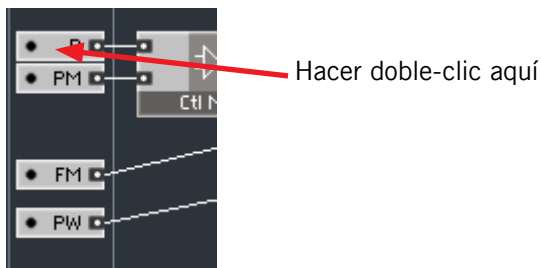
Puedes reducirlas haciendo clic en el botón derecho sobre el fondo de la pantalla y seleccionando el comando *Compact Board*.




Ahora que ya hemos aprendido a mover las cosas e incluso a reorganizar el orden del puerto de una célula core, probemos con otras opciones.

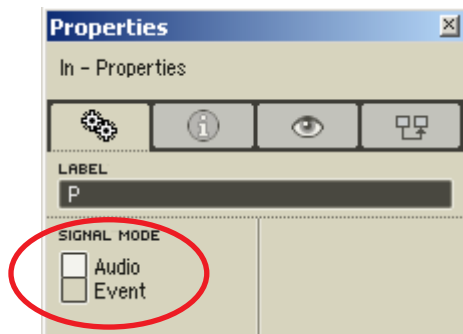
Para una célula core que *tenga salidas de audio* puedes alternar los tipos de entradas entre audio y evento (más adelante encontrarás una explicación detallada). En el ejemplo de arriba hemos usado un módulo *MultiWave Osc* que tiene todas sus entradas y salidas como audio. Pero en nuestro caso, en realidad no los necesitamos como audio, puesto que lo único conectado al oscilador es un knob de tono. ¿Ahorraríamos más CPU teniendo al menos uno de los puertos ajustado a evento? La respuesta, por supuesto, es sí. Vamos a hacerlo.

Estamos sugiriendo que cambies al menos las entradas *P* y *PM* al modo evento. Parece que esto mejoraría la carga de CPU. Para ello haz doble clic en el módulo del puerto *P* para abrir la ventana de propiedades de ese puerto:



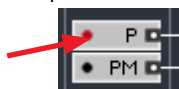
Conecta la ventana de propiedades a la página de la función si es necesario haciendo clic en el botón .

Ahora deberías ver la propiedad Signal Mode:

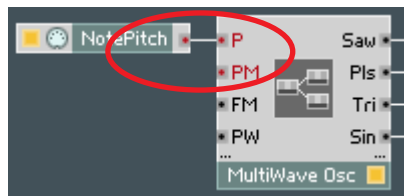


Cámbialo a evento. Nota cómo el punto grande que hay a la izquierda del módulo de entrada cambia de negro a rojo indicando que la entrada está ahora en modo evento (se verá mejor cuando de-selecciones el puerto – simplemente haz clic en cualquier otra parte).

El punto se pone rojo



Ahora haz clic en la entrada *PM* para seleccionarlo y cámbialo también al modo evento. Si quieres, puedes cambiar las dos entradas que todavía quedan al modo evento. Tras terminar esta labor, haz doble clic sobre el fondo de la estructura para regresar al nivel primario y observar cómo los colores de los puertos han cambiado a rojo y la carga de CPU ha reducido.



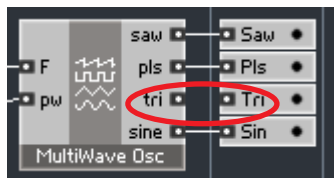
Algunas veces no tiene sentido cambiar un puerto de un tipo a otro. Por ejemplo, no tiene ninguna lógica cambiar una entrada que recibe una señal de audio real (con un sonido real, no sólo una señal de control como una envolvente con la velocidad del audio) por la velocidad del evento. Además de no tener sentido, en algunos casos podría arruinar la funcionalidad del módulo. Otro caso es el que tampoco nos interesaría hacerlo es cuando tenemos una entrada de evento realmente susceptible al evento, por ejemplo una entrada de evento activado de una envolvente (como las entradas de la Puerta de las envolventes del nivel primario de Reaktor). Si tratas de cambiarlo a audio, no funcionará bien nunca más.

Al margen de que en algunos casos no tenga sentido cambiar el tipo de puerto, hay otros en los que sí puede interesarnos, pero puede haber módulos que no funcionen correctamente al cambiar sus tipos de puerto. No obstante, estos casos son excepcionales, o pueden deberse a algún error en el diseño de implementación del módulo. Pero por lo general, cambiar los tipos de puerto debería funcionar. De todas formas, te mostramos una regla básica al respecto:

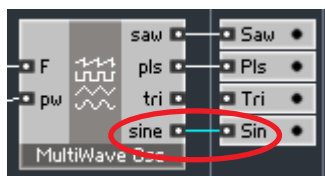
En una célula core bien diseñada, una entrada de audio de control de velocidad se puede cambiar al modo evento sin ningún problema. Una entrada de evento se puede cambiar a audio sólo si no incluye la palabra ‘trigger’ (accionar).

Otra cosa que puedes hacer para ahorrar CPU, es desconectar las salidas que no utilices, y de esta forma desactivar las partes inútiles de la estructura de Reaktor Core. Tendrás que hacerlo también desde dentro – las conexiones externas no tienen efecto a la hora de desactivar los elementos de la estructura core.

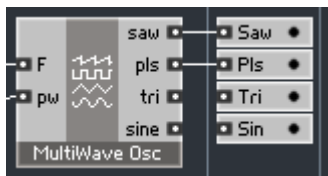
Digamos, en nuestro ejemplo, que decidimos que no vamos a necesitar las cuatro salidas, sólo el diente de sierra y el pulso. Podemos ir dentro de *MultiWave Osc* y desconectar las salidas que no vayamos a utilizar. En Reaktor Core, desconectar es muy sencillo. Haz clic en el puerto Input de la conexión, arrastra el ratón hasta cualquier parte donde no haya un puerto de salida y suéltalo. Empecemos por la salida ‘Tri’. Haz clic en el puerto de entrada de la salida ‘Tri’, y arrastra el ratón hasta un espacio vacío en el fondo de la pantalla.



Hay otra forma de borrar una conexión. Haz clic en el cable que hay entre la salida “sine” del *MultiWave Osc* y la salida “Sin” de la célula core, para seleccionarlo (lo sabrás por el color):

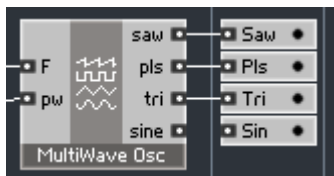


Ahora puedes presionar la tecla *Delete* para borrar el cable:



Después de borrar los dos cables, el medidor de CPU debería bajar un poco más.

¿Qué pasa si cambias de idea y decides conectarlos otra vez? También es sencillo. Haz clic sobre la entrada o la salida que quieres conectar, arrastra el ratón al otro puerto que participa en la conexión y suéltalo allí. Por ejemplo, haz clic sobre la salida “tri” del *MultiWave Osc* y arrástralo hasta la entrada del módulo de salida “Tri”. La conexión ha vuelto:



Por supuesto, las posibilidades de reajuste de las células core no se limitan a lo que acabamos de describir arriba. Aprenderás mucho más, si sigues leyendo el texto.

Entra en Reaktor Core

Células core de evento y de audio

Las células core pueden ser de dos tipos: de *Evento* y de *Audio*. Las células core de evento sólo pueden recibir señales de evento del nivel primario en sus entradas, y sólo producen señales de evento de nivel primario en sus salidas en respuesta a dichos eventos de entrada. Las células core de audio pueden recibir señales tanto de audio como de evento en sus entradas, pero sólo proporcionan salidas de audio:

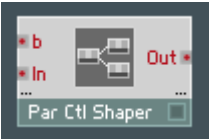
Tipo	Entradas	Salidas	Fuentes de Reloj
Evento	Evento	Evento	Deshabilitado
Audio	Evento/Audio	Audio	Habilitado

Por consiguiente las células de audio pueden implementar osciladores, envolventes, efectos y demás, mientras que las células de evento sólo son adecuadas para tareas de procesamiento de eventos.

Los módulos *HighShelf EQ* y *MultiWave Osc* con los que ya te habrás familiarizado, son ejemplos de células core de audio (lo sabrás porque tienen salidas de audio):



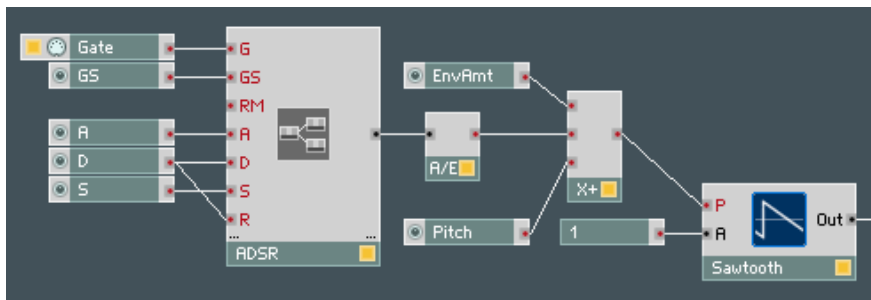
Y aquí se muestra un ejemplo de una célula core de evento:



Este módulo es un modelador parabólico de control de señales, que se puede usar para implementar, por ejemplo, curvas de velocidad o diseño de señales LFO.

Hemos dicho antes que las células core de evento se limitan a las tareas de procesamiento de eventos. Debido a que las fuentes de reloj están deshabilitadas en su interior (mira la tabla superior), no pueden generar sus propios

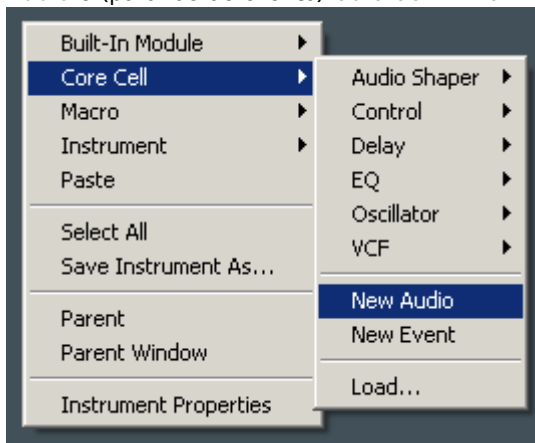
eventos y por lo tanto, no pueden implementar módulos como la velocidad del evento LFOs o envolventes. Si alguna vez necesitas algo así, te sugerimos que cojas una célula core de audio y conviertas su salida a frecuencia de evento usando uno de los convertidores audio/evento del nivel primario:



La estructura de arriba usa una célula core de audio implementando una envolvente ADSR y la convierte en frecuencia de evento para modular un oscilador.

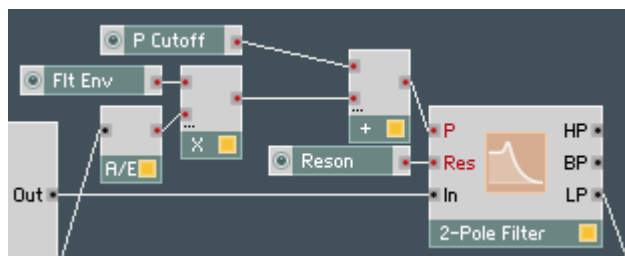
Crea tu primera célula core

Podrás crear nuevas células core haciendo clic derecho sobre el fondo de la pantalla en una estructura de nivel primario y seleccionando *Core Cell > New Audio* o (para las de evento) *Core Cell > New Event*:



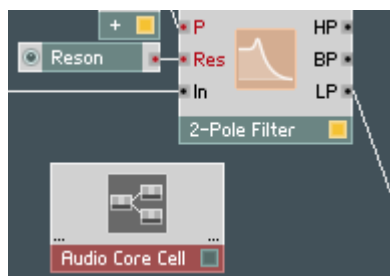
Vamos a construir una nueva célula core aprovechando el mismo *One Osc.ens* que hemos utilizado antes. Usaremos la versión modificada de este ensemble con el nuevo oscilador y chorus que ya hemos construido juntos. Si no lo has guardado, no te preocupes, puedes hacer los mismos pasos usando el *One Osc.ens* original.

Como puedes ver en este ensemble, estamos modulando el filtro en la entrada P, que sólo acepta señales de evento. No vamos a usar la versión FM del mismo filtro porque: a) funciona peor a altas frecuencias de corte y b) la escala de modulación en la entrada FM es lineal, lo que para la modulación por una envolvente ofrece resultados menos musicales (típicamente, aunque no de forma correcta, conocido como “envolventes lentas”:

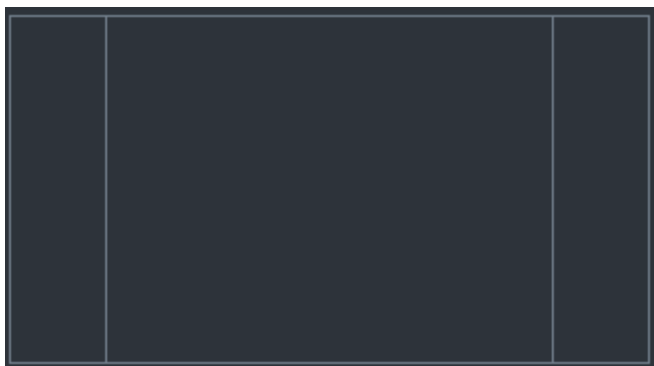


Puesto que necesitamos modular en la entrada de evento, tendremos que convertir la envolvente en señal de evento, lo que llevaremos a cabo usando el convertidor A/E. Como resultado, nuestra frecuencia de control es bastante baja. Por supuesto, podríamos haber usado un convertidor que funcionara con una frecuencia mucho mayor (consumiendo así mucha más CPU), pero lo que vamos a hacer en lugar de eso, es reemplazar este filtro por otro que construiremos como célula core. También podríamos haber cogido uno de la librería de células core, pero entonces no lo pasaríamos tan bien creando nuestra primera estructura de Reaktor Core, de forma que no elegiremos el camino más fácil.

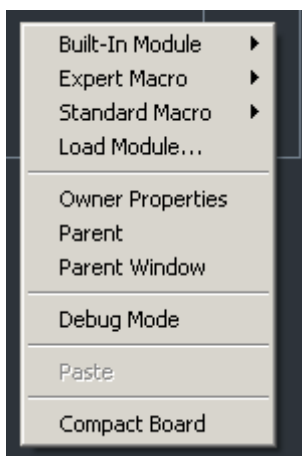
Empezaremos creando una nueva célula core de audio, así que selecciona *Core Cell > New Audio*. Aparecerá una célula core de audio vacía:



Haz doble clic sobre ella para ver la estructura interna de Reaktor Core, que obviamente está vacía. Como seguramente recordarás ahora, las tres áreas son para módulos de entrada, de salida y normales:



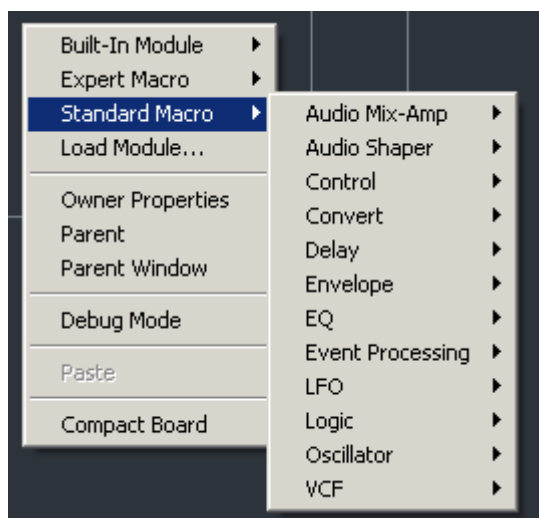
Atención: ¡Vamos a insertar nuestro primer módulo en una estructura core!
Haz clic en el botón derecho dentro del área normal para que aparezca el menú de creación de módulos:



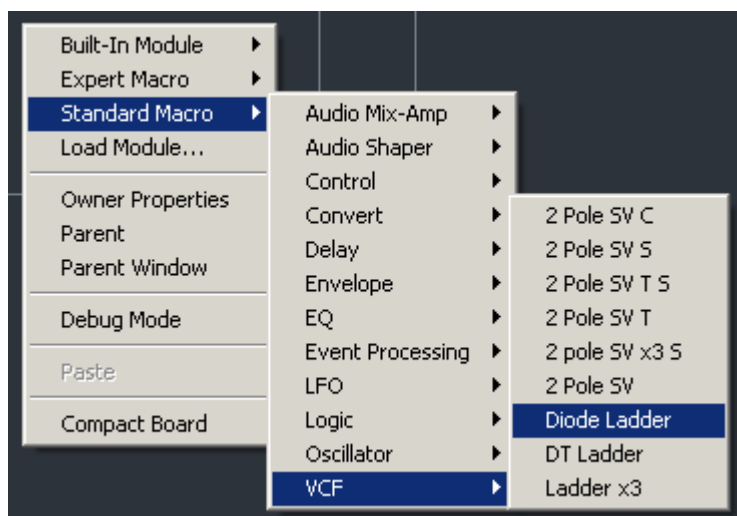
El primer sub-menú se llama *Built In Module* y te permite el acceso a los módulos internos de Reaktor Core, que normalmente están diseñados para trabajar material de nivel bajo y de los que hablaremos más tarde.

El segundo sub-menú se llama *Expert Macro* que contiene macros, y está diseñado para usar junto con los módulos incorporados en material de nivel bajo.

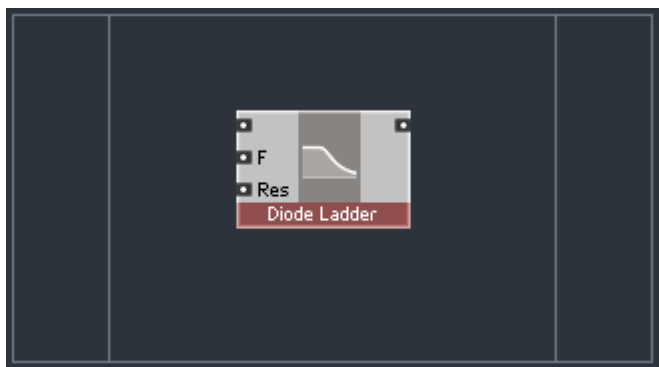
El tercer sub-menú se llama *Standard Macro* y parece que es el correcto:



La sección VCF es prometedora, echemos un vistazo al interior:

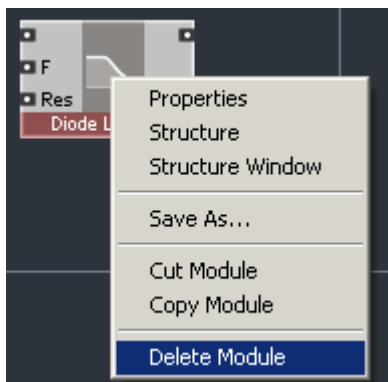


¿Deberíamos coger *Diode Ladder*? Hagámoslo:



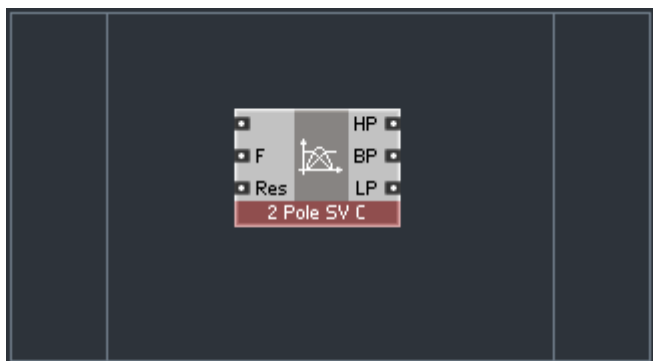
Bien, quizá no ha sido la mejor idea, porque el *Diode Ladder* debería sonar muy distinto del módulo de filtro del nivel primario que estamos intentando reemplazar. Como mínimo, el *Diode Ladder* es un filtro de 4-pole (24dB/octava) y el que queremos reemplazar es un filtro de 2-pole (12dB/octava).

Será mejor borrarlo y coger otro. Tienes dos opciones. Una es hacer clic derecho sobre el módulo y seleccionar *Delete Module*:



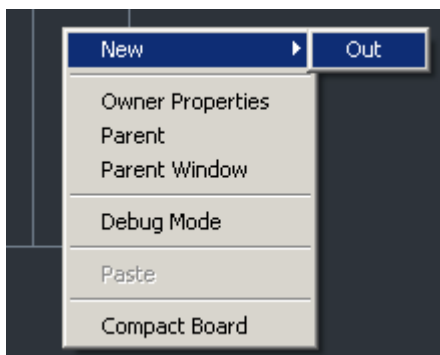
La otra opción es seleccionar el módulo haciendo clic sobre él y simplemente presionar la tecla *Delete*.

Después de borrar el *Diode Ladder* vamos a insertar en su lugar un filtro *2 Pole SV C* desde el mismo menú:



Este es un filtro de 2-pole, por eso se le llama filtro de estado variable, y es similar al que estamos reemplazando (hay alguna diferencia pero se parecen bastante). Lo importante es que este lo podemos modular la tasa de audio. Evidentemente, necesitaremos algunas entradas y salidas para nuestra célula core.

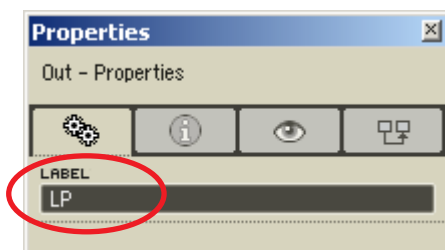
Para ser exactos, seguramente sólo necesitaremos una salida – para la señal LP. Para crearla haz clic con el botón derecho sobre el área de salidas:



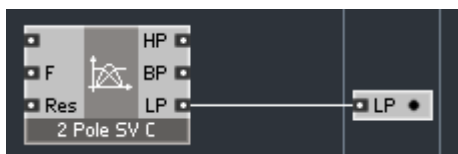
Sólo hay un tipo de módulo que puedes crear ahí, así que selecciónalo. Así es como aparecerá la estructura:



Haz doble clic sobre el módulo de salida para abrir la ventana de propiedades (si no está ya abierta). Teclea “LP” en el campo de texto:



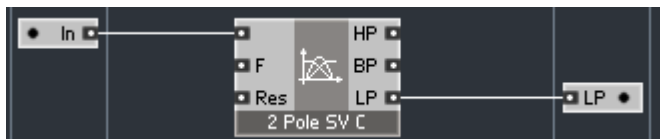
Ahora conecta la salida *LP* del filtro al módulo de salida:




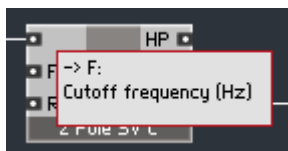
Vamos con las entradas. La primera entrada será una entrada de señal de audio. Haz clic con el botón derecho sobre el fondo de pantalla del área de las entradas y selecciona *New > In:*



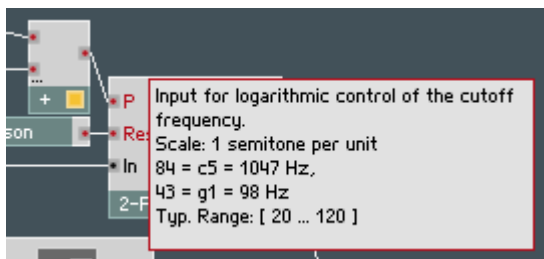
Ya hemos creado la entrada con el tipo correcto – una entrada de audio, como puedes ver por el gran punto negro. Renombra esta entrada como “In” del mismo modo que lo has hecho con la salida “LP”. Conéctala a la primera entrada del módulo de filtro:



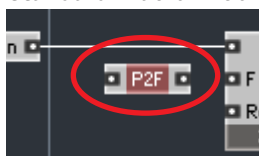
La segunda entrada es otra historia – algo más compleja. Como puedes ver la segunda entrada del módulo de filtro de Reaktor Core se llama “F”. Esto significa Frecuencia. De hecho, si mantienes el ratón durante unos instantes sobre esta entrada (comprueba que el botón  está activado) verás un texto que dice: “Cutoff frequency (Hz)”:



Como ya sabemos, la frecuencia de corte de nuestro módulo de filtro de nivel primario está controlado por una entrada llamada “P”, y como verás en el texto de información, la señal está usando la escala de semitonos.



Obviamente, necesitamos convertir de semitonos a Hz. Podemos hacerlo tanto en el nivel primario (usando el módulo *Expon. (F)*) como dentro de la estructura de Reaktor Core. Puesto que estamos aprendiendo a construir estructuras de Reaktor Core, será mejor escoger la segunda opción. Haz clic en el botón derecho sobre el fondo de pantalla del área normal y selecciona *Standard Macro > Convert > P2F*:



Como indica su nombre (y el texto de información), este módulo convierte entre escalas “P” y “F” – exactamente lo que necesitamos. Vamos a crear una segunda entrada llamada “P” y la conectaremos al módulo P2:

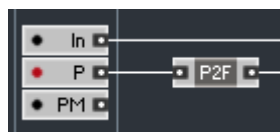


Debería funcionar, pero... ¡Espera!, en nuestro instrumento tenemos un Knob “P Cutoff” que define la base de corte del filtro y entonces se añade a la señal de modulación desde la envolvente que tenemos que convertir a señal de evento en el nivel primario para poder alimentar la entrada “P” del filtro. Puesto que ahora esta conversión ya no es necesaria, podemos eliminar el

módulo A/E y conectar la señal de audio a la entrada de audio “P” de nuestro nuevo filtro. No tendremos ningún problema durante este proceso, pero por si acaso, te sugerimos otra manera de hacerlo.

Vamos a mantener nuestra entrada “P” en modo evento y a disponer de otra entrada de modulación en modo audio. Si recuerdas nuestra discusión sobre las “envolventes lentas” comprenderás que hayamos sugerido llamar a este módulo “PM” y no “FM”, y que tengamos la modulación en escala de semitonos (“Pitch”). Es exactamente lo que está ocurriendo en nuestro instrumento – añadimos a nuestra señal de envolvente con la señal “P Cutoff” y conectamos la suma de las dos a la entrada “P”.

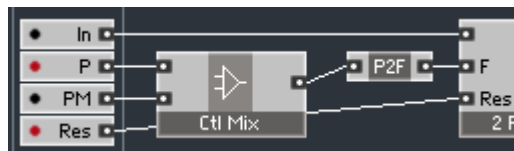
De modo que cambiamos nuestra entrada “P” al modo evento (ya hemos descrito antes cómo hacerlo) y añadimos otra entrada “PM” que, obviamente, está en modo audio:



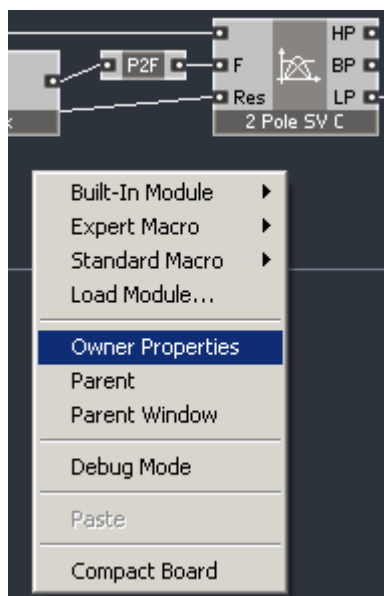
Como usuario del nivel primario de Reaktor, seguramente estés esperando que ahora unamos las dos señales, y de hecho podríamos hacerlo, pero en Reaktor Core el Add se considera un módulo de bajo nivel y normalmente requiere un conocimiento de los principios fundamentales de trabajo de bajo nivel de Reaktor Core. No son muy complejos y los describiremos más tarde, pero por el momento no necesitas saberlos. Simplemente usa un mezclador de señal de control, por ejemplo, *Standard Macro > Control > Ctl Mix*:



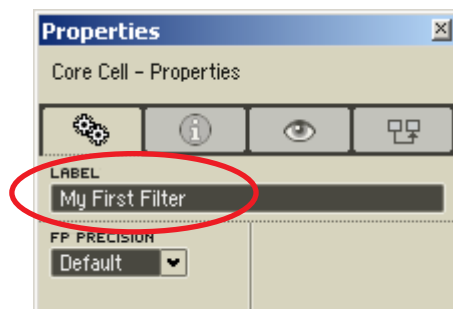
La última entrada que necesitamos es una entrada de resonancia. No la necesitamos en tasas de audio, así que usaremos una de evento:



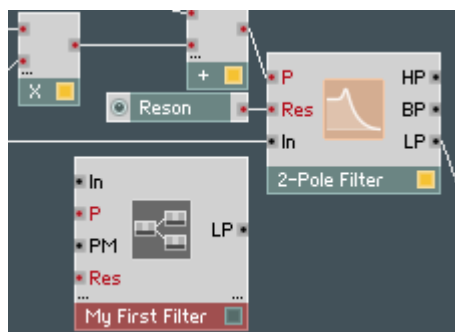
Otra cosa que tendremos que hacer es darle un nombre a nuestra célula core. Para abrir sus propiedades haz clic sobre el fondo de la pantalla si la ventana de Propiedades está ya abierta. Si no lo está, haz clic con el botón derecho sobre el fondo de la pantalla y selecciona el comando *Owner Properties*:



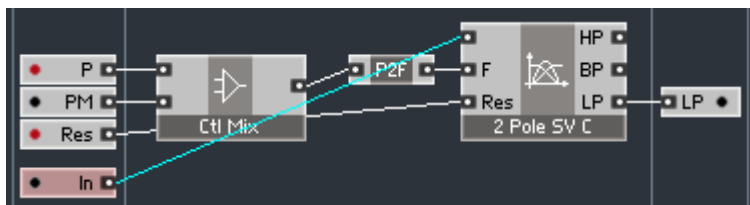
Ahora podrás escribir el nombre dentro del campo de texto:



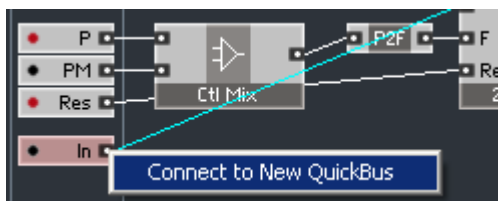
Haz doble clic sobre el fondo de la pantalla para ver los resultados:



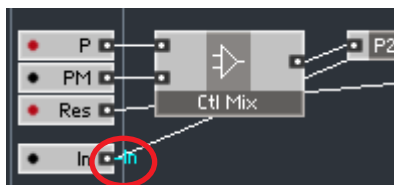
¡Guau! Parece que está muy bien, pero la entrada de la señal de audio está sobre la célula core, mientras que la del módulo de filtro del nivel primario está debajo. Bien, no creas que es un gran problema, si quieres que encajen sólo tienes que saber cómo hacerlo. Vamos a hacerlo juntos. Te enseñaremos una nueva función sobre la marcha. Vamos a volver dentro y lo primero que haremos es arrastrar la entrada de la señal de audio hasta abajo:



Debería ser la solución, pero un cable que recorra toda la estructura no queda muy bonito. De eso nos encargamos ahora. Haz clic con el botón derecho en la salida del módulo de entrada “In” y selecciona el comando *Connect to New QuickBus*:



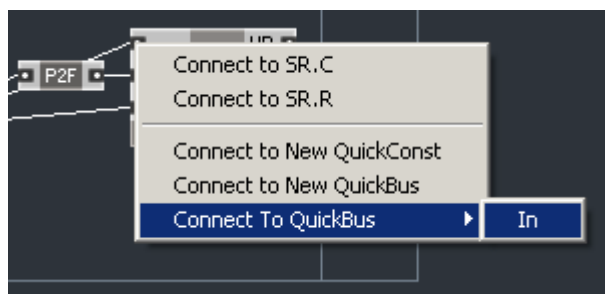
Así es como tendrías que verlo ahora:



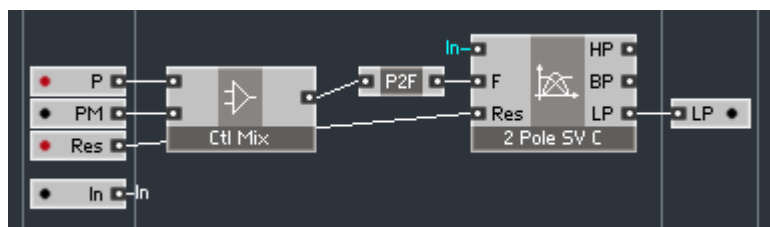
La ventana de Propiedades tendría que abrirse y mostrar las propiedades del QuickBus que acabas de crear. Lo más interesante de las propiedades del QuickBus es, por supuesto, que puedes cambiar el nombre (hay otras propiedades más avanzadas, así que de momento, no las toques). Después podrás abrir también la ventana de Propiedades con doble clic sobre QuickBus.

Aunque puedes renombrar este QuickBus si quieres, nosotros pensamos que el nombre es perfecto, porque se corresponde con el nombre de la entrada conectada a este QuickBus. Los QuickBusses en esta estructura son locales, así que tampoco tendrás que preocuparte por posibles confusiones de nombre si hay otra estructura usando un QuickBus con el mismo nombre. A continu-

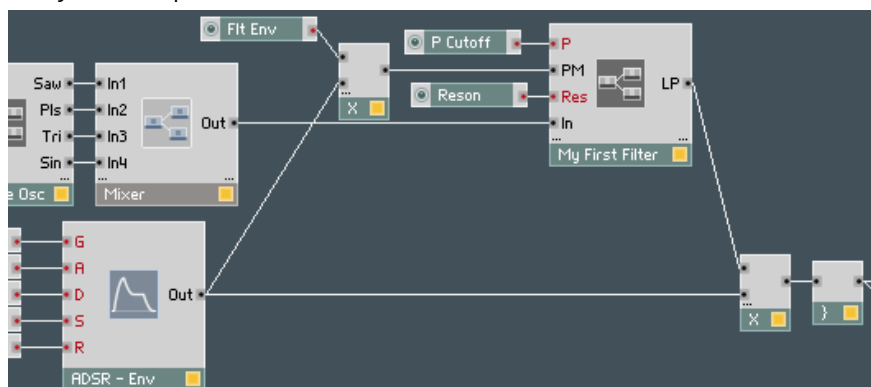
acción tendrías que hacer clic con el botón derecho sobre la entrada superior del módulo de filtro *2 Pole SV C* y seleccionar *Connect to QuickBus > In*:



En el menú de arriba, “In” no es sino el nombre del QuickBus al que te estás conectando. No queremos crear otro nuevo QuickBus, sino conectar uno ya existente, que es lo que estamos haciendo. Así es como debería verse tu estructura ahora:



En lugar de una vista caótica, ahora disponemos de unas referencias perfectas, empezando por el hecho de que están conectadas a través de un QuickBus cuyo nombre es “In”. Ya podemos regresar al nivel primario y modificar nuestra estructura para usar el nuevo filtro que acabamos de construir. Los módulos A/E y Add se pueden eliminar. Este es nuestro resultado final:



La carga de CPU ha subido un poco, ¿no? Bien, no olvides que este filtro está modulado en la tasa de audio en la escala de tono. Si no te gusta, puedes volver a la estructura anterior o usar el módulo de filtro *Multi 2 pole FM* del nivel primario (“envolventes lentas”, ¿recuerdas?), aunque esperamos que te guste. Incluso si no es así, hay unos cuantos filtros con nuevas características que quizá te agraden más. Y si no te gustan los nuevos filtros e Reaktor Core, hay otros muchos módulos de Reaktor que puedes probar.

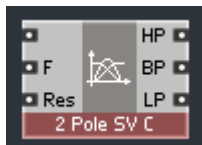
Señales de audio y de control

Antes de seguir es importante que revisemos una convención particular utilizada en *Standard Macros* de la librería de Reaktor Core. Los módulos que encontrarás en esta área se entienden mejor descritos en términos de señales de diferentes tipos: audio, control, evento, y lógicas. Hablaremos de las señales de evento y las lógicas más adelante. Por ahora vamos a centrarnos en las de audio y control.

Las señales de audio son, obviamente, las señales que transportan información de audio. Esto incluye las señales tomadas de las salidas de osciladores, filtros, amplificadores, delays, y demás. También los módulos como filtros, amplificadores, saturadores, delays, o similares, normalmente reciben una señal de audio entrante para procesarla.

Las señales de control no transportan audio, se usan simplemente para controlar algunos módulos. Por ejemplo, las salidas de envolventes, LFOs, tonalidad del teclado y velocidad, son señales que no llevan audio, pero se usan para controlar la frecuencia de corte de filtro o resonancia, un tiempo de delay, y demás. Respectivamente, un puerto de entrada del corte o resonancia de filtro, o un puerto de entrada de tiempo de delay son los que reciben las señales de control.

He aquí un ejemplo de un módulo de filtro de Reaktor Core que ya conoces:



La entrada superior del filtro recibirá la señal de audio que ha de filtrarse, y por lo tanto, se espera que sea de audio. Las salidas del filtro producen diferentes tipos de señales de audio, de forma que todas estas señales tienen audio.

Por otro lado, un módulo oscilador de sierra tiene sólo una entrada de control (para la frecuencia) y una salida de audio:



Y si echamos un vistazo al módulo *Rect LFO*, veremos que tiene dos entradas de control – para controlar la frecuencia y la anchura de pulso (la tercera entrada es del tipo evento) – y una salida de control (que se usaría para controlar cosas como el corte de filtro o niveles VCA, etc.):



Algunos tipos de procesamiento, como por ejemplo la mezcla, tienen sentido tanto para señales de audio como de control. En este caso encontrarás versiones de macros dedicadas al procesamiento de audio y versiones dedicadas al procesamiento de señales de control. Por ejemplo, hay mezcladores de audio y mezcladores de control, amplificadores de audio y amplificadores de control, etc. Generalmente no es una buena idea utilizar un módulo para procesar tipos de señales para los cuales no se diseñó, en cuyo caso tendrás que saber lo que estás haciendo.

Dicho lo cual, es importante que sepas que con frecuencia, las señales de audio se pueden re-utilizar como señales de control. El ejemplo más típico sería una señal de audio que modula la frecuencia de un oscilador o el corte de filtro. Esto está perfectamente bien, puesto que en este caso están usando una señal de audio como una de control. Damos por hecho que no vas a tratar de usar una de control por una de audio, lo cual sería bastante extraño.

No has de confundir esta división entre señales de audio, control, evento y lógicas, con la división entre evento/audio del nivel primario de Reaktor. La clasificación evento/audio del nivel primario especifica algo así como la “velocidad de procesamiento”, siendo más “rápido” el procesamiento de las señales de audio y más pesado para la CPU. Como seguramente también sabrás, las señales de evento del nivel primario tienen diferentes reglas de propagación que las de audio. La diferencia entre las señales de audio, control, y evento

en la terminología de Reaktor Core es puramente semántica, definiendo el “significado” de la señal en lugar del tipo de procesamiento. No existe una relación uno-a-uno entre la división audio/evento del nivel primario y la división audio/control/evento/logic de Reaktor Core, pero intentaremos explicarte esta relación de todas formas:

- Una señal de audio de nivel primario normalmente puede corresponderse tanto con una señal de audio de Reaktor Core (por ejemplo la salida de un oscilador, o un filtro de audio) como con una señal de control de Reaktor Core (por ejemplo, la salida de una envolvente).
- Una señal de evento del nivel primario es, normalmente, una señal de control en términos de Reaktor Core. Un ejemplo de dicha señal sería una salida de un LFO, o un Knob, o una fuente MIDI de tono o velocidad.
- Algunas veces, una señal de evento del nivel primario se corresponde con una señal de evento de Reaktor Core. El ejemplo más típico de esto sería una puerta MIDI (como hemos prometido, más tarde describiremos las señales de evento de Reaktor Core)
- Algunas veces, una señal de evento del nivel primario *se asemeja* a una señal lógica de Reaktor Core, aunque no sea completamente compatible y tenga que haber una conversión explícita entre ellas (también hablaremos de esto más adelante). Los ejemplos serían señales procesadas por *Logic AND* o módulos similares del nivel primario.

Es importante comprender que cuando seleccionas un tipo de puerto de entrada para una célula core, escogerás entre señales de audio y evento del nivel primario, no entre señales de audio y evento de Reaktor Core. Los puertos de las células core son el lugar donde los dos mundos se ponen en contacto y por lo tanto, usan parte de la terminología del nivel primario.

Vamos a aprender un poco más sobre este concepto mientras intentamos construir una emulación del efecto eco de cinta. Empezaremos construyendo una simple delay digital y luego la realzaremos para emular las características de una de cinta.

Lo primero será crear una célula core vacía. Ahora puedes meterte dentro y nombrarla como “Echo”.

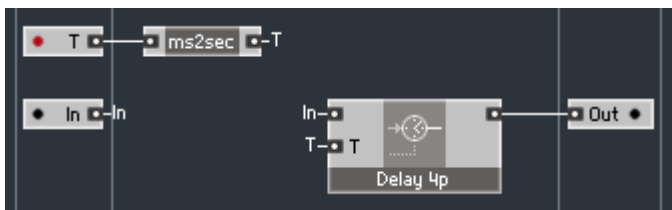
El primer módulo que vamos a poner dentro de la estructura es un módulo de delay. Cogemos un delay interpolador de 4-puntos, porque da mejor calidad que uno 2-puntos y porque un delay no-interpolador no sería apropiado para nuestra emulación de cinta: *Standard Macro > Delay > Delay 4p*:



Obviamente necesitamos una entrada de audio y una salida de audio para nuestra célula core de delay. Usaremos una conexión QuickBus para la entrada y una normal para la salida:

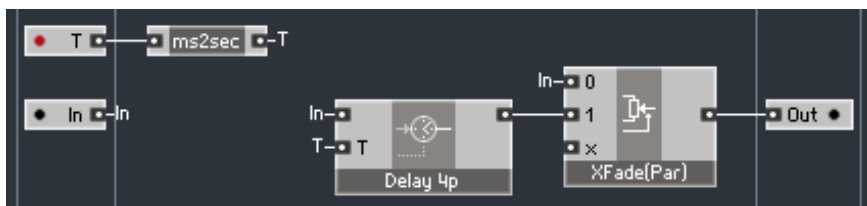


También necesitaremos una entrada de evento para controlar el tiempo de delay. Hay que tener en cuenta que en el nivel primario, el tiempo de delay se expresa normalmente en milisegundos, mientras que en las macros de delay de la librería de Reaktor Core se supone que expresarán en segundos. No hay problema, dispones de un módulo de conversión *Standard Macro > Convert > ms2sec*:

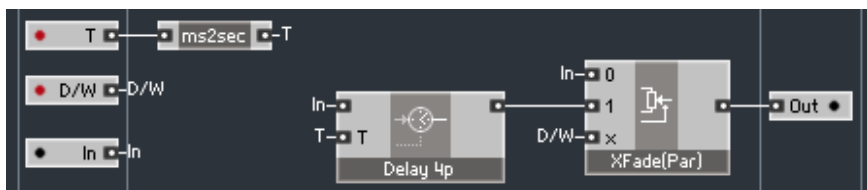


Pero hasta aquí sólo tenemos un simple delay, así que sería genial escuchar la señal original, no sólo la parte del retardo. Para ajustar la señal original a la salida tendremos que mezclarla con la señal que lleva delay.

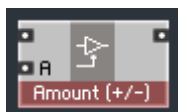
Puesto que ahora estamos mezclando señales de audio, necesitaremos un mezclador de audio (si recuerdas, hemos usado un mezclador de control para mezclar señales de control cuando construíamos una célula core de filtro). Incluso mejor, podemos usar un tipo de mezclador de audio particular, especialmente diseñado para hacer fundidos cruzados entre las dos señales: *Standard Macro > Audio Mix-Amp > XFade (par)*:



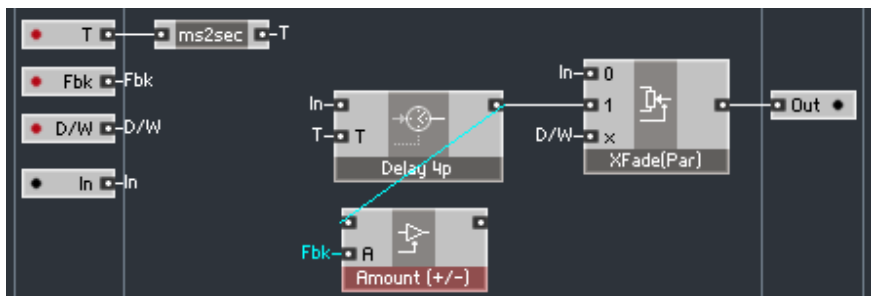
“(par)” está como parabólica – proporcionando un crossfade más natural que uno normal. La entrada de control (“x”) del crossfade todavía está desconectada, así que la conectaremos a una nueva entrada de evento para controlar la mezcla entre la señal con delay y la señal sin procesar. Cuando la señal de control sea 0, sólo escucharemos la señal original; cuando sea 1 sólo escucharemos la parte con delay.



Mucho mejor. Ahora podemos escuchar la señal original y el eco. Pero todavía hay solamente una repetición. Para tener muchos, tendremos que realimentar una fracción de la señal con delay por la entrada del delay. Así que lo primero será atenuar la señal de delay, siguiendo las mismas directrices – usa un amplificador de audio para atenuar la señal de audio – cogemos *Standard Macro* > *Audio Mix-Amp* > *Amount*.

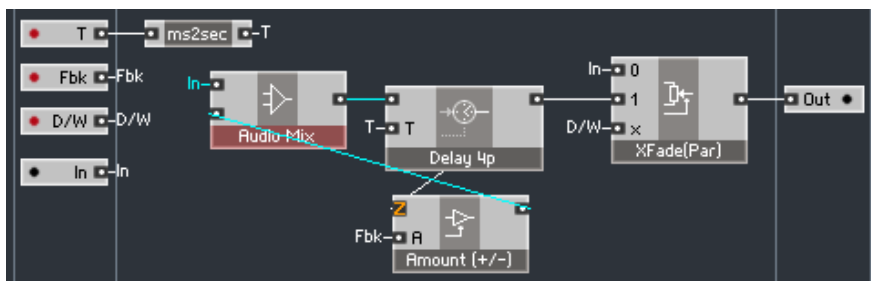


Hemos usado el amplificador *Amount* porque queremos controlar la cantidad de señal retroalimentada. Este amplificador también nos permitirá invertir a señal usando unos ajustes de cantidad negativos. Por ejemplo, un Amp (dB) estaría muy bien para controlar el volumen de la señal, pero en este caso no procede, puesto que no nos permite invertir las señales. Conectaremos la entrada de control de amplitud del amplificador a una entrada de evento para controlar la cantidad de respuesta.



Una cantidad razonable de respuesta en nuestro caso sería [0.9..0.9]. Si quieres probar este delay ya en el siguiente paso, ten cuidado con la cantidad, ya que es muy fácil obtener niveles de señal muy altos (todavía no hay saturación en nuestro circuito). Podríamos haber implantado un cómodo ajustador de cantidad de respuesta dentro de nuestra célula core de delay, pero como aplicaremos algo de saturación más tarde, no va a ser necesario. Por el contrario, tendrías que escuchar niveles altos de respuesta y saturación de delay.

Tendremos que mezclar la señal procesada con la señal de entrante. Un mezclador de audio (*Standard Macro > Audio Mix Amp > Audio Mix*) sería la elección natural:



Te preguntarás qué le ha pasado a la entrada de arriba del módulo *Amount*, que ahora muestra una “Z” naranja gigante.



De hecho, dependiendo de la versión software y otras condiciones, esta “Z” podría aparecer en otra entrada dentro de la estructura, pero no debería preocuparte mucho. Este signo indica que se ha producido una retroalimentación

digital dentro de la estructura y está pensado para el diseño de estructura avanzado, donde dicha información puede ser un indicio importante para quien diseña la estructura.

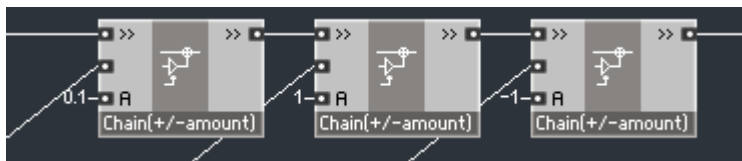
Para estructuras simples como la de arriba, uno no tendría por qué preocuparse de ese signo. Su presencia simplemente muestra que habrá un delay de 1-sample (sobre 0.02 ms a 44.1 KHz, incluso menos a mayores frecuencias de muestreo) en ese punto. Suponemos que si el tiempo de delay estuviese 0.02ms fuera del valor específico, no te percatarías.

Volvamos a nuestra estructura. Ahora puede producir series de repeticiones que decaen. De hecho ya estaría bien para una delay digital, pero queremos enseñarte otra característica de la librería que puedes usar como truco para hacer más pequeña tu estructura.

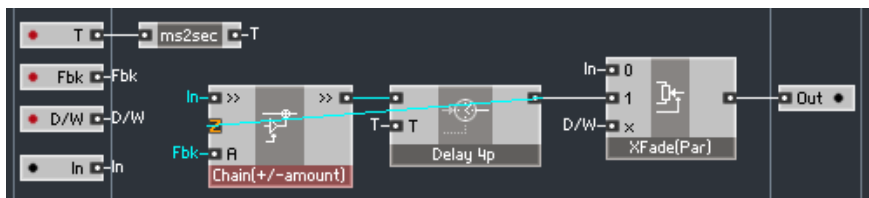
Entre los amplificadores de audio hay unos llamados “Chain”. Estos amplificadores son capaces de amplificar una señal y mezclarla en otra señal con “Chain”. Uno de ellos es el amplificador *Audio Mix Amp > Chain(amount)* que funciona de forma similar al *Amount* excepto porque adicionalmente también hace una mezcla “asociada”



La señal de la segunda entrada de este módulo se atenuará de acuerdo con la cantidad dada en la entrada “A” y se mezclará con la señal en la entrada asociada (“>>”). La señal de la entrada asociada no está atenuada. Estos amplificadores se pueden usar para construir enlaces de mezcla, donde las conexiones de los puertos “>>” se comportan como una especie de bus de mezcla:



En nuestro caso no necesitamos un bus de mezcla, pero podemos usar este módulo para reemplazar nuestros módulos *Audio Mix* y *Amount*. La señal retroalimentada se atenuará por la cantidad especificada por la entrada Fbk y se mezclará en la señal de entrada exactamente como antes:

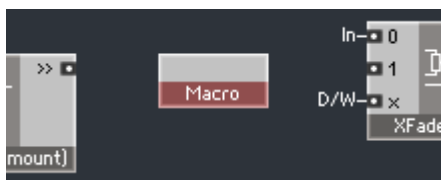


Felicidades. Has construido un efecto de delay digital simple. En el siguiente paso añadiremos un matiz de cinta.

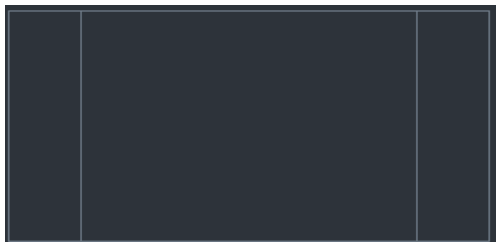
Construye tus primeras macros de Reaktor Core

En el efecto de delay que acabamos de construir, hemos usado una macro de *Delay 4p* de la librería, que nos proporciona un delay digital de calidad razonablemente alta. Con o sin alta calidad, lo cierto es que sigue sonando demasiado digital. Podríamos darle más calidez añadiendo distintas características que podemos encontrar en un delay de cinta, como saturación o agitación. Así pues podemos sustituir la macro *Delay 4p* por una macro delay de cinta que vamos a construir ahora.

Borraremos pues la macro de delay de la estructura y crearemos uno vacío en su lugar. Haz clic con el botón derecho sobre el fondo de pantalla y selecciona *Built-In Module > Macro*:

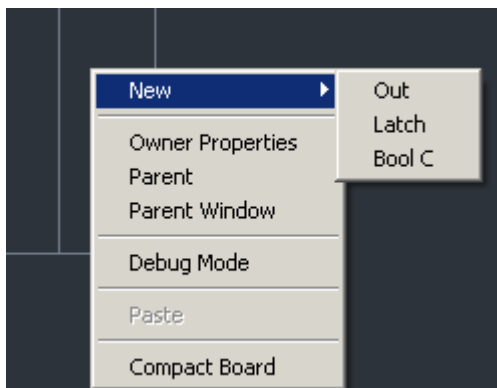


Haz doble clic sobre él para entrar. Vas a ver una estructura vacía similar a la anterior.



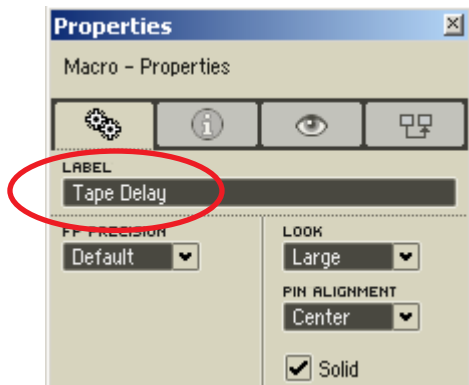
También funciona de manera similar, pero hay algunas diferencias importantes. Tras las anteriores había una estructura de una célula Reaktor Core, mientras que esta es una estructura interna de una macro de Reaktor Core.

De hecho estas diferencias tienen que ver con un set de módulos de entrada y salida disponibles. Son simplemente diferentes:



Los tipos de puertos *Latch* y *Bool C* los explicaremos más adelante en el manual, ya que se usan para asuntos avanzados. Ahora estamos interesados sobre todo en el primero, llamado “Out” (o “In” para las entradas). Es un tipo de puerto general que puede aceptar señales de audio, control, evento, o lógicas. Esta diferencia es importante para ti como usuario, ya que describe cómo se va a usar la señal, pero para Reaktor Core son iguales. No hay diferencia entre audio/evento/ entradas/salidas como en la estructura anterior, ya que ahora no tenemos señales de nivel primario de Reaktor fuera, sino que es puro Reaktor Core.

Lo primero que vamos a hacer es ponerle nombre a la macro. Puedes hacerlo de la misma forma que las células core, haciendo clic con el botón derecho sobre el fondo de pantalla y seleccionando *Owner Properties*, donde podrás escribir el nombre:



Las propiedades restantes de la macro controlan varios aspectos de la apariencia y procesamiento de la señal.

Aunque eres libre de experimentar con el resto de las propiedades, te advertimos que no apagues el parámetro Solid, y que cambies la FP Precision con moderación. El significado de estos parámetros se describirá en la sección avanzada de este manual.

Lo siguiente será crear un grupo de entradas y salidas para nuestra macro *Tape Delay*:

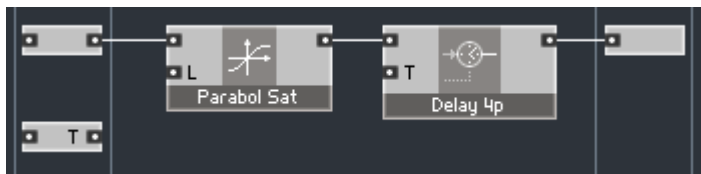


La entrada superior recibirá la entrada de audio, la inferior recibirá el parámetro de tiempo. Quizá hayas notado que hay puertos extra en la parte izquierda de los módulos de entrada; hablaremos de ellos más tarde.

Como parte central de nuestra macro usaremos el mismo módulo *Delay 4p*:



Se puede hacer una emulación del efecto de saturación fácilmente, simplemente conecta el módulo saturador antes del delay. El saturador es un tipo de modulador de señal, así que lo buscaremos entre los moduladores de audio (puesto que es saturador de audio). *Standard Macro > Audio Shaper > Parabol Sat*:

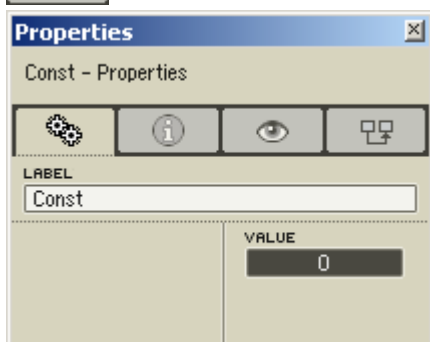


Ahora la señal de entrada se saturará en un rango de $-1..+1$. De hecho, este campo se controla por la entrada "L" del módulo saturador. Si está desconectada se pone en un valor por defecto de 1. Quizá te sorprendas, ya que

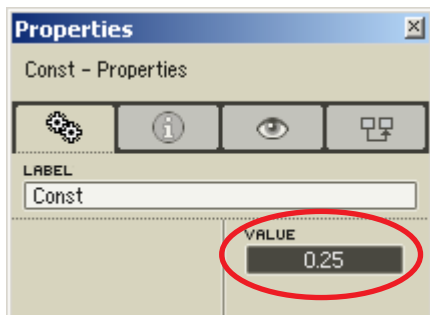
estarás acostumbrado a que las entradas desconectadas se traten como si no recibiesen señal, o dicho de otra forma, señal cero. Bien, no es exactamente el caso en las estructuras Reaktor Core. Los módulos tienen la posibilidad de especificar un tratamiento especial de las entradas desconectadas, como este saturador, que especifica que la entrada “L” tenga un valor por defecto de 1. Ahora vamos a aprender a hacer lo mismo, especificando un nuevo valor por defecto para nuestra entrada “T”. Digamos que si esta entrada “T” estuviese desconectada, nos gustaría que se tratase como si su valor fuese de 0.25 seg. Muy fácil. Haz clic con el botón derecho sobre el puerto a la izquierda del módulo de la entrada “T” y selecciona *Connect to New QuickConst*. Esto es lo que tendrías que ver:



Además, la ventana de propiedades debería mostrar las propiedades de esta constante (si muestra otra página, comprueba que has presionado el botón



En el campo de valores teclea un nuevo valor de 0.25:

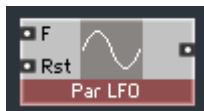


Así es como el QuickConst tendría que verse ahora en la estructura:



Te explicaremos lo que hemos hecho. El puerto de la izquierda del módulo de entrada especifica la llamada “señal por defecto”. Esto significa que la entrada no está conectada (en la parte externa de la macro). Esta señal por defecto se tomará como fuente de entrada. En nuestro caso, si la entrada T de la macro Tape Delay no estuviese conectada, se comportaría como si tuviese conectado un valor constante de 0.25.

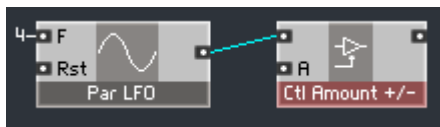
Una conexión al QuickConst no es, por supuesto, la única conexión posible para esta entrada de señal por defecto. Puedes conectarla a cualquier otro módulo en la estructura, incluyendo otros módulos de entrada. Ahora que ya tenemos saturación y un valor por defecto de la entrada “T”, vamos a emular la agitación del efecto de cinta. Una sencilla manera de hacerlo es modulando el tiempo de delay con un LFO. Puedes experimentar con diferentes moduladores LFOs para conseguir un efecto mejor, pero te sugerimos que ahora cojas cualquiera de los disponibles en la librería: *Standard Macro > LFO > Par LFO*:



Es un LFO con forma parabólica, que produce una señal similar en la forma a la senoidal, pero necesita menos CPU. Su entrada “F” ha de recibir una señal especificando la frecuencia de oscilación. Podemos usar un QuickConst otra vez. Una frecuencia de 4 Hz parece razonable:



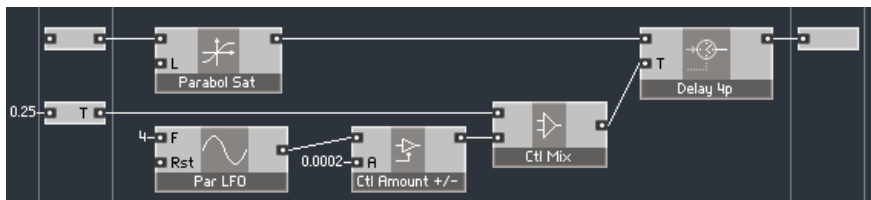
La entrada “Rst” se usa para reiniciar el LFO, no lo necesitamos por ahora. Tenemos que especificar una cantidad de modulación y aplicarla a la salida del LFO, ya que actualmente la señal de salida del LFO varía en el campo -1..1 y eso es mucho. Recuerda que estamos trabajando con señales de control, por lo que vamos a usar un módulo de cantidad de control, similar al amplificador *Amount* que hemos usado para el audio. *Standard Macro > Control > Ctl Amount*:



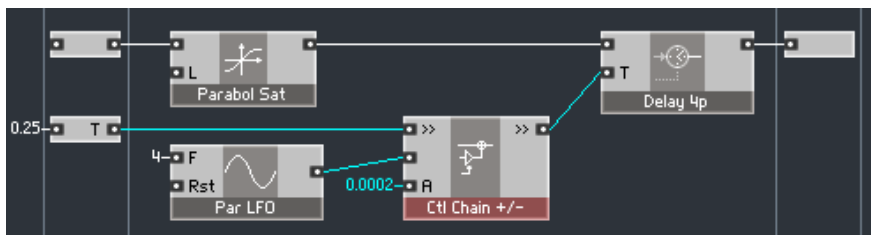
Una amplitud de modulación de 0.0002 estaría bien, así que cambiaremos la señal a esa cantidad:




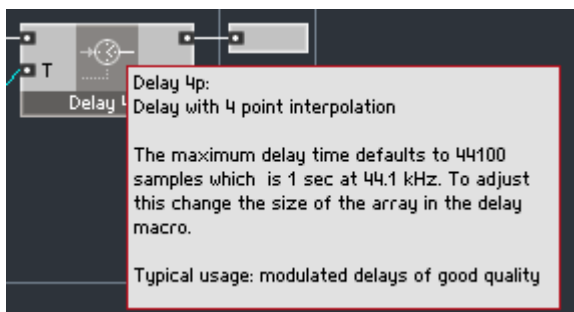
Para acabar podemos mezclar las dos señales de control (la de la entrada “T” y la del módulo *Ctl Amount*) y las incluiremos en la entrada “T” del módulo de delay. Aquí podemos volver a usar un familiar módulo *Ctl Mix*:



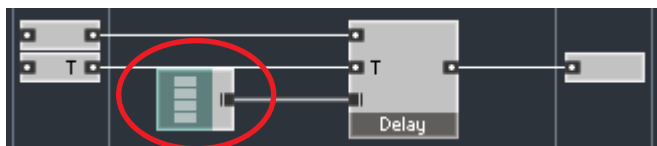
De hecho, tenemos un mezclador de control de tipo “asociado”, similar al que hemos usado para las señales de audio. Podríamos usarlo para reemplazar los módulos *Ctl Amount* y *Ctl Mix* igual que hemos hecho antes en las rutas de audio. *Standard Macro > Control > Ctl Chain*:




Un último toque para nuestra macro – vamos a cambiar el tamaño del buffer para el delay, que define el tiempo máximo de delay posible. Si mantienes el cursor del ratón sobre la macro Delay 4p (y compruebas que el botón  está conectado), podrás leer en el texto que el tamaño del buffer por defecto corresponde a 1 seg. de delay a 44.1 KHz:

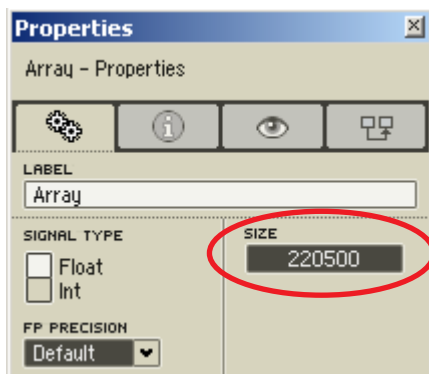


Vamos a incrementar la cantidad a 5 seg. $44100 \times 5 = 220500$ (cada sample emplea 4 bites, lo que haría $220500 \times 4 = 882000$ bites, que es casi 1 MB). Haz doble clic sobre la macro *Delay 4p*:



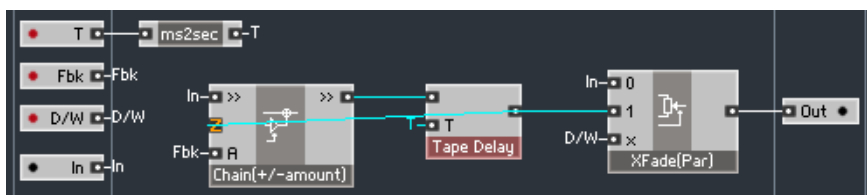
El módulo de la izquierda es el módulo de buffer de delay. Haz doble clic sobre él (o clic con el botón derecho y selecciona Show Properties) para editar sus propiedades.

Selecciona la página  en la que deberías encontrar la propiedad *Size*. Cámbiala a 220500 samples.

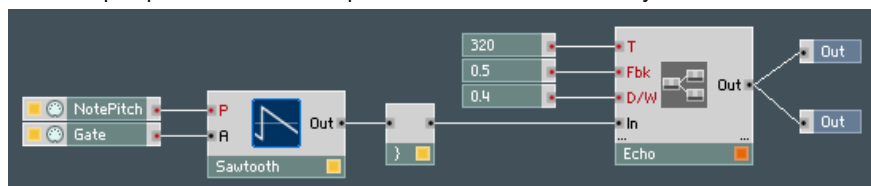


Como hemos visto, un buffer de delay de 5 segundos emplea casi 1 MB de memoria, así que ten cuidado al cambiar los parámetros. Sobre todo si los delays se van a usar en áreas polifónicas de la estructura, donde el tamaño del buffer se multiplicará por el número de voces.

Ahora podemos salir de la macro *Delay 4p* y luego de la macro *Tape Delay* que acabamos de crear (haz doble clic sobre el fondo de pantalla) para realizar las conexiones externas:



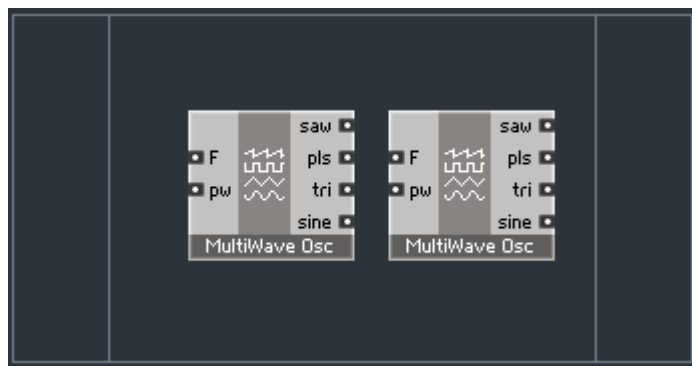
Si todavía no lo has hecho, te sugerimos que pruebes el módulo de eco que hemos construido. He aquí una estructura de prueba del nivel primario, lo más simple posible (observa que el módulo Echo está ajustado *mono*):



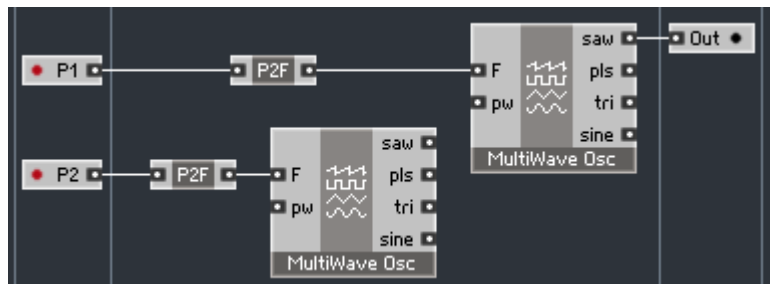
También puedes realzarlo de diversas maneras, por ejemplo, aplicando knobs que controlen los parámetros del eco, o utilizando un sintetizador real como fuente de la señal, etc.

El audio como señal de control

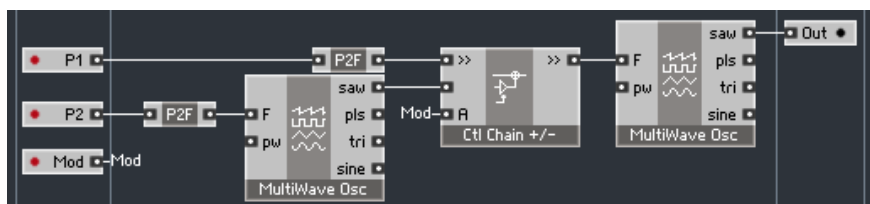
Ya hemos mencionado antes que se puede usar una señal de audio como una de control. Hemos pensado que podríamos darte un ejemplo de ello. Vamos a crear una célula de Reaktor Core implementando un par de osciladores donde uno modulará al otro. Cojamos dos osciladores multi-onda:



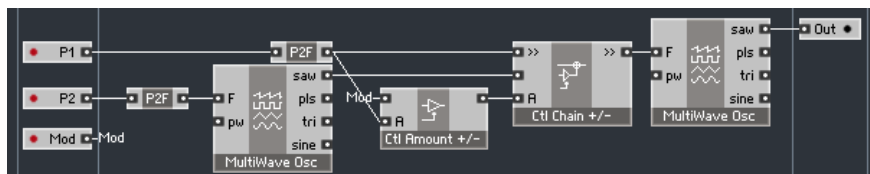
Necesitaremos control de tono para ambos osciladores, y lo dirigiremos a la salida del segundo, así que vamos a crear las entradas y salidas necesarias:



Ahora queremos usar la salida del oscilador de la izquierda para modular la frecuencia del de la derecha:



La entrada *Mod* controla la cantidad de modulación. Observa que estamos mezclando la señal de modulación después del convertidor *P2F*, de modo que la modulación tomará forma en la escala de frecuencia (también se puede modular en la escala de “tono”). En realidad es mejor escalar la cantidad de modulación de acuerdo con la frecuencia base de oscilación:



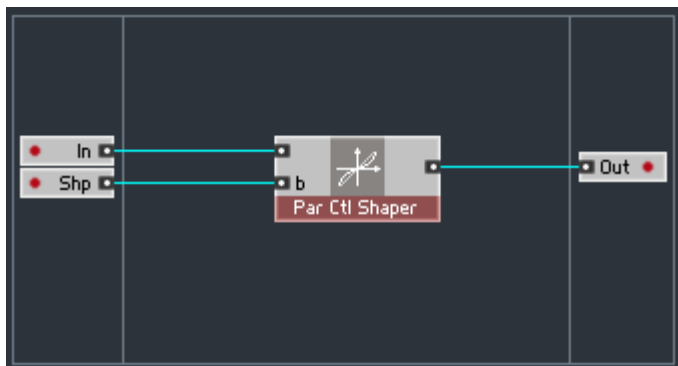
Ahora, si analizas la estructura de arriba desde el punto de vista de señales de audio y control, comprobarás que todas las señales de la estructura excepto las salidas de los osciladores son señales de control. Las salidas de ambos osciladores son, obviamente, señales de audio. Sin embargo, estamos “usando indebidamente” la salida del oscilador de la izquierda como señal de control, desde el momento en que la estamos conectando al mezclador *Ctl Chain*.

Event signals

Tal y como hemos dicho antes, hay diferentes significados para las palabras “señal de evento”. Ya deberías estar familiarizado con las señales de evento del nivel primario de Reaktor. Una señal de evento de nivel primario puede tener varias formas de uso. Una de ellas es como señal de control (por ejemplo, salida LFO, salida knob, etc), simplemente porque utiliza menos CPU que las señales de audio de nivel primario. En este caso, seguramente podrías haber utilizado una señal de audio consiguiendo más o menos el mismo efecto. Otra idea es usarlas cuando el audio no puede funcionar, y es precisamente aquí donde no sólo estás interesado en el valor de lo que transporta la señal, sino también en los momentos “*en los que*” el nuevo valor se envía por el cable. En otras palabras, *cuándo* está siendo enviado *el evento*. Un ejemplo de esto sería una señal de la puerta de envolvente del nivel primario; la envolvente se

activará en el momento en que el evento *llegue* a la entrada de la puerta. . Cuando hablábamos de señales de audio, control, evento, y lógicas en Reaktor Core, realmente no estábamos hablando técnicamente de tipos de señales distintas (técnicamente son iguales en Reaktor Core), sino que nos referíamos a diferentes formas de uso de una señal. Como podemos ver ahora, una señal de evento del nivel primario de Reaktor se puede usar como señal de control, de evento, o incluso como señal lógica (teniendo en cuenta que una señal de audio del nivel primario de Reaktor se puede usar como señal de audio o de control – como recordarás de nuestros pasos previos).

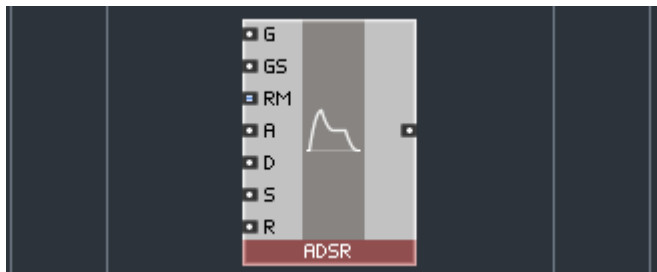
Ya hemos aprendido a alimentar señales de evento del nivel primario dentro de las estructuras de Reaktor Core y a usarlas como señales de control. Las entradas en modo evento de una célula core de *audio* implementando un filtro que hemos construido antes, es un buen ejemplo de ello. También hay casos en los que tendrías que usar una célula core de *evento* para procesar señales de evento del nivel primario, que de hecho son señales de control. Aquí tienes un ejemplo de una célula core de evento que envuelve una macro core de modulación de control:



Este modulador recibe una señal de control de frecuencia del evento desde el nivel primario (por ejemplo, una señal MIDI de velocidad, o una señal LFO del nivel primario), cambia de acuerdo con el parámetro “Shp” y reenvía el resultado a la salida.

Una restricción importante de las células core de evento mencionadas anteriormente es que todas las fuentes de reloj están deshabilitadas en su interior. Esto significa que dentro de las células core de evento no funcionan ni osciladores, ni filtros, ni envolventes ni LFOs. Estos módulos están limitados a recibir eventos del nivel primario de Reaktor, procesándolos y enviándolos fuera, de forma parecida a los ejemplos que acabamos de ver.

De forma alternativa, las señales derivadas del exterior de los eventos del nivel primario, se pueden usar como auténticas señales de evento dentro de las estructuras de Reaktor Core. Ahora vamos a echar un vistazo a un par de casos sencillos en los que usar eventos dentro de Reaktor Core. Como habrás deducido por la restricción de las células core de evento que describíamos antes, ha de ser una célula de audio. Así que vamos a crear una nueva célula core de audio usando *Standard Macro > Envelope > ADSR*:

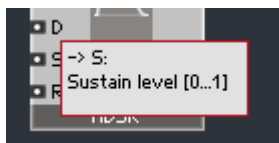


La entrada superior de la envolvente es una entrada de puerta que funciona de forma similar a las puertas de las envolventes del nivel primario, es decir, abre o cierra la envolvente en respuesta a los *eventos* entrantes. No hay problema, creamos una entrada de evento para nuestra célula core:

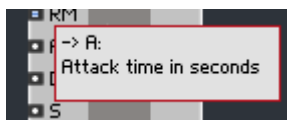


Esta entrada volcará los eventos de puerta de nivel primario entrantes dentro de los eventos core.

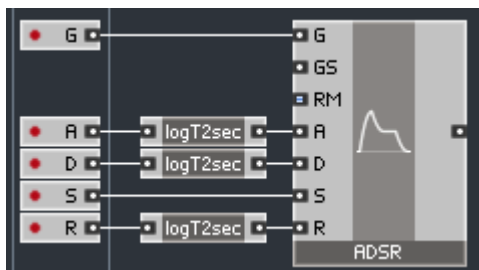
Ahora vamos a fijarnos en las entradas A, D, S y R. La entrada “S” (nivel de sostenido) funciona de forma parecida al primer nivel, es decir, espera que la señal entrante esté en el campo 0...1:



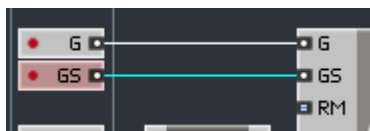
Las entradas A, D y R son diferentes en cuanto que, al contrario que las envolventes del nivel primario, esperan que el tiempo esté especificado en segundos:



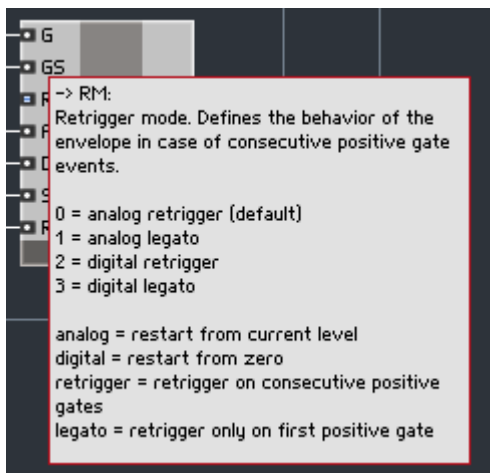
Esto se puede solucionar usando *Standard Macro > Convert > logT2sec*, , que convierte los tiempos de la envolvente del nivel primario en segundos:



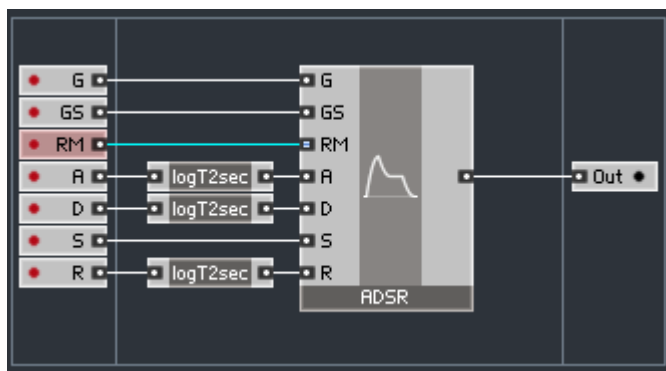
Al margen de que todas las entradas de la estructura superior están en modo de evento, desde el punto de vista semántico, la primera de estas entradas produce una señal de evento, mientras que las otras producen señales de control. Nuestra envolvente todavía tiene dos puertos sin conectar. El puerto GS ajusta la cantidad de sensibilidad de la puerta. A 0, la envolvente ignorará por completo el nivel de la puerta y estará a una amplitud completa. A 1, el nivel de la puerta tendrá un efecto máximo, como en el nivel primario de Reaktor. Podremos controlar esta cantidad desde el exterior a través de otra entrada:



El puerto *RM* especifica el modo de reactivación de la envolvente:

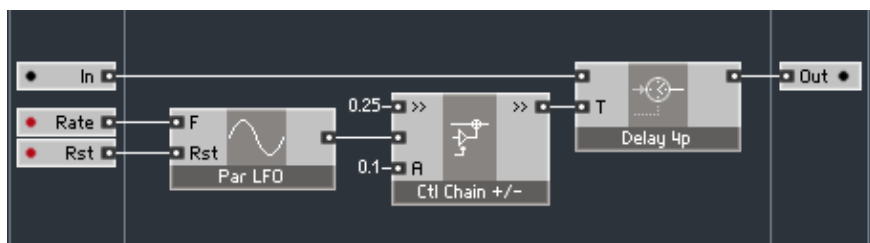


El aspecto de este puerto es diferente de los otros, ya que espera valores de números enteros en la entrada. Normalmente, si consideramos por ejemplo una entrada de tiempo de ataque, podría esperar un tiempo de ataque de 1 segundo, o de 1.5 segundos, o de 0.2 segundos. Por el contrario, el puerto en modo de reactivación, no espera un valor de 1.2 o 3.1. Viene indicado por su aspecto diferente, pero esto no significa que no podamos conectar nuestras señales normales a este puerto, simplemente tendríamos que usar otra entrada de evento:



Si los valores que entran desde el exterior no fuesen exactamente números enteros, pero de alguna forma, se acercasen a un valor entero, tomaría el valor entero. Por ejemplo, 1.2 se consideraría como 1.

Ahora echemos un vistazo a otro ejemplo en el uso de una auténtica señal de evento:



La estructura de arriba implementa un efecto de modulación de tono. El efecto está producido por un delay cuyo tiempo varía en un campo de 250 ± 100 ms. La frecuencia de esta variación está controlado por la entrada Rate, encargada de controlar la frecuencia del LFO de modulación (el valor está en Hz); una auténtica señal de control. La entrada Rst es una auténtica señal de evento que se puede usar para reiniciar el LFO. El valor entrante especifica la fase de reinicio, donde 0 reiniciaría el LFO al principio del ciclo, 0.5 en el medio y

1 al final. Puedes probarlo conectando un botón que envíe un valor específico a esta entrada..

Señales lógicas

Ahora que ya hemos aprendido algo sobre las señales de evento y de control, vamos a ocuparnos de otro uso diferente de las señales; las señales lógicas. He aquí un ejemplo de un módulo que está procesando señales lógicas:



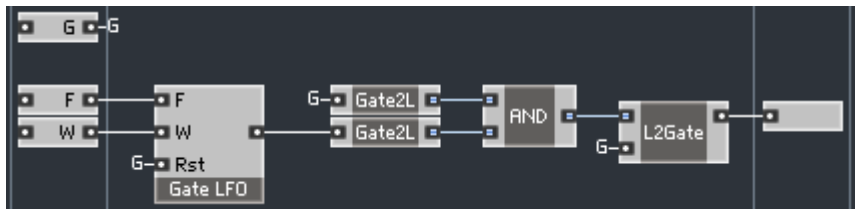
Como puedes ver, los puertos de este módulo tienen datos con valores enteros, como la entrada *RM* de la envoltura que hemos visto antes. Esto tiene que ver con el hecho de que generalmente este tipo de señales sólo transportan valores enteros, incluso “peor”, sólo transportan valores de 0 y 1.

El valor 1 representa la condición “verdadera” de la señal lógica y el valor 0 representa la condición “falsa” de la señal lógica. El significado de verdadero y falso tiene que especificarlo el usuario. Por ejemplo, podrías tener una señal lógica que te diga si una puerta en particular está abierta:



Aquí, una macro *Gate2L* comprueba la señal de puerta entrante y produce una salida verdadera (1) si la puerta se abre, y falsa (0) si se cierra.

Por lo tanto, las señales lógicas se usan para “procesamientos lógicos”. Por ejemplo, podríamos construir un procesador de puerta, que podría aplicar un “bloqueo de puerta” regular sobre una puerta MIDI:

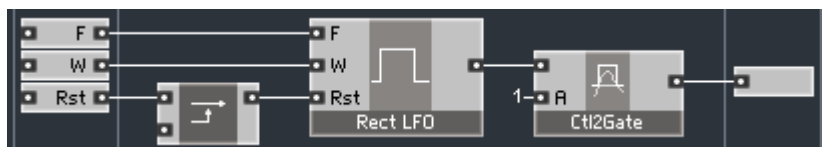


Los módulos *Gate2L*, *AND* y *L2Gate* son módulos lógicos y se pueden encontrar en el menú *Standard Macro > Logic*. El *L2Gate* es una macro que hemos construido para este procesador. Genera una señal de puerta, abriéndose y cerrándose a intervalos regulares.

La entrada de la puerta y la salida del LFO están conectadas a los convertidores *Gate2L*, que convierten las señales de puerta en señales lógicas, transformando las puertas abiertas en *verdaderas*, y las cerradas en *falsas*.

El módulo AND enviará una señal verdadera sólo si ambas puertas están abiertas a la vez. En otras palabras, la salida del módulo AND será verdadera única y exclusivamente si el usuario mantiene una tecla, y al mismo tiempo, el LFO devuelve una puerta abierta. Esto significa que mientras el usuario mantiene la tecla, se alternarán valores verdaderos y falsos en la salida del módulo AND, siendo definida la velocidad de la alternación por la frecuencia del LFO. La salida del módulo AND se procesa de nuevo en la señal de la puerta, y la amplitud de la señal de la puerta se toma de la entrada de la puerta, de modo que el nivel de la puerta se mantiene inalterado.

Aquí tienes un ejemplo de nuestra macro Gate LFO:



La entrada F define la frecuencia de repeticiones de la puerta; la entrada W define la duración de las puertas abiertas (a 0 están al 50% del período de la puerta, a -1 están al 0% y a 1 al 100%), y la entrada Rst reinicia el LFO en respuesta a los eventos entrantes (así el LFO se reiniciará cada vez que haya un evento de puerta en la entrada principal de la puerta).

El módulo conectado a la entrada Rst del Rect LFO se llama *Value* y se puede encontrar en *Standard Macro > Event Processing*. Comprueba que el LFO se reinicia en la fase cero reemplazando todos los valores de los eventos entrantes por el valor de la entrada de abajo, que es cero. La salida del LFO se convierte en una señal de puerta usando un convertidor *Ctl2Gate*, que también podrás encontrar en *Standard Macro > Event Processing*.

Como recordarás, los LFOs no funcionan dentro de las células core de evento, de forma que si quieres probar esta estructura tendrás que usar una célula core de audio.

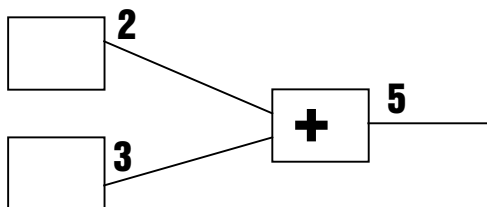
Fundamentos de Reaktor Core:

modelo de señal corel

Valores

Muchas salidas de los módulos de Reaktor Core producen valores. “Producir” un valor significa que en cualquier momento del tiempo hay un valor asociado a esa salida. Este valor está disponible para los módulos cuyas entradas estén conectadas a esta salida.

En el siguiente ejemplo, un módulo de suma recibe valores de 2 y 3 desde los otros dos módulos cuyas salidas están conectadas a sus entradas y producen un valor de 5 en su salida.

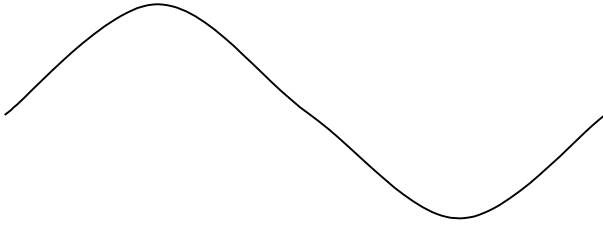


Si quieres imaginar una similitud en el mundo del hardware, piensa en los valores como niveles de señal (voltajes), sobre todo si trabajas con módulos de gran escala como osciladores, filtros, envolventes, etc. No obstante, los valores no se limitan a esto. Después de todo, son valores, y se pueden usar para implementar cualquier algoritmo de procesamiento, no necesariamente el modelado de voltajes.

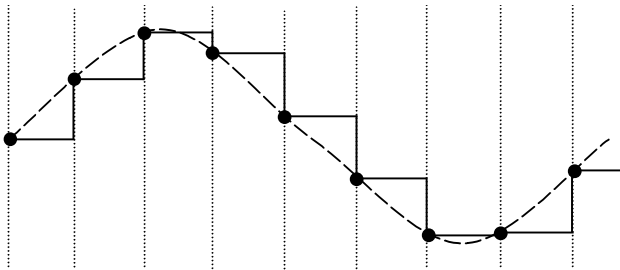
Eventos

En el mundo digital, el tiempo no es continuo, sino discontinuo. Probablemente, el ejemplo más típico es que una grabación digitalmente almacenada no guarda la información completa sobre una señal de audio que está cambiando constantemente en el tiempo, sino que sólo registra la información sobre el nivel de la señal en puntos del tiempo regularmente espaciados de modo discontinuo. El número de estos puntos por segundo es la famosa *frecuencia de muestreo*.

He aquí un dibujo de una señal continua:

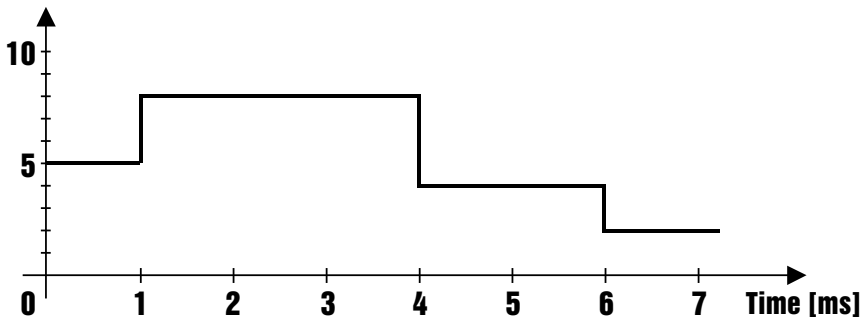


Y su representación digital:

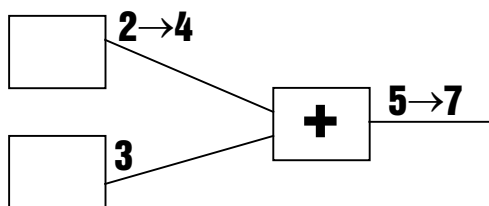


Esto significa que puesto que *estamos* en el mundo digital, las salidas de nuestros módulos no pueden cambiar sus valores continuamente. Por otro lado, no tenemos que limitarnos a forzar las salidas para que cambien sus valores en “puntos del tiempo regularmente espaciados”, es decir, que no tenemos que mantener una frecuencia de muestreo regular sobre todas nuestras estructuras. Lo que es más, en ciertas áreas de nuestras estructuras no tenemos por qué mantener ninguna frecuencia de muestreo en absoluto, es decir, que nuestros cambios no tienen por qué ocurrir en “puntos del tiempo regularmente espaciados”.

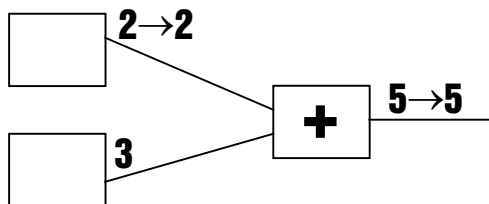
Por ejemplo, en un tiempo de cero, la salida de nuestro módulo de suma tenía un valor de 5. El primer cambio podría ocurrir en un tiempo de 1 milisegundo. El segundo cambio en 4 ms. El tercero en 6 ms:



En el dibujo superior podemos ver los cambios de la salida de nuestro módulo de suma ocurriendo durante un tiempo de 0 a 7 ms. En el momento en que la salida cambia su valor, se genera un evento. Un *evento* significa que la salida “informa” sobre un cambio en su estado, es decir, que tiene un nuevo valor. En el siguiente ejemplo, el módulo superior de la izquierda ha cambiado su valor de salida de 2 a 4, generando un evento. En respuesta, el módulo de suma también cambiará su valor de salida y también generará un evento en su salida.



De forma alternativa, el módulo superior de la izquierda podría haber generado un nuevo evento con el mismo valor que el anterior. El módulo de suma hubiera igualmente generado un nuevo evento, obviamente sin cambios en su valor de salida.



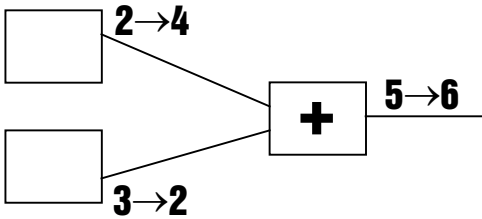
El nuevo valor que aparece en la salida no tiene porqué ser diferente del anterior. No obstante, la única forma de que una salida cambie su valor es generando un evento.

Como has podido ver en los ejemplos anteriores, si un evento tiene lugar en la salida de un módulo, será detectado—módulos conectados, que en respuesta producirán más eventos (recuerda que el módulo de suma producía un evento de salida en respuesta al evento de entrada). Estos nuevos eventos, también serán detectados por los módulos conectados a las salidas correspondientes y se propagarán más y más, hasta que la propagación se detenga por una de las razones que mencionaremos más tarde en este manual.

Los eventos en Reaktor Core no son iguales que en el nivel primario de Reaktor. Se comportan de acuerdo a unas reglas diferentes, que explicaremos más adelante.

Simultaneous events

Considera la siguiente situación: los dos módulos de la izquierda en los ejemplos anteriores producen un evento *simultáneamente*:

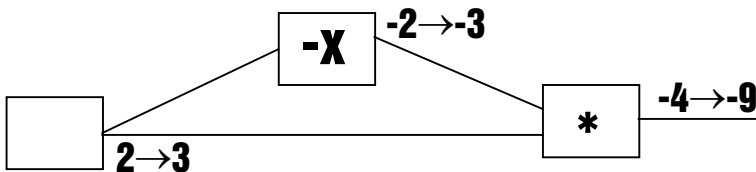


Esta es una de las características del modelo de evento de Reaktor Core – los eventos se pueden dar simultáneamente en distintos lugares. En esta situación, los eventos originados en los módulos de la izquierda, llegarán a las entradas del módulo de suma simultáneamente. Como resultado, el módulo de suma producirá en respuesta exactamente *un* evento de salida.

Esto no es igual que en el nivel primario de Reaktor, donde los eventos no pueden suceder simultáneamente y el módulo de suma (en modo evento) produciría dos eventos de salida en dicha situación.

Por supuesto, en la realidad los eventos no se producen simultáneamente por los módulos de arriba de la izquierda y la derecha, porque ambos módulos serían procesados por la misma CPU y la CPU sólo puede procesar un módulo a la vez. Pero lo importante para nosotros es que esos eventos son *lógicamente simultáneos*, que los módulos que los reciben los tratarán de forma simultánea.

Ahora vamos a tomar otro ejemplo de propagación simultánea de eventos:



En el ejemplo de arriba, el módulo que hay a la izquierda del todo está enviando un evento, cambiando su valor de salida de 2 a 3. El evento se envía simultáneamente tanto al módulo negador ($-x$), como al multiplicador ($*$). En respuesta al evento entrante, el módulo negador producirá un nuevo valor de salida -3 . Esto es importante para que veas que aunque el evento de salida

del módulo negador se ha producido en respuesta al evento enviado por el módulo de la izquierda, y que este acontecimiento debe ser posterior al evento de entrada, ambos eventos son lógicamente simultáneos. Esto significa que llegan simultáneamente a las entradas del módulo multiplicador, y que éste produce de nuevo un solo evento de salida con un valor de -9 .

En el nivel primario hubieras tenido dos eventos en la salida del módulo *Event Mult*. Tampoco estaría definido si el acontecimiento del módulo de la izquierda se envía antes al módulo de inversión o al multiplicador (aunque esto es irrelevante para dicha estructura).

En general, podrías seguir la siguiente regla para saber si determinados eventos son simultáneos o no:

Todos los eventos originados por (en respuesta a) el mismo evento son simultáneos. Todos los eventos originados desde un número arbitrario de eventos simultáneos (que ocurren en diferentes salidas pero se saben simultáneos) también son simultáneos.

El último ejemplo muestra el uso de la simultaneidad del evento. En este caso, eliminamos el proceso redundante del segundo evento con el módulo multiplicador, que hubiera llevado más tiempo a la CPU. En estructuras más grandes, con ausencia de dicho concepto, el número de eventos podría crecer al galope de forma descontrolada, a menos que el diseñador de estructuras ponga un especial cuidado en mantener reducido el número de copias de eventos.

Además de ahorrar tiempo a la CPU, este concepto encabeza las importantes diferencias en cuanto al mecanismo de construcción de estructuras, sobre todo en el caso de estructuras que implementan algoritmos DSP de bajo nivel. Comprenderás mejor estas diferencias cuando empieces a construir tus propias estructuras.

Orden de procesamiento

Como habrás visto en los anteriores ejemplos, cuando un módulo envía un evento, los módulos receptores responderán a este evento. A partir de esta idea, uno podría concluir que al margen de producir eventos “lógicamente simultáneos”, los módulos en realidad no se procesan simultáneamente. Es más, uno podría pensar que para una conexión concreta sería razonable procesar el módulo emisor de dicha conexión, antes del módulo receptor. Así que si son estas las conclusiones a las que has llegado, estás en lo cierto.

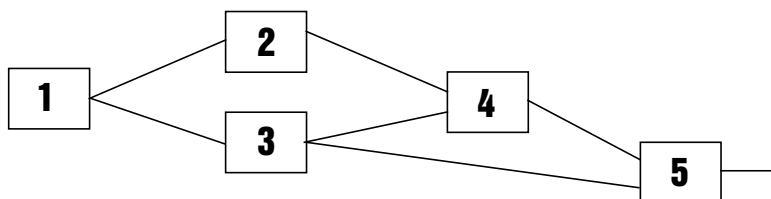
La regla general de orden de procesamiento de los módulos es la siguiente:

Si dos módulos conectados están procesando eventos “simultáneos”, el módulo emisor se procesará primero. Si los eventos no son simultáneos, el orden de procesamiento de los módulos será el orden de los eventos procesados, por supuesto.

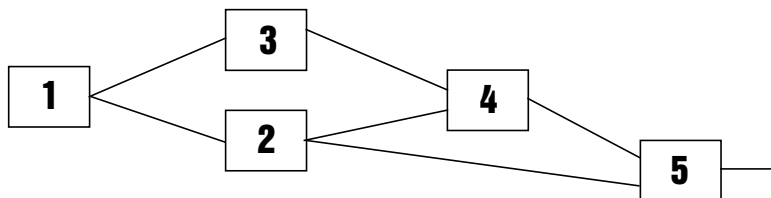
De la regla anterior podremos concluir que mientras haya una ruta de conexión de una-dirección (sea en un sentido o en el otro) desde un módulo a otro, entonces existirá un orden de procesamiento definido entre esos módulos, y que de hecho, será el emisor el que se procesará primero.

Si no hay una ruta de conexión de una-dirección entre los módulos, su orden de procesamiento, uno con respecto al otro, no está definido (para eventos simultáneos). Esto significa que este orden será arbitrario y podrá cambiar en cualquier momento. El diseñador de la estructura tendrá que tener cuidado de que esta situación se dé sólo en módulos cuyo orden de procesamiento relativo no tenga importancia, lo que normalmente será en casos en los que no esté implicada una conexión OBC (mira abajo).

He aquí un ejemplo. Los dígitos muestran el orden de procesamiento del módulo:

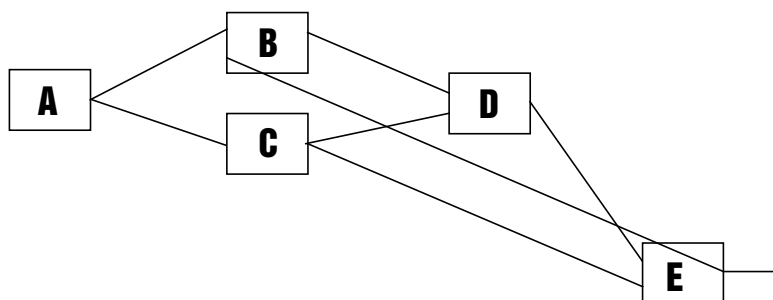


Para esta estructura, existe una alternativa válida de orden de procesamiento:



No hay que decir cuál es el que escogerá el software. Afortunadamente, mientras no uses conexiones OBC, que discutiremos más adelante, el orden relativo de los módulos en estos casos no tiene ninguna importancia.

Las reglas que acabamos de exponerte sobre el orden de procesamiento de los módulos no se pueden aplicar si hay retroalimentación (feedback) en las estructuras, ya que en ese caso, para cualquier par de módulos implicados en el loop de feedback, no podremos decir cuál de ellos es el emisor. El problema sobre la implicación del feedback y su orden de procesamiento, lo veremos más adelante.



En la estructura de arriba no es posible definir si, por ejemplo, el módulo B alimenta al módulo D o viceversa. Obviamente existe una conexión con sentido ascendente desde D hasta B, pero también hay una desde B hasta D (vía E).

Revisión de las células core de evento

Vamos a echar un vistazo a las células core de evento, desde el punto de vista de la definición concreta del concepto de evento de Reaktor Core.

As you'll remember, event core cells have event inputs and event outputs. Como recordarás, las células core de evento tienen entradas de evento y salidas de evento. Estas entradas y salidas son los puntos de conexión entre el nivel primario y el nivel Core de Reaktor. Llevan a cabo esta finalidad realizando conversiones entre los eventos del nivel primario y los eventos core (y viceversa). Las reglas de esta conversión son las siguientes:

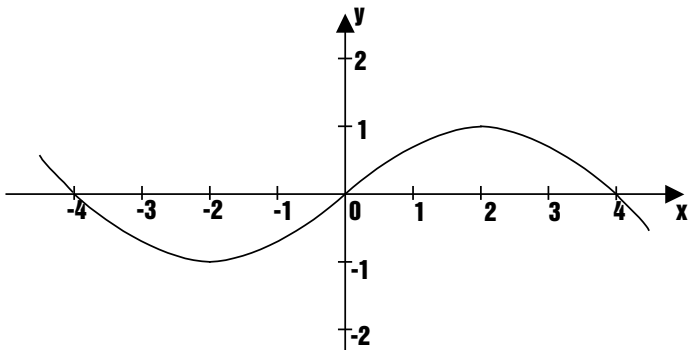
- **Las entradas de evento** envían eventos core al interior de la estructura en respuesta a los eventos del nivel primario que vienen del exterior. Puesto que los eventos del exterior del nivel primario no pueden llegar simultáneamente a las entradas, los eventos producidos internamente tampoco pueden ocurrir simultáneamente.

- **Las salidas de evento** envían eventos del nivel primario al exterior de la estructura en respuesta a los eventos core que llegan del interior. Aunque los eventos core pueden ocurrir simultáneamente en diferentes salidas, los eventos del nivel primario no se pueden enviar simultáneamente. Por lo tanto, para eventos core simultáneos, los correspondientes eventos del nivel primario se enviarán uno detrás de otro, *las salidas que haya más arriba siempre se enviarán ante de las que haya más abajo.*

Lo siguiente que vamos a hacer es poner esto en práctica.

Vamos a intentar construir un módulo de procesamiento de evento, que realice el modelado de la señal de acuerdo con la fórmula: $y = 0.25 * x * (4 - |x|)$

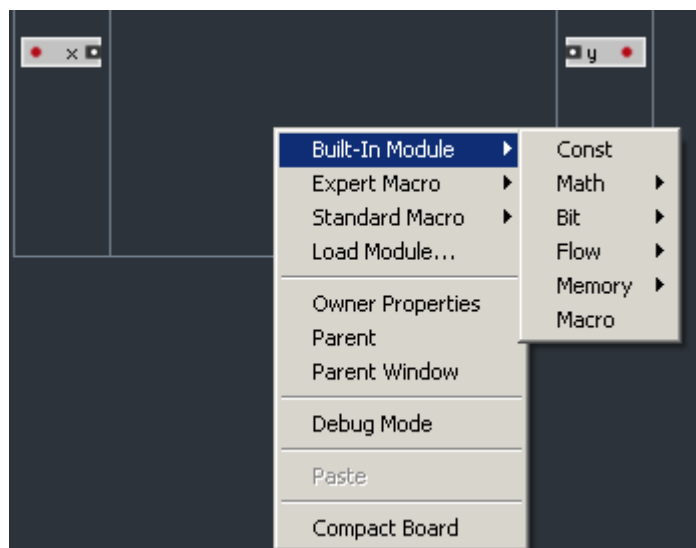
El esquema de esta función sería así:



Empezaremos creando una nueva célula core de evento con una entrada y una salidas llamadas “x” e “y” respectivamente.



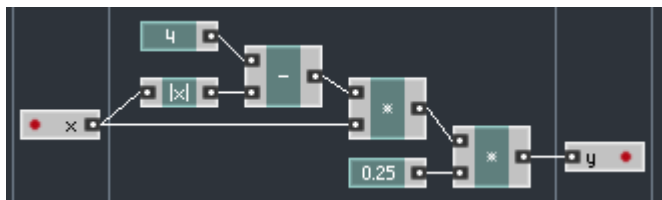
Ahora vamos a crear una estructura que calcule la fórmula. Necesitamos crear los módulos “|x|” (valor absoluto), “-” (sustractor) y dos “*” (multiplicador) en el área normal. Estas no son macros core, sino auténticos módulos internos de Reaktor Core. Para insertar estos módulos dentro de las estructuras core tendrás que hacer clic con el botón derecho sobre el fondo del área normal y seleccionar el sub-menú *Built-In Module* :



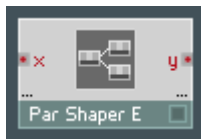
Encontrarás todos los módulos de arriba en el sub-menú
Built-In Module > Math



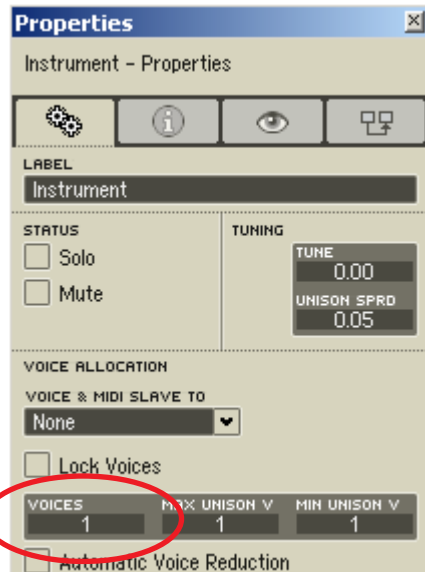
Necesitaremos valores constantes: 0.25 y 4. Podríamos usar QuickConsts igual que hicimos antes, pero también hay una opción para insertar un módulo real constante: *Built-In Module > Const* (como con QuickConst, el valor se puede especificar en la ventana de Propiedades):



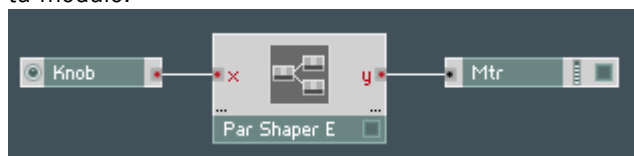
Por supuesto, en este caso particular no hay ninguna ventaja al usar los módulos Const en lugar de QuickConsts, pero puede que algunas veces lo prefieras (por ejemplo, si la misma constante se tiene que conectar a múltiples entradas, será mejor usar un módulo Const porque así sólo necesitarás uno de ellos, y tendrás un solo punto para controlar el valor). De todas formas, la estructura de arriba desempeñará correctamente la labor de dar forma a la señal como hemos mencionado. Así que quizá sería bueno poner un nombre a nuestro módulo, y volver al nivel primario:



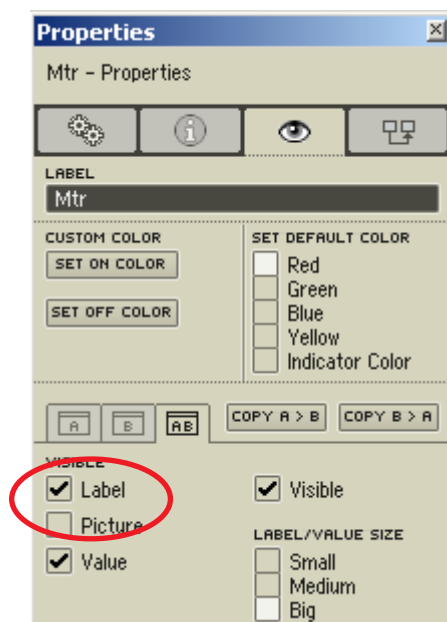
Vamos a probarlo. Ajusta el número de voces del instrumento de Reaktor a 1, para que así sea más sencillo usar un módulo Meter (medidor):



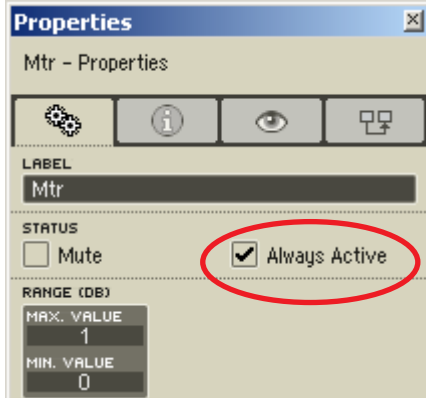
Crea un Knob y un medidor y conéctalos a la entrada y salida de tu módulo:



Ajusta las propiedades del knob y el medidor. No olvides configurar el medidor para que muestre su valor:



y comprobar la caja “Always Active” :



Ahora mueve el knob y observa cómo cambia el valor de salida.



La estructura de modelado de eventos que hemos construido tendría que funcionar perfectamente a la hora de diseñar señales de control, pero todavía le falta algo en cuanto a su comportamiento en el procesamiento de eventos. Regresaremos a este asunto y lo resolveremos más tarde

Estructuras con categoría interna

Señales de reloj

El procedimiento por el cual una célula de Reaktor Core procesa los eventos entrantes depende de este módulo. Normalmente, un módulo procesaría el valor entrante de alguna manera, pero también puede ignorarlo. El caso más común de este tipo de procesamiento son las *entradas de reloj*.

Un ejemplo de un módulo con una entrada de reloj es un *Latch* (o cerrojo). El *Latch* no es un módulo interno, sino una macro, pero es perfecto para comprender el principio de sincronía.

El *Latch* tiene dos entradas – una para el valor, y una para el reloj.



La entrada de valor (la de arriba), en respuesta al evento entrante, debería almacenar el valor que llega a su entrada dentro de la memoria interna del Latch, pero no enviaría nada por la salida. La entrada de reloj, en respuesta al evento entrante, enviaría el último valor almacenado a la salida.

La entrada de reloj, normalmente (a menos que se especifique de otra forma) ignora completamente el valor del evento entrante y responde sólo al hecho de que el evento está entrando.

Puesto que ahora estamos hablando de las señales de reloj, no de cerrojo, los ejemplos del uso del módulo *Latch* los veremos más tarde.

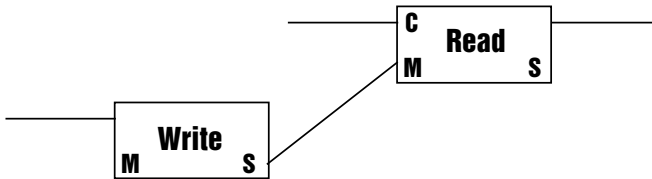
Puesto que hay módulos con entradas de reloj, está claro que algunas de las señales en la estructura no transportan ningún valor útil. Algunas señales se pueden producir incluso con el único propósito de usarlas como fuentes de reloj. Podemos llamarlas *señales de reloj*.

Un ejemplo de una señal de reloj es el reloj de la frecuencia de muestreo. Produce un evento por cada nuevo sample de audio que se genere, de forma que a 44.1 KHz de frecuencia de muestreo, “marcaría” 44100 tiempos por segundo. El valor de esta señal no tiene significado, no se tiene que usar en nada, y es siempre cero (en la implementación actual).

Conexiones de Bus de objeto

Las *conexiones de bus de objeto* (OBC) son un tipo de enlaces especiales entre módulos. Una conexión OBC entre dos módulos “anuncia” que comparten algún objeto interno. El caso más típico de módulos que utilizan conexiones OBC son los módulos de memoria *Read* y *Write*, que comparten la memoria si se conectan juntos por OBC.

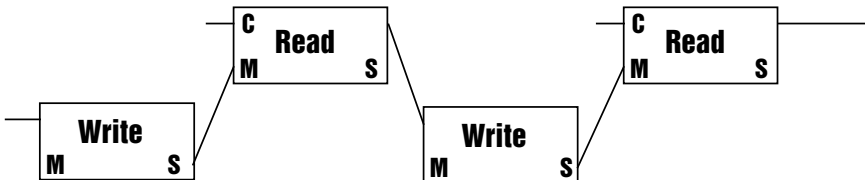
La funcionalidad del módulo *Write* es escribir un valor que está llegando a su entrada dentro de la memoria compartida por OBC. La funcionalidad del módulo *Read* es que lee el valor desde la memoria compartida por OBC, en respuesta a la señal clock entrante (entrada C). El valor de lectura se envía a la salida del módulo *Read*.



La estructura de arriba implementa la funcionalidad de la macro Latch (y de hecho, es la estructura interna de la macro Latch). Los conectores M y S de los módulos *Read* y *Write* son conectores del *tipo OBC Latch*. El conector M es la entrada de la conexión maestra, el conector S es la salida de la conexión esclava. La entrada maestra del módulo *Read* está conectada a la salida esclava del módulo *Write* (los otros conectores maestro y esclavo no se usan).

Por lo tanto, en esta estructura los módulos *Write* y *Read* comparten la memoria común.

En la siguiente estructura hay dos pares de módulos *Write* y *Read*. Cada par tiene su propia memoria. Observa que la conexión del medio (desde la salida del *Read* hasta la entrada del *Write*) no es una conexión OBC.

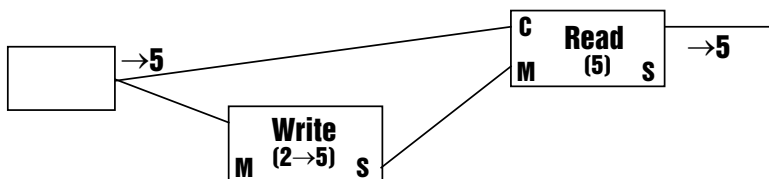


Uno podría preguntarse entonces, cuál es la diferencia entre ser maestro o esclavo. Bien, desde el punto de vista de propiedad del objeto compartido (en este caso la memoria), no hay diferencia. No obstante, como recordarás de una de las primeras secciones de este texto, hay una regla que dice que los

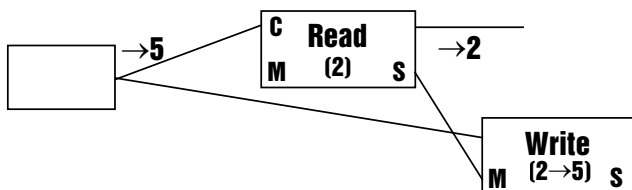
módulos emisores se procesan antes que los receptores durante el proceso de “eventos simultáneos”. Por lo tanto, en los dos últimos ejemplos, el módulo *Write* se procesaría antes que su esclavo módulo *Read* (lo que obviamente no sería igual al revés).

El orden relativo de procesamiento de módulos conectados por OBC se define usando las mismas reglas que con otros módulos: los módulos emisores se procesan antes.

De hecho, vamos a considerar dos casos. En ambos, el estado original de la memoria será 2, el mismo evento de valor 5 se enviará a los módulos *Write* y *Read*. En un caso, el módulo *Write* será el maestro, y en el otro caso, lo será *Read*.



Arriba tenemos la estructura del primer caso. El módulo de la izquierda envía un evento de valor 5, que llega primero al módulo *Write*, obligándolo así a escribir un nuevo valor de 5 dentro de la memoria compartida por la pareja de módulos *Write* y *Read*. Ahora el evento llega al módulo *Read*, funcionando como un evento de reloj y activando la operación de lectura, que de vuelta, lee el recién almacenado valor 5 y lo envía a la salida. Esta es la funcionalidad proporcionada por la macro *Latch* en la librería macro de Reaktor Core. Ahora ocupémonos del segundo caso:

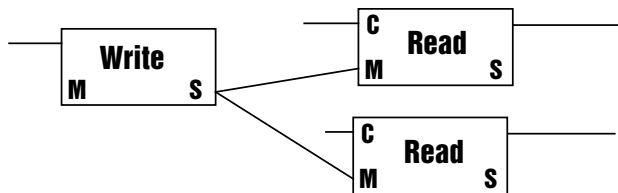


Aquí tenemos la situación contraria. Primero, el evento de reloj llega al módulo *Read*, enviando el valor almacenado de 2 a la salida. Sólo después de la lectura, el evento llega a la entrada del módulo *Write*, cambiando así el valor almacenado a 5. Esta estructura implementa la funcionalidad de un bloque Z^{-1} (un sample delay), muy usado en la teoría del DSP. De hecho, aquí el valor de salida está siempre un paso por detrás del de entrada.

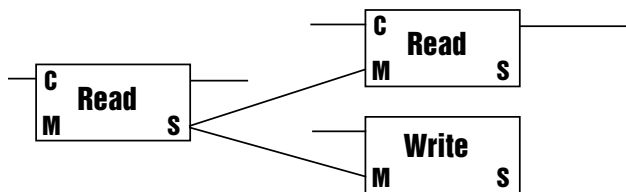
Como ya hemos dicho, la estructura superior implementa la funcionalidad Z^{-1} . Pero antes de que puedas realmente construir o usar dichas estructuras, hay algunas cosas importantes que debes saber, así que por favor, sigue leyendo.

Si tienes más de dos módulos conectados por OBC, significa que todos esos módulos comparten el mismo objeto. En este caso es imprescindible saber si el orden de las operaciones de lectura o escritura es importante, y de ser así, cuál sería ese orden.

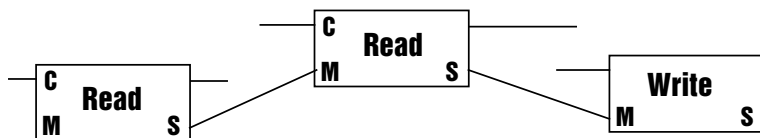
Por ejemplo, en la siguiente estructura, el orden relativo de las dos operaciones de lectura está sin definir, pero ambas ocurren antes de la operación de escritura, de forma que estaría completamente bien:



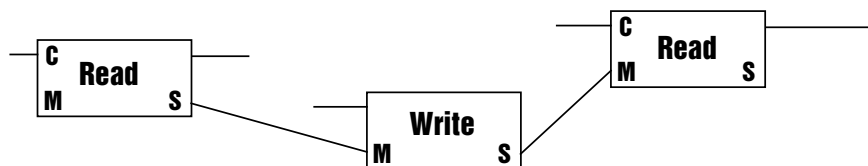
En la siguiente estructura, el orden relativo de la operación de escritura y la segunda operación de escritura está sin definir, así podría ser una estructura potencialmente peligrosa y tendrá que evitarse.



Una forma mejor de realizar la estructura de arriba es esta:



O esta:



Incluso aunque pienses que en algún lugar el orden relativo entre las operaciones de lectura o escritura es irrelevante, no te hará daño imponer un orden particular, y te ayudará a tener cierta seguridad.

El orden relativo de las operaciones de escritura es importante. El orden relativo de las operaciones de lectura no importa, siempre que su orden en las operaciones de escritura esté definido.

Las conexiones OBC no son compatibles con las conexiones de señales normales. Es más, las conexiones OBC que corresponden a tipos diferentes de objetos (por ejemplo, la distinta precisión coma flotante de capacidad de memoria) no son compatibles entre ellas. Los conectores de tipos incompatibles no se pueden conectar, por ejemplo, no puedes conectar la salida de una señal normal a una entrada OBC.

4.3. Inicialización

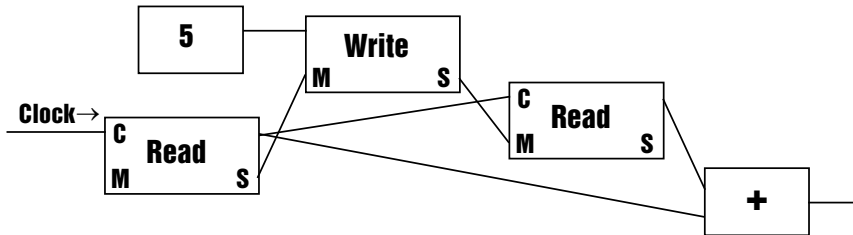
Puesto que vamos a empezar a trabajar con objetos que tienen una categoría interna (en el caso de *Read* y *Write*, la memoria compartida entre los objetos es su categoría interna), es imprescindible comprender cuál es la *categoría inicial* de la estructura que has construido. Por ejemplo, si vamos leer un valor desde la memoria (usando un módulo *Read*) ante de que se haya escrito nada, ¿cuál será el valor de lectura? Y, si no nos gusta el valor por defecto, ¿cómo podemos cambiarlo? Estas cuestiones están incluidas en el mecanismo de inicialización de Reaktor Core. La inicialización de las estructuras core se lleva a cabo del siguiente modo:

- Primero, todos los elementos de categoría se inicializan en valores por defecto, normalmente ceros. Sobre todo, toda la memoria compartida y todos los valores de salida de los módulos se ajustarán a cero, a menos que esté expresamente especificado de otra manera en la documentación.

- Segundo, un evento de inicialización se envía simultáneamente desde todas las fuentes de inicialización. Las fuentes de inicialización incluyen casi todos los módulos que no tienen una entrada: módulos Const (incluyendo QuickConsts), entradas de células core (típico), y algunos otros. Normalmente, las fuentes enviarían sus valores iniciales durante la inicialización del evento, por ejemplo, los constantes enviarían sus valores y las entradas de las células core enviarían sus valores iniciales recibidos desde el exterior de la estructura del nivel primario.

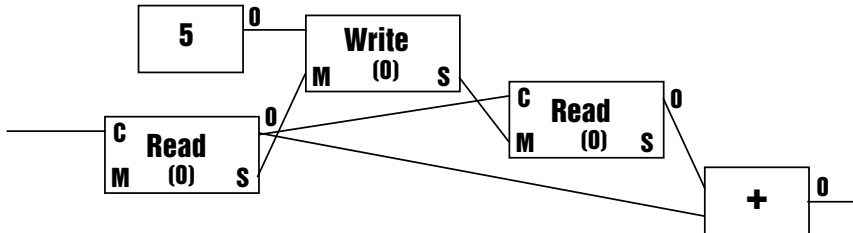
Si el módulo es una fuente de evento de inicialización, encontrarás información de este módulo en la sección de referencia. Si el módulo no es una fuente de inicialización, tratará al evento de inicialización exactamente igual que si fuera cualquier otro evento entrante. Principalmente, las fuentes de inicialización son aquellos y sólo aquellos módulos que no tienen entradas.

Echemos un vistazo a cómo funciona la inicialización en el siguiente ejemplo:

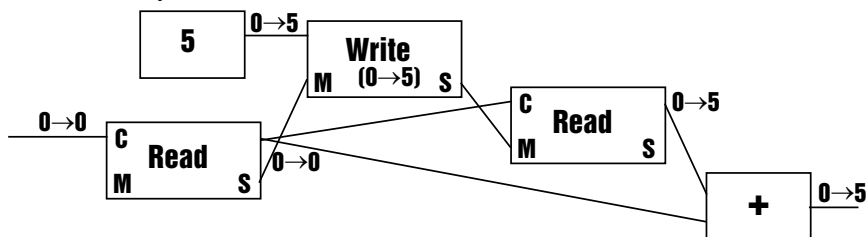


Esto es parte de la estructura. El módulo Read de la izquierda está conectado a una fuente de reloj, que a su vez envía un evento de inicialización (como harían normalmente las fuentes de reloj).

Inicialmente, todas las salidas de señal y la categoría interna de la cadena Read-Write-Read están ajustadas a cero.

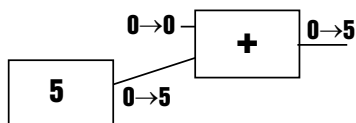


Entonces, un evento de inicialización se envía simultáneamente desde la fuente de reloj hasta el constante 5.



The *Read* module on the left is processed before the *Write* module and therefore the clock event arrives there before the new value is written into the memory, so the output of this module is zero. Then the value is written into the memory by the *Write* module. Now the second *Read* module is triggered, producing a value of 5 at the output. Lastly the adder module is processed, producing a sum of 5.

Como recordarás, las entradas desconectadas se tratan en Reaktor Core como valores cero (a menos que esté especificado en un módulo en particular). Más exactamente, se tratan como constantes cero. Esto significa que esas entradas también reciben eventos de inicialización, igual que si un módulo constante real con valor cero se hubiese conectado allí.



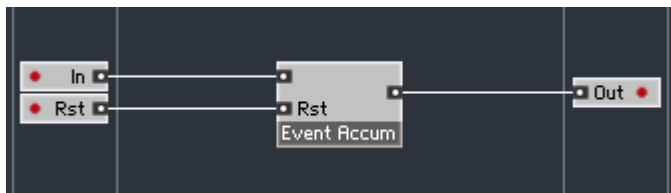
Arriba, un módulo de suma con una entrada desconectada y una conectada a un módulo constante que recibe dos eventos de inicialización simultáneos, uno desde la conexión del constante cero “por defecto” y una desde una conexión real a un constante.

Hay que prestar también una especial atención a las entradas desconectadas que no son entradas de señal (obviamente no se pueden conectar a un constante cero). Por ejemplo, una entrada maestra desconectada de un módulo *Write* significa que la cadena de memoria compartida empieza aquí y continúa en los módulos conectados a la salida esclava.

Construye un acumulador de eventos

El módulo acumulador de eventos que vamos a construir ahora tiene dos entradas, una para los valores de evento que van a ser acumulados y otra para resetear el acumulador a cero. También habrá una salida que proporcionará la suma de los eventos acumulados.

Vamos a construir este módulo en forma de macro core. Esta macro también debería de ser fácil de usar dentro de una célula core de evento.



Y esta es la apariencia del interior de nuestra macro:

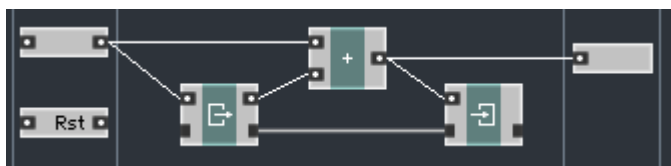


Obviamente, el módulo acumulador necesita una categoría interna donde almacenar su actual valor de acumulación. Vamos a usar los módulos *Read* y *Write* para construir el loop acumulador. Podrás encontrarlos en el sub-menú *Built In Module > Memory*:



El módulo que ves a la izquierda (con una flecha apuntando hacia fuera) es el módulo *Read* y el módulo de la derecha (con la flecha apuntando hacia dentro) es el *Write*.

En respuesta al evento entrante, el loop acumulador tendría que coger el valor antiguo y añadirle el nuevo. Así que tendremos que usar un módulo *Read* para recuperar la antigua categoría, y usar un módulo de suma para añadir el nuevo valor y un módulo *Write* para almacenar de nuevo el valor.



Observa que el módulo *Read* está sincronizado con el evento entrante y por supuesto su *Write* conectado por OBC se dirige a él de forma descendente, ya que queremos escribir después de leer. Normalmente, la estructura de arriba tendría que funcionar inmediatamente a la hora de acumular los valores entrantes y transmitir la suma total por la salida. Lo que falta es la funcionalidad reset y el cableado para asegurar la correcta categoría final.

Vamos a construir primero el circuito de reseteado. Puesto que estamos dentro del mundo de Reaktor Core, las entradas “In” y “Rst” podrían enviar un evento simultáneamente, así que en general (si queremos que esta macro core sea de uso general) deberíamos tener esto en cuenta. Así que suponemos que las entradas “In” y “Rst” producen simultáneamente un evento. ¿Qué queremos que ocurra en este caso? ¿El reseteado tendría que suceder antes o después de que el evento acumulado se procesase? (esto es muy parecido a la diferencia entre la funcionalidad Z^{-1} y *Latch*, que se diferencian únicamente en el orden de procesamiento relativo de la entrada de señal y la entrada de sincronía).

Sugerimos tomar el procedimiento *Latch*, ya que este módulo se usa mucho en las estructuras Reaktor Core, y por lo tanto este comportamiento nos resultará más intuitivo. En un *Latch* (cerrojo) la señal de reloj lógicamente llega después de la señal de valor. En nuestro caso, la señal de reseteado tendría que llegar después de la señal acumulada (forzando así la categoría y la salida a cero).

Esto significa que de alguna forma, tendríamos que reconducir la salida del acumulador con un valor inicial. Para conseguirlo, tendremos que usar un nuevo concepto que vemos a continuación.

Event merging

Hemos visto varias formas de combinación entre diferentes señales en Reaktor Core, incluyendo operaciones aritméticas y algunas otras. Lo que no hemos visto todavía es la simple asociación entre dos señales.

Asociar no es sumar. Asociar significa que vamos a tomar el último de los valores de las señales entrantes, no a sumarlos. Para asociar las señales necesitas usar el módulo *Merge*. Vamos a ver cómo funciona.

Imagina que tenemos un módulo *Merge* con dos entradas. El valor de salida inicial (antes del evento de inicialización) es, por supuesto, cero, igual que para casi todos los módulos:



Ahora, llegará un evento con un valor de 4 a la segunda entrada del módulo:



El evento pasa por el módulo y aparece en la salida. Ahora la salida del anexo (merge) tiene un valor de 4.

Luego llega un evento con valor de 5 a su primera entrada:



El evento pasa por el módulo y aparece en la salida, que cambia su valor a 5. Ahora dos eventos con valores de 2 y 8 llegan simultáneamente a ambas entradas.



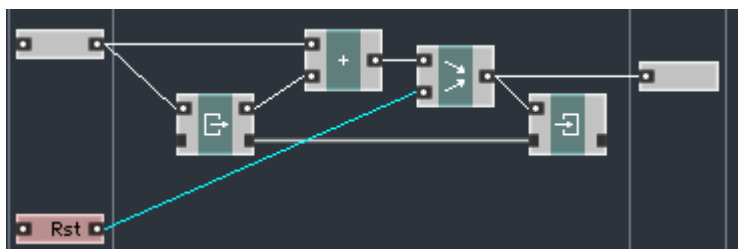
Aquí existe una regla especial para el módulo *Merge*:

Los eventos que llegan simultáneamente a las entradas del módulo *Merge* se procesan según la numeración de las entradas. De todas formas sigue habiendo un solo evento de salida generado, ya que Reaktor Core no produce varios eventos simultáneos.

En el caso de arriba, esto significa que el evento de la segunda entrada se procesará después del primer evento, siendo reconducido el 2 por el 8, que luego aparece en la salida.

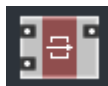
Event accumulator with reset and initialization

Así pues, para conseguir la funcionalidad de reajuste o reset necesitaremos redireccionar la salida del módulo de suma por algún valor inicial. Para ello podemos usar un módulo *Merge* (que encontrarás en el sub-menú *Built In Module > Flow*). Una manera más sencilla sería conectar la segunda entrada del módulo *Merge* a la entrada “Rst”.



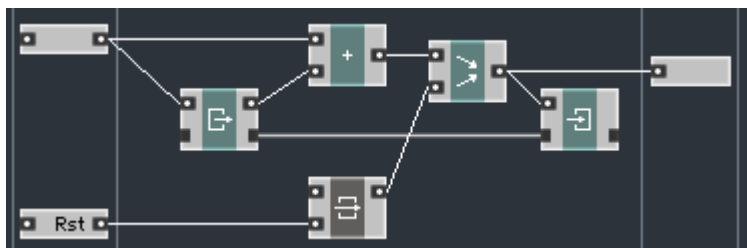
Ahora el evento de reseteo se enviará inmediatamente al módulo *Merge*, reconduciendo la salida del módulo de suma, con el evento de acumulación llegando al mismo tiempo. Desde aquí pasará a la salida y a la categoría interna del acumulador.

En la estructura de arriba, el valor que ocurre en la entrada “Rst” se usará como nuevo valor del acumulador. Quizá no es una mala idea, aunque entonces no será exactamente una funcionalidad de “reseteo” sino de “configuración” implementada en el módulo acumulador de evento estándar de Reaktor. Si queremos una auténtica funcionalidad de “reseteo” tendríamos que escribir únicamente valores cero dentro de la categoría, al margen de cuál sea el valor aparente en la entrada “Rst”. Así que lo que tenemos que hacer es enviar un valor cero al módulo *Write* cada vez que ocurra un evento en la entrada “Rst”. Enviar un evento con un valor concreto en respuesta a un evento entrante es una operación bastante común en Reaktor Core, y te sugerimos que para ello uses una macro *Latch* de la librería. *Expert Macro > Memory > Latch*:



Como ya hemos descrito, el módulo *Latch* tiene una entrada de valor (arriba) y una entrada de reloj (abajo). Necesitaríamos conectar la entrada “Rst” a la entrada de reloj del cerrojo para activar el envío de eventos a la salida del

cerrojo y también tendríamos que conectar un constante cero a la entrada de valor del cerrojo, ya que queremos enviar los eventos de valor cero todo el tiempo. O también podemos recordar que las entradas desconectadas se consideran constantes cero (a menos que se especifique otra cosa) y dejar la entrada de valor del Latch desconectada:

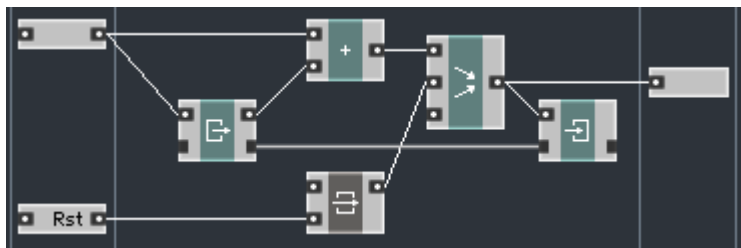


Ahora la funcionalidad de reseteo tendría que funcionar como se especifica.

El último paso sería asegurar una inicialización correcta, lo que implica definir qué sería una inicialización correcta. Echemos un vistazo a cómo se inicializaría correctamente la estructura de arriba.

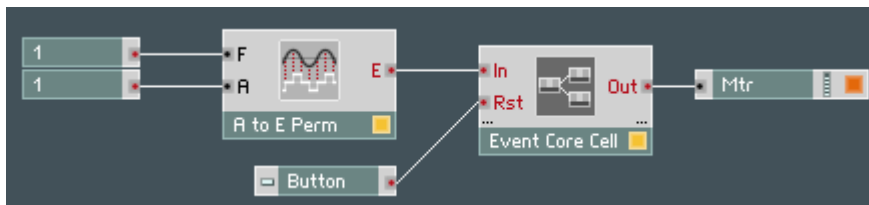
El evento de inicialización se envía simultáneamente desde las entradas “In” y “Rst” de la estructura de nivel superior de la célula core, y también desde el constante cero implícito en la entrada de valor del cerrojo. Una vez activado por la entrada “Rst” enviará un valor de cero a la segunda entrada del *Merge*, redireccionando cualquier valor que llegue a la primera entrada del *Merge*. Por lo tanto, el cero se escribirá en la categoría interna y se enviará a la salida - ¡Perfecto!

Sólo hay un problema. Podría ser que el evento de inicialización no llegase a uno o ambos puertos. Una de las razones podría ser que el evento de inicialización no llegase a la entrada correspondiente de la célula core de evento, o si esta macro se usase en una estructura más complicada de Reactor Core y no tuviese eventos de inicialización en todos los conductos (veremos cómo podemos arreglar esto más tarde). Así pues, necesitamos una modificación final en nuestra estructura para hacerla más “universal”. Ve a las propiedades del módulo *Merge* y cambia el número de entradas a 3.



Ahora, incluso aunque no hubiese un evento llegando a la entrada “Rst”, el constante cero implícito de la tercera entrada del *Merge* enviaría un evento de inicialización, provocando así una salida y unos valores de categoría iniciales correctos.

Veamos cómo funciona. Te sugerimos que construyas la siguiente estructura de nivel primario usando el módulo Event Accum creado.



El número de voces del instrumento tendría que ajustarse a 1 y el metro tendría que estar configurado para mostrar un valor y para permanecer siempre activo, como en el ejemplo anterior. El botón ha de estar ajustado al modo Trigger.

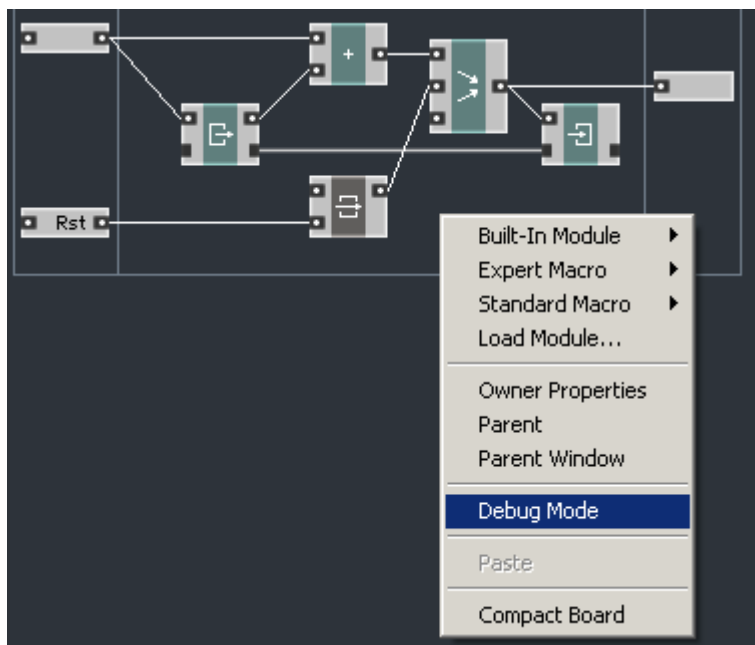



Ahora cambia al Panel y observa cómo los valores incrementan en pasos de 1, uno por segundo, y se reajustan si se presiona el botón.

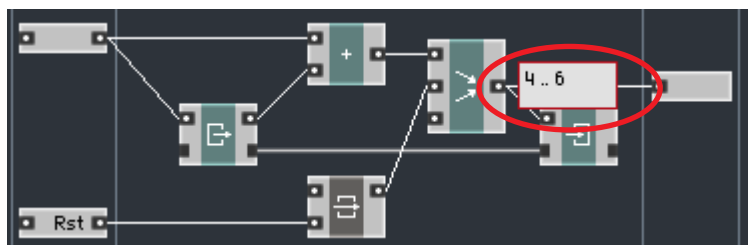


Vamos a aprovechar para presentarte el modo Debug (encargado de eliminar fallos del programa) de Reactor Core. Como probablemente habrás notado, al contrario que en el nivel primario de Reactor (donde podías ver el valor en

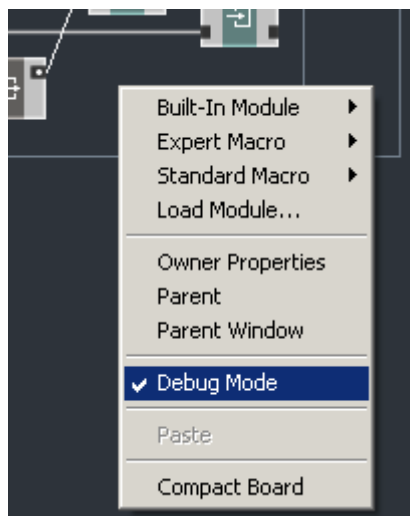
la salida de un módulo si mantenías el cursor con el ratón sobre esa salida), no ocurre lo mismo en Reaktor Core. Es una de las desventajas de la optimización interna de Reaktor Core, por la que los valores de las estructuras de Reaktor Core no están disponibles desde el exterior. Como sabíamos que ibas a quejarte, te proponemos una solución. Puedes deshabilitar la optimización para una estructura core particular y ser así capaz de ver los valores de salida. Vamos a comprobarlo en la estructura que acabamos construir. Haz clic con el botón derecho sobre le fondo y selecciona *Debug Mode*:



(Puedes hacer lo mismo usando el botón  de la barra de herramientas). Ahora, si mantienes el cursor sobre cualquier salida, podrás ver el valor (o grupo de valores):

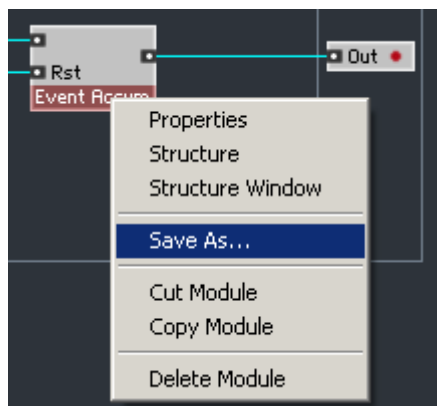


Puedes deshabilitar el modo Debug seleccionando el mismo comando (o presionando de nuevo el mismo botón):

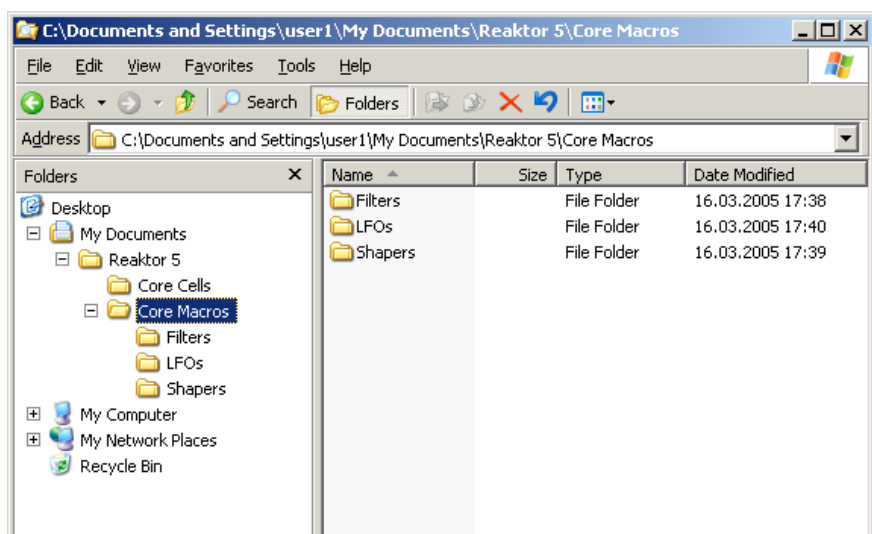


Si no, se apagará automáticamente cuando dejes esta estructura, así que puede que tengas que volver a habilitarlo en otra estructura.

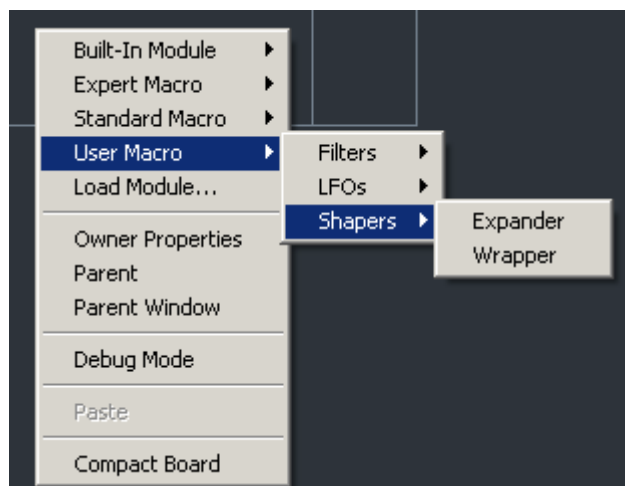
Quizá sea el momento de guardar nuestro core como un archivo independiente para su uso futuro. Hazlo presionando clic con el botón derecho sobre la macro y seleccionando “Save As...”:



Igual que con las células core tienes la opción de disponer de tus propias macros en el menú. Las macros se tienen que colocar en la sub-carpeta “Core Macros” de la carpeta de la librería de usuario de Reaktor.



Si cualquier archivo se encuentra en esta carpeta “*Core Macros*” o sus sub-carpetas, aparecerá un sub-menú dentro del menú que aparece cuando haces clic con el botón derecho.

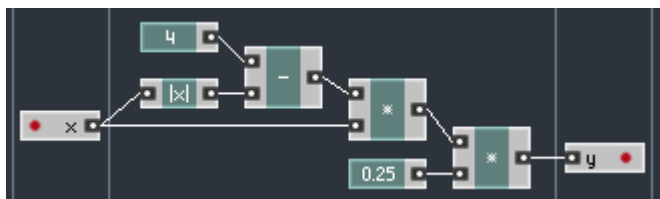


Igual que con la carpeta “*Core Cells*”, hay determinadas restricciones para la “*Core Macros*”:

- Las carpetas vacías no aparecen en el menú
- Nunca pongas tus propios archivos dentro de la librería del sistema, ponlos en la carpeta de la librería de usuario.

Arregla el modelador de eventos

Es el momento de analizar qué es lo que estaba mal en la estructura del módulo de modelado de eventos que hemos construido antes:



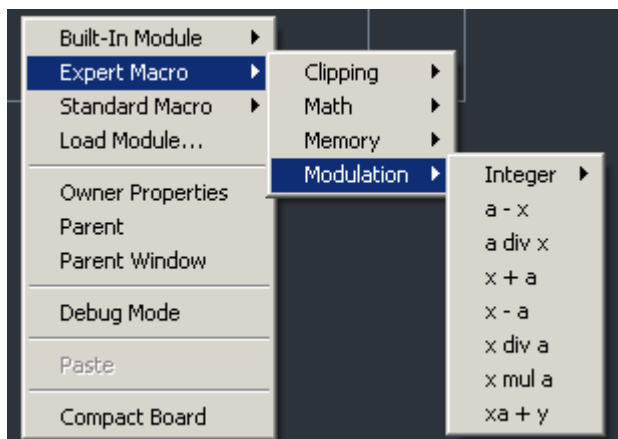
El problema es el evento de inicialización. Si observas cómo sucede la inicialización de la estructura de arriba, te darás cuenta de lo siguiente:

- La entrada “x” activa o no el evento de inicialización dependiendo de si recibe un evento de inicialización desde el exterior de la estructura de nivel primario (se trata de la regla de eventos de inicialización para entradas de evento de células core).
- Los constantes 4 y 0.25 siempre generan un evento de inicialización.

Por lo tanto, en caso de que por cualquier razón, el evento de inicialización no ocurra en la entrada del modelador, la salida de éste recibirá el evento desde el último multiplicador y lo reenviará al exterior de la estructura de nivel primario.

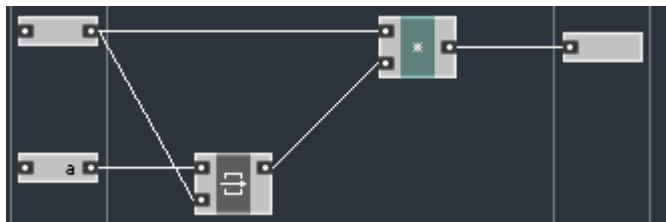
Aunque para procesar señales de control esto estaría bien (en caso de perder un evento de inicialización de entrada, la entrada se consideraría cero y el evento de inicialización de salida sería activado), no es precisamente lo que uno espera de un módulo de procesamiento de eventos. Es decir, que conseguiríamos un comportamiento más intuitivo si el módulo enviase un evento de salida sólo en respuesta a un evento entrante. El problema es que nuestros dos módulos constantes pueden estar enviando eventos en el momento equivocado (es decir, cuando no haya evento de entrada). Te sugerimos como solución que reemplaces los módulos de sustracción y multiplicación que tengan constantes a sus entradas por sus equivalentes de *Modulación*.

Las “macros de Modulación” son un grupo de módulos de la librería de Reaktor Core que encontrarás en *Expert Macro > Modulation*:

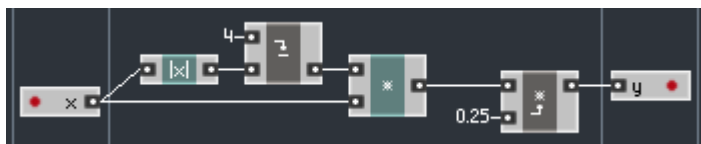


El nombre “Modulación” que no es 100% correcto, refleja su propósito al usar una señal para modular otra (hubiera sido fácil de ver más adelante, cuando hubiésemos usado señales de control para modular señales de audio en las estructuras de bajo nivel). Muchas de estas macros combinan dos señales, siendo una la “portadora” y otra la “moduladora”. Al contrario que los módulos internos core aritméticos, las “macros de modulación” generan eventos de salida sólo en respuesta a un evento en la entrada “portadora”. Los eventos de la entrada “moduladora” no activan el proceso de re-cálculo.

La implementación interna de las macros de modulación es muy sencilla: resincronizan la señal “moduladora” mediante un módulo Latch, usando la “portadora” como reloj, antes de operar entre ambas señales. He aquí un ejemplo de la estructura interna de una macro “multiplicador modulador” (la entrada llamada “a” es la moduladora):



Así pues, reemplazamos el módulo de sustracción por la macro de modulación “ $a - x$ ” y el segundo módulo de multiplicación por la macro de modulación “ $x \text{ mul } a$ ”. Así es como será nuestra estructura tras la sustitución (también hemos reemplazado los módulos constantes por QuickConst, pero eso no es importante):



Reconocerás las entradas moduladoras de las macros de modulación por sus iconos (dibujos en los módulos). La posición de una entrada moduladora se corresponde con la posición de la flecha del icono. En el caso del módulo sustractor, la flecha está arriba, por lo tanto, la entrada de la modulación está arriba. En el caso del módulo multiplicador, está al revés. También puedes observar que la salida de estos módulos está colocada contra la entrada potadora, lo que significa una fuente de información adicional. Por último, puedes mover el cursor de tu ratón sobre los módulos y sus entradas para leer sus correspondientes textos de información.

En la estructura de arriba no se enviarán eventos a menos que haya un evento en la entrada de la célula core:

- El módulo “ $|x|$ ” se activa directamente por el evento de entrada de la célula core.
- El subsiguiente módulo de sustracción se activará por la salida del módulo “ $|x|$ ”, que envía un evento sólo en respuesta al evento de entrada. El QuickConst no tiene efecto activador.
- El primer multiplicador se activará ya sea por la salida del módulo de sustracción o por el evento de entrada de la célula core, pero ya hemos visto que ambos ocurren sólo simultáneamente.
- El segundo multiplicador se activará sólo por el evento entrante y no por QuickConst.

Así pues, nuestra estructura ahora tiene un comportamiento algo más intuitivo.

Audio processing at its core

Audio signals

No existen tipos especiales de señales de audio en Reaktor Core. El audio en Reaktor Core también es un evento, lo que desde el punto de vista de la estructura, no difiere en nada de cualquier otro evento. Lo diferentes es que entre la ruta de las señales de audio, los eventos se producen en “intervalos de tiempo regularmente espaciados”, correspondiente a la “frecuencia de muestreo”.

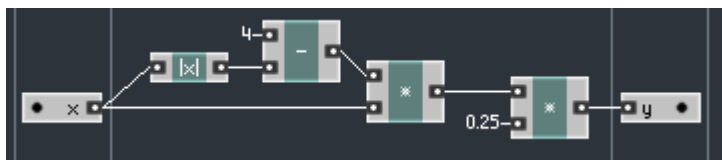
Para producir eventos regularmente espaciados (o en cuanto a eso, cualquier evento) necesitamos una fuente de evento. Parecido a las células core de evento donde las entradas del módulo eran fuentes de evento, en las célula core de audio también son fuentes de evento. Pero ahora tenemos un tipo de entrada extra disponible:

- **Las Entradas de Audio** repeatedly send core events to the inside of the structure at the rate determined by the sampling rate setting of the outside primary-level structure. The events are sent simultaneously from all audio inputs of a core-cell structure.
- **Las Entradas de audio** también envían el evento de inicialización a la estructura de la célula core. Este evento se envía al margen de lo que ocurra en el exterior de la estructura del nivel primario. No obstante, el valor enviado por estas entradas durante la inicialización depende del proceso de inicialización del exterior.

También hay un nuevo tipo de salida que ha de usarse *en lugar de* las salidas de evento.

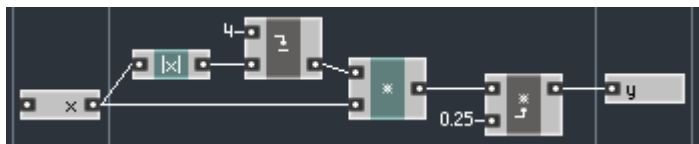
- **Las salidas de audio** deliver the last value received from the inside core structure to the outside primary-level structure. Because the audio outputs in the primary level do not send events, no events are sent to the outside.

Ahora vamos a reconstruir el mismo modelador que habíamos construido para eventos en el modo audio. Por lo tanto, crearemos una nueva célula core de audio. Generalmente, podemos usar exactamente la misma estructura, sólo que en lugar de entradas y salidas de evento, las tendremos de audio:

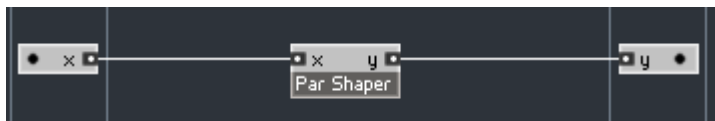


Te preguntarás por qué no hemos usado una macro de modulación en este caso. Bien, es mejor hacerlo así puesto que aquí estamos tratando procesamiento de señales de audio, y las señales de audio siempre envían eventos de inicialización. En cualquier caso podrías usar si quisieras macros de modulación. No importa.

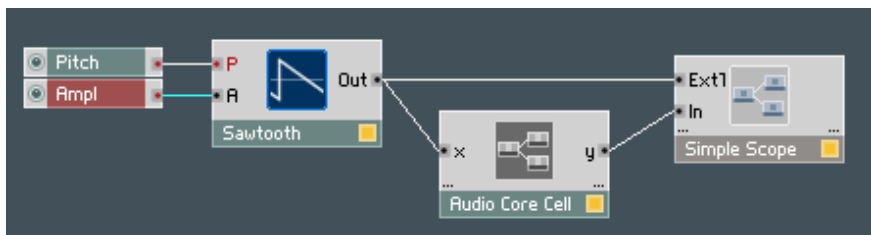
También podríamos empaquetar la estructura de arriba dentro de una macro que se pueda usar dentro de varias estructuras de Reaktor Core, tanto para procesamiento de audio como de evento. En este caso es mejor usar macros de modulación dentro, puesto que no sabemos por adelantado qué tipo de señal procesará este módulo:



Y esta es la estructura interna de la célula core en este caso:

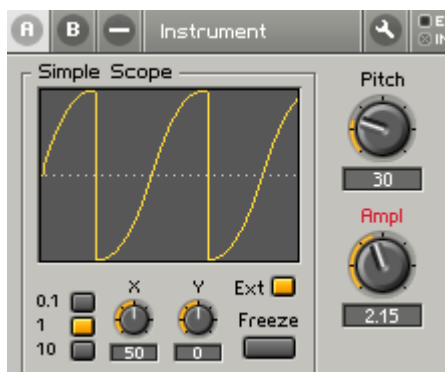


Para probarla vamos a conectar un oscilador de diente de sierra y un osciloscopio. Puedes encontrar un osciloscopio *Insert Macro > Classic Modular > OO Classic Modular – Display > Simple Scope* (para una estructura de nivel primario). No olvides asegurarte de que el número de voces para el instrumento sea 1.



Estamos usando el activador externo para el osciloscopio, porque así conseguiremos una mejor sincronización a niveles altos de distorsión (el botón

Ext del panel del osciloscopio ha de estar conectado para poder funcionar). Cambia el knob Ampl a un ajuste entre 0 y 5 para poder ver la modulación.



Bus de reloj de frecuencia de muestreo

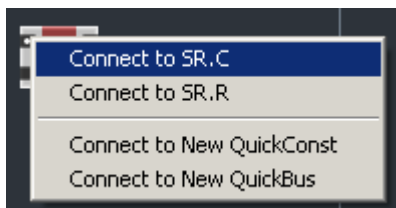
Parece que todavía nos faltan un par de características para poder construir estructuras de audio. Una tendría que permitirnos crear células core de audio que no lleven entradas de audio. Por supuesto, podemos crearlas, pero ¿de dónde sacamos una fuente de evento de audio? La segunda característica que nos falta es que muchos algoritmos DSP requieren el conocimiento de la frecuencia de muestreo actual. Vamos a por ello.

Existe una posibilidad de conexión especial disponible en toda estructura de Reaktor Core que se llama “Bus de reloj de frecuencia de muestreo”. El bus transporta dos señales: el reloj (sincronía) y la frecuencia.

Reloj es una fuente de señal donde los eventos se envían regularmente a la frecuencia de muestreo del audio. Al igual que con todas las señales de audio estándar, también envía un evento de inicialización. El valor de estos eventos es actualmente cero, pero generalmente cualquier estructura que use esta señal ignorará los valores, ya que se pueden cambiar en el futuro.

Frecuencia es una fuente de señal cuyo valor es siempre igual a la frecuencia de muestreo del audio en Hz. Los eventos se envían desde esta fuente durante la inicialización o si se dan cambios en la frecuencia de muestreo.

Tendrás acceso al bus de frecuencia de muestreo haciendo clic con el botón derecho sobre cualquier entrada de señal y seleccionando “Connect to SR.C” para la señal de reloj o “Connect to SR.R” para la señal de frecuencia.



La conexión aparecerá cerca de la entrada:



El bus de reloj de frecuencia de muestreo no funcionará dentro de las células core de evento.

Connection feedback

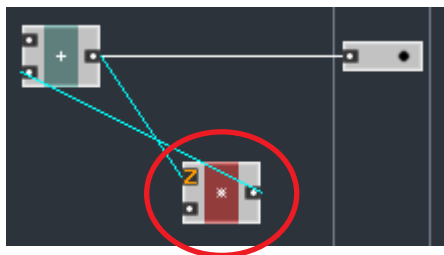
Como ya vimos, las reglas de orden de procesamiento no se pueden aplicar si hay conexiones retroalimentadas dentro de la estructura. Por lo tanto necesitaremos proporcionarte reglas adicionales que definan cómo manejar el feedback.

La regla principal es: ¡Las estructuras Reaktor Core no pueden manipular feedback.

Bien, no es exactamente así. Puedes hacer conexiones realimentadas en Reaktor Core. Pero puesto que el motor Reaktor Core no puede manejar estructuras que lleven feedback, tendrá que resolverlos. Resolver esa respuesta significa que tu estructura tendrá que modificarse (internamente, no lo verás en la pantalla) para que no haya retroalimentación.

Normalmente hay un delay de al menos un sample en la ruta de la respuesta de audio digital. Así que eso es lo que hará el motor de Reaktor Core durante la resolución del feedback – introducirá un módulo de delay de un sample (Z^{-1}) en la ruta de la retroalimentación.

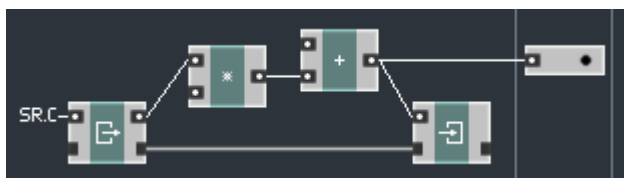
Como ya sabes, los lugares en los que se han introducido Z^{-1} 's implícitos, se indican con una gran Z naranja que aparece en el puerto:



Ya hemos visto una estructura construida o los módulos *Read* y *Write* implementando la funcionalidad Z^{-1} . Vamos a poner esa construcción de Read y Write en nuestra estructura. La pondremos en el cable donde sucede la resolución de feedback.



Antes de escribir, vamos a leer (observa que el módulo Read está sincronizado por SR.C para asegurar que la lectura ocurra una vez por cada tic de audio). Por lo tanto, el valor de lectura es siempre un sample de audio por detrás del de escritura. Y ahora no hay realimentación en la estructura. ¿No lo ves? Bien, vamos a mover los módulos un poco (no cambiaremos una sola conexión).



¿Lo ves ahora? Por supuesto.

Así que insertando un módulo Z^{-1} eliminaremos formalmente el feedback de la estructura, mientras lo conservamos lógicamente (con un delay de un sample de audio).

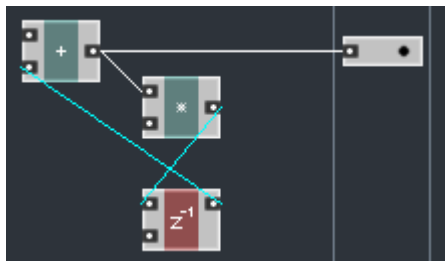
De hecho, la estructura interna de una macro Z^{-1} es algo más complicada que simplemente un par de Read y Write. Lo que es, y por qué, vamos a analizarlo en la siguiente sección.

No tienes ningún control sobre el lugar donde ocurrirá la resolución automática de feedback. Simplemente ocurre en un conector de señal arbitrario del loop de feedback. Ni siquiera está garantizado que la resolución suceda siempre en ese conector particular – se puede cambiar en la siguiente versión del software, podría cambiar en respuesta otro cambio dado en la estructura, podría ser diferente la próxima vez que cargues la estructura desde el disco.

Esta resolución automática de feedback está pensada para estructuras para las cuales no es importante el lugar exacto donde ocurra esta resolución. Particularmente, podría ser que esas estructuras las construyese un usuario que no supiese lo suficiente sobre DSP como para comprender su problemática. La resolución automática de feedback seguirá proporcionándoles unos resultados aceptables.

Si necesitas tener un control preciso sobre los puntos de resolución de feedback puedes conseguirlo insertando explícitamente módulos Z^{-1} en tu estructura. Estos módulos eliminarán formalmente el feedback y la resolución no será necesaria.

He aquí una versión de la estructura de arriba con una macro Z^{-1} insertado (lo encontrarás en el sub-menú *Expert Macro > Memory*):

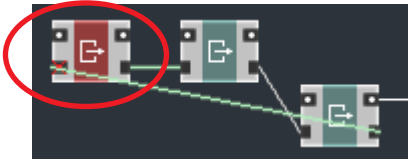


Como puedes ver, la gran Z naranja ahora se ha ido. Observa también que el punto de delay de 1-sample es diferente del que se insertó automáticamente (el automático estaba en el cable que va desde la salida del Adder a la entrada del Multiplier y ahora está en el cable que va desde la salida del Multiplier a la entrada del Adder).

El significado de la segunda entrada del módulo Z^{-1} lo explicaremos más tarde. Normalmente la dejarías desconectada.

El feedback en OBC o cualquier otro tipo de conexión sin señal (de las que hablaremos más tarde) no tiene sentido y no está permitido. Si ocurriese cualquier loop de feedback no tendría cables de señal, una de las conexiones

se marcaría como inválida y se consideraría no-existente. La marca “inválido” aparece como una X grande color rojo cruzando el puerto.



Por otro lado, los loops de feedback con tipos de conexiones “mezcladas” contienen cables de señales normales, funcionan perfectamente y se resolverán de modo normal. La resolución sucediendo en uno de los cables de señal normales:



De todas a todas, esto significa que las conexiones sin señal no se verán afectadas por la resolución de feedback, a menos que hagas un feedback sin señal, lo cual no tiene ningún sentido.

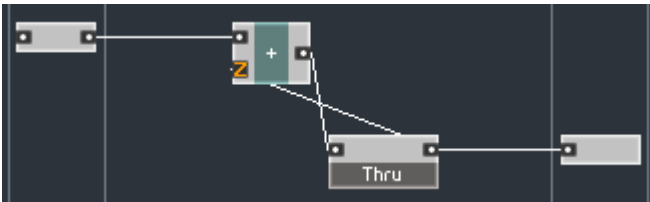
Feedback en torno a macros

Las macros, en general, se tratan de la misma forma que los módulos internos con respecto a la resolución de feedback.

Vamos a ver una macro que simplemente deja pasar la señal. Esta es la estructura de dicha macro:



Ahora supongamos que construimos una estructura de feedback usando esta macro:



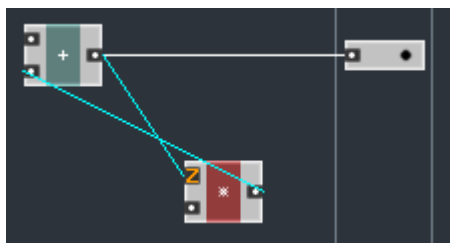
El loop de feedback pasa a través de dos cables en la estructura de arriba y a través de otro cable dentro de la macro. ¿Dónde va a ocurrir ahora la resolución? (Bien, puedes ver en el dibujo de arriba que está ocurriendo en la entrada de suma *en este caso particular*, pero como ya sabemos, también podría ocurrir en otro punto).

Imagina por un momento que *Thru* no fuese una macro sino un módulo interno. En este caso es evidente que la resolución de feedback no podría darse dentro de este módulo, tendría que ocurrir fuera.

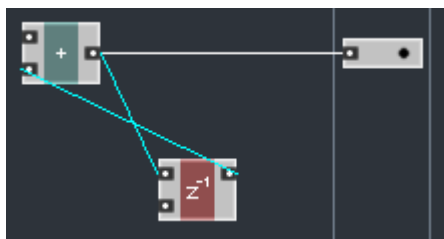
Bien, vamos a hacer lo posible para que las macros parezcan y se comporten como los módulos internos. Para ello, *por defecto*, la resolución de dicho loop de respuesta ocurrirá fuera de la macro. No está especificado el lugar donde sucederá, pero en cualquier caso, será fuera de la macro *Thru*.

La regla general es – la resolución de feedback ocurre en el nivel más alto de la estructura del loop de feedback.

No obstante, puedes cambiar este comportamiento – es decir, permitir que la resolución de feedback ocurra dentro de la macro. De hecho te habrás preguntado, ya que las macros se tratan igual que los módulos internos, cómo puede la macro Z^{-1} resolver el feedback. Queremos decir, siguiendo la siguiente estructura:

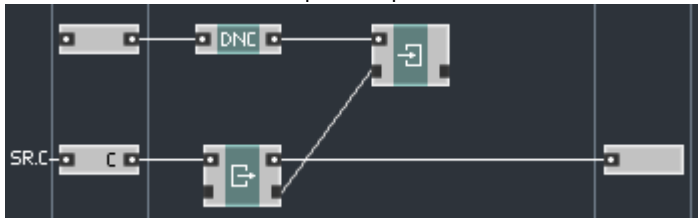


Que si las macros y los módulos internos son iguales, entonces nada debería cambiar al reemplazar el multiplicador por una macro Z^{-1} :



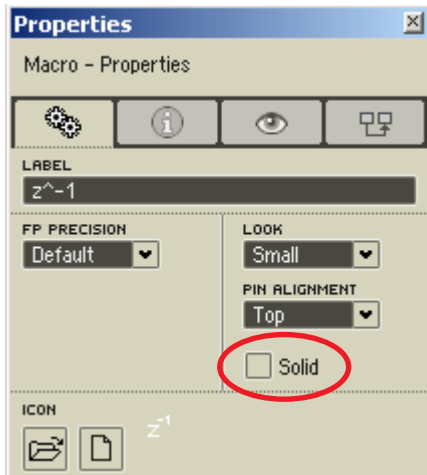
Pero es diferente, ya que el feedback implícito ahora se ha ido. Tiene que haber algo especial en esta macro Z^{-1} . Y de hecho lo hay.

Si miramos dentro de esta macro veremos una estructura casi igual a la que hemos mencionado antes para implementar la funcionalidad Z^{-1} :



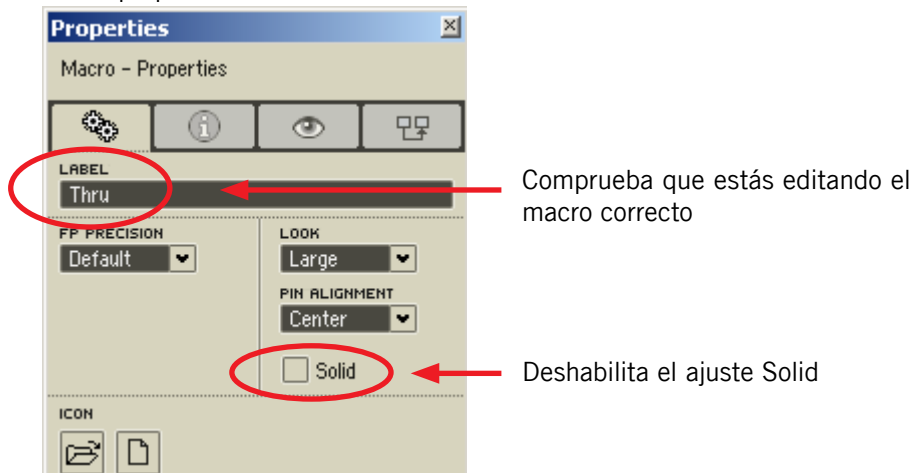
Como puedes ver, la entrada de reloj de la macro está conectada al módulo interno *Read*. La conexión por defecto de esta entrada no es un constante cero, sino el reloj *de audio*, y así lo querrás la mayoría de las veces. El módulo conectado entre la entrada superior y el módulo Write lo explicaremos más tarde. De momento, ignóralo. No hay nada especial en la estructura, excepto que parece implementar la estructura Z^{-1} que habíamos estado viendo antes. ¿Cómo sabe el motor de Reaktor Core que esta estructura sirve para resolver los loops de feedback? Es decir, obviamente el motor sabe que *puede* resolver los loops de feedback, pero ¿cómo sabe cuándo ha de hacerlo?

Esto se controla en las propiedades de la macro. Concretamente en el ajuste *Solid* de la macro:

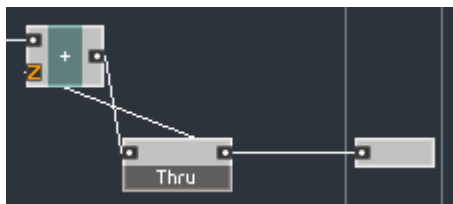


Esta propiedad le indica al motor de Reaktor Core si la macro ha de considerarse como un módulo interno “sólido” para la resolución de feedback, o si ha de ser transparente. *En el 99% de los casos, te interesará mantener esta propiedad conectada*, ya que normalmente no querrás que la resolución de feedback implícita ocurra dentro de tus macros.

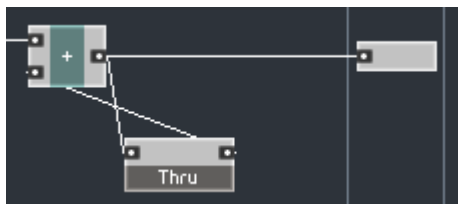
Una razón para ello es que cuando ocurre una resolución de feedback dentro de una macro, no es visible a menos que entres dentro de la macro, así que puede ser que algunos delays implícitos de feedback se te pasen por alto. Por ejemplo, podemos coger nuestra estructura previa con la macro “Thru” y deshabilitar el ajuste *Solid* (comprueba que estás editando el ajuste *Solid* en la macro correcta. Podrás saberlo por el texto “Thru” en texto de la etiqueta de las propiedades):



Ahora, probablemente, tu estructura sigue pareciendo igual (decimos probablemente porque nunca sabrás exactamente dónde va a ocurrir la resolución automática de feedback):



Pero si cambias un poco tu estructura, conectando la salida a otro módulo, sería así:



Parece que nuestro delay de resolución de feedback ya no está. Así que en una estructura más grande y complicada podríamos olvidar fácilmente que hay un delay explícito. ¿Dónde ha ido este delay? Por supuesto, ahora está dentro de nuestra macro Thru – el único sitio que no podemos ver desde fuera.



Otra razón para conservar la propiedad *Solid* es que estuviese desconectada, en algunos casos la funcionalidad de la macro podría cambiar una vez puesto en la ruta del feedback. Así que hazte un favor y quita la propiedad sólo si construyes macros pensadas para resolver feedback. No serán muchos. Regresemos al módulo Z^{-1} . Como hemos quitado la propiedad *Solid* para esta macro, el límite de esta macro es completamente transparente para la resolución de feedback. Por tanto, la macro Z^{-1} no es tratado realmente como si fuera un módulo interno y es capaz de resolver la realimentación en la forma en que ya habíamos descrito antes en este texto.

Valores anormales

Los valores de señal en las estructuras que hemos ido construyendo en las secciones anteriores se representan en el ordenador a través de un tipo de datos binarios llamados *dígitos de coma flotante*, o *flotantes* para abreviar. Este tipo de datos permiten la representación eficiente de una gran cantidad de valores. .

El término *dígitos de coma flotante* no especifica exactamente cómo se representan los números. Simplemente describe el procedimiento adoptado para representarlos, dejando todavía libertad para los detalles de implementación. Las CPUs de los ordenadores personales de hoy en día usan el estándar IEEE de coma flotante. Este estándar define exactamente cómo se han de representar los *dígitos de coma flotante* y cuáles han de ser los resultados al operar sobre ellos (por ejemplo, cómo manipular los temas de precisión limitada, etc). Particularmente, este estándar dice que para un grupo de valores de coma flotante especialmente pequeños que no se pueden representar de un modo “normal”, debido a la precisión limitada de la coma flotante, ha de usarse otra representación especial. Este modo se denomina representación “anormal”.

El campo para valores “anormales” de coma flotante de 32 bit es aproximadamente desde 10^{-38} a 10^{-45} y desde -10^{-38} to -10^{-45} . Los valores menores de 10^{-45} en magnitud absoluta no pueden representarse, y se considerarán como ceros.

Puesto que estos números adoptan una representación distinta de la “normal”, algunas CPUs tienen ciertos problemas al tratar estos números. Sobre todo, las operaciones con estos números sucederán mucho más lentas (10 veces o más) que con los números “normales”.

Una situación típica en la que los números anormales aparecen durante *prolongados períodos de tiempo* es con los valores de decaimiento exponencial, como los filtros, y algunas envolventes o estructuras de feedback. En dichas estructuras, después de que la señal de entrada se ha bajado a cero, las señales caen a cero asintóticamente. Asintóticamente significa que el nivel de la señal trata de alcanzar el cero, pero nunca lo consigue. En esta situación los números anormales pueden aparecer y quedarse en la estructura durante un tiempo prolongado creando así una considerable carga de CPU.

Otra situación en la que pueden darse los números anormales, es al cambiar la precisión de un valor de coma flotante, desde una precisión alta (64 bit) a una más baja (32 bit), ya que un valor 10^{-41} no es anormal en un flotante de precisión a 64 bit, pero lo es en un flotante de precisión a 32 bit (habaremos de los cambios de precisión más tarde).

Imaginemos que vamos a dar forma a un filtro paso-bajo con la frecuencia de corte ajustada a 20. Nuestros valores de señal digitales se corresponderán a los voltajes analógicos en voltios. Imaginemos que el nivel de la señal de entrada fuese igual a 1 V (voltio) durante un período de tiempo considerable. El voltaje de la salida del filtro también es igual a 1 V. Ahora cambiamos radicalmente el voltaje de entrada a cero. El voltaje de salida decaerá de acuerdo con la norma:

$$V_{out} = V_0 e^{-2\pi f_c t}$$

donde f_c es el corte de filtro en Hz, t es el tiempo en segundos y $V_0 = 1\text{ V}$ (voltaje inicial).

Entonces el voltaje de salida cambia de la siguiente forma:

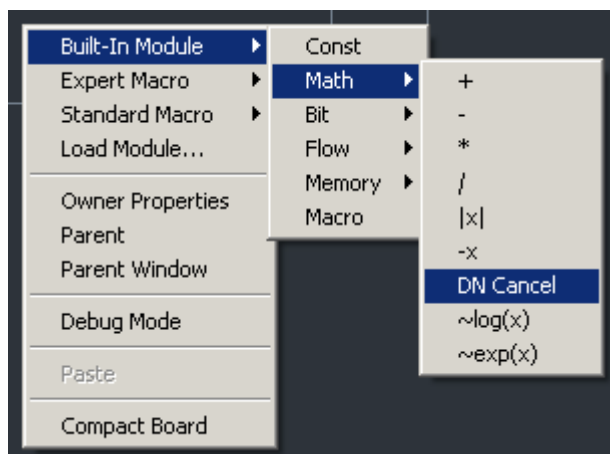
tras 0.5 sec	$V_{out} \approx 10^{-29}$ volt
tras 0.6 sec	$V_{out} \approx 10^{-33}$ volt
tras 0.7 sec	$V_{out} \approx 10^{-38}$ volt
tras 0.8 sec	$V_{out} \approx 10^{-44}$ volt

Oh, los números entre 10^{-38} y 10^{-45} están en el ámbito anormal. De forma que en el período de tiempo de 0.7 a 0.8 segundos, nuestro voltaje estaría representado por un valor anormal. Y no sólo dentro del filtro. La salida de filtro se procesará más tarde en el flujo de corriente de la estructura, provocando así que, al menos los siguientes módulos, tengan que trabajar también con valores anormales.

A una frecuencia de muestreo de 44.1 KHz, el intervalo de tiempo de 0.1 se corresponde con 4410 samples. Suponiendo que el tamaño típico de un buffer ASIO sea unos cientos de samples, tendremos que producir varios buffers con una carga de CPU considerablemente superior. Si la carga de CPU (por computación de buffers) se acerca o sobrepasa el 100% causará pérdidas de audio.

Del texto anterior has de sacar una conclusión: los valores anormales son negativos para el audio en tiempo real.

Los módulos del nivel primario de Reaktor se han programado prestando atención a los valores anormales que puedan ocurrir en su interior. ¿Cómo? Modificando el uso del algoritmo DSP, en cuanto a que generalmente no debería producir valores anormales. Si estás diseñando tus propias estructuras DSP de nivel bajo en Reaktor Core, también deberás tener cuidado con los valores anormales. Para ayudarte en esta tarea te ofrecemos el módulo *Denormal Cancel* disponible en el sub-menú *Built-In Module > Math*:



El módulo *Denormal Cancel* lleva una entrada y una salida y modifica lentamente el valor entrante, ya que los valores anormales no pueden llegar a la salida:



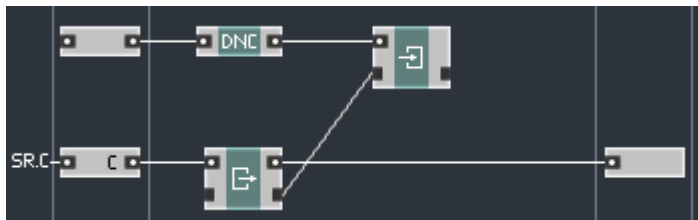
El modo en que este módulo modifica la señal no es fijo, y puede cambiar de una versión software a otra, o incluso de un lugar de la estructura a otro. Actualmente sucede añadiendo un constante muy pequeño al valor de entrada. Puesto que la precisión pierde esta suma, no modificará valores demasiado grandes (un valor de 10-10 no se modificará en absoluto), por lo tanto, el bastante improbable que el resultado de una adición pueda ser un valor anormal (en la mayor parte de los casos es incluso imposible).

Si por cualquier razón el módulo *Denormal Cancel* no funcionase para tu estructura, eres libre de usar tus propias técnicas de cancelación de valores anormales. Pero el problema puede ser que la técnica utilizada funcione en una plataforma y en otra no. Mientras que nosotros vamos a adaptar el algoritmo interno *DN Cancel* para cada plataforma. Así que por favor, siempre que sea posible, usa el módulo *DN Cancel*. Vamos incluso a considerar el hecho de construir algoritmos alternativos dentro de este módulo – puedes darnos tu opinión en los Foros.

Algunas CPUs ofrecen una opción para quebrantar el estándar IEEE deshabilitando la producción de valores anormales. Puesto que las estructuras de Reaktor Core están diseñadas para poder usarse sea cual sea su plataforma, es importante que siempre prestes atención a la cancelación de los valores anormales dentro de tus estructuras, incluso aunque tu sistema particular no los padezca.

Puesto que una de las situaciones comunes en las que podrían aparecer es en los loops de feedback con decay exponencial (lo que incluye – aunque no se limita a – filtros, y estructuras de loops de feedback con delays), hemos decidido construir la cancelación de anormales dentro de la macro estándar Z^{-1} .

Como recordarás, el interior de esta macro era así:



Te preguntará qué hace aquí el módulo Denormal Cancel. Puesto que normalmente usarías la macro Z^{-1} dentro de estructuras con feedback, hay bastantes probabilidades de que ocurran anormales. Por lo tanto, hemos decidido poner este módulo dentro de la estructura de la macro Z^{-1} .

Existe otra versión de esta macro llamada $Z^{-1} \text{ ndc}$ que no realiza cancelaciones de anormales (ndc = no denormal cancel). Puedes usarlos en estructuras en las que estás seguro de que no se van a generar anormales (por ejemplo, filtros FIR):

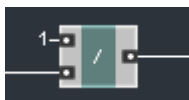


Otras cifras peligrosas

Los números anormales no son el único tipo negativo de números que pueden introducirse en la categoría interna de la estructura, sobre todo en los Loops de feedback. Hay unos cuantos: INFs, NaNs y QNaNs. No vamos a entrar en detalle hablando de ellos, esta información puedes encontrarla en muchos sitios, incluyendo Internet. Lo importante es prevenirte de que ocurran en tus estructuras. Muchas veces, estos números aparecerán como resultado de operaciones inválidas. Dividir entre cero es el caso más fácil. Otros casos tienen que ver con números demasiado grandes como para encajar en la representación de los dígitos flotantes (sería por encima de 10^{38}), o números fuera del ámbito de una operación particular.

Dichos números tienden a dañar las estructuras, todavía más que los números anormales. Por ejemplo, si añades un valor anormal a otro valor que no sea anormal, el resultado será no-anormal (a menos que el otro valor esté muy cerca de serlo). Si añades un valor normal a un INF, el resultado seguirá siendo INF. Además de tener tendencia a pegarse en las estructuras permanentemente (es decir, hasta que la estructura se reajuste), estos números también tienen la mala costumbre de requerir unos tiempos de procesamiento mucho más largos en algunas CPUs. Por lo tanto, es muy importante tener cuidado de crear ninguno.

Tener cuidado significa, por ejemplo, que si alguna vez divides dos números, tendrás que comprobar si una división entre cero es posible. El caso de inicialización requiere una atención particular en este apartado. Por ejemplo, vamos a estudiar el siguiente elemento de esta estructura:



Si por cualquier razón el evento de inicialización no llegase a la entrada inferior del módulo Divisor, ocurriría una división entre cero durante el proceso de inicialización. En este caso, podrías usar en su lugar una macro delay de “modulación” o cualquier otra solución, dependiendo de tus necesidades específicas.

Construir un filtro paso-bajo de 1-pole

Podemos construir un filtro paso-bajo de 1-pole usando una ecuación recurrente:

$$y = b * x + (1 - b) * y_{-1}$$

donde

x es el sample de entrada,

y es el nuevo sample de salida,

y_{-1} es el sample de salida previo, y

b es el coeficiente que define el corte de filtro.

El valor del coeficiente b es igual a frecuencia cíclica de corte normalizada, y podemos calcularla usando la siguiente fórmula:

$$F_c = 2 * \pi * f_c / f_{SR}$$

donde

f_c es la frecuencia corte deseada en Hz

f_{SR} es la frecuencia de muestreo en Hz

π es 3.14159...

F_c es el corte cíclico normalizado (en radianes, si quieres saberlo)

De hecho, el coeficiente b es igual al corte normalizado sólo de forma aproximada. El error incrementa en valores altos de corte, pero más o menos funcionará para nuestros propósitos, sobre todo si no necesitamos disponer de una precisión total de frecuencia de corte en nuestro filtro.

Empezaremos creando una célula core de audio con dos entradas: una para la entrada de audio y otra para el corte. En esta versión del módulo vamos a usar una entrada de evento para el corte.

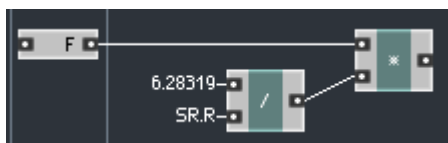


En realidad, puesto que pensamos que construir estructuras de Reaktor Core como macros core es una buena costumbre para incrementar su valor de uso,

vamos a crear nuestro filtro también como una macro. Así pues, creamos una nueva macro dentro, con las mismas entradas: a macro.



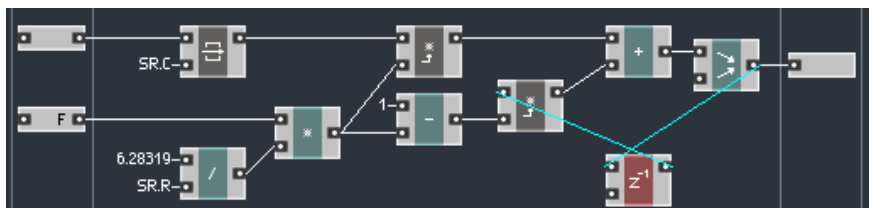
Ahora vamos a construir el circuito para convertir la frecuencia de corte en el corte cíclico normalizado:



6.28319 es $2*\pi$, que se divide por la frecuencia de muestreo, formando así el valor que se multiplicará con la frecuencia del corte. No necesitamos un multiplicador de “modulación” porque, lógicamente, “F” es una entrada de señal de control, así que realizaremos la multiplicación incluso aunque no haya evento de inicialización en la entrada “F”.

Llevaremos cabo la división antes de la multiplicación, ya que la división es una operación más pesada para la CPU y la frecuencia de muestreo no cambia tan a menudo. Si sólo cambia la frecuencia de corte, no habrá eventos enviados al módulo divisor y por lo tanto la división no se llevará a cabo. Esta es una de las optimizaciones que puedes hacer en Reaktor Core.

Vamos a construir el circuito implementando la ecuación del filtro:



La entrada de audio se cerrará en caso de que lleguen eventos asíncronos al reloj de audio estándar. Esto no tendría que ser necesario en la estructura de una célula core en la que se sabe que una entrada de audio envía eventos a tiempos correctos, pero en una macro core general, es una buena práctica.

Hay dos multiplicadores de modulación para evitar que los eventos que lleguen a la entrada “F” (lo que, hablando en general, sucede siempre) activen el cálculo en el loop de feedback.

El uso de cerrojos es una técnica estándar en Reaktor Core para impedir que los eventos entrantes activen los cálculos en el momento equivocado. También se emplea mucho en forma de macros de modulación u otras situaciones similares.

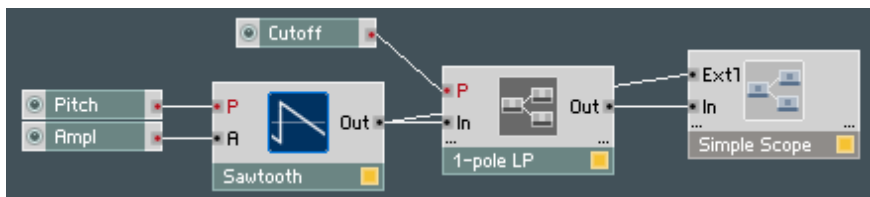
El módulo Z^{-1} se usa para almacenar el valor de salida anterior y enviar automáticamente un evento con el valor de salida anterior en cada tic del reloj de audio. También atiende posibles irregularidades con valores anormales. Aquellos familiarizados con el DSP deberían darse cuenta de que la estructura se parece bastante a los diagramas de filtro DSP estándar.

El módulo Merge en la salida del módulo de suma asegura que la categoría del filtro tras la inicialización sea todavía 0, incluso aunque la señal de entrada no tenga un valor de cero en ese momento.

No te olvides de poner el tono al convertidor de frecuencia dentro de la estructura de la célula core. Ya estamos listos para probar:



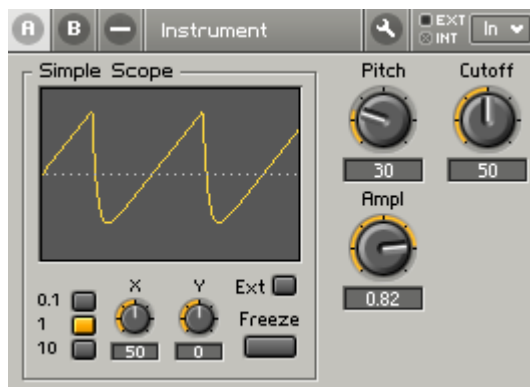
Para la prueba te sugerimos que uses la siguiente estructura (sin olvidar que el instrumento ha de tener 1 voz):



El knob Cutoff debería ajustarse entre 0 y 100, o algo similar. Ten cuidado con los valores de frecuencia de corte muy altos. Debido al incremento de error del coeficiente del filtro en frecuencias de corte muy altas, el filtro se volverá inestable con valores grandes de corte.

Un buen diseño de filtro implicaría ajustar los valores de corte de forma que el filtro se mantenga estable. En nuestro caso, podría hacerse recorriendo el coeficiente b dentro de un campo entre 0..0.99, o algo similar. Las técnicas para llevarlo a cabo las explicaremos más tarde.

Así deberías ver el panel ahora:



Mueve el knob de corte y observa cómo cambia la señal.

Procesamiento condicional

Rutas de evento

El evento en Reaktor Core no siempre ha de viajar por las mismas rutas predefinidas. Existe una posibilidad de cambiar esas rutas dinámicamente. Puedes hacerlo usando el módulo *Router* (*Built-In Module > Flow > Router*):



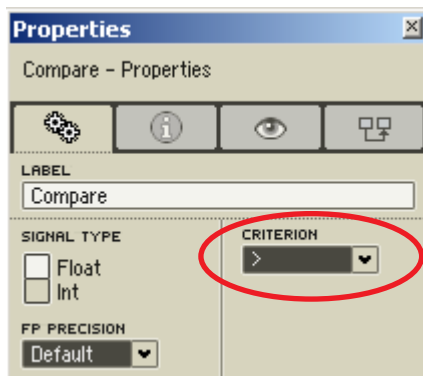
El módulo *Router* acepta eventos en sus entradas de señal (abajo) y las rutea a su salida 1 (arriba) o a su salida 0 (abajo). Las salidas por las que se ruteen dependerá de la categoría actual del Router que se controla desde la entrada *Ctl* (arriba). La entrada *Ctl* acepta un nuevo tipo de conexión que no es compatible con señales normales ni con conexiones OBC. Se trata de una señal *BoolCtl* (control Boolean). La señal *BoolCtl* se puede dar en dos categorías: verdadera o falsa (on u off, 1 ó 0). Si la señal de control está en la categoría verdadera, los eventos se rutean a la salida 1. Si la señal de control está en la categoría falsa, los eventos se rutean a la salida 0.

Las señales de control son considerablemente distintas de las señales normales en Reaktor Core: no transmiten eventos y por lo tanto no pueden activar ningún procesamiento por sí mismas.

Para controlar un Router necesitarás una fuente de señal de control. La más común sería el módulo Comparison que encontrarás en *Built In Module > Flow > Compare*:



Este módulo realiza una comparación entre dos señales entrantes y transmite el resultado en forma de señal BoolCtl. Damos por hecho que la entrada de arriba estará a la izquierda del signo de comparación y la de abajo, a la derecha. De forma que un módulo que lee '>', produce una señal de control verdadera si el valor de la entrada de arriba es mayor que el valor de la entrada de abajo. Puedes cambiar el criterio de comparación en las propiedades del módulo:

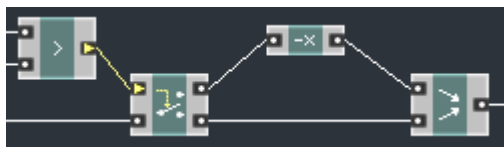


Los criterios disponibles son:

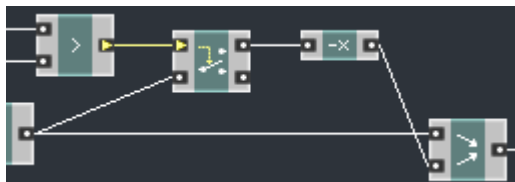
- = igual
- != no igual (\neq)
- <= menor o igual (\leq)
- < menor
- >= mayor o igual (\geq)
- > mayor

Por supuesto, es posible conectar varios ruteadores al mismo módulo de comparación. En ese caso, sus categorías cambiarían simultáneamente.

El módulo Router divide la ruta del evento en dos ramas. Con frecuencia estas ramas volverán a asociarse.



Dependiendo del resultado de la comparación, la estructura de arriba invertiría la señal de entrada o la dejaría intacta. También es posible una implementación alternativa en esta estructura.



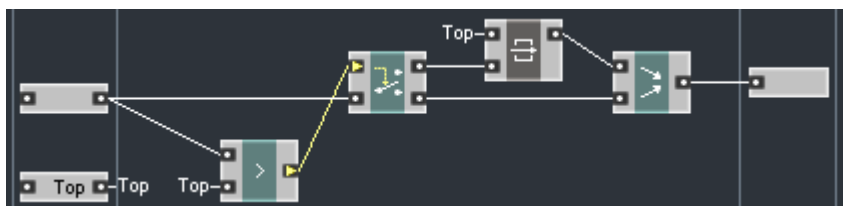
En esta versión, la salida 0 del Router está desconectada, por lo tanto el Router funciona como puerta, dejando pasar a los eventos sólo si su categoría es verdadera. El valor invertido llega a la segunda entrada del Merge, y por lo tanto anula el valor no-invertido, que siempre llega a la primera entrada. Si el ruteador estuviese en la categoría falsa, el módulo negador no recibiría un evento ni enviaría un evento a la segunda entrada del Merge, por lo tanto, a la salida del Merge llegaría la señal original sin modificar.

Las ramas se asocian normalmente con un módulo Merge. Pero en teoría, podrías usar muchos otros módulos (por ejemplo, módulos aritméticos como módulos de suma, multiplicadores, etc).

Los ruteadores tratan un evento de inicialización de la misma forma que tratan cualquier otro evento. Por lo tanto, los ruteadores se pueden usar para filtrar un evento de inicialización, para que de ese modo el evento de inicialización no apareciese en áreas particulares de la estructura.

Construir una macro de recorte de señales

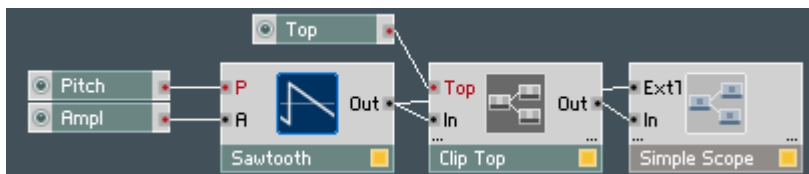
Vamos a construir una estructura macro de Reaktor Core que recorte la señal de audio entrante de arriba a un nivel específico:



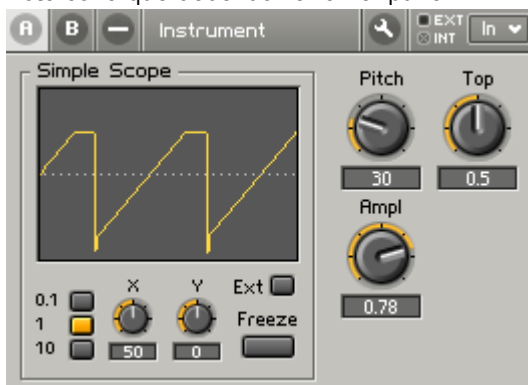
Si la señal de entrada no es mayor que el threshold se enviará a la salida 0 del Router y, a través del Merge, a la salida de la estructura. De otra forma, la señal se rutearía a la salida 1, donde activaría el cerrojo, enviando el valor de threshold al Merge. Lo mismo ocurre durante la inicialización.

Observa que esta estructura no cambia su salida en respuesta a los cambios del threshold. En lugar de eso, el nuevo valor de threshold se usará para el siguiente y los subsiguientes eventos en la señal de entrada. Se parece al comportamiento de las macros de modulación, donde los cambios de modulación no ofrecen como resultado eventos de salida.

He aquí una estructura de prueba para usar el módulo de recorte que hemos construido (se ha usado una célula core de audio):



Esto es lo que deberías ver en el panel:



De hecho, hay varias de estas macros de “modulación” por recorte en el menú Expert Macro > Clipping: *Expert Macro > Clipping* menu.

Construir un oscilador de diente de sierra simple

Vamos a construir un oscilador de diente de sierra simple, generando una forma de onda de diente de sierra de amplitud 1 y frecuencia específica. Usaremos el siguiente algoritmo: incrementa el nivel de la señal de salida a una velocidad constante y en el momento en el que el nivel sea mayor que 1, bájalo 2.

Uno podría también argumentar que en lugar de bajar 2, podríamos haber reajustado el nivel a -1 , pero en general esta es una mala idea, ya que en ese caso no seríamos capaces de mantener de forma precisa la frecuencia necesaria del oscilador.

La velocidad de incremento define la frecuencia del oscilador por la siguiente ecuación:

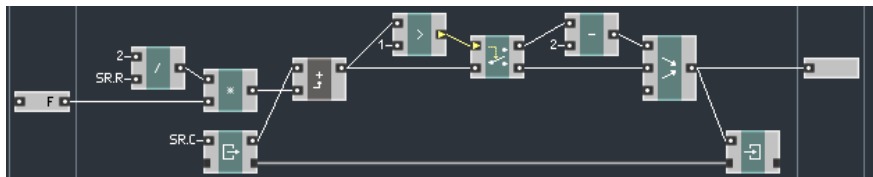
$$d = 2f / f_{SR}$$

donde d es el nivel de incremento por un sample de audio, es la frecuencia del oscilador, y f_{SR} es la frecuencia de muestreo.

Primero vamos a construir el circuito para calcular la velocidad de incremento:



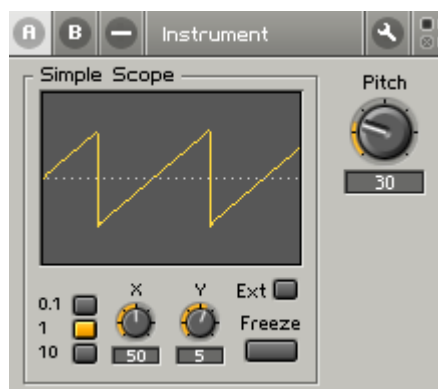
Ahora necesitamos el loop de incremento. Es una vuelta a usar un par de módulos *Read* y *Write* igual que antes hicimos con el acumulador:



El módulo *Read* activa el nivel de incremento durante cada evento de audio. La suma del nivel antiguo y el incremento se comparan contra 1 y se rutean, o bien directamente a la escritura de resultado o bien al circuito protegido. La tercera entrada del módulo *Merge* asegura la inicialización del oscilador a cero. Teóricamente, el módulo que sustrae 2 desde el nivel de la señal, tendría que haber sido una macro de modulación, pero no ha de preocuparnos mucho, puesto que el *Merge* anula el resultado de la inicialización de todas formas. He aquí la estructura que sugerimos (no olvides el convertidor P2F dentro de la célula core):



Y este es el panel:



Más tipos de señales

Señales flotantes

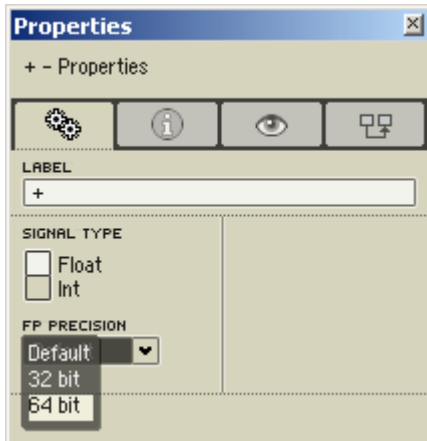
El tipo de señal más comunmente usado para el DSP (digital signal processing) en los ordenadores personales de hoy en día, son los dígitos de coma flotante (flotantes para abreviar). Los valores flotantes puede representar una gran cantidad de valores, tan grandes como 10^{38} (en modo 32 bit) o incluso 10^{308} (en modo 64 bit). Tan buenos como son, tienen un inconveniente: precisión limitada. La precisión es mayor en modo 64 bit, pero sigue siendo limitada.

Limitar la precisión de un valor flotante tiene unas razones teóricas. Si no estuviese limitada, dichos valores necesitarían una infinidad de memoria para almacenarlos. Es lo mismo por lo que no puedes escribir la representación decimal total de un número trascendental (por ejemplo π) en una página “finita” de papel. Incluso aunque conocieses todos sus dígitos (¿cómo podrías saberlos? Uno no puede recordar un número infinito de dígitos, pero en el caso de que tú fueses capaz), simplemente intenta escribirlos: 3, 1, 4, 1, 5, 9, etc. En algún momento te saldrías del papel. De igual manera, intentar procesar estos números dentro de una CPU, requeriría una infinita memoria. .

El almacenamiento de señales y memoria que has estado usando usaban números de puntos flotantes a 32 bit en su representación. Reaktor Core te ofrece la posibilidad de usar flotantes a 64 bit y también, si lo necesitas, más precisión (aunque es difícil imaginar que de 10^{-38} a 10^{38} no sea suficiente).

Por defecto todos los procesos en Reaktor Core son a 32 bit coma flotante. Esto no supone exactamente que las señales estén procesadas a 32 bit flotante, sin que al menos se utilizará como mínimo 32 coma flotante (en ocasiones se utilizarán los 64 bit flotantes para los puntos intermedios).

Podrás cambiar la precisión de dígitos flotantes en módulos individuales y en macros. Para módulos individuales lo harás en la propiedad *FP Precision* (floating point precision) de estos módulos:



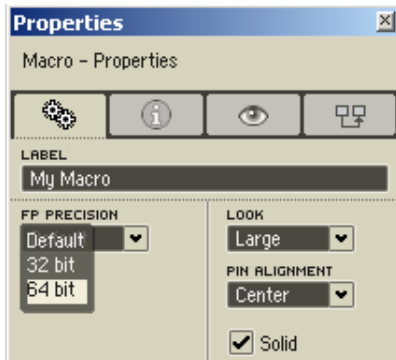
por defecto significa que usa la precisión por defecto de la estructura actual

32 bit usa 32 bits como mínimo de precisión

64 bit usa 64 bits como mínimo de precisión o

Cambiar la precisión de un módulo significa que el procesamiento dentro de ese módulo se realizará usando una precisión específica, y que el valor de salida también se generará usando la precisión específica.

También puedes cambiar la precisión por defecto para la estructura completa haciendo clic con el botón derecho sobre el fondo de la pantalla y seleccionando *Owner Properties* para abrir las propiedades del módulo:



Esta precisión por defecto se hará efectiva para todos los módulos que haya dentro de la estructura actual, incluyendo las macros, ya que ellos no definen su propia precisión (o, en caso de las macros, la nueva precisión por defecto para sus respectivas estructuras interiores).

Las señales normales de puntos flotantes de 32 y 64 bit son totalmente compatibles entre ellas y se pueden interconectar libremente. Las señales OBC de diferente precisión no son compatibles entre ellas (ya que no se puede disponer de un almacenamiento a 32 y 64 bit a la vez). También, para señales OBC, los ajustes “por defecto”, “32 bit” y “64 bit” se consideran diferentes e incompatibles, ya que la precisión efectiva por defecto se puede alterar cambiando las propiedades de una de las macros que posee.

Los módulos de entrada y salida de las estructuras de alto nivel de las células core siempre envían y reciben flotantes a 32 bit, ya que es el tipo de señal que se usa en las conexiones de audio y evento del nivel primario de Reaktor Core

Señales enteras

Existe otro tipo de dato que normalmente soportan las CPUs modernas, y de hecho, es más fundamental con respecto al mundo digital que los flotantes. Es el tipo *entero*. Los números enteros se representan y se procesan con infinita precisión. Aunque la precisión de los enteros es *infinita*, el campo de valores representables está limitado. Para enteros de 32 bit, los valores pueden llegar hasta más de 10^9 ..

La precisión infinita de almacenamiento y procesamiento en los valores enteros es posible porque no tienen dígitos decimales después de la coma, así que se pueden escribir usando un número finito de dígitos. Vamos a escribir el número de segundos en una hora: 3, 6, 0, 0, hecho. Es así de fácil. Si intentas escribir el valor de π no podrás hacerlo completamente: 3, 1, 4, 1, para. ¿No está completo? Bien, vamos a escribir un par de dígitos más: 5, 9, para. Todavía no está completo, etc. Pero con un número entero, puedes hacerlo completa y precisamente: 3600, eso es.

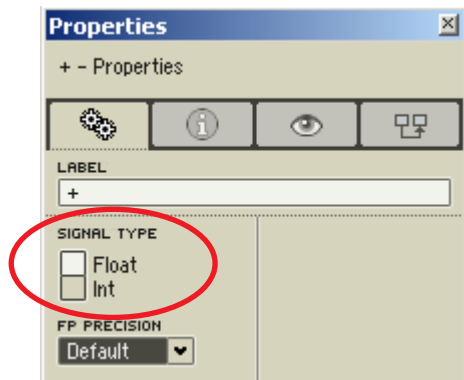
Mientras que el tipo flotante es la elección natural de los valores que están constantemente cambiando, como las señales de audio, para valores que cambian discretamente (por ejemplo, contadores), los enteros sería una elección más apropiada.

Muchos de los módulos de Reaktor Core se pueden alternar con el modo entero, en el que esperarán recibir señales enteras en su entrada, procesarlas como

enteras (es decir, con infinita precisión) y producir señales enteras en la salida. Este tipo de módulos podrían ser aritméticos, como de suma, multiplicador, sustractor. Pero hay incluso módulos que sólo se pueden usar con enteros.

Para valores enteros de Reaktor Core está garantizado un mínimo de 32 bit.

Alternar entre tipos flotantes y enteros (si el módulo puede) es algo que podrás hacer en la propiedad *Signal Type* del módulo:



Un módulo ajustado al tipo entero procesará los valores entrantes como enteros y producirá valores enteros en la salida. Podrás saber si el módulo está en la categoría de entero por el hecho de que sus entradas y salidas aparecen de forma distinta:

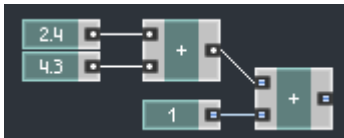


No existe el tipo de señal por defecto para macros. La razón es que normalmente, no podrías construir las estructuras que procesaran valores enteros exactamente igual que las estructuras que procesan flotantes y viceversa.

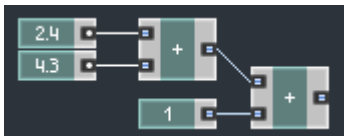
Las señales enteras se pueden interconectar libremente con flotantes, pero los cables creados entre dichas señales realizarían una conversión de señal que necesitaría mucha CPU. En el momento en que escribimos este texto, esa cantidad de CPU es bastante considerable en PCs y poco significativa en Macs. Las conexiones OBC de tipos flotantes y enteros no son compatibles entre ellas, por supuesto. Puede que se pierda información durante dicha conversión. Sobre todo, los enteros muy grandes no se pueden representar de forma muy precisa

por los flotantes, y los flotantes no se pueden representar de forma precisa por los enteros. Los enteros muy largos (más largos que el entero más largo que se pueda representar) no se pueden representar como enteros. En este caso, el resultado de la conversión será indefinido. Durante la conversión de flotante a entero, los valores se acercarán aproximadamente al entero más cercano. “Aproximadamente” significa que un resultado de alrededor de 0.5 puede ser 0 ó 1, aunque puedes fiarte del hecho de que 0.49 estará más cerca de 0 y 0.51 de 1.

Es importante comprender que cambiar el modo de procesamiento en una operación a enteros, y realizar una conversión del resultado de puntos flotantes de la misma operación a enteros, no será lo mismo. Veamos un ejemplo. Vamos a sumar dos número 2.4 y 4.3 como flotantes. El resultado es claramente 6.7, que convertido a entero, produciría 7. Así, la salida de la siguiente estructura es 8:



Ahora, si cambiamos el modo del primer módulo de suma a entero, en lugar de sumar 2.4 y 4.3, sumaría sus versiones más cercanas, es decir, 2 y 4 respectivamente, produciendo 6 en la salida. Así el resultado sería 7:



Las entradas de reloj ignoran sus valores entrantes, por lo tanto normalmente siempre serán flotantes. La conversión de tipo de señal tampoco se llevará a cabo en señales que se usen sólo como reloj.

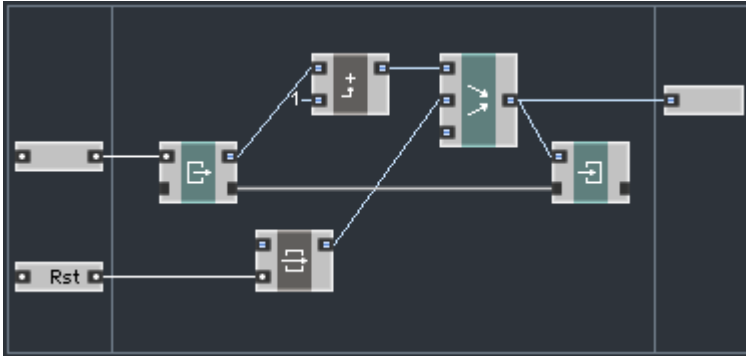


Aquí, la entrada de reloj del módulo Read todavía está negra aunque el módulo se ha ajustado al modo entero (los puertos OBC mantienen la misma apariencia aunque se cambien al modo flotante o entero).

El feedback entero se resuelve automáticamente del mismo modo que el feedback flotante – insertando un módulo Z^{-1} en modo entero (por supuesto aquí no se necesita cancelación anormal).

Construir un contador de evento

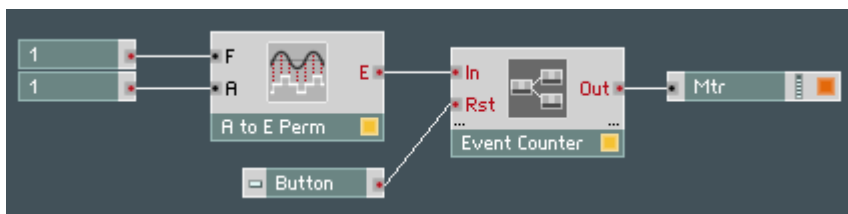
Vamos a construir un módulo contador de evento. La funcionalidad de esta macro será similar al acumulador de evento, pero en lugar de sumar acumulando los valores de los eventos, ésta simplemente los enumerará. Parece que la elección más apropiada para enumerar sería un tipo de señal entera.



Aquí, la salida y todos los módulos internos se han ajustado al modo entero. La macro ILatch se usa en lugar del Latch para reajustar el circuito. Hace exactamente lo mismo (y se puede encontrar en el mismo menú), pero trabaja con señales enteras. También se ha usado una macro de modulación entera (lo encontrarás en el menú *Expert Macro > Modulation > Integer*). Ninguna de las dos entradas necesita ajustarse al modo entero, puesto que sólo proporcionan señales de reloj. Si echamos un vistazo a la estructura de la célula core de evento que contiene esta macro:



veremos que la salida de este módulo no está ajustada al modo entero (tampoco es posible ajustarlo a él en el modo entero). Es porque, al ser la célula core que hay en el exterior un módulo del nivel primario de Reaktor, ha de sacar un evento normal del nivel primario, que es un valor flotante: He aquí una estructura de prueba para el módulo contador:



Y el panel:



Construir una macro como contador de tramos ascendentes

Ahora vamos a hablar de una técnica de comparación de signos que quizá alguna vez necesites al construir estructuras de Reaktor Core. Comparación de signos se refiere a una comparación de dos números de los que ignoras por competo su valor, y se presta atención sólo a su signo (“más” o “menos”). Naturalmente, más se considera mayor que menos. Por ejemplo:

3.1 tiene mayor signo que -1.4

2.1 tiene igual signo que 5.0

-4.5 tiene igual signo que -2.9

Ten cuidado porque el signo cero es indefinido. Esto significa que el resultado de cualquier comparación de signos que incluya un valor de cero puede ser arbitrario.

Por supuesto podrías haber implementado la funcionalidad de comparación de signos usando distintos módulos de Comparación y distintos Routers, pero éste es un método más eficiente. La comparación de signos se puede hacer en las estructuras de Reaktor Core usando el módulo *Compare Sign* (*Built-In Module > Flow > Compare Sign*):



El módulo produce una señal BoolCtl en la salida, de forma que puedes conectarlo a un Router.

Uno de los usos posibles de estos módulos sería detectar los tramos ascendentes de una señal de entrada. A continuación vamos a construir un contador de tramos ascendentes en Reaktor Core:

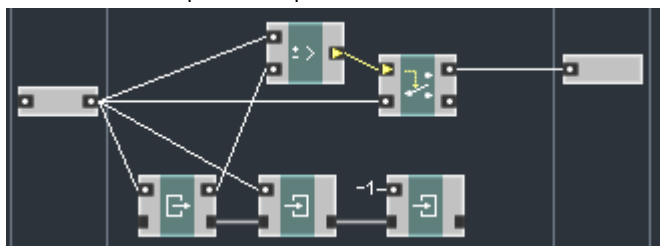


Observa que la entrada está ajustada al modo entero, puesto que la cuenta es un valor entero.

Lo primero que vamos a necesitar dentro es una macro detectora de tramos que convierta el tramo detectado en un evento:



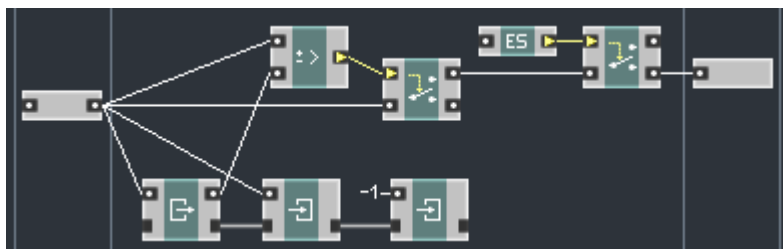
Así es como se puede implementar la macro detectora de tramos:



La cadena OBC que hay abajo guarda el valor de la señal entrante anterior. Como puedes ver, el valor se almacena después de que el antiguo ha sido leído. El último módulo Write de la cadena realiza el trabajo de inicialización en el almacenamiento del valor anterior. Inicializaremos el almacenamiento a -1 para que el primer valor positivo se cuente como tramo ascendente.

El módulo Write al final de la cadena es una alternativa diferente del Merge para inicializar el almacenamiento. Debe ser el último módulo Write de la cadena para que pueda anular los resultados almacenados por los módulos Write emisores.

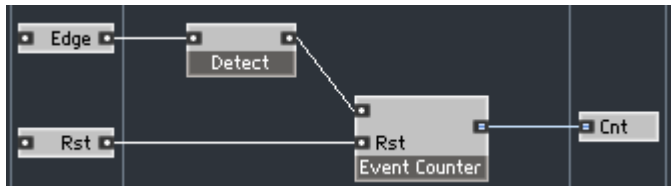
El Router controlador por el módulo Songn Comparison hará de puerta para los eventos, dejando pasar sólo aquellos en los que el cambio de signo se dé de negativo a positivo. No está claro si dicho módulo enviará un evento durante la inicialización o no, sobre todo porque el almacenamiento todavía es cero en el tiempo de procesamiento los eventos de inicialización y el signo cero es indefinido. Podemos modificar esta estructura para evitar que se envíen eventos durante la inicialización:



El módulo ES Ctl es un “control sensitivo de eventos”. La señal de control producida por el módulo es verdadera sólo si hay un evento entrante en la entrada del módulo. Puesto que esta entrada está desconectada en la estructura de arriba, lo que significa que está conectada a un constante de cero, el único momento en que la señal de control es verdadera es en la inicialización. Así pues, el ruteador bloqueará cualquier evento que pueda suceder durante la inicialización y dejará que los otros pasen.

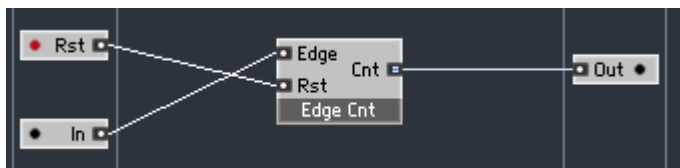
Observa que aquí tenemos un ejemplo de un módulo que no envía ningún evento desde su salida durante la inicialización.

Ahora que ya tenemos un módulo detector, podemos conectarlo al circuito de cálculo que ya teníamos:

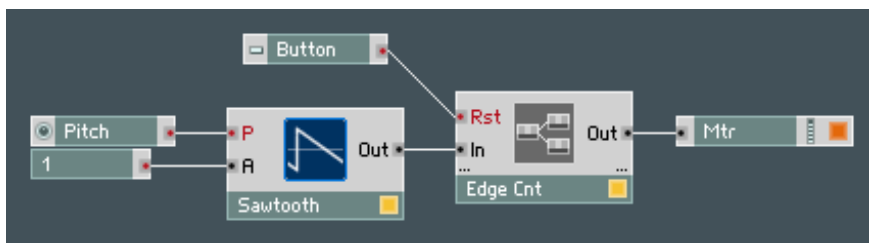


La funcionalidad del circuito de arriba ha de estar clara. El módulo Detect envía un evento cada vez que detecta un tramo ascendente. Estos eventos los cuenta el módulo Evt Cnt.

Para construir un circuito de prueba vamos a poner esta macro en una célula core de audio, y vamos a contar los tramos ascendentes de la forma de onda de diente de sierra. La estructura interna de la célula core será así:



Y la estructura de prueba del nivel primario:



(No olvides ajustar las propiedades del Meter como en los ejemplos anteriores). Esto es lo tendrías que ver en el panel:



La velocidad del cambio del número en el medidor corresponde a la frecuencia del oscilador, definida por le knob de tono. En un valor de tono de cero, la frecuencia del oscilador es aproximadamente 8 Hz, así que los números deberían incrementar a una frecuencia aproximada de 8 por segundo.

Series

Introducción a las series

Vamos a imaginar que quieres construir un módulo selector de señales de audio que, dependiendo del valor de la entrada de control, recoja la señal desde una de las cuatro entradas de señales de audio:



Un procedimiento para implementarlo sería usando módulos Router, pero también hay otra posibilidad. Podemos usar otra característica de Reaktor Core – las series.

Una *serie de una dimensión* es una *colección ordenada* de objetos de datos *del mismo tipo* que se pueden tratar según su rango en este orden o índice. Por ejemplo, aquí tenemos un grupo de 5 números flotantes:

5.2 16.1 -24.0 11.9 -0.5

En Reaktor Core, los índices de elementos de serie están basados en cero. Esto significa que el primer elemento de la serie tiene un índice de cero. Por lo tanto, el elemento con un índice de cero es 5.1, con un índice de 1 nos da 16.1, con un índice de 2 se trata como -24.0, el “tercer elemento” es 11.9 y el “cuarto elemento” es -0.5.

Aquí tienes una representación de esta serie usando una tabla:

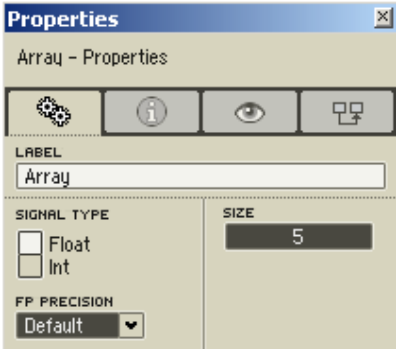
Here is a representation of this array using a table:

Índice	0	1	2	3	4
Valor	5.2	16.1	-24.0	11.9	0.5

Las series se crean en Reaktor Core usando módulos de series (*Built-In Module > Memory > Array*):



Un módulo Array tiene una salida sencilla que es de tipo *Array OBC*. El tamaño de esta serie (el número de elementos) y el tipo de datos almacenados en los elementos de la serie vienen especificados en las propiedades del módulo



Por ejemplo, para la tabla de arriba de 5 elementos, necesitaremos especificar el tipo de dato *flotante* y el tamaño los 5.

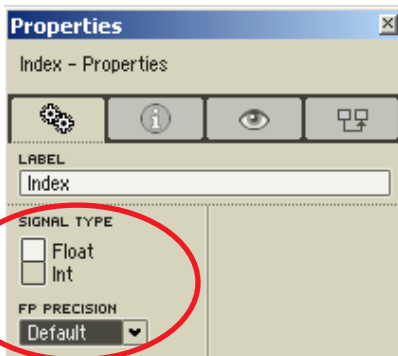
Por favor, date cuenta de que, puesto que los índices de series en Reaktor Core están basados en cero, el alcance de índices para una serie de tamaño 5, sería de 0 a 4 (puedes verlo en la tabla de arriba).

Las señales de series OBC correspondientes a diferentes tipos de datos de objetos no son compatibles.

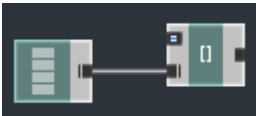
Para trabajar con un elemento de serie tendrás que especificarlo y catalogarlo. Puedes hacerlo usando el módulo *Index* (*Built-In Module > Memory > Index*):



La entrada OBC maestra (abajo) del módulo Index tendría que estar conectada a la salida esclava del módulo de serie. El tipo de conexión de entrada maestra tendrá que corresponderse con el tipo de serie. Lo primero se puede especificar en las propiedades del módulo Index:



Y ahora la conexión:

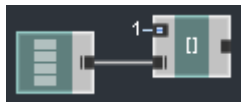


La entrada de arriba del módulo Index siempre lleva tipos enteros y acepta el valor del índice. Aquí nos encargamos del elemento de serie con el índice 1:



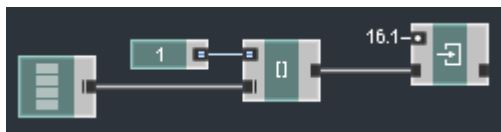
Observa que el módulo constante también se ha ajustado al modo entero (podrás saberlo por el aspecto del puerto de salida). En realidad no es necesario, puesto que de todas formas se hubiese llevado a cabo una conversión a enteros, pero así queda mejor.

También podríamos haber usado un QuickConst:



La salida del módulo Index tiene un tipo Latch OBC. Esto significa que puedes conectar módulos Read y Write (o incluso varios de ambos) a esta salida. Por supuesto, has de cuidar que los módulos Read y Write estén conectados a los mismos tipos de datos que los módulos Array e Index.

Aquí, el elemento de serie con índice de 1 se inicializará a 16.1:



Si se envía un índice fuera del alcance al módulo Index, el resultado al acceder a la serie es indefinido. La estructura no se accidentará, pero no definirá a qué elementos de la serie se puede acceder, o si habrá algún acceso de algún tipo. Esto significa que si no estás seguro del alcance del valor de índice, deberías recortarlo usando módulos o macros Routers de la librería.

Construir un selector de señales de audio

Vamos a volver a construir un módulo selector de señales de audio:



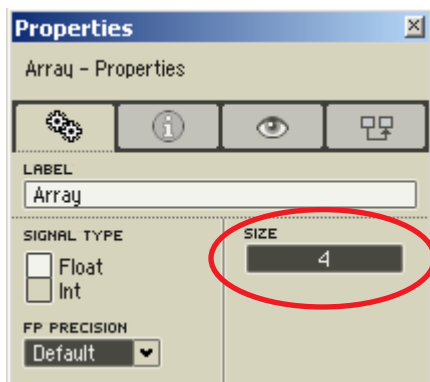
He aquí una estructura interna vacía para dicho módulo:



Vamos a usar una serie de 4 elementos flotantes para almacenar nuestras señales de audio:



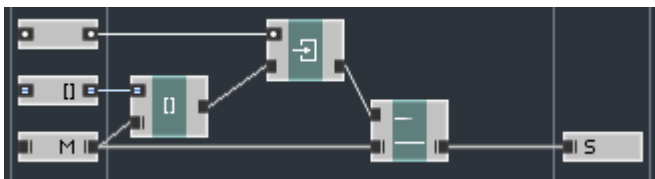
Estas son las propiedades del módulo de serie:



Para escribir los valores de entrada dentro de la serie vamos a usar una macro *Write []* estándar (*Expert Macro > Memory > Write []*):



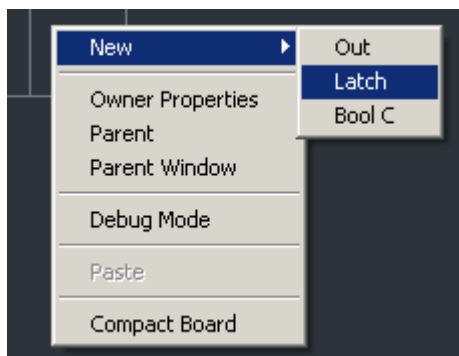
Internamente, esta macro tiene un módulo Index y un módulo Write, que sirven para escribir los elementos de la serie con un índice específico:



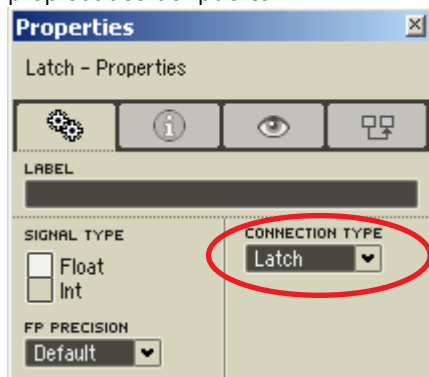
La entrada de arriba, claro está, recibe el valor que será escrito. La entrada "[]" recibe el índice en el que tendrá lugar la operación de escritura. La entrada "M" recibe la conexión OBC de la serie flotante a una precisión por defecto, y la salida "S" es una conexión "thru", similar a otros módulos OBC como *Read* y *Write*.

La entrada "M" y la salida "S" son otro tipo de puertos de macro, diferentes de los que hemos ido viendo hasta ahora. Estos puertos se pueden insertar

seleccionando la entrada “Latch” desde le menú de inserción de puerto (el tercero es un tipo de puerto macro *Bool/Ctl*):



Los puertos latch se pueden usar para conexiones latch OBC (entre módulos Read y Write) o para conexiones OBC de serie. Podrás controlar esto en las propiedades del puerto:



Ajustando el tipo de conexión a “Latch” o “Array” definirás la conexión como conexión OBC latch u OBC de serie, respectivamente. Para los puertos de la macro *Write []*, por supuesto, los hemos ajustado al tipo “Array”.

El módulo con dos líneas horizontales paralelas es el módulo *R/W Order* (*Built-In Module > Memory > R/W Order*):



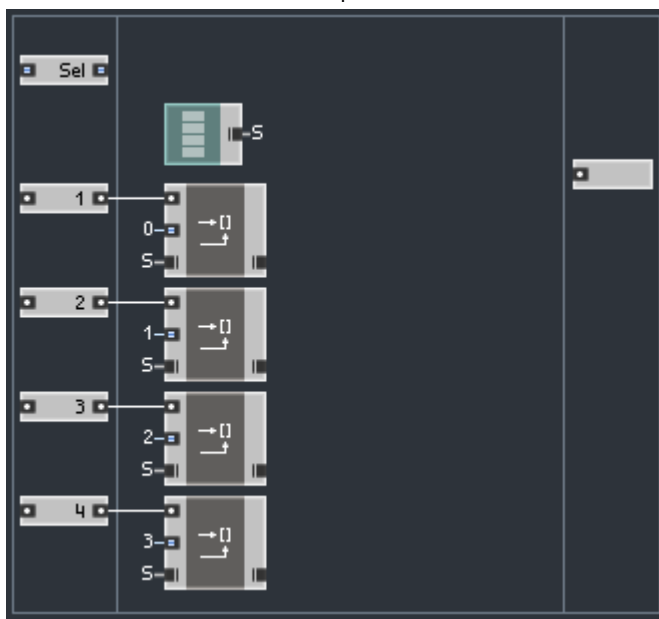
No hace nada, excepto dejar pasar la conexión de su entrada maestra (abajo) a su salida esclava. La entrada de arriba no tiene ningún efecto, pero como se trata de una conexión en su entrada afectará al orden de procesamiento

de los módulos. Por lo tanto, todo lo que esté conectado a la salida “S” de la macro se procesará después del módulo Write, que no sería el caso si el módulo R/W Order no estuviese en la estructura.

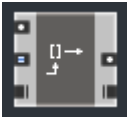
Sin del módulo R/W Order, la funcionalidad de la macro Write [] no sería fidedigna o intuitiva, ya que el usuario espera que todo aquello que esté conectado a la salida S de la macro Write [] se procesase después de esta macro. En general, este tipo de problemas aparecen sólo con conexiones OBC, en cuyo caso hay que poner módulos R/W Order cuando sea necesario dentro de las macros que diseñes.

El módulo R/W Order, al igual que los puertos OBC, tiene una propiedad *Connection Type*. En este módulo, dicha propiedad controla sólo el tipo de puertos “M” y “S”. La entrada “sidechain” siempre está en el modo “latch”. Mira la descripción del R/W Order en la sección de referencia del módulo para más detalles.

Vamos a construir un circuito para escribir las señales de entrada en la serie:

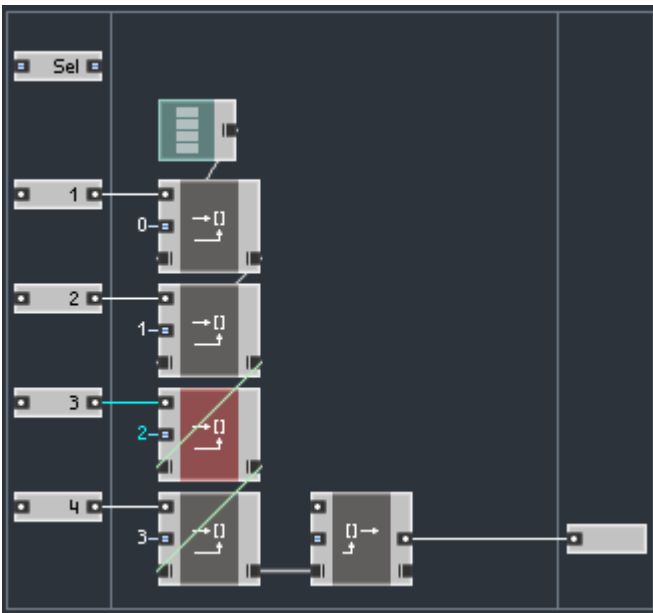


Los cuatro módulos [] se encargarán de almacenar los valores de audio entrantes en la serie. Ahora necesitamos un circuito para leer uno de los cuatro valores. Te sugerimos que uses la macro *Read []* (*Expert Macro > Memory > Read []*):

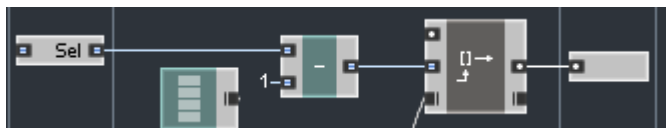


Esta macro realiza la lectura desde los elementos de una serie cuyo índice está especificado en la entrada entera del medio. La entrada de arriba es la entrada de reloj para la operación de lectura – enviará el valor leído a la salida de arriba del módulo en respuesta a un evento que llegue a su entrada. Los puertos que hay abajo son, por supuesto, las conexiones de serie esclava y maestra.

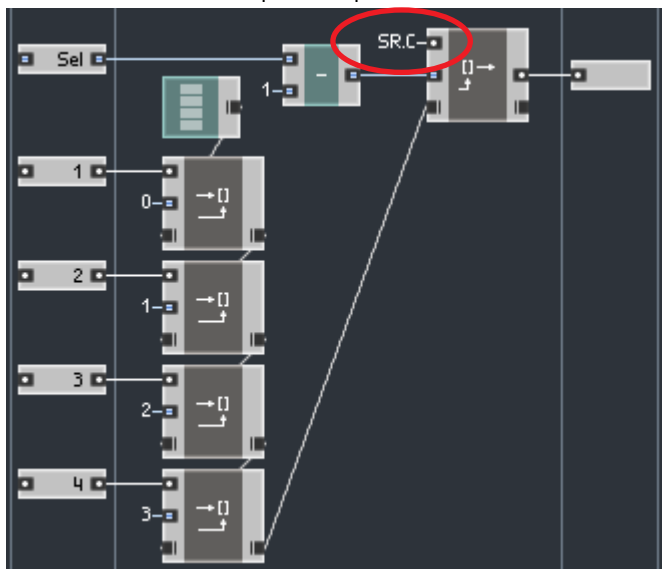
Ahora ¿a qué conectamos la entrada maestra? Obviamente no podemos conectarla directamente a un módulo de serie, ya que necesitamos que la operación de lectura se realice después de las operaciones de escritura (si no, puede producirse un retraso de un-sample, o no, pero de todos modos, es poco fiable). Tampoco podemos conectarlo a ninguno de los módulos Write [], ya que no solucionaría nuestro problema. Nuestra propuesta es que, en lugar de conectar los módulos Write [] en forma de “abanico” al módulo de serie, los conectes en serie para luego poder conectar el módulo Read [] a la salida del último módulo Write [].



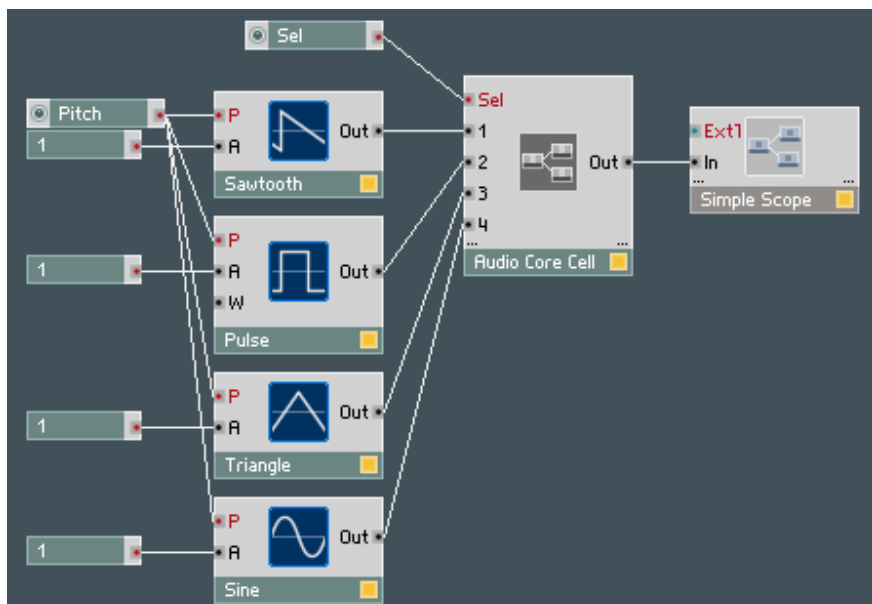
Ahora, ¿qué conectamos a la entrada índice del módulo Read []? Si queremos que el alcance de nuestro valor de selección esté entre 1 y 4, tendremos que sustraer 1 del valor de la entrada “Sel”. Observa que hemos realizado una sustracción entera (puesto que “Sel” es una entrada de control no necesitamos una macro de modulación aquí):



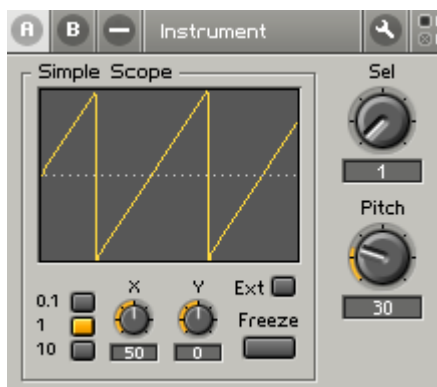
Por último, vamos a sincronizar el módulo de lectura con el reloj de la frecuencia de muestreo (puesto que se trata de un selector de audio):



En general, deberíamos haber recortado el valor de la entrada “Sel” para corregir el alcance, pero para simplificar las cosas, no vamos a hacerlo ahora. He aquí la estructura propuesta (la macro se ha puesto en una célula core de audio):



El knob Sel está ajustado para alternar entre 4 valores de 1 a 4. Ahora cambia al panel y mira las diferentes formas de onda que se corresponden con los ajustes del knob:



Construye un delay

Ahora que ya tenemos algo de experiencia con las series, vamos a construir una simple macro delay de audio. El módulo tendría que verse así:

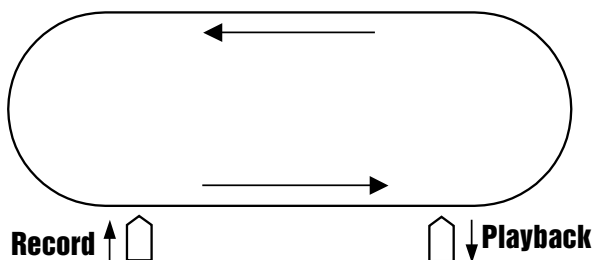


O incluso mejor, así (para ello tendremos que meternos dentro de la macro y cambiar la propiedad *Port Alignment* de la macro a “top”):

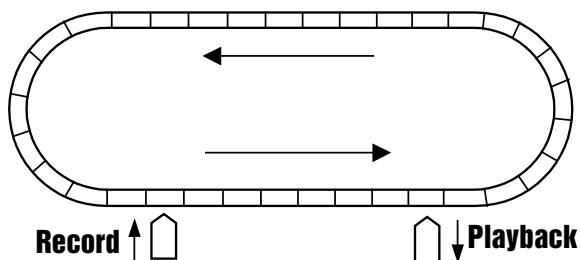


La entrada T aceptará el tiempo de delay en segundos.

Si echas un vistazo a un dispositivo analógico de delay de cinta, verás un loop de cinta combinado con cabezales de grabación y reproducción. En realidad, hay un cabezal de borrado, pero para hacerlo más sencillo, vamos a imaginar que el cabezal de grabación realiza ambos trabajos, el de borrado y el de grabación.

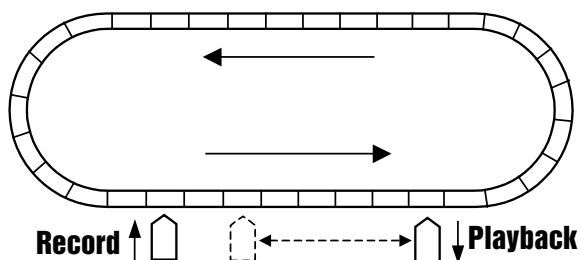


Si ahora queremos simular esto en forma digital, necesitaríamos un tipo de loop de cinta digital. Debido a la naturaleza discontinua del mundo digital, el loop de “cinta digital” soportará un número finito de samples de audio. Estos samples se grabarán y leerán en la frecuencia de muestreo del audio.



Una opción natural para un “loop en cinta digital” podría ser una serie. El tamaño de la serie sería igual al número de samples grabados en la totalidad del loop.

En un delay de cinta analógica, el tiempo del delay depende de la distancia que haya entre los cabezales de grabación y reproducción, que es fija, y la velocidad, que es variable. Se hace así por varias razones técnicas – es mucho más fácil variar la velocidad de la cinta que la distancia entre los cabezales. En el mundo digital ocurre todo lo contrario, ya que variar la velocidad de la cinta significa que tendremos que realizar una conversión de frecuencia de muestreo entre la “cinta digital” y la salida, mientras que variar la distancia entre los cabezales es mucho más sencillo. Eso es lo que vamos a hacer:



Hay otra diferencia con el mundo analógico. En el mundo analógico, la cinta se mueve. Si nosotros queremos mover nuestra “cinta digital” necesitaríamos realizar una copia de todos los elementos de la serie en sus posiciones vecinas por cada reloj de audio, lo cual puede ser muy costoso. Así que moveremos los cabezales.

De lo que acabamos de decir, podemos concluir que necesitaremos lo siguiente:

- | | |
|---------------------|--|
| serie | – para simular nuestro “loop de cinta digital” |
| índice de escritura | – será nuestro cabezal de grabación |
| índice de lectura | – será nuestro cabezal de reproducción |

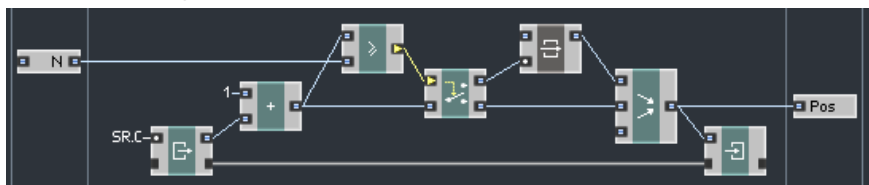
Los índices de lectura y escritura se moverán por la serie sample a sample. En el momento en que alguno de ellos llegue al final de la serie debería reajustarse al principio (ocurre por conectar los finales abiertos de la cinta al loop). La diferencia entre las posiciones de lectura y escritura corresponden al tiempo de delay medido en samples.

Esta es una técnica muy común en la programación y aquí se llama “buffer circular” o “buffer en anillo”.

Vamos a empezar programando el “cabezal de grabación”. Esta funcionalidad es muy parecida al oscilador de diente de sierra que hemos programado antes, excepto por que las operaciones se hacen en modo entero. El incremento del valor es uno por tic de audio y el alcance del valor de salida es desde 0 a $N-1$ donde N es el tamaño de la serie. Pongamos la computación de circuito del índice de escritura en el módulo “RecordPos”:

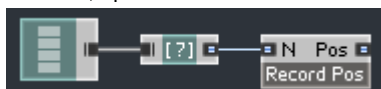


La entrada “N” aceptará el número de elementos de la serie, y la salida “Pos” transmitirá la posición de escritura actual (índice). Así es como podrás implementar esta macro (compáralo con la implementación del oscilador de diente de sierra):

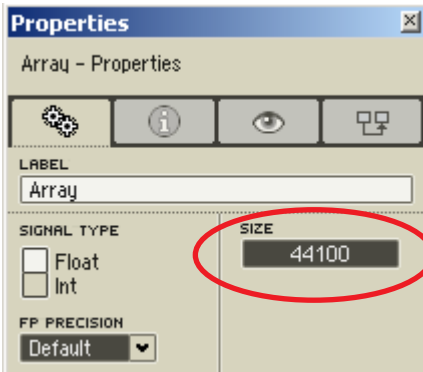


Observa que el módulo de comparación está ajustado a “>=”. Esto no era importante para el oscilador de diente de sierra (podríamos haber usado “>=” o “>”), pero en cálculos enteros es crucial. Usando la condición “>=” nos aseguramos de que el índice de escritura nunca alcance el valor de N .

En el nivel superior, creamos un módulo de serie y lo conectamos al Record-pos a través del módulo `Size []` (disponible en *Built-In Module > Memory > Size []*), que comunica el tamaño de la serie:



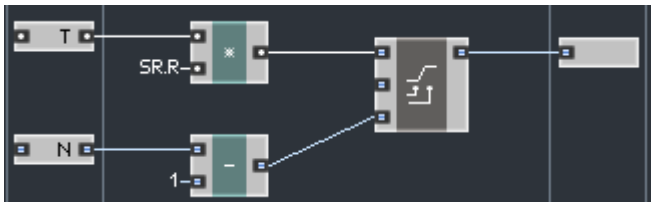
La propiedad de tamaño de la serie se puede ajustar a 44100. Esto nos permitirá hasta 1 segundo de delay (actualmente un sample menos) a una frecuencia de muestreo de 44.1KHz.



Ahora necesitamos calcular el índice de lectura. Lo haremos construyendo dos macros. La primera convertirá el tiempo de delay requerido en distancia de samples:



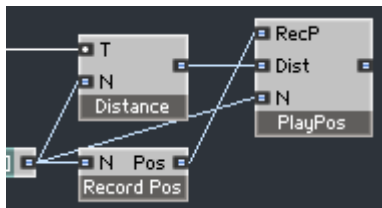
Se puede hacer multiplicando el tiempo en segundos por la frecuencia de muestreo en Hz. No deberíamos olvidar recortar el resultado. Para ello sería bueno el módulo *clipp Expert Macro > Clipping > IClipMinMax*:



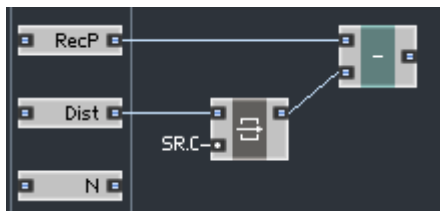
Recortamos a N-1 porque es la distancia máxima entre dos elementos de la serie. Observa que la conversión a enteros se realizará después de la multiplicación.

También podríamos haber recortado el valor de entrada (a un alcance diferente, por supuesto), que normalmente resultaría mejor, puesto que los valores flotantes que están fuera del alcance de la representación de los enteros produce valores enteros arbitrarios, así que nunca realizaremos un auténtico recorte.

Ahora usaremos otra macro para calcular el índice de lectura desde RecordPos y Distance.



Obviamente, la posición de reproducción debe ser *a cierta distancia* de samples por detrás de la posición de grabación, por lo tanto sustraeremos uno de otro:

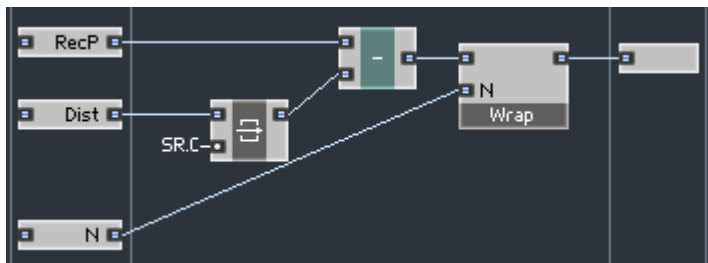


El valor de distancia es bloqueado porque se produce por una entrada de señal de control que potencialmente puede recibir eventos en cualquier momento, y nosotros no queremos que la sustracción tenga lugar en otro momento que no sea en el evento de reloj de audio.

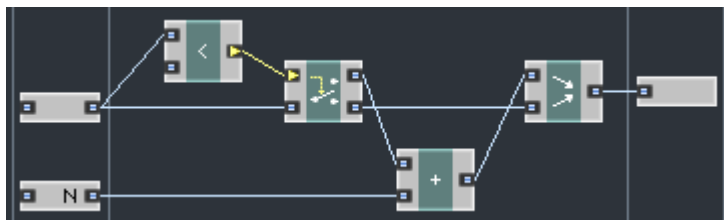
Si sólo sustraemos, es bastante fácil que la diferencia sea menor de cero. Es porque nuestra serie no es un loop, sus “finales” no están conectados entre sí. Necesitamos *ajustar* el resultado:

- 1 debe ser $N-1$,
- 2 debe ser $N-2$,
- 3 debe ser $N-3$, etc.

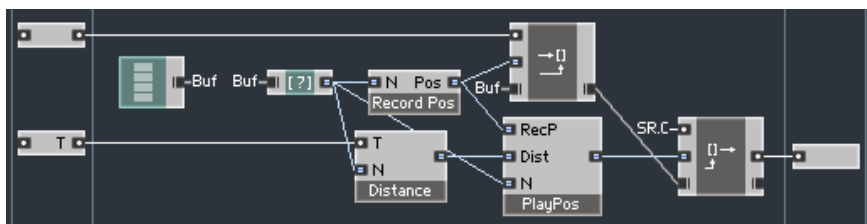
Así, pondremos otra macro para ajustar:



Puesto que sabemos que la diferencia no puede ser más pequeña que $-N+1$ (porque Recordpos está entre 0 y $N-1$ y Distance está entre 0 y $N-1$) el ajuste se puede implementar como adición de N :

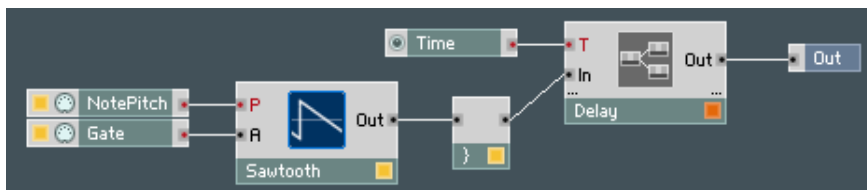


Ahora vamos a regresar a nuestro nivel superior de la estructura. Parece que tenemos nuestros índices de escritura y lectura, así que ya sólo necesitamos escribir y leer:



Observa que la lectura sucede después de la escritura y que está sincronizada por el reloj de frecuencia de muestreo.

Te proponemos una estructura de prueba. No olvides poner un convertidor de milisegundos a segundos dentro de la célula core Delay y ajustarlo al modo monofónico:



De hecho, será buena idea cambiar el delay al modo monofónico cuanto antes, ya que para cada voz podría consumir cerca de 200 K de memoria. 44100 samples, usando 4 bytes (32 bit) cada uno:
 $44100 * 4 = 176400 \text{ bytes} = \text{un poco más de } 172 \text{ K}$
 (un kilobyte tiene 1024 bytes)

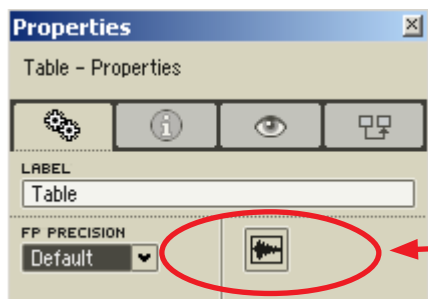
Para probar la estructura de arriba toca notas en tu teclado MIDI y escúchalas con una cantidad de tiempo de retraso especificado por el knob Time.

Tablas

Hay un módulo muy parecido al *Array*. Su nombre es *Table* y podrás encontrarlo en el sub-menú *Built-In Module > Memory > Table*.

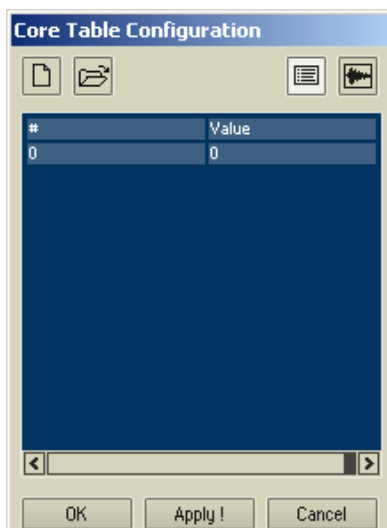



Se parecen porque estas tablas son series en cierto modo. La diferencia es que desde la tabla, no podrás escribir, sólo leer. Los valores en una tabla se pre-inicializan usando las propiedades del módulo. Para acceder a la lista de valores presiona el botón de la ventana de propiedades:

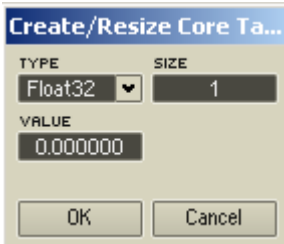


Haz clic aquí
para editar los valores


Tendría que aparecer un nuevo diálogo:



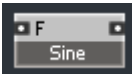
Lo que ves ahora es una tabla vacía. Consiste en un elemento sencillo con valor cero. Ahora podrás empezar a teclear nuevos valores manualmente o importándolos desde un archivo. Si te decides por la opción manual, haz clic sobre el botón . Aparecerá el siguiente diálogo:



Tendrás que seleccionar el tipo de valores almacenados en la tabla, el tamaño de la tabla (número de elementos en la tabla) y un valor para inicializar todos los elementos de la tabla.

Si no, también puedes importar la tabla desde un archivo. El archivo puede ser WAV/AIFF, un archivo de texto (TXT/ASC), o un archivo Native Table File (NTF). Para importar presiona el botón . Aparecerá un diálogo pidiéndote que selecciones un archivo. Luego aparecerá otro diálogo pidiéndote que selecciones el tipo de dato para los valores de la tabla.

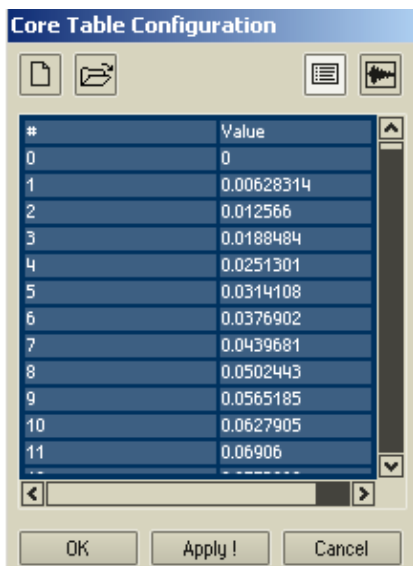
Vamos a usar la tabla. Construiremos una macro oscilador senoidal usando un procedimiento de búsqueda de tabla:





Dentro de esta macro vamos a crear un módulo de tabla:



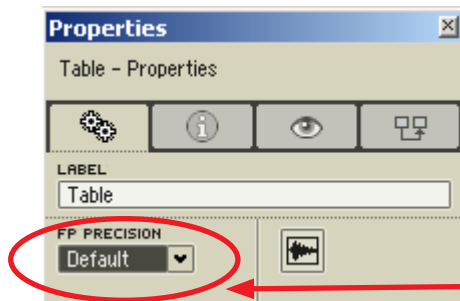
Y lo inicializaremos desde el archivo *sinetable.txt* que hemos preparado para ti en la carpeta “Core Tutorial Examples” dentro de la instalación de Reaktor Core. Se trata de un archivo que contiene valores para un período y un sample de función senoidal. Impórtalo como valores del tipo Float32:



También puedes ver los valores cargados como forma de onda. Los botones  y  cambiarán entre la visualización en forma de lista o de onda, respectivamente.

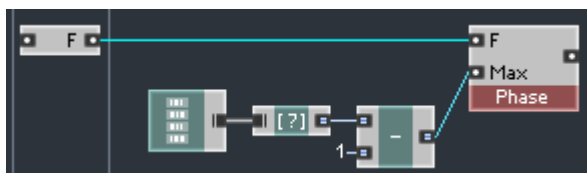
Ahora presiona OK para cerrar el diálogo y traer los valores cargados a la tabla.

También hay un ajuste de la propiedad *FP Precision* en la tabla. En realidad no controla la precisión de los valores en la tabla (esa deberías haberla seleccionado al importar el archivo o al crear manualmente la lista de valores), sino que controla el tipo de precisión “formal” de la salida del módulo de tabla. Debes tenerla en “default”:

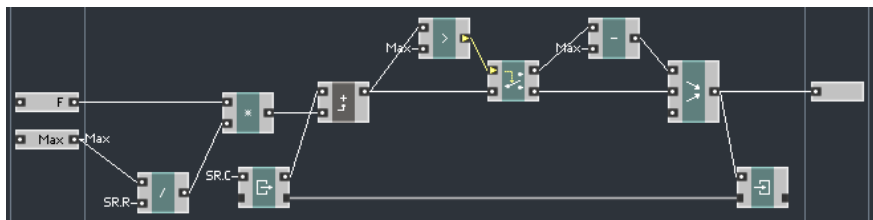


Precisión formal de salida

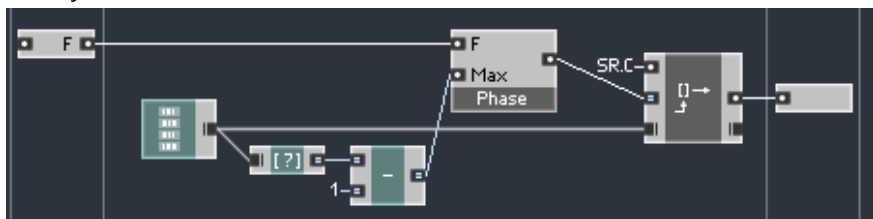
Ahora que ya tenemos la tabla podemos seguir construyendo el oscilador. En su interior vamos a poner un “oscilador de fase”, que genere una señal de diente de sierra de rampa ascendente desde 0 hasta el tamaño de la tabla menos uno:



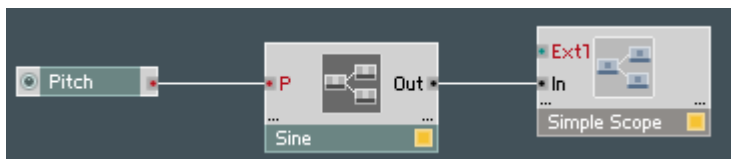
La implementación del oscilador de fase es muy parecida a un diente de sierra o a la posición de grabación en el delay:



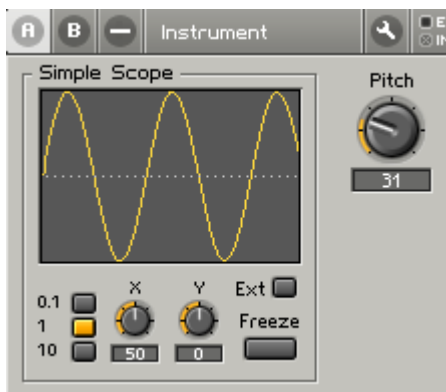
Un módulo *Read []* conectado al oscilador de fase y sincronizado por el reloj de la frecuencia de muestreo, nos permitirá acceder a los elementos de la tabla y transmitir en la salida su valor.



La estructura que te sugerimos es la siguiente (no olvides el convertor P2F en la célula core):



Y este es el panel



Por supuesto, no se trata de un sonido de senoidal muy claro, ya que no hemos puesto interpolación. Si quieres hacer una versión interpolada, lo dejamos en tus manos.

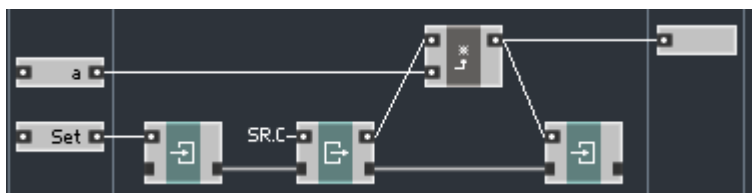
Construir estructuras óptimas

Como regla general, ninguna herramienta es ideal. La tecnología de Reaktor Core no es una excepción. Bien, esto no significa que sea mala, en absoluto. De hecho nosotros creemos que es fabulosa ☺. *No ser ideal* significa que es *real*. También significa que tendrás que saber algunas cosas para conseguir los mejores resultados de esta tecnología. Llámalo “trucos y consejos” o como quieras. A continuación vamos a tratar estos asuntos.

Macros

Usa Latches (cerrojos) y/o macros de modulación en todos los casos para asegurarte de que los eventos esperen hasta que los valores que transmiten tengan que ser procesados.

He aquí una estructura que usa una macro de modulación para la multiplicación en el loop de iteración del audio. Usando la macro de modulación evitaremos que los eventos activen el procesamiento en la entrada “a”:



Hemos encontrado muchos ejemplos de esta técnica anteriormente en el tutorial.

Usar cerrojos siempre implica una optimización de la ejecución y exactitud en tus estructuras. Pueden darse algunos errores típicos en la programación de la estructura como por ejemplo, enviar eventos a ciertos módulos en momentos inadecuados.

No te preocupes de que los cerrojos ralenticen la ejecución de tus estructuras. No son muy pesados y no consumirán capacidad de tu CPU *en absoluto*.

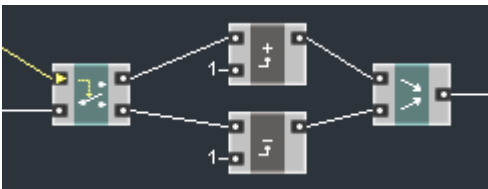
Normalmente, los cerrojos son mejores que el ruteo para filtrar los eventos por su bajo peso. Usa ruteadores sólo cuando la lógica del procesamiento lo dicte.

Rutear y asociar

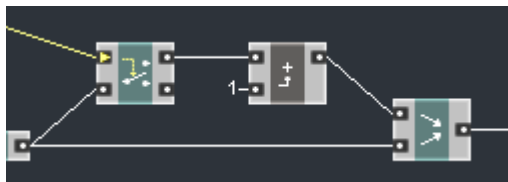
Rutear puede ser más o menos costoso dependiendo de la situación y de la plataforma. Si no puedes rutear sin tener que añadir complicadas operaciones a tu estructura, evítalo.

Algunas veces, el ruteo ES Ctl se puede sustituir por cerrojos. Si puedes, reemplázalo.

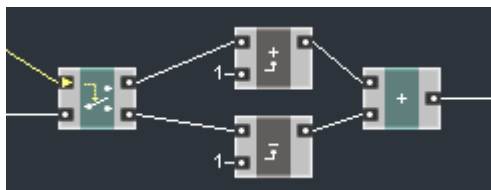
Si divides la ruta del evento en dos ramas usando un Router, es mejor que asocies las ramas generadas en la salida de este Router:



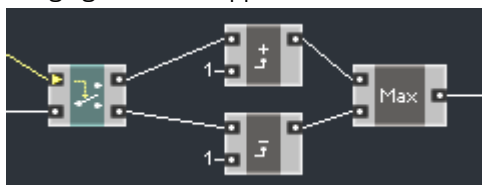
También es bueno asociar el evento entrante (sin dividir) al Router:



No es necesario usar un módulo Merge para asociar. Cualquier módulo de aritmética o similar podría hacer el trabajo:



Merging can also happen inside a macro (depending on its internal structure):



Puede que necesitemos asociar las ramas generadas por diferentes Routers, pero en ese caso hay que tener cuidado con la carga de CPU.

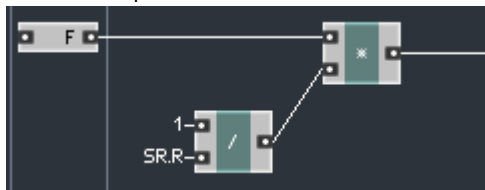
Operaciones numéricas

La adición de dígitos flotantes, multiplicación, sustracción, valores absolutos y negación son, normalmente, las operaciones de flotantes más pesadas. La adición de enteros y negación son las operaciones de enteros menos pesadas. Un valor absoluto entero también está bien, más o menos. *DN Cancel* es, como recordarás, una adición simple.

La división de flotantes y la multiplicación y división de enteros son, de promedio, considerablemente más pesadas.

Es aconsejable agrupar las operaciones de forma que las más pesadas se evalúen lo menos posible. Por ejemplo, si necesitas calcular la frecuencia normalizada dividiendo la frecuencia en Hz por la frecuencia de muestreo, podrías calcular la frecuencia de muestreo recíproca primero y multiplicar la

frecuencia por el resultado:



En la estructura de arriba la división se realizará sólo en caso de que la frecuencia de muestreo cambie, lo que sería bastante raro. Los cambios en la frecuencia deberían activar sólo la multiplicación.

Compara la sencilla implementación de la misma fórmula:

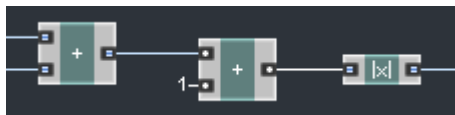


donde la división se ejecutará en respuesta a cada cambio de frecuencia.

Conversión entre flotantes y enteros

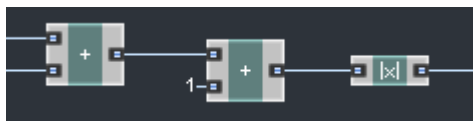
En general, evita conversiones innecesarias entre números flotantes y enteros. Dependiendo de la plataforma, estas conversiones podrían consumir enormes cantidades de CPU. Las conversiones necesarias, por supuesto han de hacerse.

Aunque la siguiente estructura podría funcionar, hay dos conversiones innecesarias entre tipos flotantes y enteros:



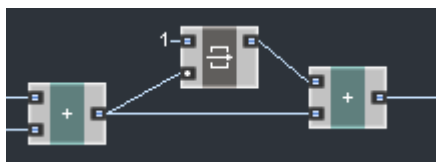
La primera conversión sucede en la entrada del módulo de adición del medio. Este módulo está ajustado al modo flotante, pero recibe una entrada de señal entera. Por lo tanto, se llevará a cabo una conversión de entero a flotante. La segunda ocurre en la entrada del módulo de valor absoluto, que está ajustada al modo entero, pero recibe una entrada flotante. Por lo tanto, ocurrirá una conversión de flotante a entero.

De esta forma sería mucho mejor:



Todos los módulos están ajustados al modo entero, por lo tanto, no hay conversiones.

Las señales de reloj llevan, generalmente, tipos flotantes, pero dará problemas si se usa una señal entera:



Al margen de que la entrada de reloj del ILatch es flotante, está sincronizada por una señal entera. Puesto que el valor del reloj no importa, no se necesita una conversión.

Apéndice A. El interfaz de usuario de Reaktor Core

A.1. Células core

Una célula core se puede crear desde la estructura del nivel primario de Reaktor (excepto la estructura ensemble) haciendo clic con el botón derecho sobre el fondo de la pantalla y seleccionando *Core Cell > New Audio or Core Cell > New Event*.

La librería de células core (del sistema y del usuario) las encontrarás en el menú “Core Cell”. También puedes cargar células core usando el comando *Core Cell > Load...*

Para borrar una célula core selecciona y presiona la tecla *Delete*, o haz clic con el botón derecho sobre la célula core y selecciona el comando *Delete*. También puedes hacer selección múltiple para borrar.

Para guardar una célula core en un archivo, haz clic con el botón derecho sobre la célula core y selecciona el comando *Save Core Cell As...*

Para editar la estructura interna de una célula core haz doble clic sobre la célula core. Para volver atrás haz doble clic sobre el fondo.

Para editar las propiedades exteriores de la célula core, haz clic con el botón derecho sobre la célula y selecciona *Properties*. Si la ventana de propiedades ya está abierta, será suficiente con hacer clic sobre la célula.

Para editar las propiedades internas tendrás que ir a la estructura interna, hacer clic con el botón derecho sobre el fondo y seleccionar *Owner Properties*. Si la ventana de propiedades ya está abierta, será suficiente con hacer clic sobre el fondo.

A.2. Módulos/macros core

Para crear un módulo o una macro core normal, haz clic sobre el área central (la más larga) de la estructura core y selecciona uno de los módulos/macros desde los menús *Built In Module*, *Expert Macro*, *Standard Macro*, o *User Macro*. También puedes cargar un módulo/macro haciendo clic con el botón derecho sobre el fondo de la pantalla y seleccionando el comando *Load Module...*

Se puede crear una macro vacía desde el menú *Built In Module*.

Para guardar un módulo/macro core en un archivo, haz clic con el botón derecho sobre él y selecciona el comando *Save As...*

Para borrar un módulo/macro core, selecciona y presiona la tecla *Delete*, o haz clic con el botón derecho sobre él y selecciona el comando *Delete*. También puedes hacer selección múltiple para borrar.

Para editar la estructura interna de una macro core haz doble clic sobre la macro. Para regresar haz doble clic sobre el fondo.

Para editar las propiedades de un módulo/macro core, tendrás que ir a la estructura interna, hacer clic con el botón derecho sobre el fondo de la pantalla y seleccionar *Owner Properties*. Si la ventana de propiedades ya está abierta, será suficiente con hacer clic sobre el fondo.

También puedes acceder a las propiedades del módulo/macro desde el exterior, haciendo clic con el botón derecho sobre él y seleccionando *Properties*. Si la ventana de propiedades ya está abierta, será suficiente con hacer clic sobre el módulo/macro.

A.3. Puertos Core

Para crear un puerto core haz clic con el botón derecho en el área izquierda (entradas) o derecha (salidas) de la estructura core y selecciona uno desde los distintos tipos disponibles en el submenú *New*.

Para borrar un puerto core, selecciónalo y presiona la tecla *Delete*, o haz clic con el botón derecho sobre él y selecciona el comando *Delete*. También puedes hacer selección múltiple para borrar (incluyendo selecciones mezcladas de módulos/puertos).

A.4. Edición de estructuras core

Para mover un módulo core haz clic sobre él y arrástralo a la posición deseada. Los puertos se pueden arrastrar sólo verticalmente, siendo su orden vertical el que define su apariencia externa.

Para crear una conexión entre la entrada de un módulo y la salida de otro, haz clic sobre uno de ellos y arrastra hasta el otro.



Para eliminar una conexión haz clic en el cable de conexión para seleccionarlo y presiona la tecla *Delete*. También puedes arrastrar desde la entrada hasta el fondo de la estructura.

Para crear un QuickConst, haz clic con el botón derecho en la entrada de un módulo y selecciona *Connect to New QuickConst*. Para acceder a las propiedades del QuickConst, haz clic sobre él.

Para crear un QuickBus haz clic con el botón derecho en la entrada o salida de un módulo y selecciona *Connect to New QuickBus*. Para conectar la entrada o salida de un módulo a un QuickBus existente, haz clic con el botón derecho en esta entrada o salida y selecciona uno de los buses disponibles en el menú *Connect to QuickBus*.

Apéndice B. Conceptos de Reaktor Core

B.1. Señales y eventos

Hay señales de tipo flotante o entero, Los puertos flotantes son así:  y los enteros son así: .

Las señales propagan las conexiones desde las salidas hasta las entradas en forma de eventos. Un evento es una acción básica que ocurre en una salida particular, dando como resultado un cambio en el valor de la salida (como caso particular, también puede darse que el resultado se quede igual).

Todos los eventos que se originan desde la misma fuente de evento se consideran “simultáneos”. “Simultáneo” significa que si dos de estos eventos llegan a distintas entradas del mismo módulo, lo harán a la vez.

“La misma fuente” quiere decir la misma salida. En ciertas circunstancias, varias salidas pueden considerarse “la misma fuente de evento”. Por ejemplo, todas las entradas de audio y conexiones estándar de reloj de frecuencia de muestreo, se consideran la misma fuente de evento. Durante la inicialización, todas las salidas que envíen eventos se consideran la misma fuente de evento. “La misma fuente” no significa “el mismo valor” en este contexto, sino “simultaneidad”.

A menos que un módulo en concreto sea una fuente de evento, lo único que puede activarlo para procesar los valores entrantes, es uno o varios eventos que llegan a sus entradas. En caso de que haya múltiples eventos, sólo se generará un evento de salida, puesto que los eventos de entrada llegan a la vez.

B.2. Inicialización

La inicialización de las estructuras se realiza así. Primero, todos los valores se reajustan a cero. Luego, el evento de inicialización se envía simultáneamente desde todas las fuentes de inicialización, que generalmente son constantes, entradas de células core (no siempre) y relojes.

B.3. Conexiones OBC

OBC (Object Bus Connections) son conexiones entre módulos que no envían ninguna señal, pero proclaman que los módulos comparten una categoría común (memoria). El ejemplo más típico de este tipo de conexión es entre módulos *Write* y *Read*, accediendo al mismo valor almacenado.

B.4. Routing

Puedes usar módulos *Router* para dirigir la corriente de eventos entre dos rutas posibles. En caso de que el *Router* elija una ruta de salida para los eventos entrantes, la otra salida no recibirá eventos (esto significa que el valor de esta otra salida no puede cambiar).

El *Router* se controla a través de una conexión de entrada del tipo *BoolCtl*. Al otro lado de esta conexión, normalmente pondrías un módulo *Compare* (algunas veces puedes poner módulos intermediarios entre el *Router* y el *Compare*, como por ejemplo, puertos macro *BoolCtl*)

Con los *Routers* podrás habilitar o deshabilitar dinámicamente evaluaciones en partes de la estructura.

Normalmente, si usas un *Router* para dividir una ruta de señal en dos, asociarías las dos ramas con un módulo *Merge* u otro similar. Lo normal sería asociar una de las ramas a la señal original sin dividir. Aunque eres libre de hacer lo que quieras, siempre que tengas cuidado con los problemas de ejecución.

B.5. El cerrojo

El uso de cerrojos es, probablemente, la técnica más común en Reaktor Core, ya que al usar los módulos *Latch* estamos evitando que los eventos se envíen en un momento erróneo. Por ejemplo, no querrás que un evento de señal de control active el cálculo en el loop de audio.

También puedes usar macros del grupo *Expert Macros > Modulation*, que son un grupo de combinaciones típicas de cerrojos con módulos aritméticos de procesamiento.

B.6. Clocking

Clocks are sources of events. The clock events typically occur at regular time intervals corresponding to the clock rate. You normally need clocks to drive various modules, such as oscillators, filters, and so on. Most of the implementations of such modules do not require an explicit clock connection from the outside, but implicitly use a standard clock source available in core structures. That source is the sample rate clock, which is running at the default audio rate. Note, that in event core cells, although the connection to the sample rate clock is available, the clock signal itself is not available, therefore most oscillators, filters, and similar modules will not run in event core cells.

Apéndice C. Puertos macro core

C.1. Entrada



Acepta un evento entrante desde el exterior y lo reenvía sin modificar a su propia salida interna. La conexión de entrada interna se puede usar para desestimar el propósito por defecto (desconectado) de este puerto.

C.2. Salida



Acepta un evento entrante en la entrada interna y lo reenvía sin modificar al exterior.

C.3. Cerrojo (entrada)



Envía una conexión OBC desde el exterior de la macro al interior de la macro. La conexión de entrada del interior se puede usar para desestimar el propósito por defecto (desconectado) de este puerto.

C.4. Cerrojo (salida)



Envía una conexión OBC desde el interior de la macro al exterior de la macro.

C.5. Bool C (entrada)



Envía una conexión BoolCtl desde el exterior de la macro al interior de la macro. La conexión de entrada del interior se puede usar para desestimar el propósito por defecto (desconectado) de este puerto.

C.6. Bool C (salida)



Envía una conexión BoolCtl desde el interior de la macro al exterior de la macro.

Apéndice D. Puertos de células core

D.1. Entrada (modo audio)



Proporciona acceso a la señal de audio desde el exterior del módulo. Envía eventos regulares a la frecuencia de muestreo (sincronizado al reloj de frecuencia de muestreo global SR.C).

EVENTO DE INICIALIZACIÓN: envía un evento de inicialización. El valor se define por la inicialización del exterior.

D.2. Salida (modo audio)



Hace que los valores recibidos en la entrada interna estén disponibles en el exterior del módulo. En un momento dado, el último valor recibido se entregará al exterior.

D.3. Entrada (modo evento)



Convierte los eventos del nivel primario de Reaktor que llegan desde el exterior en eventos de Reaktor Core y los reenvía al interior.

EVENTO DE INICIALIZACIÓN: envía un evento de inicialización si en el exterior se recibe un evento de inicialización.

D.4. Salida (modo evento)



Convierte los eventos de Reaktor Core que llegan desde el interior en eventos del nivel primario de Reaktor y los reenvía al exterior. Si hay varias salidas en modo evento que reciben simultáneamente eventos de Reaktor Core, los eventos correspondientes del nivel primario de Reaktor se enviarán en orden desde las salidas superiores a las inferiores.

Apéndice E. Buses incorporados

E.1. SR.C

Envía eventos de reloj regulares a la frecuencia de muestreo

EVENTO DE INICIALIZACIÓN: siempre envía un evento de inicialización

E.2. SR.R

Proporciona la actual frecuencia de muestreo en Hz. Envía eventos con nuevos valores en respuesta a los cambios de frecuencia de muestreo.

EVENTO DE INICIALIZACIÓN: siempre envía un evento de inicialización con frecuencia de muestreo

Apéndice F. Módulos internos

F.1. Const



Produce una señal con un valor constante. El valor aparece en el módulo.

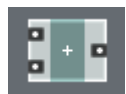
EVENTO DE INICIALIZACIÓN: durante la inicialización, envía el evento del valor especificado a la salida. Es el único momento en que este módulo envía un evento.

PROPIEDADES:

Valor

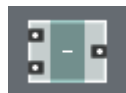
el valor para ser enviado a la salida

F.2. Math > +



Produce la suma de las señales entrantes en la salida. El evento de salida se envía cada vez que haya un evento en cualquiera de las entradas o en ambas simultáneamente.

F.3. Math > -



Produce la resta o diferencia de las señales entrantes en la salida (la señal de la entrada inferior es sustraída de la señal superior). El evento de salida se envía cada vez que haya un evento en cualquiera de las entradas o en ambas simultáneamente.

F.4. Math > *



Genera el producto de la multiplicación de las señales en su salida. El evento de salida se envía cada vez que hay un evento en cualquiera de las entradas o en ambas simultáneamente.

F.5. Math > /



Genera el cociente de las señales entrantes en su salida (la señal de arriba se divide por la señal de abajo). El modo entero realiza una división con remanente, siendo éste descartado. El evento de salida se envía cada vez que haya un evento en cualquiera de las entradas o en ambas simultáneamente.

F.6. Math > |x|



Produce un valor absoluto de la señal entrante en su salida. El evento de salida se envía cada vez que haya un evento en su entrada.

F.7. Math > -x



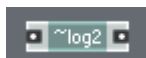
Produce el valor invertido (cambio de signo) de la señal entrante en su salida. El evento de salida se enviará cada vez que haya un evento en su entrada.

F.8. Math > DN Cancel



Modifica la señal entrante en cuanto que elimina los números anormales. Habitualmente se implementa añadiendo un constante muy pequeño. Funciona sólo con números de dígitos flotantes. El evento de salida se envía cada vez que haya un evento en su entrada.

F.9. Math > ~log

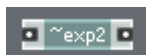


Calcula una aproximación de un algoritmo del valor entrante. El evento de salida se envía cada vez que haya un evento en su entrada.

PROPIEDADES:

Base	base del algoritmo
Precisión	precisión aproximada (mejores precisiones requieren más CPU)

F.10. Math > ~exp



Calcula una aproximación de un exponente del valor entrante. El evento de salida se envía cada vez que haya un evento en su entrada.

PROPIEDADES:

Base	base del exponente
Precisión	precisión aproximada (mejores precisiones requieren más CPU)

F.11. Bit > Bit AND



Realiza la conjunción con precisión de bit de las señales entrantes. Sólo funciona con enteros. El evento de salida se envía cada vez que haya un evento en cualquiera de sus entradas o en ambas simultáneamente.

F.12. Bit > Bit OR



Realiza la disyunción con precisión de bit de las señales entrantes. Sólo funciona con enteros. El evento de salida se envía cada vez que haya un evento en cualquiera de las entradas o en ambas simultáneamente.

F.13. Bit > Bit XOR



Realiza la disyunción exclusiva con precisión de bit de las señales entrantes. Sólo funciona con enteros. El evento de salida se envía cada vez que haya un evento en cualquiera de las entradas o en ambas simultáneamente.

F.14. Bit > Bit NOT



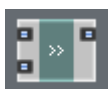
Realiza la inversión con precisión de bit de la señal entrante. Sólo funciona en enteros. El evento de salida se envía cada vez que haya un evento en su entrada.

F.15. Bit > Bit <<



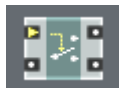
Convierte los bits del valor de la entrada superior de la izquierda (a bits menos significativos). La cantidad de bits a la que hay que convertirlo se especifica a través de la entrada inferior. El resultado para $N < 0$ y $N > 31$ es indefinido (significa que has de usarla sólo para $0 \leq N \leq 31$). Sólo funciona con enteros. El evento de salida se envía cada vez que haya un evento en cualquiera de las entradas o en ambas simultáneamente.

F.16. Bit > Bit >>



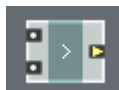
Convierte los bits del valor de la entrada superior de la derecha (a bits menos significativos). No se realiza extensión de signo. La cantidad de bits a la que hay que convertirlo se especifica a través de la entrada inferior. El resultado para $N < 0$ y $N > 31$ es indefinido (significa que has de usarla sólo para $0 \leq N \leq 31$). Sólo funciona con enteros. El evento de salida se envía cada vez que haya un evento en cualquiera de las entradas o en ambas simultáneamente.

F.17. Flow > Router



Rutea la señal de entrada (la de debajo) a una de las dos salidas dependiendo de la categoría de la señal de control (la entrada de arriba). Si la señal de control es verdadera, rutea a la salida 1 (la de arriba) y si es falsa, rutea a la salida 0 (la de debajo). El evento de salida se envía a una de las salidas cada vez que haya un evento en la entrada de señal.

F.18. Flow > Compare

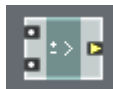


Produce una señal BoolCtl en la salida que indica el resultado de la comparación de los valores de entrada. El valor de la entrada de arriba se coloca a la izquierda del signo de comparación y el valor de la entrada de abajo a la derecha (para que el módulo del dibujo de arriba compruebe si el valor de arriba es mayor que el de abajo).

PROPIEDADES:

Criterio la operación de comparación que ha de realizarse

F.19. Flow > Compare Sign



Produce una señal BoolCtl en la salida que indica el resultado de la comparación de signos de los valores de entrada. El valor de la entrada de arriba se coloca a la izquierda del signo de comparación, y el valor de la entrada de abajo a la derecha (para que el módulo del dibujo de arriba compruebe si el signo del valor de arriba es mayor que el de abajo).

La comparación de signos se define así:

- + es igual a +
- es igual a –
- + es mayor que –

El signo de valor cero es indefinido, así que, si uno de los valores comparados es cero, se pueden producir resultados arbitrarios.

PROPIEDADES:

Criterio la operación de comparación que ha de realizarse

F.20. Flow > ES Ctl



Produce una señal BoolCtl en la salida que indica la presencia momentánea de un evento en la entrada (es decir, la señal de control es verdadera si hay un evento en la entrada de este módulo en un momento dado).

F.21. Flow > ~BoolCtl



Produce una señal BoolCtl en la salida que es una inversión de la señal de entrada BoolCtl (verdadero cambia falso y viceversa).

F.22. Flow > Merge

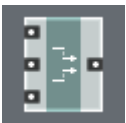


Envía un evento de salida cada vez que hay un evento en cualquiera de sus entradas o en varias de ellas simultáneamente. Si sólo una entrada recibe un evento en un momento, el valor del evento de salida será igual al valor de este evento de entrada. Si varias entradas reciben un evento simultáneamente, se seleccionará el valor de la entrada de debajo (entra las que reciben el evento en ese momento). Así, si el evento se recibe en la segunda y la tercera (desde arriba), se tomará el valor de la tercera.

PROPIEDADES:

Cuenta de entrada número de entradas del módulo

F.23. Flow > EvtMerge



La funcionalidad es similar a la del módulo Merge, excepto que todos los valores de entrada se ignoran. El valor del evento de salida es indefinido. Este módulo sirve para generar señales que vayan a usarse como reloj. Sólo funciona en modo de punto flotante, puesto que se supone que el valor no va a usarse en ningún modo.

PROPIEDADES:

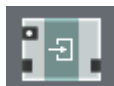
Cuenta de entrada número de entradas del módulo

F.24. Memory > Read



Lee el valor almacenado de la memoria asociada con la cadena OBC a la que pertenece este módulo. La lectura sucede en respuesta a un evento en la entrada de arriba (reloj) y se envía a la salida de arriba. Los puertos de abajo son las conexiones OBC maestra y esclava respectivamente.

F.25. Memory > Write



Escribe el valor que llega a la entrada de arriba en la memoria asociada con la cadena OBC a la que pertenece este módulo. La escritura ocurre en respuesta a un evento en la entrada de arriba. Los puertos de abajo son las conexiones OBC maestra y esclava respectivamente.

F.26. Memory > R/W Order



Este módulo no realiza ninguna acción. Se puede insertar dentro de una estructura para controlar el orden de procesamiento de módulos conectados por OBC. Los puertos OBC de abajo son las conexiones maestra y esclava, que están conectadas internamente mediante “thru”. La entrada OBC de arriba es la conexión “sidechain”, que permite colocar este módulo después del módulo conectado a la entrada sidechain.

La conexión sidechain sólo se puede conectar a módulos del tipo OBC Latch normales. Por otro lado, los puertos maestro y esclavo se pueden conectar a módulos del tipo Latch o Array OBC, dependiendo de los ajustes de las propiedades del módulo R/W Order. En cualquier caso, el tipo de señal y la precisión han de ser las mismas para todas las conexiones de este módulo (por ejemplo, no puedes conectar por sidechain a un Read entero y el maestro y el esclavo a módulos flotantes a la vez).

PROPIEDADES:

Tipo de Conexión de puerto de conexión “Thru” (Latch o Array)

F.27. Memory > Array



Define un objeto de memoria en serie. El módulo por sí mismo no realiza ninguna acción. Todas las operaciones de serie han de ser realizadas por los módulos conectados a la salida de la serie, que es una conexión esclava del tipo serie.

PROPIEDADES:

Tamaño de elementos de la serie

F.28. Memory > Size []



Comunica el tamaño del objeto de serie conectado a la entrada. El tamaño es un valor constante entero.

EVENTO DE INICIALIZACIÓN: durante la inicialización envía el evento con el valor del tamaño de la serie a la salida. Es el único momento en el que este módulo envía un evento.

F.29. Memory > Index



Proporciona acceso a un elemento de serie sencillo. El acceso se proporciona en forma de conexión OBC latch asociado con el elemento de la serie. La asociación se establece y/o cambia enviando un evento a la entrada de arriba (índice) del módulo Index, que está *basado en cero* y siempre está en el modo entero. La entrada de abajo es la conexión OBC maestra de la serie. La salida proporciona la conexión latch OBC al elemento de serie seleccionado por la entrada de índice. La precisión y el tipo de valor base son, por supuesto, los mismos para ambas conexiones OBC (entrada y salida) y se controlan en las propiedades del módulo.

F.30. Memory > Table



Define una serie sólo de lectura preinicializada. El módulo por sí mismo no realiza ninguna acción. Todas las operaciones de la tabla han de ser realizadas por los módulos conectados a la salida de la tabla, que es una conexión OBC esclava del tipo serie.

PROPIEDADES:



Precisión FP

edita los valores en la tabla

la precisión formal de la conexión de salida

F.31. Macro



Proporciona un contenedor para una estructura interna. El número de entradas y salidas no es fijo y viene definido por la estructura interna.

PROPIEDADES:

Precisión FP

controla la precisión formal de la conexión de salida

Apariencia

cambia entre *Large* (los nombres de la etiqueta y el puerto son visibles) y *Small* (no son visibles)



Alineación de conexiones [pin]

controla la alineación de los puertos en el aspecto externo de la macro

Solidez

controla el tratamiento de la macro en el motor de core. Si está desconectada, los límites de la macro son transparentes para resolución de feedback y otras cosas. Déjala en ON a menos que *realmente*, realmente sepas lo que estás haciendo.

Icono

el botón  carga un nuevo icono para la macro, y el botón  elimina el icono (no hay icono asignado).

Apéndice G. Macros expertas

G.1. Clipping > Clip Max / IClip Max



La señal de la entrada superior está recortada desde arriba por el valor de threshold de la entrada de abajo. Los cambios en el threshold no generan eventos.

G.2. Clipping > Clip Min / IClip Min



La señal de la entrada superior está recortada desde abajo por el valor de threshold de la entrada inferior. Los cambios en el threshold no generan eventos.

G.3. Clipping > Clip MinMax / IClipMinMax



La señal en la entrada de arriba está recortada desde abajo por el valor de threshold de la entrada del medio y por arriba por el valor de threshold de la entrada inferior. Los cambios en el threshold no generan eventos.

G.4. Math > 1 div x



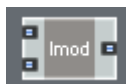
Calcula lo recíproco del valor de entrada

G.5. Math > 1 wrap



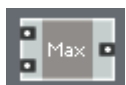
Acota la señal entrante dentro del campo [-0.5..0.5] (el período de acotación es 1).

G.6. Math > Imod



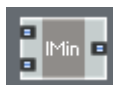
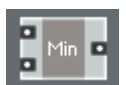
Calcula el resto de la división del valor superior por el valor inferior. El evento de salida se envía cada vez que haya un evento en cualquiera de sus entradas o en ambas simultáneamente.

G.7. Math > Max / IMax



Calcula el máximo de los valores de entrada. El evento de salida se envía cada vez que haya un evento en cualquiera de sus entradas o en ambas simultáneamente.

G.8. Math > Min / IMin



Calcula el mínimo de los valores de entrada. El evento de salida se envía cada vez que haya un evento en cualquiera de sus entradas o en ambas simultáneamente.

G.9. Math > round



Redondea el valor entrante al entero más cercano. El resultado al redondear valores que están exactamente en la mitad de dos enteros no está definido. Por ejemplo, 1.5 podría redondearse tanto a 1 como a 2.

G.10. Math > sign +/-



Saca tanto 1 como -1 dependiendo del signo de la entrada (números positivos producen 1, negativos -1, el cero nunca se saca).

G.11. Math > sqrt (>0)



Aproximación a la raíz cuadrada. Funciona sólo para entradas mayores que 0.

G.12. Math > sqrt



Aproximación a la raíz cuadrada.

G.13. Math > x(>0)^y



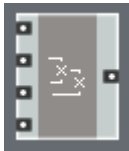
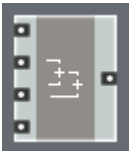
Un aproximación de x^y . x ha de ser >0 . El evento de salida se envía cada vez que haya un evento en cualquiera de sus entradas o en ambas simultáneamente.

G.14. Math > x^2 / x^3 / x^4



Calcula la $2^a, 3^a, 4^a$ potencia de x .

G.15. Math > Chain Add / Chain Mult



Añade/multiplica las señales juntas en un orden de arriba abajo. El evento de salida se generará si hay uno o más eventos en cualquiera de sus entradas.

G.16. Math > Trig-Hyp > 2 pi wrap



Acota el valor entrante dentro del campo $[-\pi.. \pi]$ (el período de acotación es 2π).

G.17. Math > Trig-Hyp > arcsin / arccos / arctan



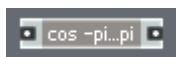
Aproximación al arcoseno/arccoseno/arcotangente

G.18. Math > Trig-Hyp > sin / cos / tan



Aproximación al seno, coseno, tangente.

G.19. Math > Trig-Hyp > sin $-\pi..pi$ / cos $-\pi..pi$ / tan $-\pi..pi$



Aproximación al seno, coseno, tangente (sólo funciona en el alcance $[-\pi..pi]$).

G.20. Math > Trig-Hyp > tan $-\pi/4..pi/4$



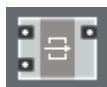
Aproximación a la tangente (sólo funciona en el alcance $[-\pi/4..pi/4]$).

G.21. Math > Trig-Hyp > sinh / cosh / tanh



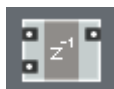
Aproximación hiperbólica al seno/coseno/tangente/.

G.22. Memory > Latch / lLatch



Bloquea, corta el paso a la señal de la entrada superior hasta que llega un evento de reloj a la entrada inferior. Si ambos eventos llegan simultáneamente, la señal entrante podrá pasar inmediatamente.

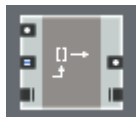
G.23. Memory > z^{-1} / $z^{-1} \text{ ndc}$



Envía el último valor que se ha recibido en la entrada superior antes de que llegue el evento de reloj a la entrada inferior en respuesta al evento de reloj. Si la entrada de reloj está desconectada, el módulo usará el reloj de audio estándar (SR.C) y funcionará como un retardo de un sample.

Ambos módulos pueden resolver automáticamente loops de feedback, no obstante, sólo la versión z^{-1} proporciona cancelación anormal. La versión $z^{-1} ndc$ se usa sólo en los lugares en que no se espere que haya anormales.

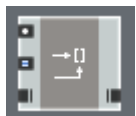
G.24. Memory > Read []



Lee un valor desde una serie a un índice dado (especificado por la entrada del medio) en respuesta a un evento de reloj entrante (en la entrada superior). La entrada inferior (OBC) es la conexión de serie.

Usa la salida OBC del módulo para crear cadenas OBC y serializar así las operaciones de acceso a series.

G.25. Memory > Write []



Escribe un valor (recibido por la entrada superior) en una serie a un índice dado (especificado por la entrada del medio). La operación de escritura se activa por un valor entrante. La entrada inferior (OBC) es la conexión de serie.

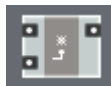
Usa la salida OBC del módulo para crear cadenas OBC y serializar así las operaciones de acceso a series.

G.26. Modulation > $x + a$ / Integer > $lx + a$



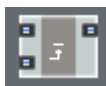
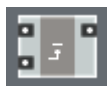
Añade un parámetro (entrada inferior) a la señal (entrada superior) en respuesta a un evento de la señal entrante. Los cambios del parámetro no generan eventos.

G.27. Modulation > $x * a$ / Integer > $lx * a$



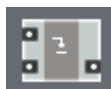
Multiplca la señal (entrada superior) por un parámetro (entrada inferior) en respuesta a un evento de la señal entrante. Los cambios de parámetro no generan eventos.

G.28. Modulation > $x - a$ / Integer > $lx - a$



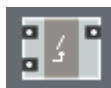
Sustrae un parámetro (entrada inferior) de la señal (entrada superior) en respuesta a un evento de la señal entrante. Los cambios de parámetro no generan eventos.

G.29. Modulation > $a - x$ / Integer > $la - x$



Sustrae la señal (entrada inferior) de un parámetro (entrada superior) en respuesta a un evento de la señal entrante. Los cambios de parámetro no generan eventos.

G.30. Modulation > x / a



Divide la señal (entrada superior) por un parámetro (entrada inferior) en respuesta a un evento de la señal entrante. Los cambios de parámetro no generan eventos.

G.31. Modulation > a / x



Divide un parámetro (entrada superior) por la señal (entrada inferior) en respuesta a un evento de la señal entrante. Los cambios de parámetro no generan eventos.

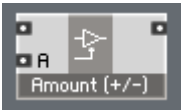
G.32. Modulation > xa + y



Multiplica la señal de la entrada superior por el parámetro de ganancia (entrada del medio) y suma el resultado a la señal de la entrada inferior. Los eventos en cualquiera o ambas de las entradas generan un nuevo valor de salida; los eventos en la entrada del parámetro, no.

Apéndice H. Macros estándar

H.1. Audio Mix-Amp > Amount



Proporciona un control lineal invertible de la cantidad (amplitud) de una señal de audio.

- A = 0 mutea la señal
- A = 1 deja la señal intacta
- A = -1 invierte la señal

Uso típico: controlar la cantidad de feedback de audio

H.2. Audio Mix-Amp > Amp Mod



Modula la amplitud de la señal de audio por una cantidad dada (AM) en la escala lineal.

- AM = 1 dobla la amplitud
- AM = 0 no cambia
- AM = -1 mutea la señal

Uso típico: tremolo, AM.

H.3. Audio Mix-Amp > Audio Mix



Mezcla dos señales de audio juntas.

H.4. Audio Mix-Amp > Audio Relay



Alterna entre dos señales de audio de entrada. Si "x" es mayor que 0, cogerá la señal 1, si no, la señal 0.

H.5. Audio Mix-Amp > Chain (amount)

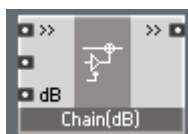


Cambia la amplitud de la señal de audio por una cantidad lineal dada (A) y la mezcla en la señal de audio encadenada[chained] (>>).

- A = 0 la señal se mutea
- A = 1 la señal no cambia
- A = -1 la señal se invierte

Uso típico: cadenas de mezcla de audio, control de cantidad de feedback de audio

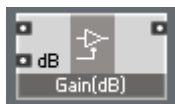
H.6. Audio Mix-Amp > Chain (dB)



Cambia la amplitud de una señal de audio por una cantidad dada de dB y la mezcla en la señal de audio encadenada (>>).

Uso típico: cadenas de mezcla de audio

H.7. Audio Mix-Amp > Gain (dB)



Cambia la amplitud de una señal de audio por una cantidad dada en dB.

- +6 dB dobla la amplitud
- 0 dB no cambia
- 6 dB divide por la mitad la amplitud

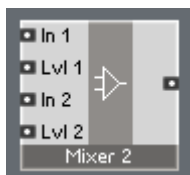
Uso típico: control de volumen de la señal en la escala

H.8. Audio Mix-Amp > Invert



Invierte la polaridad de una señal de audio

H.9. Audio Mix-Amp > Mixer 2 ... 4



Mezcla las señales de audio entrantes (In 1, In 2, ...) atenuando sus niveles por cantidades especificadas en dB (Lvl1, Lvl2, ...).

H.10. Audio Mix-Amp > Pan



Panoramiza la señal de audio entrante usando curva parabólica.

- 1 izquierda
- 0 centro
- 1 derecha

H.11. Audio Mix-Amp > Ring-Amp Mod



La señal de audio portadora (en la entrada de arriba) se modula por la señal de audio de la entrada 'Mod'. El tipo de modulación se controla por la entrada 'R/A', que suavemente varía entre la modulación de anillo y de amplitud.

R/A = 0 modulación de anillo

R/A = 1 modulación de amplitud

(para una auténtica modulación de amplitud, la amplitud del modulador no debería exceder de 1).

H.12. Audio Mix-Amp > Stereo Amp



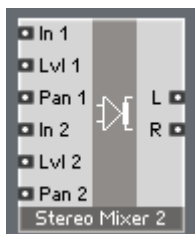
Amplifica una señal de audio monofónica por una cantidad dada en dB y la panoramiza a una posición determinada. La posición de panoramización se define así:

-1 izquierda

0 centro

1 derecha

H.13. Audio Mix-Amp > Stereo Mixer 2 ... 4



Mezcla las señales de audio entrantes (In 1, In 2, ...) atenuando sus niveles por una cantidad dada en dB (Lvl1, Lvl2, ...) y las panoramiza a posiciones determinadas (Pan1, Pan2, ...). Las posiciones de panormización se definen así:

- 1 izquierda
- 0 centro
- 1 derecha

H.14. Audio Mix-Amp > VCA



Amplificador de audio con control lineal directo para la amplitud.

- A = 0 mutea la señal
- A = 1 deja la señal sin cambios

Uso típico: conectar la envolvente de amplitud a la entrada A.

Nota: para una amplificación invertible usa el módulo Audio Amount.

H.15. Audio Mix-Amp > XFade (lin)

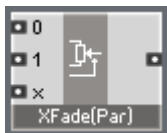


Crossfade de audio con curva lineal.

- x = 0 sólo se oye la señal 0
- x = 0.5 mezcla igual para ambas señales
- x = 1 sólo se oye la señal 1

Nota: un crossfade parabólico normalmente proporciona mejores resultados sonoros.

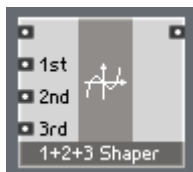
H.16. Audio Mix-Amp > XFade (par)



fundido cruzado de audio con curva parabólica. Normalmente proporciona mejores resultados sonoros que el fundido cruzado lineal.

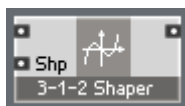
- x = 0 sólo se oye la señal 0
- x = 0.5 mezcla igual para la dos señales
- x = 1 sólo se oye la señal 1

H.17. Audio Shaper > 1+2+3 Shaper



Proporciona modelado controlable de la señal de audio de 2º y 3º orden. La entrada '1st' especifica la cantidad de la señal original en la salida (1=sin cambio, 0=nada). Las entradas '2nd' y '3rd' especifican las cantidades de distorsión de orden 2º y 3º, respectivamente.

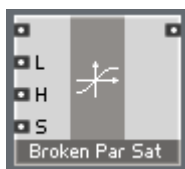
H.18. Audio Shaper > 3-1-2 Shaper



Modelador de señales de audio con cantidad variable de distorsión de orden 2º y 3º. LA cantidad de distorsión y tipo se controla en la entrada "Shp":

- Shp = 0 no hay modelado
- Shp > 0 modelado de orden 3º
- Shp < 0 modelado de orden 2º

H.19. Audio Shaper > Broken Par Sat

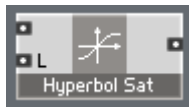


Saturador parabólico roto. Tiene un segmento lineal alrededor del nivel 0. La entrada "L" especifica el nivel de salida de la "saturación total" (por defecto = 1).

La entrada "H" especifica la dureza (alcance 0...1). Los valores mayores se corresponden con los segmentos lineales más largos del medio.

La entrada "S" controla la simetría de la curva de modelado (alcance -1...1). En 0 a curva es simétrica.

H.20. Audio Shaper > Hyperbol Sat



Saturador hiperbólico simple. La entrada “L” especifica el nivel de salida de “saturación total” (por defecto = 1). No obstante, la “saturación total” nunca se consigue con este tipo de saturador.

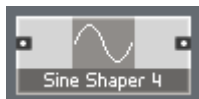
H.21. Audio Shaper > Parabol Sat



Saturador parabólico simple. La entrada “L” especifica el nivel de salida de “saturación total” (por defecto = 1).

Nota: la saturación total se consigue con el nivel de entrada igual a 2L.

H.22. Audio Shaper > Sine Shaper 4 / 8



Modelador de senoidales de 4º y 8º orden. El modelador de 8º orden tiene una mejor aproximación a la senoidal, pero consume más CPU.

H.23. Control > Ctl Amount



Control lineal invertible de la cantidad (amplitud) de la señal de control.

- A = 0 cierra la señal
- A = 1 deja la señal intacta
- A = -1 invierte la señal

Uso típico: controlar la cantidad de modulación

H.24. Control > Ctl Amp Mod



Modula la amplitud de la señal de control por una cantidad dada (AM) en escala lineal.

AM = 1	dobla la amplitud
AM = 0	no cambia
AM = -1	mutea la señal

H.25. Control > Ctl Bi2Uni



Cambia una señal bipolar $-1...1$ en una unipolar. La entrada “a” controla la cantidad de cambio. A 0 no hay cambio, a 1 hay un 100% de cambio (por defecto es 1)

Uso típico: conectar inmediatamente después de un LFO para ajustar la polaridad de la modulación.

H.26. Control > Ctl Chain



Cambia la amplitud de la señal de control por una cantidad lineal dada (A) y la mezcla en la señal de control encadenada (>>).

A = 0	la señal se cierra
A = 1	la señal no cambia
A = -1	la señal se invierte

Uso típico: cadenas de control de mezcla

H.27. Control > Ctl Invert



Invierte la polaridad de la señal de control

H.28. Control > Ctl Mix



Mezcla dos señales de control

H.29. Control > Ctl Mixer 2



Mezcla dos señales de control (In 1, In2) juntas usando los factores de ganancia específicos (A 1, A 2).

A = 0 no hay señal
A = 1 sin cambio
A = -1 invertida

H.30. Control > Ctl Pan



“panoramiza” una señal de control usando curva parabólica.

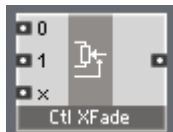
Pos = -1 izquierda
Pos = 0 centro
Pos = 1 derecha

H.31. Control > Ctl Relay



Alterna entre dos señales de control. Si $x > 0$ recoge la señal 1, si no, la señal 0.

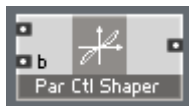
H.32. Control > Ctl XFade



Realiza crossfade entre dos señales de control usando curva lineal.

- $x = 0$ solo pasa la señal 0
- $x = 0.5$ misma mezcla para ambas señales
- $x = 1$ sólo pasa la señal 1

H.33. Control > Par Ctl Shaper



Aplica una doble curva parabólica a una señal controladora. La señal de entrada debe estar en el alcance $-1..0..1$. La señal de salida también tendrá el alcance $-1..0..1$. El grado de desviación se controla en la entrada "b" (el alcance también es $-1..0..1$).

- $b = 0$ no hay desviación (curva lineal)
- $b = -1$ desviación máxima "hacia" el eje X
- $b = 1$ desviación máxima "hacia" el eje Y

También puedes usar este modelador para señales cuyo alcance sea $0..1$, en cuyo caso sólo se usará la mitad de la curva.

Uso típico: velocidad y otros modelados de control

H.34. Convert > dB2AF



Convierte una señal de control desde la escala dB a un factor de ganancia de amplitud lineal.

- 0 dB → 1.0
- 6 dB → 0.5
- etc.

H.35. Convert > dP2FF



Convierte una señal de control de un intervalo en escala de tono (diferencia de tonalidad en semitonos) a un ratio de frecuencia.

12 semitonos	→	2
-12 semitonos	→	-2
etc.		

H.36. Convert > logT2sec



Converts Reaktor primary level logarithmic time (used for envelopes) to seconds.

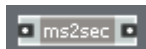
0	→	0.001 sec
60	→	1 sec
etc.		

H.37. Convert > ms2Hz



Convierte un período de tiempo de milisegundos a la frecuencia correspondiente en Hz. Por ejemplo 100ms → 10 Hz.

H.38. Convert > ms2sec



Convierte un tiempo especificado en milisegundos a un tiempo especificado en segundos. Por ejemplo, 500 ms → 0.5 sec.

H.39. Convert > P2F



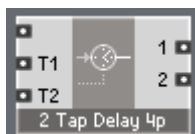
Convierte una señal de control desde la escala de tono a la escala de frecuencia. Por ejemplo, 69 → 440 Hz.

H.40. Convert > sec2Hz



Convierte un período de tiempo de segundos en la frecuencia correspondiente en hz. Por ejemplo, 0.1 sec → 10 Hz.

H.41. Delay > 2 / 4 Tap Delay 4p



Tap Delay de 2/4 pasos con 4 puntos de interpolación. Las entradas T1...T2 especifican el tiempo de delay en segundos para cada uno de los pasos. El máximo tiempo de delay por defecto es 44100 samples, que es 1 seg a 44.1 KHz. Para ajustar el tiempo, cambia el tamaño de la serie en la macro delay.

H.42. Delay > Delay 1p / 2p / 4p



Delay 1-punto (no interpolado)/2-puntos (interpolado)/4-puntos (interpolado). La entrada T especifica el tiempo de delay en segundos. El tiempo máximo de delay por defecto es 44100 samples, que es 1 seg a 44.1 KHz. Para ajustar el tiempo, cambia el tamaño de la serie en la macro delay. Usa las versiones interpoladas de delay para delays modulados. Para delays no-modulados (tiempo fijo), normalmente es mejor usar una versión no interpolada.

H.43. Delay > Diff Delay 1p / 2p / 4p



Delay de difusión de 1-punto (no interpolado)/2-puntos (interpolado)/4-puntos (interpolado). La entrada T especifica el tiempo de delay en segundos. La entrada Dffs ajusta el factor de difusión. El tiempo máximo de delay por defecto es 44100 samples, que es 1 seg a 44.1 KHz. Para ajustar el tiempo, cambia el tamaño de la serie en la macro delay.

H.44. Envelope > ADSR



Genera una Envolvente ADSR.

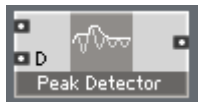
- A, D, R especifica los tiempos de ataque, decay y release en segundos
- S especifica el nivel de sustain (alcance 0..1, a 1 el nivel de sustain es igual al nivel de pico)
- G entrada de puerta. Los eventos entrantes positivos (re-)inician la envolvente. Los eventos cero o negativos cierran la envolvente
- GS sensibilidad de la puerta. A sensibilidad cero, el pico de la envolvente tiene siempre amplitud de 1. A sensibilidad igual a uno, el nivel de pico es igual al nivel positivo de la puerta.
- RM modo re-activación [retrigger]. Selecciona entre el modo analógico/digital y el modo retrigger/legato. En el modo “digital”, la envolvente se reinicializa desde su nivel de salida actual. En el modo “retrigger”, los eventos de puerta positivos consecutivos reinicializarán la envolvente, mientras que el modo “legato” se reinicializará sólo cuando la puerta cambie de negativo/cero a positivo. Los valores RM permitidos son los siguientes:
 - RM = 0 retrigger analógico (por defecto)
 - RM = 1 legato analógico
 - RM = 2 retrigger digital
 - RM = 3 legato digital

H.45. Envelope > Env Follower



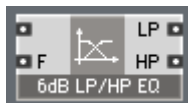
Transmite una señal de control que “sigue” la envolvente de una señal de audio entrante. Las entradas A y D especifican el tiempo de los parámetros de ataque y de decay en segundos.

H.46. Envelope > Peak Detector



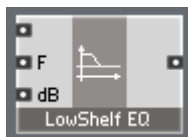
Transmite el “último” nivel de peak del audio entrante como señal de control. La entrada D especifica el nivel de tiempo en la salida del parámetro decay en segundos.

H.47. EQ > 6dB LP/HP EQ



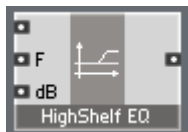
EQ paso-bajo / paso-alto de 1-pole (6dB/octava). La entrada F especifica el corte de frecuencia (en Hz) para las salidas LP y HP.

H.48. EQ > 6dB LowShelf EQ



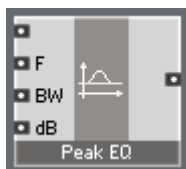
EQ de rampa de graves de 1-pole. La entrada dB especifica el refuerzo de graves en dB (los valores negativos atenuarán las frecuencias), y la entrada F especifica la frecuencia central de corte en Hz.

H.49. EQ > 6dB HighShelf EQ



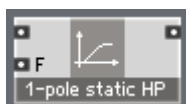
EQ de rampa de agudos de 1-pole. La entrada dB especifica el refuerzo de agudos en dB (los valores negativos atenuarán las frecuencias), y la entrada F especifica la frecuencia central de corte en Hz.

H.50. EQ > Peak EQ



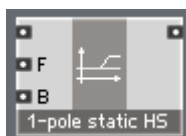
EQ pico / valle de 2-pole. La entrada F especifica la frecuencia central en Hz. La entrada BW especifica el ancho de banda de la EQ en octavas y la entrada dB especifica la altura de pico (los valores negativos crearán un valle).

H.51. EQ > Static Filter > 1-pole static HP



Filtro estático paso-alto de 1-pole. La entrada F especifica la frecuencia de corte en Hz.

H.52. EQ > Static Filter > 1-pole static HS



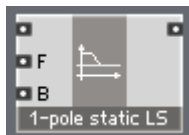
Filtro estático de rampa de agudos de 1-pole. La entrada F especifica la frecuencia de corte en Hz y la entrada B especifica el refuerzo de frecuencias agudas en dB.

H.53. EQ > Static Filter > 1-pole static LP



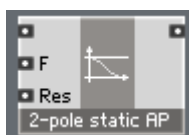
Filtro estático paso-bajo de 1-pole. La entrada F especifica la frecuencia de corte en Hz.

H.54. EQ > Static Filter > 1-pole static LS



Filtro estático de rampa de graves de 1-pole. La entrada F especifica la frecuencia de corte en Hz y la entrada B especifica el refuerzo de frecuencias graves en dB.

H.55. EQ > Static Filter > 2-pole static AP



Filtro estático paso-todo [allpass] de 2-pole. La entrada F especifica la frecuencia de corte en Hz y la entrada Res especifica la resonancia (0..1).

H.56. EQ > Static Filter > 2-pole static BP



Filtro estático paso-banda de 2-pole. La entrada F especifica la frecuencia de corte en Hz y la entrada Res especifica la resonancia (0...1).

H.57. EQ > Static Filter > 2-pole static BP1



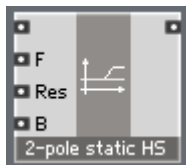
Filtro estático paso-banda de 2-pole. La entrada F especifica la frecuencia de corte en Hz y la entrada Res especifica la resonancia (0...1). La ampliación en la frecuencia de corte es siempre 1 independientemente de la resonancia.

H.58. EQ > Static Filter > 2-pole static HP



Filtro estático paso-alto de 2-pole. La entrada F especifica la frecuencia de corte en Hz y la entrada Res especifica la resonancia (0...1).

H.59. EQ > Static Filter > 2-pole static HS



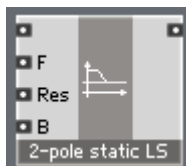
Filtro estático de rampa de agudos de 2-pole. La entrada F especifica la frecuencia de corte en Hz, la entrada Res especifica la resonancia (0...1), y la entrada B especifica el refuerzo de frecuencias agudas en dB.

H.60. EQ > Static Filter > 2-pole static LP



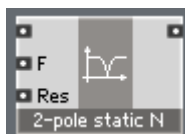
Filtro estático paso-bajo de 2-pole. La entrada F especifica la frecuencia de corte en Hz y la entrada Res especifica la resonancia (0...1).

H.61. EQ > Static Filter > 2-pole static LS



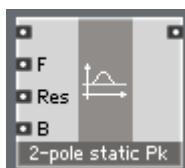
Filtro estático de rampa de graves de 2-pole. La entrada F especifica la frecuencia de corte en Hz, la entrada Res especifica la resonancia (0...1), y la entrada B especifica el refuerzo de frecuencias graves en dB.

H.62. EQ > Static Filter > 2-pole static N



Filtro estático de valle de 2-pole. La entrada F especifica la frecuencia de corte en Hz y la entrada Res especifica la resonancia (0...1).

H.63. EQ > Static Filter > 2-pole static Pk



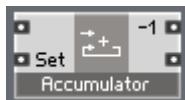
Filtro estático de pico de 2-pole. La entrada F especifica la frecuencia de corte en Hz, la entrada Res especifica la resonancia (0...1), y la entrada B especifica el refuerzo de frecuencias medias en dB.

H.64. EQ > Static Filter > Integrator



Integra la señal de audio entrante usando un método de suma rectangular. Un evento en la entrada Rst reajusta [resets] la salida del integrador al valor de este evento.

H.65. Event Processing > Accumulator



Calcula la suma de los valores de la entrada superior. Un evento en la entrada Rst reajusta la salida por el valor de este evento. El valor de la salida inferior es la suma de todos los eventos previos; el valor de salida superior es la suma de todos los eventos excepto el último.

H.66. Event Processing > Clk Div



Divisor de frecuencia de reloj. Los eventos de reloj que llegan a la entrada superior se filtrarán, permitiendo sólo que pasen los eventos 1st, $N+1$ th, $2N+1$ th etc. (N es el valor de la entrada inferior y especifica la razón de la división).

H.67. Event Processing > Clk Gen



Genera eventos de reloj a la frecuencia especificada en la entrada (en Hz). Este módulo sólo funciona dentro de las células core de audio.

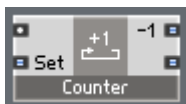
H.68. Event Processing > Clk Rate



Estima la frecuencia y el período de los eventos de reloj entrantes. La salida F es la frecuencia en Hz y la salida T es el período en segundos. Este módulo sólo funciona dentro de las células core de audio.

El valor inicial de período es cero y la frecuencia es un valor muy grande. Conseguirás una salida razonable sólo después del segundo evento de reloj.

H.69. Event Processing > Counter



Cuenta el número de eventos en la entrada superior. Un evento en la entrada “Rst” reajusta la salida por el valor de este evento. El valor de la salida inferior es la cuenta de todos los eventos previos; el valor de la salida superior es la cuenta de todos los eventos excepto el último.

H.70. Event Processing > Ctl2Gate



Convierte una señal de control (o de audio) en la entrada superior en una señal de puerta con una amplitud definida por la entrada inferior. Los cruces [crossings] cero positivos abrirán la puerta; los cero negativos la cerrarán.

H.71. Event Processing > Dup Flt / IDup Flt



Filtra eventos con valores duplicados (sólo podrán pasar los eventos con valores diferentes de los previos).

H.72. Event Processing > Impulse



Genera un impulso de un sample de amplitud 1 en respuesta al evento entrante. Este módulo sólo funciona dentro de las células core.

H.73. Event Processing > Random



Genera números aleatorios [random] en respuesta a los relojes entrantes. El alcance de salida es $-1..1$. Un evento en la entrada Seed reajustará el generador con el valor de ese evento.

H.74. Event Processing > Separator / ISeparator



Los eventos de la entrada superior mayores que el valor de Thld se rutearán a la entrada Hi. El resto, se rutearán a la salida Lo.

H.75. Event Processing > Thld Crossing



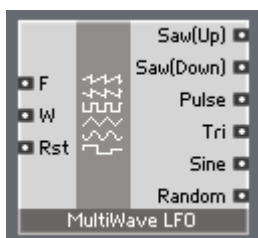
Cada vez que una señal ascendente en la entrada superior cruce el threshold especificado por la entrada inferior, se enviará un evento desde la salida Up. Cada vez que una señal descendente cruce el threshold, se enviará un evento desde la salida Dn.

H.76. Event Processing > Value / IValue



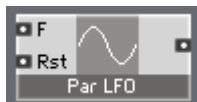
Cambia el valor de un evento que llega a la entrada superior por el valor disponible en ese momento en la entrada inferior.

H.77. LFO > MultiWave LFO



Transmite simultáneamente varias formas de onda de baja frecuencia sincronizadas en fase. La entrada F especifica la frecuencia en Hz, la entrada W controla la amplitud del pulso (alcance -1..0..1, sólo afecta a la salida Pulse), los eventos de la entrada Rst reajustan el LFO a la fase especificada por el valor del evento (alcance 0..1).

H.78. LFO > Par LFO



Genera una señal de control de baja frecuencia parabólica. La señal F especifica la frecuencia en Hz, los eventos en la entrada Rst reajustan el LFO a la fase especificada por el valor del evento (alcance 0..1).

H.79. LFO > Random LFO



Genera una señal de control escalonada de baja frecuencia aleatoria ("random sample-and-hold"). La entrada F especifica la frecuencia en Hz, los eventos de la entrada Rst reajustan el LFO a la fase especificada por el valor del evento (alcance 0..1).

H.80. LFO > Rect LFO



Genera una señal de control rectangular de baja frecuencia. La entrada F especifica la frecuencia en Hz, la entrada W controla la amplitud del pulso (alcance -1..0..1), los eventos de la entrada Rst reajustan el LFO a la fase especificada por el valor del evento (alcance 0..1).

H.81. LFO > Saw(down) LFO



Genera una señal de control de diente de sierra descendente de baja frecuencia. La entrada F especifica la frecuencia en Hz, los eventos en la entrada Rst reajustan el LFO a la fase especificada por el valor del evento (alcance 0..1).

H.82. LFO > Saw(up) LFO



Genera una señal de control de diente de sierra ascendente de baja frecuencia. La entrada F especifica la frecuencia en Hz, los eventos en la entrada Rst reajustan el LFO a la fase especificada por el valor del evento (alcance 0..1).

H.83. LFO > Sine LFO



Genera una señal de control senoidal de baja frecuencia. La entrada F especifica la frecuencia en Hz, los eventos en la entrada Rst reajustan el LFO a la fase especificada por el valor del evento (alcance 0..1).

H.84. LFO > Tri LFO



Genera una señal de control triangular de baja frecuencia. La entrada F especifica la frecuencia en Hz, los eventos en la entrada Rst reajustan el LFO a la fase especificada por el valor del evento (alcance 0..1).

H.85. Logic > AND



Realiza una conjunción de dos señales lógicas (la salida es 1 sólo si ambas entradas son 1). Para valores de entrada que no sean 0 ó 1 el resultado es indefinido.

H.86. Logic > Flip Flop



La salida alterna entre 0 y 1 cada vez que la entrada de reloj recibe un evento.

H.87. Logic > Gate2L



Convierte las señales de puerta en señales lógicas. La puerta abierta produce un valor 1 en la salida; la puerta cerrada produce un valor 0 en la salida.

H.88. Logic > GT / IGT



Compara los dos valores de entrada flotantes/enteros y saca 1 si el valor de arriba es mayor que el de abajo, si no, saca 0.

H.89. Logic > EQ



Compara los dos valores enteros entrantes y saca 1 si ambos valores son iguales, si no, saca 0.

H.90. Logic > GE



Compara los dos valores enteros entrantes y saca 1 si el valor de arriba es mayor que el de abajo, si no, saca 0.

H.91. Logic > L2Clock



Convierte una señal lógica en una señal de reloj. Al cambiar la señal de entrada de 0 a 1, envía un evento de reloj. Para valores de entrada que no sean 0 ó 1 la funcionalidad es indefinida.

H.92. Logic > L2Gate



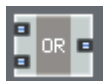
Convierte una señal lógica en una señal de puerta. Al cambiar la señal de entrada de 0 a 1 se abre la puerta, y al revés, se cierra. El nivel de la puerta abierta se define por el valor de la entrada inferior (por defecto = 1). Para valores de entrada que no sean 0 y 1 la funcionalidad es indefinida.

H.93. Logic > NOT



Convierte 1 en 0 y viceversa. Para valores de entrada que no sean 0 y 1 el resultado es indefinido.

H.94. Logic > OR



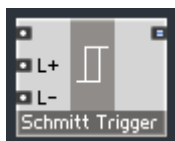
Realiza una disyunción de dos señales lógicas (la salida es 1 si al menos una de las entradas es 1). Para valores de entrada que no sean 0 y 1 el resultado es indefinido.

H.95. Logic > XOR



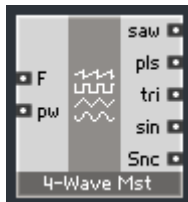
Realiza una disyunción exclusiva de dos señales lógicas (la salida es 1 si una de las entradas es igual a 1 y la otra es igual a 0). Para valores de entrada que no sean 0 y 1 el resultado es indefinido

H.96. Logic > Schmitt Trigger



Cambia la salida a 1 si el valor de entrada se hace más largo que L+ (por defecto =0.67); y cambia a salida a 0 si el valor de entrada se hace menos largo que L- (por defecto =0.33).

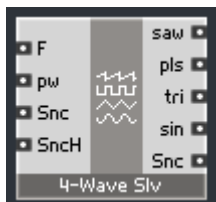
H.97. Oscillators > 4-Wave Mst



Genera 4 formas de onda de audio sincronizadas en fase. La frecuencia está especificada en la entrada F (en Hz). La amplitud del pulso está especificada por la entrada 'pw' (alcance -1..0..1, afecta sólo a la forma de onda del pulso).

Este oscilador puede oscilar en frecuencias negativas y además ofrece salida de sincronización para el oscilador *4 Wave Slv*.

H.98. Oscillators > 4-Wave Slv



Genera formas de onda de audio. La frecuencia está especificada en la entrada F (en Hz). La amplitud del pulso está especificada por la entrada 'pw' (alcance -1..0..1, afecta sólo a la forma de onda del pulso).

Este oscilador puede oscilar en frecuencias negativas y se puede sincronizar con otro oscilador 4 Wave Mst/Slv. La entrada SncH controla la dureza de sincronización (0= no hay sincronización, 1 = sincronización dura, 0..1 = varios grados de sincronización). También proporciona una salida de sincronización para otro oscilador *4 Wave Slv*.

H.99. Oscillators > Binary Noise



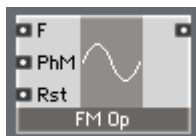
Generador de ruido blanco binario. Entrega valores aleatorios que alternan entre 1 y -1. La llegada de un evento a la entrada Seed (re-)inicializaría el generador aleatorio con el valor dado a esa entrada.

H.100. Oscillators > Digital Noise



Operador FM clásico. Saca una onda senoidal cuya frecuencia está definida por la entrada F (en Hz). La fase de la onda senoidal se puede modular en la entrada PhM (en radianes). Un evento entrante por la entrada Rst reajustaría el oscilador a la fase definida por el valor de ese evento (alcance 0..1).

H.101. Oscillators > FM Op



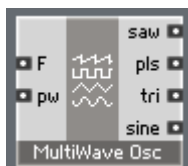
Operador FM clásico. Saca una onda senoidal cuya frecuencia está definida por la entrada F (en Hz). La fase de la onda senoidal se puede modular en la entrada PhM (en radianes). Un evento entrante por la entrada Rst reajustaría el oscilador a la fase definida por el valor de ese evento (alcance 0..1).

H.102. Oscillators > Formant Osc



Genera una forma de onda con una frecuencia fundamental especificada por la entrada F (en Hz) y la frecuencia de formantes especificada por la entrada Fmt (en Hz).

H.103. Oscillators > MultiWave Osc



Genera 4 formas de onda de audio sincronizadas en fase. La frecuencia está especificada por la entrada F (en Hz). La amplitud del pulso está especificada por la entrada 'pw' (alcance -1..0..1, afecta sólo a la forma de onda del pulso).

Este oscilador no puede oscilar en frecuencias negativas.

H.104. Oscillators > Par Osc



Genera una forma de audio parabólica. La entrada F especifica la frecuencia en Hz.

H.105. Oscillators > Quad Osc



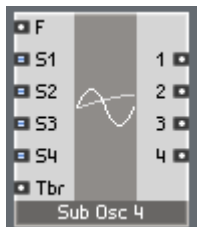
Genera un par de formas de onda senoidales con una diferencia de fase de 90 grados. La entrada F especifica la frecuencia en Hz.

H.106. Oscillators > Sin Osc



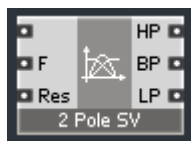
Genera una onda senoidal. La entrada F especifica la frecuencia en Hz.

H.107. Oscillators > Sub Osc 4



Genera 4 subarmónicos sincronizados en fase. La frecuencia “fundamental” está especificada por la entrada F (en Hz). Los números subarmónicos están especificados por las entradas S1..S4 (alcance 1..120). La entrada Tbr controla el contenido armónico de la forma de onda saliente (alcance 0..1).

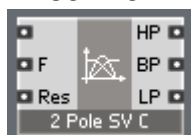
H.108. VCF > 2 Pole SV



Filtro de estado variable de 2-pole. La entrada F especifica el corte en Hz, y la entrada Res especifica la resonancia (alcance 0..0.98).

Las salidas HP/BP/LP producen señales pasa-agudos, pasa-bandas y pasa-graves respectivamente.

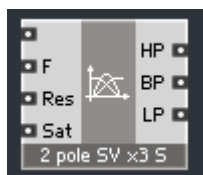
H.109. VCF > 2 Pole SV C



Filtro de estado variable de 2-pole (versión compensada). Ofrece un comportamiento mejorado en ajustes altos de corte. La entrada F especifica el corte en Hz y la entrada Res especifica la resonancia (alcance 0..0.98). También puedes usar valores negativos, lo que forzará aún más la pendiente.

Las señales HP/BP/LP producen señales paso-alto, paso-banda y paso-bajo respectivamente.

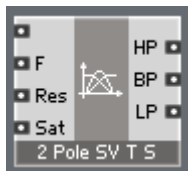
H.110. VCF > 2 Pole SV (x3) S



Filtro de estado variable de 2-pole con saturación y sobre-muestreo opcional (versión x3). La entrada F especifica el corte en Hz, la entrada Res especifica la resonancia (alcance 0..1), y la entrada Sat especifica el nivel de saturación (alcance típico 8..32).

Las señales HP/BP/LP producen señales paso-alto, paso-banda y paso-bajo respectivamente.

H.111. VCF > 2 Pole SV T (S)



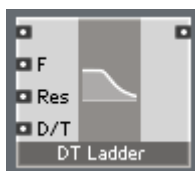
Filtro de estado variable de 2-pole con compensación de tabla y saturación opcional (versión S). Ofrece un comportamiento mejorado en ajustes altos de corte, pero es algo diferente de la versión 2 Pole SV C. La entrada F especifica el corte en Hz la entrada Res especifica la resonancia (alcance 0..1), y la entrada Sat especifica el nivel de saturación (alcance típico 8..32). Las señales HP/BP/LP producen señales pasa-agudos, pasa-bandas y pasa-graves respectivamente.

H.112. VCF > Diode Ladder



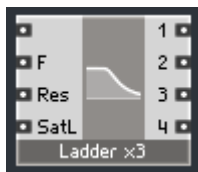
Emulación lineal de filtro de escalera de diodo. La entrada F especifica el corte en Hz y la entrada Res especifica la resonancia (alcance 0..0.98).

H.113. VCF > D/T Ladder



Emulación lineal de filtro de escalera. Se puede alternar entre el comportamiento de diodo y de transistor. La entrada F especifica el corte en Hz, la entrada Res especifica la resonancia (alcance 0..0.98), y la entrada D/T alterna entre diodo y transistor (0 = diodo, 1= transistor).

H.114. VCF > Ladder x3



Una emulación de un filtro tipo escalera de saturación por transistor (sobremuestreado x3 veces). La entrada F especifica el corte en Hz, la entrada Res especifica la resonancia (alcance 0..0.98), y la entrada SatL especifica el nivel de saturación (alcance típico 1..32).

Las salidas 1-4 se cogen de los pasos correspondientes de la escalera emulado. Usa el 4º paso para conseguir un sonido de filtro de escalera 'clásico'.

Apéndice I. Core cell library

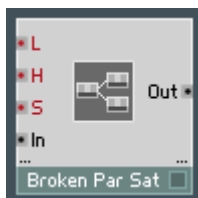
I.1. Audio Shaper > 3-1-2 Shaper



Modelador de señal de audio con cantidad variable de distorsión de 2º y 3º orden. La cantidad de distorsión y el tipo se controlan por la entrada “Shp”:

Shp = 0	no hay modelado
Shp > 0	modelado 3º orden
Shp < 0	modelado 2º orden

I.2. Audio Shaper > Broken Par Sat



Saturador parabólico roto. Tiene un segmento lineal alrededor del nivel cero. La entrada “L” especifica el nivel de salida de “saturación total” (por defecto = 1). La entrada “H” especifica la dureza (alcance 0..1). Los valores más largos corresponden a los segmentos más largos del medio. La entrada “S” controla la simetría de la curva de modelado (alcance -11). En 0, la curva es simétrica.

I.3. Audio Shaper > Hyperbol Sat



Saturador hiperbólico simple. La entrada “L” especifica el nivel de salida de “saturación total (por defecto = 1). No obstante, la “saturación total” nunca se consigue con este tipo de saturador.

I.4. Audio Shaper > Parabol Sat



Saturador parabólico simple. La entrada “L” especifica el nivel de salida de “saturación total” (por defecto = 1).

Nota: La saturación total se consigue con el nivel de entrada igual a 2L.

I.5. Audio Shaper > Sine Shaper 4/8



Modelador de senoidales de 4º y 8º orden. El modelador de 8º orden tiene una aproximación senoidal mejor pero consume más CPU.

I.6. Control > ADSR



Genera una envolvente ADSR.

- A, D, R especifica los tiempos de ataque, decay y release en segundos.
- S especifica el nivel de sustain (alcance 0..1, en 1 el nivel de sustain es igual al nivel de pico).
- G entrada de puerta. Los eventos positivos entrantes (re-)inician la envolvente. Los eventos con valor cero o negativos cierran la envolvente.
- GS sensibilidad de la puerta. A sensibilidad cero, el pico de la envolvente tiene siempre amplitud de 1. A sensibilidad igual a uno, el nivel de pico es igual al nivel positivo de la puerta.

RM modo re-activación [retrigger]. Selecciona entre el modo analógico/digital y el modo retrigger/legato. En el modo “digital”, la envolvente se reinicializa desde su nivel de salida actual. En el modo “retrigger”, los eventos de puerta positivos consecutivos reinicializarán la envolvente, mientras que el modo “legato” se reinicializará sólo cuando la puerta cambie de negativo/cero a positivo. Los valores RM permitidos son los siguientes:

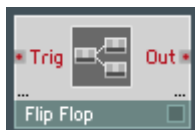
RM = 0	retrigger analógico (por defecto)
RM = 1	legato analógico
RM = 2	retrigger digital
RM = 3	legato digital

I.7. Control > Env Follower



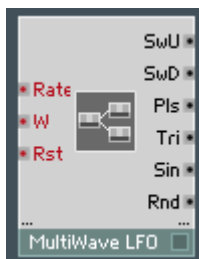
Transmite una señal de control que “sigue” la envolvente de una señal de audio entrante. Las entradas A y D especifican el tiempo de los parámetros de ataque y de decay en segundos.

I.8. Control > Flip Flop



La salida alterna entre 0 y 1 cada vez que la entrada Trigger recibe un evento.

I.9. Control > MultiWave LFO



Transmite simultáneamente varias formas de onda de baja frecuencia sincronizadas en fase. La entrada Rate especifica la frecuencia en Hz, la entrada W controla la amplitud del pulso (alcance $-1..0..1$, sólo afecta a la salida de Pulse), los eventos en la entrada Rst reajustan el LFO a la fase especificada por el valor del evento (alcance $0..1$).

I.10. Control > Par Ctl Shaper



Aplica una doble curva parabólica a una señal controladora. La señal de entrada ha de estar en el alcance $-1..0..1$. La señal de salida también ha de estar en el alcance $-1..0..1$. El grado de desviación se controla por la entrada “b” (el alcance también es $-1..0..1$).

$b = 0$ no hay desviación (curva lineal)

$b = -1$ máxima desviación posible “hacia” el eje X

$b = 1$ máxima desviación posible “hacia” el eje Y

También puedes usar este modelador para señales cuyo alcance sea $0..1$, en cuyo caso sólo se usará la mitad de la curva.

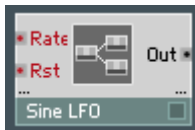
Uso típico: modelado de velocidad y otros controles

I.11. Control > Schmitt Trigger



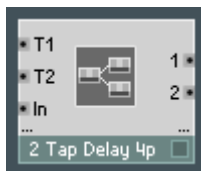
Cambia la salida a 1 si el valor de entrada se hace más largo que L+ (por defecto 0.67), y a 0 si el valor de entrada se hace menor que L- (por defecto 0.33).

I.12. Control > Sine LFO



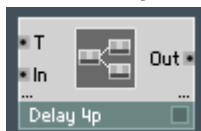
Genera una señal de control de baja frecuencia de forma senoidal. La entrada Rate especifica la frecuencia en Hz, y los eventos de la entrada Rst reinician el LFO a una fase especificada por el valor del evento (alcance $0..1$).

I.13. Delay > 2/4 Tap Delay 4p



Tap Delay de 2 / 4 pasos con 4 puntos de interpolación. Las entradas T1..T4 especifican el tiempo de delay en milisegundos para cada uno de los pasos. El tiempo máximo de delay es por defecto de 44100 samples, lo que es 1 segundo a 44.1KHz. Para ajustar el tiempo, cambia el tamaño de la serie en la macro delay.

I.14. Delay > Delay 4p



Delay interpolado de 4-puntos. La entrada T especifica el tiempo de delay en milisegundos.

El tiempo máximo de delay por defecto es 44100 samples, lo que es 1 segundo a 44.1KHz. Para ajustar el tiempo, cambia el tamaño de la serie en la macro delay.

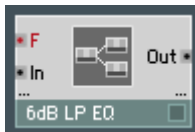
I.15. Delay > Diff Delay 4p



Delay de difusión interpolado de 4-puntos. La entrada T especifica el tiempo de delay en milisegundos. La entrada Dffs ajusta el factor de difusión.

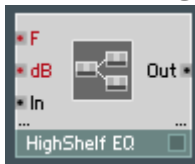
El tiempo máximo de delay por defecto es 44100 samples, lo que es 1 segundo a 44.1KHz. Para ajustar el tiempo, cambia el tamaño de la serie en la macro delay.

I.16. EQ > 6dB LP/HP EQ



EQ paso-bajo / paso-alto de 1-pole (6dB/octava). La entrada F especifica la frecuencia de corte (en Hz).

I.17. EQ > HighShelf EQ



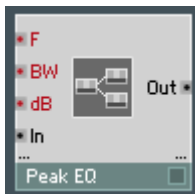
EQ de rampa de agudos de 1-pole. La entrada dB especifica la ampliación de frecuencias altas en dB (los valores negativos cortarán las frecuencias), y la entrada F especifica la frecuencia central de corte en Hz.

I.18. EQ > LowShelf EQ



EQ de rampa de graves de 1-pole. La entrada dB especifica la ampliación de frecuencias graves en dB (los valores negativos cortarán las frecuencias), y la entrada F especifica la frecuencia central de corte en Hz.

I.19. EQ > Peak EQ



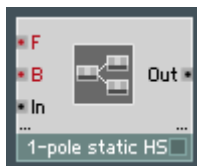
EQ pico / valle de 2-pole. La entrada F especifica las frecuencias centrales en Hz, la entrada BW especifica el ancho de banda de la EQ en octavas y la entrada dB especifica la altura del pico (los valores negativos producirán un valle).

I.20. EQ > Static Filter > 1-pole static HP



Filtro estático paso-alto de 1-pole. La entrada F especifica la frecuencia de corte en Hz.

I.21. EQ > Static Filter > 1-pole static HS



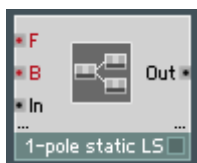
Filtro estático de rampa de agudos de 1-pole. La entrada F especifica la frecuencia de corte en Hz y la entrada B especifica el refuerzo de frecuencias altas en dB.

I.22. EQ > Static Filter > 1-pole static LP



Filtro estático paso-bajo de 1-pole. La entrada F especifica la frecuencia de corte en Hz.

I.23. EQ > Static Filter > 1-pole static LS



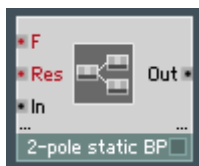
Filtro estático de rampa de graves de 1-pole. La entrada F especifica la frecuencia de corte en Hz y la entrada B especifica el refuerzo de frecuencias graves en dB.

I.24. EQ > Static Filter > 2-pole static AP



Filtro estático paso-todo de 2-pole. La entrada F especifica la frecuencia de corte en Hz y la entrada Res especifica la resonancia (0..1).

I.25. EQ > Static Filter > 2-pole static BP



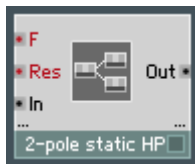
Filtro estático paso-banda de 2-pole. La entrada F especifica la frecuencia de corte en Hz y la entrada Res especifica la resonancia (0..1).

I.26. EQ > Static Filter > 2-pole static BP1



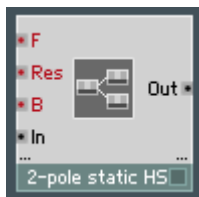
Filtro estático paso-banda de 2-pole. La entrada F especifica la frecuencia de corte en Hz y la entrada Res especifica la resonancia (0..1). La amplificación en la frecuencia de corte es siempre 1 al margen de la resonancia.

I.27. EQ > Static Filter > 2-pole static HP



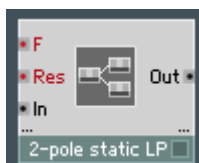
Filtro estático paso-alto de 2-pole. La entrada F especifica la frecuencia de corte en Hz y la entrada Res especifica la resonancia (0..1).

I.28. EQ > Static Filter > 2-pole static HS



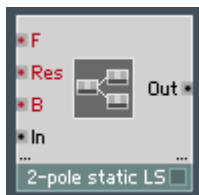
Hz, la entrada Res especifica la resonancia (0..1), y la entrada B especifica la ampliación de las frecuencias altas en dB.

I.29. EQ > Static Filter > 2-pole static LP



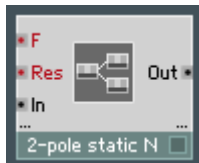
Filtro estático paso-bajo de 2-pole. La entrada F especifica la frecuencia de corte en Hz y la entrada Res especifica la resonancia (0..1).

I.30. EQ > Static Filter > 2-pole static LS



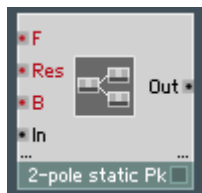
Filtro estático de rampa de graves de 2-pole. La entrada F especifica la frecuencia de corte en Hz, la entrada Res especifica la resonancia (0..1), y la entrada B especifica el refuerzo de frecuencias graves en dB.

I.31. EQ > Static Filter > 2-pole static N



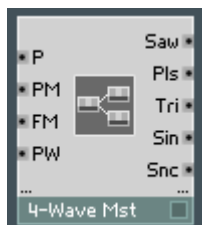
Filtro estático de valle de 2-pole. La entrada F especifica la frecuencia de corte en Hz y la entrada Res especifica la resonancia (0..1).

I.32. EQ > Static Filter > 2-pole static Pk



Filtro estático de pico de 2-pole. La entrada F especifica la frecuencia de corte en Hz, la entrada Res especifica la resonancia (0..1), y la entrada B especifica la ampliación de frecuencias centrales en dB.

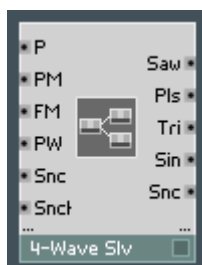
I.33. Oscillator > 4-Wave Mst



Genera 4 formas de onda de audio sincronizadas en fase. El tono del oscilador está especificado por la entrada P (como número de nota MIDI), se puede modular por la entrada PM (en semitonos, exponencial) y por la entrada FM (en Hz, lineal). La amplitud del pulso está especificada por la entrada 'pw' (alcance -1..0..1, sólo afecta a la forma de onda del pulso).

Este oscilador puede oscilar en frecuencias negativas y además ofrece una salida de sincronización para el oscilador *4 Wave Slv*.

I.34. Oscillator > 4-Wave Slv

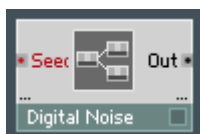


Genera 4 formas de onda de audio sincronizadas en fase. El tono del oscilador está especificado por la entrada P (como número de nota MIDI), se puede

modular por la entrada PM (en semitonos, exponencial) y por la entrada FM (en Hz, lineal). La amplitud del pulso está especificada por la entrada 'pw' (alcance -1..0..1, sólo afecta a la forma de onda del pulso).

Este oscilador puede oscilar en frecuencias negativas y se puede sincronizar con otro oscilador 4 Wave Mst/Slv. La entrada SncH controla la dureza de sincronización (0 = no hay sincronización, 1 = sincronización dura, 0..1 = varios grados de sincronización). También proporciona una salida de sincronización para otro módulo 4 Wave Slv

I.35. Oscillator > Digital Noise



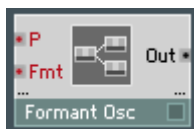
Generador de ruido blanco digital. Transmite valores aleatorios en el alcance -1..1. La llegada de un evento a la entrada Seed (re-)inicializaría el generador aleatorio con el valor dado a esa entrada.

I.36. Oscillator > FM Op



Operador FM clásico. Transmite una onda senoidal cuyo tono está especificado por la entrada P (como número de nota MIDI). La fase de la senoidal se puede modular por la entrada PhM (en radianes). Un evento que llega a la entrada Rst reiniciaría el oscilador a una fase especificada por el valor de este evento (alcance 0..1).

I.37. Oscillator > Formant Osc



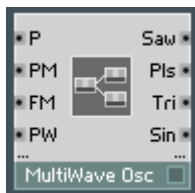
Genera una forma de onda con una frecuencia fundamental especificada por la entrada P (como número de nota MIDI) y la frecuencia de formantes especificada por la entrada Fmt (en Hz).

I.38. Oscillator > Impulse



Genera un impulso de un sample de amplitud 1 en respuesta a un evento entrante.

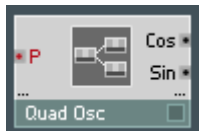
I.39. Oscillator > MultiWave Osc



Genera 4 formas de onda de audio sincronizadas en fase. El tono del oscilador está especificado por la entrada P (como número de nota MIDI), se puede modular por la entrada PM (en semitonos, exponencial) y por la entrada FM (en Hz, lineal). La amplitud del pulso está especificada por la entrada 'pw' (alcance -1..0..1, sólo afecta a la forma de onda del pulso).

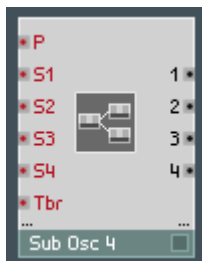
Este oscilador no puede oscilar en frecuencias negativas

I.40. Oscillator > Quad Osc



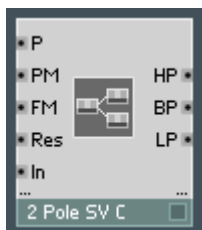
Genera un par de formas de onda senoidales con una diferencia de fase de 90 grados. La entrada P especifica el tono (como número de nota MIDI).

I.41. Oscillator > Sub Osc



Genera 4 subarmónicos sincronizadas en fase. La frecuencia “fundamental” está especificada por la entrada P (como número de nota MIDI). Los números de subarmónicos están especificados por las entradas S1..S4 (alcance 1..120). La entrada Tbr controla el contenido armónico de la forma de onda saliente (alcance 0..1).

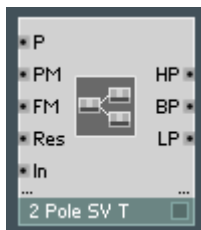
I.42. VCF > 2 Pole SV C



Filtro de estado variable de 2-pole (versión de compensación). Ofrece un comportamiento mejorado en ajustes altos de corte. La frecuencia de corte del filtro está especificada por la entrada P (como número de nota MIDI), se puede modular por la entrada PM (en semitonos, exponencial) y por la FM (en Hz, lineal). La entrada Res especifica la resonancia (alcance 0..0.98). También puedes usar valores negativos de resonancia, lo que forzará aún más la pendiente.

Las salidas HP/BP/LP producen señales pasa-agudos, pasa-bandas y pasa-graves respectivamente.

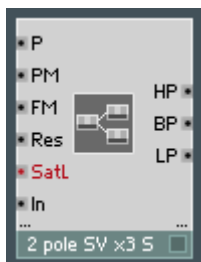
I.43. VCF > 2 Pole SV T



Filtro de estado variable de 2-pole con compensación de tabla. Ofrece un comportamiento mejorado en ajustes altos de corte, pero es algo diferente de la versión 2 Pole SV C. La frecuencia de corte del filtro está especificada por la entrada P (como número de nota MIDI) y se puede modular por la entrada PM (en semitonos, exponencial) y por la FM (en Hz, lineal). La entrada Res especifica la resonancia (alcance 0..1).

Las salidas HP/BP/LP producen señales pasa-agudos, pasa-bandas y pasa-graves respectivamente.

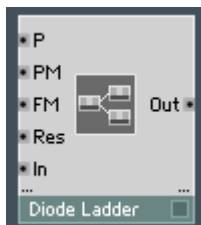
I.44. VCF > 2 Pole SV x3 S



Filtro de estado variable de 2-pole con saturación y sobre-muestreo opcional. La frecuencia de corte del filtro está especificada por la entrada P (como número de nota MIDI) y se puede modular por la entrada PM (en semitonos, exponencial) y por la FM (en Hz, lineal). La entrada Res especifica la resonancia (alcance 0..1) y la entrada Sat especifica el nivel de saturación (alcance típico 8..32).

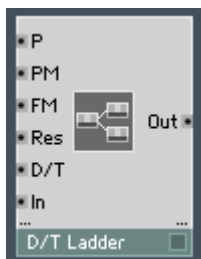
Las salidas HP/BP/LP producen señales pasa-agudos, pasa-bandas y pasa-graves respectivamente

I.45. VCF > Diode Ladder



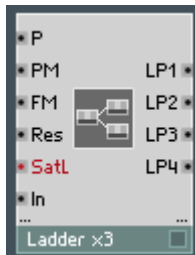
Emulación de filtro lineal de escalera de diodo. La frecuencia de corte del filtro está especificada por la entrada P (como número de nota MIDI) y se puede modular por la entrada PM (en semitonos, exponencial) y por la FM (en Hz, lineal). La entrada Res especifica la resonancia (alcance 0..0.98).

I.46. VCF > D/T Ladder



Emulación de filtro lineal de escalera. Se puede alternar entre escalera de diodo y de transistor. La frecuencia de corte del filtro está especificada por la entrada P (como número de nota MIDI) y se puede modular por la entrada PM (en semitonos, exponencial) y por la FM (en Hz, lineal). La entrada Res especifica la resonancia (alcance 0..0.98) y la entrada D/T alterna entre diodo y transistor (0=diodo, 1=transistor).

I.47. VCF > Ladder x3



Una emulación de un filtro tipo escalera de saturación por transistor (sobremuestreado x3 veces). La frecuencia de corte del filtro está especificada por la entrada P (como número de nota MIDI) y se puede modular por la entrada PM (en semitonos, exponencial) y por la FM (en Hz, lineal). La entrada Res especifica la resonancia (alcance 0..1) y la entrada SatL especifica el nivel de saturación (alcance típico 1..32).

Las salidas 1-4 se toman de los pasos correspondientes de la escalera emulada. Usa el 4º paso para conseguir un sonido de filtro de escalera clásico.

Glosario

Symbols

1/96 Clock	51
1 / Square Root	60
12-Step	109
16-Step	109
4-Ramp	84, 125
4-Step	83
5-Ramp	126
6-Ramp	128
6-Step	108
8-Step	109

A

$a * b + c$	54
Accumulator	163
AD - Env	118
ADBDR - Env	121
ADBDSR-Env	122
Add	53
ADSR - Env	119, 120, 123, 124
Allpass 1-Pole	131
Amp	65
Appendix	198
AR - Env	118
Arccos	61
Arcsin	61
Arctan	61
A to E	180
A to E (Perm)	181
A to E (Trig)	180
A to Gate	181
Audio Modifier	152
Audio Outputs	277
Audio Smoother	183
Audio Table	161
Audio Voice Combiner	179
Auxiliary	175

B

Bad values	287, 291
Beat Loop	104
Bi-Pulse	81
Bi-Saw	68
Button	24

C

Cells	<i>See</i> Core cells
Ch. Aftertouch	44, 49
Chopper	155
Clipper	153
Clock Oscillator	86
Clock signals	258
Compare	56
Compare/Equal	57
Constant	53
Controller	43, 49
Core cells	
(re-)naming	220
basic editing of	205
creating	212
event	252
event and audio	211
using	203
Core events.	<i>See</i> Events
Core macros	
building	231
Core modules	
integer mode of	306
processing order of	250
Counter	163
Crossfade	63
Ctrl. Shaper 1 BP	165
Ctrl. Shaper 2 BP	165
Ctrl. Shaper 3 BP	166

D

D - Env	114
DBDR - Env.....	116
Default signals of inputs.	<i>See</i> Inputs
Delay.....	145
Denormal Cancel module	289
Denormal values	287
Differentiator	144
Diffuser Delay.....	147
Distributor.....	64
Divide x/y.....	55
DR - Env.....	114
DSR - Env.....	115
Dynamic ports	20

E

Envelope.....	111
ES Ctl module	309
Event Processing.....	163
Events	246
routing of	295
simultaneous	249
Event Smoother	184
Event Table	173
Event V.C. All.....	179
Event V.C. Max	179
Event V.C. Min	180
Expon. (A).....	58
Expon. (F).....	58

F

Fader.....	21
Feedback	280
around macros.....	283
indication of	280
Filter	130
Frequency Divider	160, 164
From Voice.....	182

G

Gate	24, 42
Geiger.....	87
Grain Cloud	102
Grain Cloud Delay	150
Grain Delay	148

H

H - Env.....	113
High Shelf EQ.....	142
High Shelf EQ FM	143
Hold	172
HP/LP 1-Pole.....	130
HP/LP 1-Pole FM	131
HR - Env.....	113
Hybrid modules	19

I

IC Receive.....	196
IC Send	196
Impulse	82
Impulse FM.....	82
Impulse Sync.....	82
Initialization	262
In Port.....	194
Inputs.....	<i>See</i> Ports
Integrator	145
Invert	53
Iteration.....	168

K

Knob	24
------------	----

L

Ladder Filter.....	139
Ladder Filter FM	140
Lamp.....	28
Latch module	306

Level Lamp	29
Level Meter	31
LFO.....	111
Log (A)	58
Log (F)	59
Logic AND	166
Logic EXOR.....	167
Logic NOT.....	167
Logic OR.....	167
Low Shelf EQ.....	143
Low Shelf EQ FM	144

M

Macros.	<i>See</i> Core macros
Master Tune/Level.....	185
Math	52
Merge.....	170
Meter	30
MIDI Channel Info.....	188
MIDI In.....	41
MIDI Out.....	48
Mirror 1 Level.....	154
Mirror 2 Levels	155
Mixer.....	65
Mod. Clipper	154
Modules.	<i>See</i> Core modules
Modulo	55
Mouse Area	38
Multi-Ramp	84
Multi-Sine	76
Multi-Step.....	83
Multi-Tap Delay.....	147
Multi/HP 4-Pole.....	137
Multi/HP 4-Pole FM	138
Multi/LP 4-Pole	135
Multi/LP 4-Pole FM.....	136
Multi/Notch 2-Pole.....	133
Multi/Notch 2-Pole FM	134
Multi 2-Pole	132
Multi 2-Pole FM.....	132

Multi Display	36
Multi Picture	32
Multiplex 16	110
Multiply	54
Multi Text.....	33

N

Noise.....	87
Note Pitch.....	41
Note Pitch/Gate	48
Note Range Info.....	187

O

Object Bus Connections (OBC)	259, 311
Off Velocity	43
On Velocity.....	43
Order.....	168
Oscillator	66
OSC Receive	197
OSC Send	197
Out Port.....	194
Outputs.	<i>See</i> Ports

P

Panner.....	64
Parabol	71
Par FM	72
Par PWM	73
Par Sync	72
Peak Detector.....	159
Peak EQ.....	141
Peak EQ FM	141
Picture.....	31
Pitchbend	41, 48
Poly Aftertouch.....	44
Poly Display.....	36
Ports	
audio	277
event	252

Power x y	59
Precision floating point. <i>See</i> FP Precision	
Pro-52 Filter	139
Program Change	45, 50
Pulse	77
Pulse 1-ramp.....	80
Pulse 2-ramp.....	80
Pulse FM	78
Pulse Sync	79

Q

Quantize	57
QWERTY.....	198

R

Random	87
Randomizer	164
Receive.....	194
Reciprocal.....	54
Rect./Sign	56
Rectifier.....	56
Relay 1,2	63
Router 1,2	171
Router 1->M	171
Router M->1	170
Router module.....	295, 331

S

Sample & Hold	160
Sample Lookup.....	106
Sampler	90
Sampler FM	91
Sampler Loop	92
Sampling-rate clock.....	279, 318
Saturator.....	152
Saturator 2.....	153
Saw FM	67
Saw Pulse	68

Saw Sync	67
Sawtooth	66
Scanner	62
Scope	35
Sel. Note Gate	43
Sel. Poly AT.....	45, 50
Selector	62
Send	194
Separator	169
Sequencer.....	107, 108
Set Random	191
Shaper 1 BP	156
Shaper 2 BP	156
Shaper 3 BP	157
Shaper Cubic.....	158
Shaper Parabolic.....	158
Signal Path	62
Signals	
audio	224, 277
clock.	<i>See</i> Clock signals
control	224
event	239
float.....	301, 333
integer	303, 333
logic	244
Sine	60, 74
Sine/Cos	60
Sine FM	74
Sine Sync.....	75
Single Delay	145
Single Trig. Gate	42
Slew Limiter	159
Slow Random	112
Snapshot	188
Snap Value.....	191
Song Pos	47, 51
Square Root	59
Stacked Macro.....	40
Start/Stop	45, 50
Step Filter	170
Stereo Amp	65

Stereo Pan	64
Subtract.....	53
Switch	26
Sync Clock	46
System Info.....	187

T

Table module.....	326
Tapedeck 1-Ch	175
Tapedeck 2-Ch	178
Tempo Info.....	186
Terminal.....	194
Text.....	33
Timer	172
Toggle	24
To Voice.....	182
Transposing.....	198
Tri/Par Symm	70
Triangle.....	69
Tri FM	69
Tri Sync	70
Tuning Info.....	186

U

Unison Spread.....	191
Unit Delay.....	152

V

Value	169
Voice Info.....	186
Voice Shift	183

W

Write [] macro	314
----------------------	-----

X

XY.....	33
---------	----