



REAKTOR 5

Référence core et modules

Les informations contenues dans ce document peuvent être modifiées à tout moment sans préavis et n'engagent pas la responsabilité de Native Instruments Software Synthesis GmbH. Le Logiciel décrit dans ce document est soumis à l'acceptation d'une Licence d'Utilisation et ne doit pas être copié sur d'autres supports. Aucune partie de ce manuel ne peut être copiée, reproduite, transférée ou enregistrée, sous quelque forme que ce soit et pour quelque usage que ce soit, sans l'accord écrit explicite de Native Instruments Software Synthesis GmbH. Tous les noms de produits et d'entreprises sont des marques déposées par leurs propriétaires respectifs.

En outre, le fait que vous lisiez ce texte signifie que vous êtes propriétaire d'une version légale plutôt que d'une copie illégalement piratée. C'est grâce à l'honnêteté et à la loyauté de personnes comme vous que NATIVE INSTRUMENTS GmbH peut continuer à créer et à développer des logiciels audio innovants. Nous vous en remercions au nom de la société toute entière.

Manuel d'utilisation écrit par: NATIVE INSTRUMENTS et Len Sasso.

Remerciements spéciaux à l'équipe de bêta-testeurs, dont l'aide nous fut précieuse non seulement pour trouver et corriger les bogues, mais aussi pour rendre ce produit encore meilleur.



NATIVE INSTRUMENTS

© NATIVE INSTRUMENTS GmbH, 2007. Tous droits réservés.

Allemagne

NATIVE INSTRUMENTS GmbH

Schlesische Str. 28-30

D-10997 Berlin

Germany

info@native-instruments.de

www.native-instruments.de

États-Unis

NATIVE INSTRUMENTS North America, Inc.

5631 A Hollywood Boulevard

Los Angeles, CA 90028

USA

info@native-instruments.com

www.native-instruments.com

Table des matières

Manuel de référence des modules.....	19
Panneaux.....	21
Fader.....	21
Knob	24
Button	24
List.....	25
Switch	27
Lamp.....	28
Level Lamp	29
RGB Lamp	29
Meter.....	30
Level Meter	30
Picture.....	31
Multi Picture.....	31
Text.....	32
Multi Text.....	33
XY	33
Scope	34
Multi Display et Poly Display	36
Mouse Area	39
Stacked Macro	41
Snap Value Array.....	..
Auxiliary.....	..
MIDI In.....	42
Note Pitch.....	42
Pitchbend.....	42
Gate	43
Single Trig. Gate.....	43
Sel. Note Gate	44
On Velocity.....	44
Off Velocity	44
Controller	45
Ch. Aftertouch	45
Poly Aftertouch	46
Sel. Poly AT.....	46
Program Change.....	46
Start/Stop	47
1/96 Clock.....	47

Sync Clock	48
Song Pos	48
Channel Message	49
MIDI Out.....	50
Note Pitch/Gate	50
Pitchbend.....	50
Controller	51
Ch. Aftertouch	51
Poly Aftertouch	51
Sel. Poly AT.....	52
Program Change.....	52
Start/Stop	52
1/96 Clock	53
Song Pos	53
Channel Message	53
Math	55
Constant	55
Add	55
Subtract.....	56
Invert, -X.....	56
Multiply.....	56
$a * b + c$	57
Reciprocal $1/x$	57
Divide x/y	57
Modulo $x \% y$	58
Rectifier	58
Rect./Sign	58
Compare	59
Compare/Equal	59
Quantize	60
Expon. (A).....	60
Expon. (F).....	60
Log (A)	61
Log (F).....	61
Power x^y	61
Square Root	62
$1 / \text{Square Root}$	62
Sine.....	62
Sin/Cos	63
Arcsin	63
Arccos	64

Arctan.....	64
Signal Path	65
Selector/Scanner	65
Relay 1,2	66
Crossfade.....	66
Distributor/Panner	67
Stereo Pan	67
Amp/Mixer	68
Stereo Amp/Mixer.....	68
Oscillator	70
Sawtooth.....	70
Saw FM	70
Saw Sync	71
Saw Pulse	72
Bi-Saw.....	72
Triangle.....	73
Tri FM.....	73
Tri Sync	74
Tri/Par Symm	74
Parabol	75
Par FM	75
Par Sync	76
Par PWM.....	77
Sine.....	77
Sine FM	78
Sine Sync.....	78
Multi-Sine	79
Pulse	80
Pulse FM.....	81
Pulse Sync	81
Pulse 1-ramp.....	82
Pulse 2-ramp.....	83
Bi-Pulse.....	84
Impulse.....	84
Impulse FM	85
Impulse Sync.....	85
Multi-Step	86
4-Step	86
5-Step	87
6-Step	87
8-Step	87

Multi-Ramp	87
4-Ramp	87
5-Ramp	88
6-Ramp	88
8-Ramp	88
Ramp.....	88
Clock Oscillator	89
Noise.....	90
Random	90
Geiger.....	90
Echantillonneur	92
Sampler	94
Sampler FM.....	94
Sampler Loop	96
Grain Resynth	98
Grain Pitch Former	102
Grain Cloud	106
Beat Loop	108
Sample Lookup	110
Séquenceur	112
Sequencer.....	112
6-Step	112
8-Step	113
12-Step	113
16-Step	113
Multiplex16	114
LFO, Enveloppe	116
LFO	116
Slow Random	117
H-Env	117
HR-Env	118
D-Env	119
DR-Env	119
DSR-Env	120
DBDR-Env.....	121
DBDSR-Env.....	122
AD-Env	123
AR-Env	123
ADR-Env	124
ADSR-Env	125
ADBDR-Env.....	126

ADBDJR-Env	127
AHDSR - Env	128
AHDBDR - Env.....	129
4-Ramp	130
5-Ramp	131
6-Ramp	133
Filtres.....	135
HP/LP 1-Pole.....	135
HP/LP 1-Pole FM	136
Allpass 1-Pole.....	136
Multi 2-Pole	137
Multi 2-Pole FM	138
Multi/Notch 2-Pole	139
Multi/Notch 2-Pole FM.....	140
Multi/LP 4-Pole.....	141
Multi/LP 4-Pole FM	142
Multi/HP 4-Pole	143
Multi/HP 4-Pole FM	144
Pro-52 Filter.....	145
Ladder Filter.....	145
Ladder Filter FM	146
Peak EQ.....	147
Peak EQ FM	147
High Shelf EQ.....	148
High Shelf EQ FM	149
Low Shelf EQ.....	149
Low Shelf EQ FM	150
Differentiator	150
Integrator	151
Delay	152
Single Delay	152
Multi-Tap Delay	153
Diffuser Delay.....	154
Grain Delay.....	155
Grain Cloud Delay	156
Unit Delay	158
Audio Modifier	159
Saturator.....	159
Saturator 2.....	159
Clipper.....	160
Mod. Clipper.....	160

Mirror 1 Level	161
Mirror 2 Levels.....	161
Chopper	162
Shaper 1 BP.....	162
Shaper 2 BP.....	163
Shaper 3 BP.....	163
Shaper Parabolic.....	164
Shaper Cubic.....	165
Slew Limiter	165
Peak Detector	166
Sample & Hold	166
Frequency Divider.....	166
Audio Table	167
Event Processing.....	169
Accumulator	169
Counter.....	169
Randomizer	170
Frequency Divider.....	170
Ctrl. Shaper 1 BP.....	171
Ctrl. Shaper 2 BP.....	171
Ctrl. Shaper 3 BP.....	172
Logic AND.....	173
Logic OR.....	173
Logic EXOR.....	173
Logic NOT.....	174
Order	174
Iteration	175
Separator	175
Value	176
Merge	176
Step Filter	176
Router M->1	177
Router 1,2.....	177
Router 1->M	178
Timer.....	178
Hold	178
Event Table.....	179
Auxiliary.....	181
Tapedeck 1-Ch.....	181
Tapedeck 2-Ch.....	184
Audio Voice Combiner	185

Event V.C. All	185
Event V.C. Max.....	185
Event V.C. Min	185
A to E	186
A to E (Trig).....	186
A to E (Perm).....	186
A to Gate.....	187
To Voice	187
From Voice	188
Voice Shift	188
Audio Smoother	189
Event Smoother	189
Master Tune/Level	190
Tempo Info.....	190
Voice Info.....	190
Tuning Info	191
System Info	191
Note Range Info.....	192
MIDI Channel Info	192
Snapshot.....	193
Set Random	195
Unison Spread	195
Snap Value	195
Snap Value Array.....	196
Terminals	198
In Port.....	198
Out Port.....	198
Send.....	198
Receive	198
IC Send	200
IC Receive.....	200
OSC Send	201
OSC Receive.....	201
Annexe	202
Premiers pas avec Reaktor Core.....	203
Qu'est-ce que Reaktor Core ?.....	203
Utilisation des cellules core	204
Utilisation des cellules core dans un exemple réel	207
Édition basique des cellules core	209
Pénétrer dans l'environnement Reaktor Core	215
Cellules core évènement et audio	215

Création de votre première cellule core	217
Les signaux audio et les signaux de contrôle.....	229
Utiliser l'audio comme signal de contrôle	243
Les signaux évènements.....	245
Les signaux logiques.....	249
Fondements de Reaktor Core: le modèle du signal core	252
Les valeurs	252
Les évènements	252
Évènements simultanés.....	255
L'ordre de traitement.....	257
Retour aux cellules core évènements	258
Les structures avec état interne	264
Les signaux d'horloge	264
Les Object Bus Connections (OBC).....	265
Initialisation.....	269
Construire un accumulateur d'évènements.....	271
Le mélange d'évènements	273
L'accumulateur d'évènements avec reset et initialisation	275
Réparons le modeleur d'évènements.....	281
Le traitement de l'audio en détail	284
Les signaux audio.....	284
Le bus d'horloge du taux d'échantillonnage	286
Réinjection d'une connexion	287
Réinjection et macros	291
Les valeurs dénormales.....	295
Les autres mauvais nombres.....	300
Construction d'un filtre passe-bas à 1 pôle.....	301
Le traitement conditionnel	304
Le routage des évènements	304
Construction d'un limiteur de signal.....	307
Construction d'un oscillateur en dents de scie simple	308
Les autres types de signaux	310
Les signaux flottants.....	310
Les signaux entiers.....	312
Construction d'un compteur d'évènements	315
Building a rising edge counter macro	316
Les tableaux.....	320
Introduction aux tableaux	320
Construction d'un détecteur de signal audio	323
Construction d'un délai	330
Les tables	336

Construction de structures optimisées.....	341
Les Latches et les macros à modulation.....	341
Routage et mélange.....	342
Les opérations numériques.....	343
Conversions entre flottants et entiers.....	344
Annexe A. Interface utilisateur de Reaktor Core.....	346
A.1. Les cellules core.....	346
A.2. Les macros et modules core.....	346
A.3. Les ports core.....	347
A.4. Modification de la structure core.....	347
Annexe B. Le concept de Reaktor Core.....	348
B.1. Les signaux et les évènements.....	348
B.2. L'initialisation.....	349
B.3. Les connexions OBC.....	349
B.4. Le routage.....	349
B.5. Le "latching".....	349
B.6. Les horloges.....	350
Annexe C. Les ports des macros core.....	350
C.1. In.....	350
C.2. Out.....	350
C.3. Latch (entrée).....	351
C.4. Latch (sortie).....	351
C.5. Bool C (entrée).....	351
C.6. Bool C (sortie).....	351
Annexe D. Les ports des cellules core.....	352
D.1. In (mode audio).....	352
D.2. Out (mode audio).....	352
D.3. In (mode évènement).....	352
D.4. Out (mode évènement).....	352
Annexe E. Les bus intégrés.....	353
E.1. SR.C.....	353
E.2. SR.R.....	353
Annexe F. Les modules d'usine.....	353
F.1. Const.....	353
F.2. Math > +.....	354
F.3. Math > -.....	354
F.4. Math > *.....	354
F.5. Math > /.....	354
F.6. Math > x	355
F.7. Math > -x.....	355

F.8. Math > DN Cancel	355
F.9. Math > ~log	355
F.10. Math > ~exp	356
F.11. Bit > Bit AND	356
F.12. Bit > Bit OR	356
F.13. Bit > Bit XOR	356
F.15. Bit > Bit <<	357
F.16. Bit > Bit >>	357
F.17. Flow > Router	357
F.18. Flow > Compare	358
F.19. Flow > Compare Sign	358
F.20. Flow > ES Ctl	359
F.21. Flow > ~BoolCtl	359
F.22. Flow > Merge	359
F.23. Flow > EvtMerge	359
F.24. Memory > Read	360
F.25. Memory > Write	360
F.26. Memory > R/W Order	360
F.27. Memory > Array	361
F.28. Memory > Size []	361
F.29. Memory > Index	361
F.30. Memory > Table	362
F.31. Macro	362
Annexe G. Les macros expertes	363
G.1. Clipping > Clip Max / IClip Max	363
G.2. Clipping > Clip Min / IClip Min	363
G.3. Clipping > Clip MinMax / IClipMinMax	363
G.4. Math > 1 div x	363
G.5. Math > 1 wrap	363
G.6. Math > lmod	364
G.7. Math > Max / lMax	364
G.8. Math > Min / lMin	364
G.9. Math > round	364
G.10. Math > sign +-	364
G.11. Math > sqrt (>0)	365
G.13. Math > $x(>0)^y$	365
G.14. Math > $x^2 / x^3 / x^4$	365
G.15. Math > Chain Add / Chain Mult	365
G.16. Math > Trig-Hyp > 2 pi wrap	365
G.17. Math > Trig-Hyp > arcsin / arccos / arctan	366
G.18. Math > Trig-Hyp > sin / cos / tan	366

G.19. Math > Trig-Hyp > sin $-\pi..pi$ / cos $-\pi..pi$ / tan $-\pi..pi$	366
G.20. Math > Trig-Hyp > tan $-\pi4..pi4$	366
G.21. Math > Trig-Hyp > sinh / cosh / tanh.....	366
G.22. Memory > Latch / lLatch.....	366
G.23. Memory > z^{-1} / z^{-1} ndc.....	367
G.24. Memory > Read []	367
G.25. Memory > Write []	367
G.26. Modulation > $x + a$ / Integer > $lx + a$	368
G.27. Modulation > $x * a$ / Integer > $lx * a$	368
G.28. Modulation > $x - a$ / Integer > $lx - a$	368
G.29. Modulation > $a - x$ / Integer > $la - x$	368
G.30. Modulation > x / a	368
G.31. Modulation > a / x	369
G.32. Modulation > $xa + y$	369
Annexe H. Les macros standard.....	369
H.1. Audio Mix-Amp > Amount	369
H.2. Audio Mix-Amp > Amp Mod	369
H.3. Audio Mix-Amp > Audio Mix	370
H.4. Audio Mix-Amp > Audio Relay	370
H.5. Audio Mix-Amp > Chain (amount).....	370
H.6. Audio Mix-Amp > Chain (dB)	371
H.7. Audio Mix-Amp > Gain (dB)	371
H.8. Audio Mix-Amp > Invert.....	371
H.9. Audio Mix-Amp > Mixer 2 ... 4	371
H.10. Audio Mix-Amp > Pan	372
H.11. Audio Mix-Amp > Ring-Amp Mod	372
H.13. Audio Mix-Amp > Stereo Mixer 2 ... 4	373
H.14. Audio Mix-Amp > VCA.....	373
H.15. Audio Mix-Amp > XFade (lin)	373
H.16. Audio Mix-Amp > XFade (par)	374
H.17. Audio Shaper > 1+2+3 Shaper	374
H.18. Audio Shaper > 3-1-2 Shaper	374
H.19. Audio Shaper > Broken Par Sat.....	375
H.20. Audio Shaper > Hyperbol Sat	375
H.21. Audio Shaper > Parabol Sat.....	375
H.22. Audio Shaper > Sine Shaper 4 / 8.....	376
H.23. Control > Ctl Amount.....	376
H.24. Control > Ctl Amp Mod.....	376
H.25. Control > Ctl Bi2Uni.....	376
H.26. Control > Ctl Chain.....	377
H.27. Control > Ctl Invert.....	377

H.28. Control > Ctl Mix	377
H.29. Control > Ctl Mixer 2	377
H.30. Control > Ctl Pan	378
H.31. Control > Ctl Relay	378
H.32. Control > Ctl XFade	378
H.33. Control > Par Ctl Shaper	378
H.34. Convert > dB2AF	379
H.35. Convert > dP2FF	379
H.36. Convert > logT2sec	379
H.37. Convert > ms2Hz	379
H.38. Convert > ms2sec	380
H.39. Convert > P2F	380
H.40. Convert > sec2Hz	380
H.41. Delay > 2 / 4 Tap Delay 4p	380
H.42. Delay > Delay 1p / 2p / 4p	380
H.43. Delay > Diff Delay 1p / 2p / 4p	381
H.44. Envelope > ADSR	381
H.45. Envelope > Env Follower	382
H.46. Envelope > Peak Detector	382
H.47. EQ > 6dB LP/HP EQ	382
H.48. EQ > 6dB LowShelf EQ	383
H.49. EQ > 6dB HighShelf EQ	383
H.50. EQ > Peak EQ	383
H.51. EQ > Static Filter > 1-pole static HP	383
H.52. EQ > Static Filter > 1-pole static HS	384
H.53. EQ > Static Filter > 1-pole static LP	384
H.54. EQ > Static Filter > 1-pole static LS	384
H.55. EQ > Static Filter > 2-pole static AP	384
H.56. EQ > Static Filter > 2-pole static BP	384
H.57. EQ > Static Filter > 2-pole static BP1	385
H.58. EQ > Static Filter > 2-pole static HP	385
H.59. EQ > Static Filter > 2-pole static HS	385
H.60. EQ > Static Filter > 2-pole static LP	385
H.61. EQ > Static Filter > 2-pole static LS	386
H.62. EQ > Static Filter > 2-pole static N	386
H.63. EQ > Static Filter > 2-pole static Pk	386
H.64. EQ > Static Filter > Integrator	386
H.65. Event Processing > Accumulator	387
H.66. Event Processing > Clk Div	387
H.67. Event Processing > Clk Gen	387
H.68. Event Processing > Clk Rate	387

H.69. Event Processing > Counter.....	388
H.70. Event Processing > Ctl2Gate	388
H.71. Event Processing > Dup Flt / IDup Flt	388
H.72. Event Processing > Impulse.....	388
H.74. Event Processing > Separator / ISeparator	389
H.75. Event Processing > Thld Crossing.....	389
H.76. Event Processing > Value / IValue.....	389
H.77. LFO > MultiWave LFO	389
H.78. LFO > Par LFO	390
H.79. LFO > Random LFO.....	390
H.80. LFO > Rect LFO.....	390
H.81. LFO > Saw(down) LFO	391
H.82. LFO > Saw(up) LFO	391
H.83. LFO > Sine LFO.....	391
H.84. LFO > Tri LFO	391
H.85. Logic > AND	392
H.86. Logic > Flip Flop.....	392
H.87. Logic > Gate2L	392
H.88. Logic > GT / IGT	392
H.89. Logic > EQ.....	392
H.90. Logic > GE	392
H.91. Logic > L2Clock.....	393
H.92. Logic > L2Gate.....	393
H.94. Logic > OR.....	393
H.95. Logic > XOR	393
H.97. Oscillators > 4-Wave Mst.....	394
H.98. Oscillators > 4-Wave Slv	394
H.99. Oscillators > Binary Noise	395
H.100. Oscillators > Digital Noise	395
H.101. Oscillators > FM Op	395
H.102. Oscillators > Formant Osc	395
H.103. Oscillators > MultiWave Osc.....	396
H.104. Oscillators > Par Osc	396
H.105. Oscillators > Quad Osc.....	396
H.106. Oscillators > Sin Osc	396
H.107. Oscillators > Sub Osc 4	396
H.108. VCF > 2 Pole SV	397
H.109. VCF > 2 Pole SV C.....	397
H.110. VCF > 2 Pole SV (x3) S	397
H.111. VCF > 2 Pole SV T (S)	398
H.112. VCF > Diode Ladder.....	398

H.113. VCF > D/T Ladder.....	398
H.114. VCF > Ladder x3	399
Annexe I. Core cell library	400
I.1. Audio Shaper > 3-1-2 Shaper.....	400
I.2. Audio Shaper > Broken Par Sat.....	400
I.3. Audio Shaper > Hyperbol Sat.....	400
I.4. Audio Shaper > Parabol Sat.....	401
I.5. Audio Shaper > Sine Shaper 4/8	401
I.6. Control > ADSR	401
I.7. Control > Env Follower.....	402
I.8. Control > Flip Flop	402
I.9. Control > MultiWave LFO.....	403
I.10. Control > Par Ctl Shaper.....	403
I.11. Control > Schmitt Trigger.....	403
I.12. Control > Sine LFO	404
I.13. Delay > 2/4 Tap Delay 4p	404
I.14. Delay > Delay 4p	404
I.15. Delay > Diff Delay 4p.....	404
I.16. EQ > 6dB LP/HP EQ	405
I.17. EQ > HighShelf EQ.....	405
I.18. EQ > LowShelf EQ	405
I.19. EQ > Peak EQ	405
I.20. EQ > Static Filter > 1-pole static HP.....	406
I.21. EQ > Static Filter > 1-pole static HS.....	406
I.22. EQ > Static Filter > 1-pole static LP	406
I.23. EQ > Static Filter > 1-pole static LS.....	406
I.24. EQ > Static Filter > 2-pole static AP.....	407
I.25. EQ > Static Filter > 2-pole static BP.....	407
I.26. EQ > Static Filter > 2-pole static BP1.....	407
I.27. EQ > Static Filter > 2-pole static HP.....	407
I.28. EQ > Static Filter > 2-pole static HS	408
I.29. EQ > Static Filter > 2-pole static LP.....	408
I.30. EQ > Static Filter > 2-pole static LS.....	408
I.31. EQ > Static Filter > 2-pole static N.....	408
I.32. EQ > Static Filter > 2-pole static Pk	409
I.33. Oscillator > 4-Wave Mst	409
I.34. Oscillator > 4-Wave Slv	409
I.35. Oscillator > Digital Noise.....	410
I.36. Oscillator > FM Op	410
I.37. Oscillator > Formant Osc	410
I.38. Oscillator > Impulse	411

I.39. Oscillator > MultiWave Osc	411
I.40. Oscillator > Quad Osc	411
I.41. Oscillator > Sub Osc	412
I.42. VCF > 2 Pole SV C.....	412
I.43. VCF > 2 Pole SV T.....	412
I.44. VCF > 2 Pole SV x3 S.....	413
I.45. VCF > Diode Ladder	413
I.46. VCF > D/T Ladder	414
I.47. VCF > Ladder x3.....	414
Index.....	415

Manuel de référence des modules

Les modules sont les composants les plus élémentaires d'un ensemble REAKTOR. Cette partie du manuel est avant tout destinée aux utilisateurs de REAKTOR qui souhaitent construire entièrement leurs ensembles. Si ce n'est pas votre cas, ou si vous pensez que votre expérience en la matière est insuffisante, REAKTOR vous permet d'exploiter ses qualités différemment :

- Si l'élaboration de vos propres ensembles ne vous intéresse en rien, travaillez au niveau Ensemble.
- Si vous regroupez vos instruments préférés dans un ensemble et souhaitez les utiliser conjointement, travaillez au niveau Instrument.
- Si vous souhaitez assembler vos propres instruments avec des éléments préfabriqués, travaillez au niveau Macro.
- Si vous souhaitez garder le contrôle des différents paramètres d'un ensemble, travaillez au niveau Module.

Ce manuel de référence présente tous les modules de REAKTOR et les décrit en détail. Leur description contient les réglages des propriétés, dans la mesure de leur disponibilité, et une liste des ports d'entrée et de sortie des modules.

Un champ Label existe dans les propriétés de tous les objets de REAKTOR (modules, macros, instruments et ensembles). Cette vignette apparaît également sur l'icône de l'objet, dans la fenêtre de structure. Dans le réglage standard, la vignette représente également la fonction du module. Modifiez donc le nom des modules avec prudence, en particulier si vous souhaitez que d'autres utilisateurs de REAKTOR comprennent la structure que vous avez créée.

Modules hybrides

Un certain nombre de modules de REAKTOR sont des **hybrides**. Une fois ajouté, un tel module apparaît d'abord comme un module traitant des événements, ce qui est signalé par une vignette rouge. Un point vert sur un port indique que vous pouvez y connecter soit un câble audio, soit un câble d'événement.

Lorsque vous connectez un câble audio à l'un des ports d'entrée, le module se transforme en un module traitant les signaux audio, ce qui est indiqué par des points et vignettes noirs du port correspondant. Si vous y connectez un câble d'événement, le point vert du port se transforme en point rouge.

Un module hybride présente les caractéristiques suivantes :

- Si vous connectez à la fois des câbles audio et d'événements aux entrées, ou seulement des câbles audio, le module fonctionne au taux audio.
- Si vous connectez uniquement des câbles d'événements, le module fonctionne au taux d'événement.
- La connexion d'une sortie de module à une entrée d'événement d'un autre module le fait devenir automatiquement un module de traitement d'événements, il vous est impossible de connecter des câbles audio à ses entrées.
- Lorsque vous connectez la sortie avec l'entrée audio d'un autre module, il fonctionne soit au taux audio, soit au taux d'événement, selon les câbles connectés à ses entrées.
- Lorsque, à cause des connexions de ses entrées, le module fonctionne au taux audio, il est impossible d'en connecter la sortie à l'entrée d'événements d'un autre module.

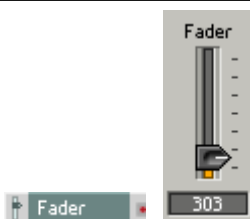
Gestion dynamique des ports

Les modules qui le requièrent sont équipés d'une fonction de gestion **dynamique** des ports d'entrée et/ou de sortie. Pour ajouter des entrées au module, tirez des câbles supplémentaires dans la zone de ses ports tout en maintenant la touche **Ctrl** appuyée. Pour visualiser l'endroit où le port est ajouté, maintenez le pointeur sur le module cible. La position verticale du pointeur détermine l'emplacement du nouveau port, vous permettant d'en ajouter entre ceux qui existent déjà.

Panneaux

Les modules panneaux fournissent les éléments de commande des instruments de REAKTOR. Vous les positionnez et les visualisez dans les panneaux A et B indépendamment les uns des autres. Certains modules panneaux servent à afficher des valeurs. Lamps, Meters et Scopes, par exemple, visualisent des processus de Reaktor, et les modules graphiques et de texte permettent d'embellir l'interface utilisateur. Les autres modules créent ou assignent les données de traitement des signaux dans REAKTOR. Ce sont par exemple des atténuateurs, des boutons, des touches, des commutateurs, des menus et un champ de commande XY (pouvant faire office d'affichage et d'élément de commande).

Fader



Panel

Le curseur de ce panneau permet de régler la valeur que le module émet comme Audio et Event dans la structure. Le signal est monophonique, et donc toutes les voix d'une entrée polyphonique auront la même valeur.

Page Properties - Function

Plage de valeur (**Range**) : selon la position du bouton, le signal de sortie est compris entre **Min** et **Max**. Le paramètre **Step** sert à régler la valeur du pas avec lequel les valeurs affichées et émises sont quantifiées.

Ou alors, vous pouvez utiliser le champ Num Steps pour régler la résolution de l'atténuateur. Les valeurs des deux atténuateurs, Stepsize et Num Steps, sont fonction l'une de l'autre. La modification de l'une des valeurs entraîne donc celle de l'autre. Stepsize contient une plage de valeur par pas, alors que Num Steps contient le nombre de pas de la plage de valeurs de l'atténuateur.

La résolution maximum d'un atténuateur est de 127 000 pas, que vous obtenez en entrant 0 dans **Stepsize** ou 127 000 dans **Num Step**.

Vous pouvez définir, dans **Num Steps**, une résolution supérieure à la résolution MIDI standard de 128. Gardez à l'esprit que REAKTOR est certes en mesure de calculer en interne une résolution supérieure, mais ne peut échanger des données avec du matériel ou des logiciels externes que dans la résolution MIDI. Dans certaines circonstances, il se peut que la fonctionnalité ou la sonorité d'un ensemble diffère selon les environnements d'utilisation. En règle générale, ceci ne pose aucun problème car la résolution MIDI est largement suffisante pour contrôler les paramètres. Mais, dans certains cas (par ex. lorsque vous utilisez un filtre à haute résonance et modifiez la fréquence limite), vous souhaitez une résolution plus élevée, et REAKTOR vous en donne la possibilité.

Mouse Res permet de définir la distance, en pixel, que le pointeur parcourt entre les valeurs maximum et minimum de l'atténuateur. Par exemple, si vous entrez dans **Mouse Res** une valeur deux fois supérieure à la valeur en pixel du champ **Y** de la page **Apperance** des propriétés, il faudra déplacer la souris de deux fois la longueur de l'atténuateur pour passer du minimum au maximum.

Si **Num Steps** est supérieur à **Mouse Res**, il est impossible d'atteindre tous les pas avec la souris. Au contraire, lorsque **Mouse Res** est supérieur à **Num Steps**, vous pouvez régler le nombre de pixel par pas sur une valeur supérieure à 1.

La valeur **Default** est utilisée dans tous les cas d'initialisation de la commande. Ce sont les situations suivantes :

- Actionnez la touche Default de la fenêtre snapshot réinitialise tous les éléments de commande de l'Ensemble/Instrument.
- Lorsque « default » s'affiche dans la fenêtre snapshot comme étant l'un des buts du morphing, vous pouvez morpher entre un snapshot et les réglages par défaut de tous les éléments de commande.
- Si vous ajoutez un élément de commande à un instrument contenant déjà une liste de snapshots, la valeur par défaut du nouvel élément est utilisée dans tous ceux-ci jusqu'à ce que vous affectiez une nouvelle valeur aux snapshots ou activiez **Snap Isolate**.

Lorsque **Snap Isolate** est activé, l'atténuateur ne réagit pas aux appels de snapshots.

REAKTOR utilise le numéro **ID** pour gérer les snapshots. Les numéros ID s'inscrivent toujours automatiquement lorsque vous ajoutez un élément de commande. Si vous les modifiez (par exemple pour importer des snapshots depuis un instrument aux éléments de commande similaires), les snapshots déjà existants ne reconnaissent plus, et donc ignorent le panneau. Donc méfiez-vous, modifiez le numéro ID en toute connaissance de cause.

Page Properties - Info (infos propriétés)

Vous pouvez entrer un texte d'information expliquant la fonction dans **Info**. Ce texte est une aide rapide (Hint) qui s'affiche tant que le pointeur repose sur l'atténuateur, lorsque la fonction Show-Hints est activée.

Page Properties - Appearance

La vignette que vous entrez dans cette page est visible, au-dessus de l'atténuateur, uniquement lorsque **Label** est activé, dans la section **Visible**. La valeur réglée, de la même manière, s'affiche sous forme numérique, en dessous de l'atténuateur, uniquement lorsque **Value** est activé. Il est également possible de masquer le bitmap de l'atténuateur, pour que seul le champ **Value** soit visible. Vous pouvez alors continuer à l'utiliser en cliquant, dans le panneau, sur le champ Value, puis en déplaçant la souris vers le haut ou le bas.

Les atténuateurs sont représentés à l'horizontale ou à la verticale. Cochez le champ correspondant de la section **Form**, afin d'obtenir l'apparence souhaitée.

Vous pouvez régler la longueur de l'atténuateur dans le champ **Pixel in Y** de la section **Size**. Les options **Big**, **Medium** et **Small** vous permettent d'en définir la largeur.

Page Properties - Connection

La page Connection des propriétés des éléments de commande est présentée en détail dans la Section *Connection Properties des éléments de commande*, à partir de la page - 138.



Similaire à l'atténuateur, mais représenté comme régleur rotatif (bouton).



Ces touches à pression servent à sélectionner la valeur Event et Audio que le module émet dans la structure. Le signal est monophonique, et donc toutes les voix d'une entrée polyphonique auront la même valeur.

Page Properties - Function

Range : définissez les valeurs émises lorsque vous appuyez sur la touche et lorsque vous la relâchez dans les champs **On Value** et **Off Value**.

Mode : vous pouvez choisir entre les modes de fonctionnement **Trigger**, **Gate** et **Toggle**. En mode Trigger, un événement est émis uniquement lorsque vous enfoncez la touche. En mode Gate, un événement de valeur 0 est émis en plus lorsque vous relâchez la touche. En mode Toggle, le logiciel permute entre les deux états à chaque fois que vous appuyez sur la touche.

Default = On : règle la valeur par défaut utilisée par REAKTOR dans différentes circonstances sur On.

Activer le commutateur **Snap Isolate** empêche la touche de réagir aux appels de snapshots.

Activer le commutateur **Random Isolate** empêche la touche de réagir à la fonction aléatoire des snapshots.

Page Properties - Info (infos propriétés)

Vous pouvez entrer un texte d'information expliquant la fonction de la touche dans **Info**. Ce texte est une aide rapide (Hint) qui s'affiche tant que le pointeur repose sur la touche, lorsque la fonction Show Hints est activée.

Page Properties - Appearance

La vignette apparaît dans le panneau, au-dessus de la touche, lorsque **Label** est activé dans le dialogue Properties. La valeur réglée, de la même manière, s'affiche en dessous de la touche uniquement lorsque **Value** est activé dans Properties.

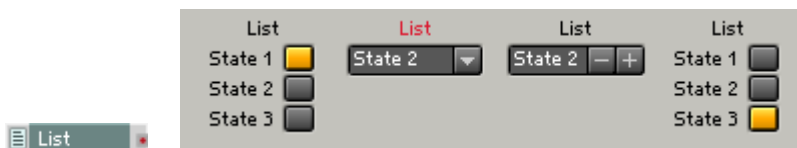
Vous avez trois tailles au choix pour la touche : **Small**, **Medium** et **Big**.

Page Properties - Connection

La page Connection des propriétés des éléments de commande est présentée en détail dans la Section *Connection Properties des éléments de commande*, à partir de la page - 138.

List

Panel



Le module List permet de concevoir les listes des panneaux, les menus déroulants, les textes des touches et les affichages déroulants.

Page Properties - Function

La page **Function** des propriétés contient une liste dans laquelle vous pouvez joindre (**Append**), insérer (**Insert**) et effacer (**Delete**) des entrées. Vous pouvez également définir le nombre d'entrées par une valeur numérique. Les différentes entrées de la liste sont représentées dans le panneau en fonction du style d'affichage choisi. Vous pouvez saisir une valeur pour chaque entrée, dont la sélection provoque l'envoi à la sortie du module.

La page **Function** contient un générateur de valeur. Cet outil vous permet de générer simultanément les valeurs de plusieurs éléments d'une liste. Exemple : vous souhaitez un pas de 4 au lieu de 1 entre les différentes entrées. Pour ce faire, entrez « 4 » dans **Stepsize**, dans le générateur Value, puis appuyez sur la touche **Apply**. Les valeurs situées dans la colonne **Value** de la liste sont modifiées en fonction des réglages du générateur.

Le réglage **Mouse Resolution** a de l'importance uniquement lorsque, pour l'élément de commande, le style **Spin** est sélectionné dans la page **Appearance**, ce qui vous permet de cliquer sur l'entrée de cet élément, puis de déplacer la souris vers le haut ou vers le bas pour la modifier.

Page Properties - Appearance

La représentation du panneau de l'élément de commande se modifie en fonction du style sélectionné dans les propriétés, à la page **Appearance**. Les styles suivants sont disponibles :

- **Button** : chaque port d'entrée d'un module génère une touche. Toutes les touches sont ordonnancées verticalement dans le panneau d'instrument, formant une barre. L'entrée sélectionnée est représentée de la couleur caractéristique de l'instrument.
- **Menu** : chaque port d'entrée du module génère une nouvelle entrée dans une liste déroulante.
- **Text Panel** : chaque port d'entrée du module génère une nouvelle entrée dans la liste, qui en affiche plusieurs simultanément. Si vous générez plus d'entrées que l'affichage, dont vous définissez la taille à l'aide des champs **Size X** et **Size Y** de la page **Appearance** des propriétés, ne peut en contenir, il se munit de barres de déroulement.
- **Spin** : chaque port d'entrée du module génère une nouvelle entrée dans une liste. Les touches **+** et **-** situées à droite de l'affichage vous permettent de naviguer parmi les entrées.

Les champs **Size X** et **Size Y** servent à régler la taille de l'affichage de l'élément de commande dans le panneau.

Ports

- **Out** : sortie d'événement pour la valeur correspondant à l'entrée sélectionnée.



Il sert à commuter le parcours des signaux. Il ne sert pas à proprement parler au traitement des signaux, il se contente d'établir la liaison entre des modules. Appuyer sur une touche ou sélectionner une entrée de la liste relie la sortie à l'entrée ainsi sélectionnée. Tous les autres ports d'entrée sont désactivés.

Les modules dont les sorties sont libres sont désactivés automatiquement afin d'éviter toute contrainte inutile du processeur. Le voyant d'état des modules désactivés est éteint. Le module est hybride, donc en mesure, en fonction du câblage, de traiter des signaux audio ou événements, et il présente une gestion dynamique des entrées.

Page Properties - Function

Enable Switch Off : Lorsque cette option est activée, le commutateur peut prendre l'état dans lequel aucune entrée n'est sélectionnée. Lorsque le style **Button** est sélectionné, pour l'élément de commande, sur la page **Appearance**, un deuxième clic sur le commutateur sélectionné vous permet de désactiver toutes les entrées. Dans tous les autres styles d'affichage, vous disposez du point « Off » en plus. Le module dispose d'une gestion dynamique des ports d'entrée. Définissez le nombre d'entrées en utilisant **Min Num Port Groups** de la page **Function** des propriétés.

Le réglage **Mouse Resolution** a de l'importance uniquement lorsque, pour l'élément de commande, le style **Spin** est sélectionné dans la page **Appearance**, qui vous permet de cliquer sur l'entrée de cet élément, et de déplacer la souris vers le haut ou vers le bas pour la modifier.

Page Properties - Appearance

La vignette apparaît dans le panneau, au-dessus du commutateur, lorsque **Label** est activé dans les propriétés. L'élément de commande peut être représenté en trois tailles : **Small**, **Medium** et **Big**.

Si vous utilisez un commutateur à seulement deux entrées, seule une touche (celle d'en haut) s'affiche dans le panneau, en style **Button**, lorsque l'option **1 Toggle Button** est activée : lorsque la touche est activée, l'entrée 1 est sélectionnée, si elle est désactivée, l'entrée 2.

La représentation du panneau de l'élément de commande se modifie en fonction du style sélectionné dans les propriétés, à la page **Appearance**. Les styles suivants sont disponibles :

- **Button** : chaque port d'entrée d'un module génère une touche. Elles sont ordonnancées verticalement dans le panneau d'instrument, formant une barre. L'entrée sélectionnée est représentée dans la couleur caractéristique de l'instrument.
- **Menu** : chaque port d'entrée du module génère un nouveau point dans une liste déroulante.
- **Text Panel** : chaque port d'entrée du module génère une nouvelle entrée dans la liste, qui en affiche plusieurs simultanément. Si vous générez plus d'entrées que l'affichage, dont vous définissez la taille à l'aide des champs **Size X** et **Size Y**, à la page **Appearance** des propriétés, ne peut en contenir, il se munit de barres de déroulement.
- **Spin** : chaque port d'entrée du module génère une nouvelle entrée dans une liste. Les touches **+** et **-** situées à droite de l'affichage vous permettent de naviguer parmi les entrées.

Lamp



Panel

Le voyant est allumé tant que le signal d'entrée (échantillonné à 25 Hz) se trouve dans la plage définie, dans le dialogue Properties, via **Min** et **Max**, donc tant que la valeur est supérieure à **Min** et inférieure ou égale à **Max**. Lorsque vous avez activé le mode **Continuous**, dans les propriétés, la couleur du voyant s'intensifie et s'atténue entre les valeurs minimum et maximum.

Vous pouvez choisir la couleur du voyant dans les propriétés (rouge, vert, bleu, jaune ou couleur caractéristique). L'option **Indicator Color** permet de donner la couleur Indicator au voyant définie dans les propriétés de l'instrument.

Pour définir des couleurs spécifiques pour les positions on et off du voyant, utilisez les touches **Set On Color** et **Set Off Color**. L'option **Has Frame** non activée, vous pouvez positionner des voyants dans le panneau, les uns à côté des autres, sans intervalle et au pixel près.

La vignette apparaît dans le panneau, au-dessus du voyant, uniquement lorsque **Label** a été activé dans les propriétés.

Level Lamp

Panel



Le voyant du panneau s'allume dès que le niveau sonore, à l'entrée, est compris dans la plage réglée avec **Min** et **Max** (en dB) dans les propriétés, donc lorsqu'il est supérieur à **Min** et inférieur ou égal à **Max**.

Lorsque vous avez activé le mode **Continuous**, dans les propriétés, la couleur du voyant s'intensifie et s'atténue entre les valeurs minimum et maximum.

Vous pouvez choisir la couleur du voyant dans les propriétés (rouge, vert, bleu, jaune ou couleur caractéristique). L'option **Indicator Color** permet de donner la couleur Indicator au voyant définie dans les propriétés de l'instrument.

Pour définir des couleurs spécifiques pour les positions on et off du voyant, utilisez les touches **Set On Color** et **Set Off Color**.

L'option **Has Frame** non activée, vous pouvez positionner des voyants dans le panneau, les uns à côté des autres, sans intervalle et au pixel près.

La vignette apparaît dans le panneau, au-dessus du voyant, uniquement lorsque **Label** a été activé dans les propriétés.

RGB Lamp

Panel

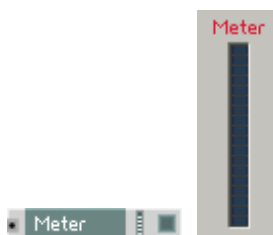


Ce module sert à générer un affichage de taille réglable, dont la couleur est le mélange des couleurs des trois entrées de module correspondantes. Réglez les dimensions horizontale et verticale en pixel, sur la page **Appearance** des propriétés.

- **R** : entrée audio pour l'intensité du rouge. Plage : 0 - 1.
- **G** : entrée audio pour l'intensité du vert. Plage : 0 - 1.
- **B** : entrée audio pour l'intensité du bleu. Plage : 0 - 1.

Meter

Panel



Le signal existant est échantillonné (à 25 Hz) et représenté sur une échelle linéaire. Réglez la plage représentée, dans les propriétés, avec **Min** et **Max**.

Vous pouvez choisir la couleur de la barre dans les propriétés (rouge, vert, bleu, jaune ou couleur caractéristique). L'option **Indicator Color** permet de donner la couleur Indicator au voyant définie dans les propriétés de l'instrument.

Pour définir des couleurs spécifiques pour les sections supérieure et inférieure de la barre, utilisez les touches **Set On Color** et **Set Off Color**.

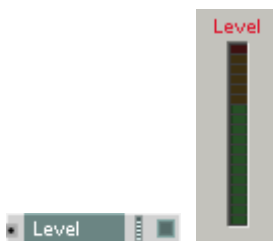
Number Of Segments définit le nombre d'éléments composant la barre.

Size X (segment) et **Size Y (segment)** servent à définir la taille d'un élément de la barre. La hauteur totale de la barre est égale à **Size Y (segment)** multiplié par **Number of Segments** (nombre de segments).

La vignette apparaît dans le panneau, au-dessus de la barre, lorsque **Label** est activé dans les propriétés.

Level Meter

Panel



L'amplitude du signal audio existant est représentée sur une échelle logarithmique. Réglez la plage représentée, dans les propriétés, avec **Min** et **Max**, en dB. Vous pouvez choisir la couleur de la barre dans les propriétés (rouge, vert, bleu, jaune ou couleur caractéristique). L'option **Indicator Color** permet de donner la couleur Indicator au voyant définie dans les propriétés de l'instrument.

Pour définir des couleurs spécifiques des sections supérieure et inférieure de la barre, utilisez les touches **Set On Color** et **Set Off Color**.

Number Of Segments définit le nombre d'éléments composant la barre.

Size X (segment) et **Size Y (segment)** servent à définir la taille d'un élément de la barre. La hauteur totale de la barre est égale à **Size Y (segment)** multiplié par **Number of Segments** (nombre de segments).

La vignette apparaît dans le panneau, au-dessus de la barre de niveau, lorsque **Label** est activé dans les propriétés.

Picture

Panel

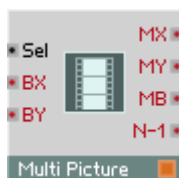


Permet d'importer un bitmap de décoration pour le panneau, en format TGA (extension *.tga). Le module bitmap ne présente ni entrée ni sortie. La taille de l'affichage se règle sur la taille originale du bitmap chargé. Vous pouvez mettre le cadre visible à l'échelle, en pixel, dans les propriétés. REAKTOR ne permet pas de mettre l'image à l'échelle, un logiciel de traitement d'images est nécessaire.

Activez l'option **Save bitmap with ensemble**, dans les propriétés, pour enregistrer les données d'image sous forme d'une partie d'un ensemble de REAKTOR.

Multi Picture

Panel



Le module Multi Picture est un élément de commande bi-dimensionnel (similaire au module XY) qui donne la position de la souris et l'état du bouton aux sorties du module. Ce module supporte une animation multi-frame lorsqu'elle est contenue dans une seule image. Pour effectuer l'animation, dans la fenêtre **Picture Properties**, entrez le nombre de frames (animations) et leur orientation (horizontale ou verticale).

Vous pouvez inclure des formats image 24 bit BMP et 32 bit Targa (non comprimé) à différents endroits de REAKTOR : comme arrière-plan des panneaux d'instruments et de macros, dans les icônes de structure des instruments et macros, dans les modules Picture et Multi-Picture.

Le format Targa présente l'avantage de supporter un canal alpha, qui peut servir de masque pour les zones visibles de l'image (les zones non masquées apparaissant transparentes). Ceci est utile par exemple pour créer des boutons sur un fond rectangulaire.

Le menu déroulant **Select Picture**, dans les propriétés de l'objet concerné, permet de charger des images. Ce faisant, la fenêtre **Picture Properties** s'ouvre automatiquement, et vous pouvez y procéder à tous les réglages importants de l'image. Toutes les images sont automatiquement partagées et disponibles pour tous les modules appropriés.

- **Sel** : entrée audio pour sélectionner la section d'image par un chiffre. Plage 0 - numéro du dernier frame.
- **MX** : sortie d'événement pour position horizontale de la souris (X) lorsque celle-ci se trouve dans la section visible de l'image.
- **MY** : sortie d'événement pour position verticale de la souris (Y) lorsque celle-ci se trouve dans la section visible de l'image.
- **MB** : sortie d'événement pour l'état du bouton de la souris (0=relâché, 1=appuyé)
- **N-1** : sortie d'événement pour le nombre de sections d'image défini dans Picture Properties (Num animations) -1.

Text

Panel



Le module de texte ne traite aucun signal, il sert uniquement à ajouter des textes explicatifs à la structure. Vous pouvez par ex. y noter le nom de l'auteur et la date de création, et expliquer le mode de fonctionnement.

Multi Text

Panel

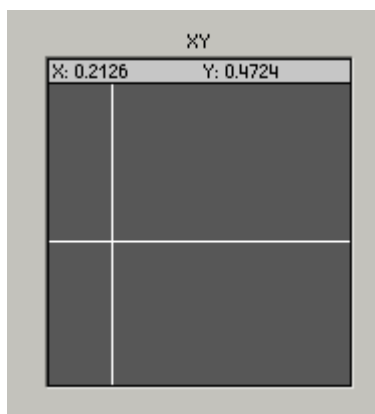


Le module Multi Text permet un affichage modifiable de texte dans le panneau de commande. Vous pouvez ajouter un nombre quelconque de textes dans les propriétés du module. Leur édition ou leur suppression y est également possible.

In : entrée audio pour le numéro du texte à afficher.

XY

Panel



Le champ de commande XY a deux fonctions : il affiche les signaux audio d'entrée et fait office d'élément bi-dimensionnel de commande des déplacements de la souris.

Page Properties - Function

L'option **Always Active** active la capacité du module à libérer le parcours du signal connecté à un des ports d'entrée du module.

Le mode **Incremental Mouse** permet à l'élément de commande de réagir comme un bouton. Vous pouvez alors cliquer à n'importe quel endroit du panneau d'affichage du module et déplacer la souris sans réinitialiser directement la valeur de sa position. Au contraire, la valeur suit le pointeur avec un décalage constant.

Page Properties - Appearance

Les propriétés du module servent à régler le type d'affichage destiné à la visualisation du signal audio. Les entrées **X1** et **Y1** pilotent la position de l'objet visible (**Pixel** ou **Cross**). Lorsqu'il s'agit de **Bar** ou de **Rectangle**, **X1** et **Y1** pilotent l'un des angles de l'objet visualisé, et **X2** et **Y2** son coin opposé.

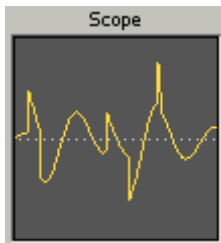
Pour afficher des données audio se modifiant rapidement, choisissez le mode **Scope**. Les autres modes évaluent l'entrée à un taux d'échantillonnage faible.

Vous pouvez également régler la taille de la croix qui symbolise la position du pointeur.

Tous ce qui s'affiche dans le champ s'évanouit plus ou moins rapidement en fonction de la valeur de réglage de **Fade Time**, dans les propriétés (valeur maximum : 99).

- **X1** :entrée audio pour la coordonnée **X1** de l'objet visualisé.
- **Y1** :entrée audio pour la coordonnée **Y1** de l'objet visualisé.
- **X2** :entrée audio pour la coordonnée X2 de l'objet visualisé.
- **Y2** :entrée audio pour la coordonnée Y2 de l'objet visualisé.
- **MX** :sortie d'événement pour la position X du pointeur lorsque vous appuyez sur le bouton de la souris, pointeur dans l'affichage.
- **MY** :sortie d'événement pour la position Y du pointeur lorsque vous appuyez sur le bouton de la souris, pointeur dans l'affichage.
- **MB** :état du bouton gauche de la souris (1=appuyé, 0=relâché).

Scope



Panel

Oscilloscope de représentation des signaux à modification rapide, en particulier audio.

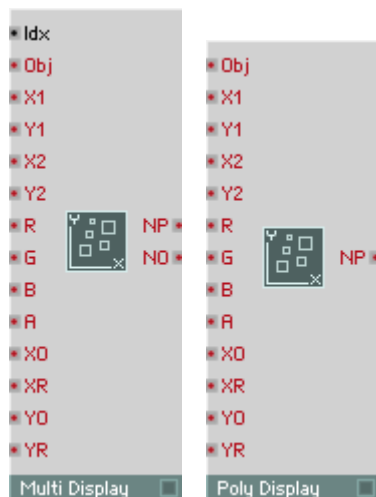
Chaque fois que l'entrée de déclenchement (**Trg**) reçoit un événement, le module commence à enregistrer le signal audio établi à l'entrée (**In**) et le représente dans le panneau, sous forme de courbe. En l'absence d'événement de déclenchement, le signal enregistré demeure affiché ; vous pouvez en modifier agrandissement et position via les valeurs des entrées de commande correspondantes.

Réglez la taille du graphique à la page **Appearance** des propriétés, avec **Pixel in X** et **Pixel in Y**.

A la page **Function**, réglez le tampon utilisé pour la représentation Scope, en ms.

En général, on connecte un module **A to E Trig** à l'entrée **Trg** pour synchroniser l'oscilloscope avec un signal audio.

- **Trg** : entrée d'événement mono pour le signal de déclenchement qui démarre la représentation.
- **TP** : entrée d'événement mono pour le décalage du temps (Time Position) de la courbe, en millisecondes. La courbe débute du côté gauche de l'affichage, **TP** ms après l'événement de déclenchement.
- **TS** : entrée d'événement mono pour l'échelle du temps (Time Scale) de la courbe. **TP** ms du signal est représenté du bord gauche au bord droit du graphique.
- **YP** : entrée d'événement mono du décalage de l'amplitude (Y Position) du graphique. **YP** = -1 correspond au bord inférieur de l'affichage, +1 est le bord supérieur.
- **YS** : entrée d'événement mono d'échelle de l'amplitude (Y Scale) de la courbe. La différence entre le signal au bord supérieur et le signal au bord inférieur est 2 **YS**. Lorsque **YP** = 0, les valeurs affichées sont comprises entre +**YS** et -**YS**.
- **In** : entrée audio mono pour le signal à représenter.



Les modules Multi Display et Poly Display permettent l’affichage et la manipulation de plusieurs objets graphiques (croix, barres, images, animations, etc.). Une série de paramètres (type, position, taille et couleur) peuvent être définis individuellement pour chaque objet graphique.

La principale différence entre les deux modules est que, pour le module Multi Display, le nombre d’objets graphiques est spécifié par le champ Number of Objects, alors que pour le module Poly Display, le nombre d’objets graphiques est déterminé par le nombre de voix de l’instrument.

L’avantage du Multi Display est qu’il peut afficher un nombre quelconque d’objets graphiques, indépendamment du nombre de voix polyphoniques. Chaque objet peut alors être appelé individuellement via l’entrée Idx (pour “index”). De son côté, le Poly Display peut sembler plus facile à programmer, car il ne nécessite pas de procédure de gestion de l’index, tous les objets pouvant être appelés simultanément grâce au traitement polyphonique en parallèle. Cependant, le nombre d’objets est limité au nombre de voix de l’instrument.

Les propriétés de ces modules comprennent diverses options de personnalisation de l’affichage, comme la couleur et l’image de fond.

Lorsqu'ils sont utilisés en conjonction avec les modules Mouse Area et Snap Value Display, Multi Display et Poly Display permettent de construire des éléments personnalisés d'interface très sophistiqués (p.ex. des séquenceurs).

Toutes les entrées du Multi Display reçoivent des signaux événements monophoniques. L'entrée **Idx** peut aussi recevoir des signaux audio monophoniques.

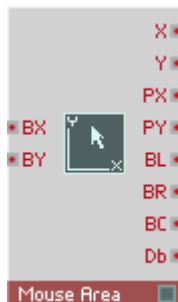
- **Idx** : index de l'objet graphique à appeler. L'index débute à 1 : l'ID du premier objet est 1 (et non 0). Les valeurs fractionnaires sont arrondies à l'entier le plus proche. Lorsque les valeurs d'**Idx** sont en dehors de l'intervalle [1.. N], le comportement dépend de l'option **Index Behaviour** des propriétés. L'ordre d'empilement des objets graphiques est déterminé par leurs valeurs **Idx** : les objets avec un **Idx** faible apparaissent au-dessus des objets avec un **Idx** élevé. Il est indispensable de fixer la valeur d'**Idx** avant de transmettre des événements à d'autres ports.
- **Obj** : type d'objet graphique, ou sélection d'une image particulière dans une animation.
 - 4 : croix.
 - 3 : ligne de (X1, Y1) au (X1, Y1) de l'objet suivant du même type.
 - 2 : ligne de (X1, Y1) to (X2, Y2).
 - 1 : barre.
 - 0 : rectangle.
 - 1 ... NP : index spécifiant une image particulière de la série d'images constituant une animation (1, 2, 3, ..., NP), où NP est le nombre total d'images dans l'animation.

Les valeurs fractionnaires sont arrondies à l'entier le plus proche.

Si l'option **Ignore Index Obj** (dans les propriétés) est activée, tous les objets graphiques sont fixés au même type.

- **X1, Y1** :coordonnées du premier coin de la zone de l'objet graphique.
Si l'option **Center to X1, Y1** est sélectionnée dans les propriétés, ces valeurs définissent les coordonnées du centre de l'objet.
- **X2, Y2** :coordonnées du second coin de la zone de l'objet graphique.
- **R, G, B** :ces entrées définissent la quantité des composantes rouge (**R**), verte (**G**) et bleue (**B**) dans la couleur de l'objet (intervalle de 0 à 1). Si l'option **Ignore Index RGB** des propriétés est activée, la même couleur est utilisée pour tous les objets graphiques.
- **A** :transparence de l'objet (0 = complètement transparent, 1 = complètement opaque).
- **X0, Y0** :valeurs X et Y minimales (pour le défilement). Ces entrées prennent le pas sur les réglages des champs **X Origin** et **Y Origin** dans les propriétés.
- **XR, YR** :plage de visibilité horizontale/verticale (pour le zoom). Ces entrées prennent le pas sur les réglages des champs **X Range** et **Y Range** dans les propriétés.
- **NP** :sortie transmettant le nombre d'images dans l'animation.
- **NO** :sortie transmettant le nombre d'objets graphiques.

Les entrées et les sorties de **Poly Display** sont les mêmes que celles de **Multi Display**, si ce n'est que **Poly Display** n'a pas d'entrée **Idx** ni de sortie **NO**. Toutes les entrées de **Poly Display** reçoivent des signaux évènements polyphoniques sauf **X0, XR, Y0** et **YR**, qui reçoivent des signaux évènements monophoniques.



Le module **Mouse Area** détecte et transmet les actions de la souris : clics du bouton, glissements et changements de position. Le module **Mouse Area** peut avoir ses propres couleur de bordure et de remplissage, et peut ainsi être lui-même utilisé comme élément de l'interface du panneau. Cependant, il est le plus souvent utilisé comme superposition invisible (transparente) au-dessus d'autres modules, notamment les modules **Multi Display** et **Poly Display**, afin de construire des éléments d'interface hautement perfectionnés.

Les sorties **X** et **Y** de **Mouse Area** travaillent en mode absolu ou en mode incrémental (vous le spécifiez dans les propriétés). Lorsque **Incremental Mode** est sélectionné, les positions **X** et **Y** sont ajustées de façon incrémentale à chaque nouvelle action de glisser, comme pour les potentiomètres et autres tirettes de REAKTOR. Particulièrement, lorsque l'utilisateur commence un nouveau glisser, les sorties **X** et **Y** transmettent les valeurs relativement à la position à laquelle le dernier glisser est arrivé, par opposition à la position absolue du curseur de la souris. Les entrées **BX** et **BY** ne concernent que le mode incrémental ; elles servent à fixer la "base" incrémentale pour les glissers suivants (prenant le pas sur le dernier glisser de la souris). Typiquement, ces entrées sont connectées à des modules **Snap Value** pour garantir que la base incrémentale est fixée en fonction du dernier mouvement de souris enregistré dans un snapshot.

Dans les propriétés, **Outline Style** spécifie l'apparence de la bordure de la boîte de **Mouse Area**. Si **Rectangle** est sélectionné, une ligne de 1 pixel de large entoure la boîte de **Mouse Area**, tandis que si **Bar** est sélectionné, la boîte est remplie avec une couleur pleine.

L'option **Active State** spécifie l'action entraînant le passage de la bordure de la boîte de l'état **inactive** à l'état **active**. Coisissez entre **Selection** (l'état actif de la boîte est enclenché lorsque la boîte de **Mouse Area** est sélectionnée) et les options **Left/Right/Center button** (l'état actif de la boîte est enclenché

lorsque le bouton correspondant de la souris est appuyé). Les propriétés **Outline Color** permettent de définir les réglages de couleur et de transparence séparément pour les états actif et inactif.

Parmi les propriétés se trouvent également **X offset** et **Y offset**, qui permettent de décaler la position de **Mouse Area** sur le panneau de l'instrument par rapport au quadrillage 4 x 4 de REAKTOR.

BX : entrée pour fixer la valeur de base des changements incrémentaux à la sortie **X**. Les valeurs autorisées sont celles à l'intérieur de **Range X**, tel que spécifié dans les propriétés.

BY : entrée pour fixer la valeur de base des changements incrémentaux à la sortie **Y**. Les valeurs autorisées sont celles à l'intérieur de **Range Y**, tel que spécifié dans les propriétés.

Notez que les entrées **BX** et **BY** ont une influence sur les sorties **X** et **Y** uniquement si l'option **Incremental Mode** est activée dans les propriétés.

X : sortie transmettant la position horizontale de la souris, redimensionnée et limitée par les valeurs **Range X** (dans les propriétés). La sortie **X** transmet seulement les positions se trouvant à l'intérieur de la boîte **Mouse Area** (dont la taille est définie par **Size X** et **Size Y** dans les propriétés).

Y : sortie transmettant la position verticale de la souris, redimensionnée et limitée par les valeurs **Range Y** (dans les propriétés). La sortie **Y** transmet seulement les positions se trouvant à l'intérieur de la boîte **Mouse Area** (dont la taille est définie par **Size X** et **Size Y** dans les propriétés).

PX : position horizontale (en pixels) de la souris par rapport à la ligne d'origine des X (côté gauche de **Mouse Area**). Un déplacement de la souris à gauche de la ligne d'origine des X génère des valeurs négatives, un déplacement de la souris à droite de cette ligne génère des valeurs positives. Contrairement à la sortie **X**, qui est limitée aux mouvements à l'intérieur de la boîte de **Mouse Area**, **PX** transmet les valeurs de l'extrémité gauche à l'extrémité droite de l'écran.

PY : position verticale (en pixels) de la souris par rapport à la ligne d'origine des Y (côté haut de **Mouse Area**). Un déplacement de la souris au-dessus de la ligne d'origine des Y génère des valeurs négatives, un déplacement de la souris au-dessous de cette ligne génère des valeurs positives. Contrairement à la sortie **Y**, qui est limitée aux mouvements à l'intérieur de la boîte de **Mouse Area**, **PY** transmet les valeurs de l'extrémité haute à l'extrémité basse de l'écran.

BL : état du bouton gauche de la souris : 1 s'il est appuyé, 0 sinon.

BR : état du bouton droit de la souris : 1 s'il est appuyé, 0 sinon.

BC : état du bouton central de la souris : 1 s'il est appuyé, 0 sinon.

Db : génère un événement unique de valeur 1 à chaque fois qu'un double-clic survient. La valeur reste à 1 après le double-clic, donc testez les événements Db et non leur valeur !

Stacked Macro

Panel



Les **Stacked Macros** («macros empilées») permettent à plusieurs objets graphiques de partager la même zone du panneau de l'instrument. Les module **Panel Index** permet alors de contrôler à tout instant quel objet est affiché dans la zone.

Après avoir inséré une **Stacked Macro** dans votre structure, placez-la au bon endroit dans le panneau et donnez-lui la taille voulue (dans les propriétés). Puis insérez deux macros ou plus (des macros «normales», pas des **Stacked Macros**) dans la **Stacked Macro**, ainsi qu'un module **Panel Index**. Une seule macro «normale» à la fois sera visible. La valeur d'entrée du module **Panel Index** détermine la macro visible. Pour découvrir le numéro d'index d'une macro, faites un clic droit sur cette macro dans le panneau de l'instrument – le numéro d'index apparaît dans le menu contextuel.

MIDI In

Les modules MIDI In servent à rendre disponibles, dans REAKTOR, des données MIDI provenant d'appareils extérieurs. Les modules sont distincts pour les différents types de données MIDI, par exemple Notes, Velocity, Pitchbend, Mono et Poly Aftertouch, Controller, Program Change et MIDI Clock. REAKTOR crée aussi, à partir des informations de notes, des données Gate disponibles sous forme de différents modules. Certains modules MIDI In sont utilisables pour des connexions internes ou OSC.

Note Pitch

MIDI In



Source d'événement polyphonique pour la hauteur de son d'événements MIDI Note On.

Un événement Note On affecte à la sortie la valeur correspondant au numéro de la touche (Note Number). Réglez la plage du signal de sortie, dans le dialogue de propriétés, avec **Min** et **Max**. La résolution comporte 128 niveaux. Un événement Note Off n'a aucun effet.

Lorsque vous contrôlez, dans la plage de sortie standard de **Min** = 0 à **Max** = 127, l'entrée **P** d'un oscillateur (commande logarithmique de la hauteur de son = pitch), le son rendu est le ton tempéré classique. Une unité correspond à un pas d'un demi-ton. Le Do (C) central correspond à 60. Un réglage différent de **Min** et **Max** permet de décaler et d'échelonner la hauteur de son en conséquence, par exemple pour jouer des quarts de ton.

Pitchbend

MIDI In



Source monophonique d'événement pour pitchbend MIDI (Pitch Bender). La résolution comporte 16 384 niveaux. Lorsque le Pitch Bender est en position médiane, la valeur de sortie est toujours 0. Réglez la plage de valeur de sortie séparément vers le haut et vers le bas. Pour un réglage vers le bas, utilisez **Min** et vers le haut, utilisez **Max**.

Lorsque vous contrôlez l'entrée **P** d'un oscillateur (commande logarithmique de la hauteur de son), par exemple après avoir ajouté une valeur de pitch de note, vous pouvez modifier la hauteur de son d'un demi-ton vers le haut ou vers le bas, avec une plage de sortie de **Min** = -1 et **Max** = 1. Une plage de -12 à +12 signifie un pitchbend de ± 12 demi-tons, c'est-à-dire ± 1 octave.



Source d'événements polyphoniques pour événements MIDI Note On et Note Off.

Un événement Note On affecte à la sortie la valeur correspondant à la vitesse (intensité de l'attaque). Réglez la plage du signal de sortie, dans le dialogue de propriétés, avec **Min** et **Max**. La résolution comporte 128 niveaux. Un événement Note Off règle la sortie sur 0. Si vous souhaitez désactiver la vitesse, réglez **Min** et **Max** sur la même valeur.

Single Trig. Gate



Source d'événements monophoniques pour événements MIDI Note On et Note Off à déclenchement unique.

Un événement Note On affecte à la sortie la valeur correspondant à la vitesse (intensité de l'attaque). Réglez la plage du signal de sortie, dans le dialogue de propriétés, avec **Min** et **Max**. La résolution comporte 128 niveaux. Un événement Note Off règle la sortie sur 0. Si vous souhaitez désactiver la vitesse, réglez **Min** et **Max** sur la même valeur.

Seule la première note déclenche un événement et lance ainsi des enveloppes connectées depuis leur origine. Les notes jouées pendant que d'autres sont maintenues (légato) ne créent, au contraire de **Gate** aucun événement, et ne redéclenchent donc pas les enveloppes.

Sel. Note Gate

MIDI In



Source d'événements monophoniques pour événements MIDI Note On et Note Off sélectionnés.

Un événement Note On au numéro de touche (Note Number) sélectionné affecte à la sortie la valeur correspondant à la vélocité (intensité de l'attaque). Réglez la plage du signal de sortie, dans le dialogue de propriétés, avec **Min** et **Max**. La résolution comporte 128 niveaux. Un événement Note Off au numéro de touche (Note Number) sélectionné règle la sortie sur zéro. Si vous souhaitez désactiver la vélocité, réglez **Min** et **Max** sur la même valeur.

On Velocity

MIDI In



Source d'événements polyphoniques pour la vélocité des événements MIDI Note On. Un événement Note On affecte à la sortie la valeur correspondant à la vélocité (intensité de l'attaque). Réglez la plage du signal de sortie, dans le dialogue de propriétés, avec **Min** et **Max**. La résolution comporte 128 niveaux.

Off Velocity

MIDI In



Source d'événements polyphoniques pour la vélocité des événements MIDI Note Off. Un événement Note Off règle la sortie sur une valeur correspondant à la vitesse à laquelle vous relâchez la touche. Réglez la plage du signal de sortie, dans le dialogue de propriétés, avec **Min** et **Max**. La résolution comporte 128 niveaux. Remarque : seuls quelques claviers permettent de créer de tels événements.



Source d'événements monophoniques pour événements de contrôleur MIDI (Control Change) Un événement règle la sortie sur une valeur correspondant à la position de l'élément de commande (Controller). Réglez-en le numéro (1–128), dans le dialogue de propriétés, avec **Note No** et la plage du signal de sortie avec **Min** et **Max** . La résolution comporte 128 niveaux.

Vous pouvez enregistrer les positions actuelles des éléments de commande (et atténuateurs) d'un instrument dans un snapshot, puis les recharger sous cette forme. Activer le commutateur **Snap Isolate**, dans les propriétés du Controller, empêche la modification de sa valeur lors du rappel du snapshot. Le signal de sortie de ce module est utilisable comme signal d'événement ou audio.

Certaines des commandes MIDI fréquentes portent les numéros suivants :

- 1 molette de modulation (Mod Wheel)
- 7 Volume
- 10 Panorama (Pan)
- 64 Entretien (sustain)
- 65 Portamento
- 66 Maintien (Sostenuto)

Consultez les graphiques de mise en œuvre MIDI de votre instrument pour découvrir quelles commandes MIDI il peut transmettre.

Ch. Aftertouch

MIDI In



Source d'événements pour événements MIDI monophoniques Aftertouch (Channel's After Touch). Un événement règle la sortie sur la valeur correspondant à la pression exercée sur la touche. Réglez la plage du signal de sortie, dans le dialogue de propriétés, avec **Min** et **Max**. La résolution comporte 128 niveaux.

Poly Aftertouch

MIDI In



Source d'événements pour ce que l'on appelle événements MIDI polyphoniques Aftertouch (Key Aftertouch). Un événement règle la sortie sur la valeur correspondant à la pression exercée sur la touche. La valeur est définie uniquement pour la voix de l'instrument qui joue la note correspondant à la touche. Réglez la plage du signal de sortie, dans le dialogue de propriétés, avec **Min** et **Max**. La résolution comporte 128 niveaux.

Remarque : seuls de très rares claviers sont en mesure de générer un aftertouch polyphonique.

Sel. Poly AT

MIDI In



Source d'événements monophoniques pour des événements Aftertouch polyphoniques (Key Aftertouch).

Un événement portant le numéro de touche (Note Number) sélectionné règle la sortie à une valeur correspondant à la pression exercée sur la touche. Réglez le numéro de la touche (1–128), dans le dialogue de propriétés, avec **Controller No** et la plage du signal de sortie avec **Min** et **Max**. La résolution comporte 128 niveaux. Vous pouvez enregistrer les positions actuelles des commandes (et atténuateurs) d'un instrument dans un snapshot, puis les recharger sous cette forme. Activer le commutateur **Snap Isolate**, dans les propriétés du contrôleur, empêche la modification de sa valeur lors du rappel du snapshot.

Remarque : seuls de très rares claviers sont en mesure de générer un aftertouch polyphonique.

Program Change

MIDI In



Source d'événements monophoniques pour événements MIDI de changement de programme. Un événement règle la sortie sur la valeur correspondant au numéro du programme. Réglez la plage du signal de sortie, dans le dialogue de propriétés, avec **Min** et **Max**. La résolution comporte 128 niveaux.

Lorsque vous utilisez ce module, vous désactivez en règle générale, dans les propriétés de l'instrument, **Prog. Change Enable**, afin que les événements MIDI Program Change n'appellent pas aussi des snapshots.

Start/Stop

MIDI In



Source d'événements Start et Stop pour la synchronisation avec des appareils MIDI extérieurs ou l'horloge centrale interne.

Le module **Start/Stop** fournit un signal Gate monophonique à la sortie, dont vous réglez la valeur, dans les propriétés, avec **Output Value**. Le signal passe à la valeur réglée lorsque vous appuyez sur le commutateur Start ou à la réception d'un événement MIDI Start. Le signal repasse à zéro lorsque vous appuyez sur le commutateur Stop ou à la réception d'un événement MIDI Stop.

On connecte généralement ce module avec l'entrée Reset de séquenceurs et de diviseurs d'événements synchronisés avec MIDI Clock, pour en forcer le démarrage synchrone.

1/96 Clock

MIDI In



Source d'un signal de temps correspondant à l'horloge MIDI externe ou à l'horloge centrale interne.

La sortie délivre un événement par 96e de note, dont vous réglez la valeur, dans les propriétés, avec **Output Value**.

Contrairement à la source **Sync Clock**, les événements de l'**horloge 1/96 Clock** doivent être préparés avec un **Event Freq. Divider** (voir **Random**). Ceci présente l'avantage de vous permettre d'expérimenter avec différents facteurs de division.



Source d'un signal de temps dérivé de l'horloge MIDI externe ou de l'horloge centrale interne.

La sortie fournit un signal Gate dont vous réglez la valeur, dans les propriétés, avec **Output Value**. À chaque battement, la porte passe à la valeur réglée, et revient à zéro après un certain temps. Des événements Start-Stop activent et désactivent le module.

Vous réglez la vitesse du signal de mesure (quart, huitième, seizième de note, etc.) dans les propriétés, dans **Rate**.

La durée du signal de la porte (huitième, seizième de note, etc.) se règle dans les propriétés, sous **Duration**.

Song Pos

MIDI In



Module de transmission des informations concernant la position du morceau. Le décompte a lieu en 96es de notes, à partir du début d'un morceau. Normalement, on connecte le module avec l'entrée **A** du module **Modulo** et une constante égale à **6** avec l'entrée **B**, ce qui produit une grille de 16es de notes à la sortie Div.

- **96**: sortie d'événements pour le nombre de 96e de notes (24 par quart de note) écoulés depuis le début du morceau. Utilisez le module **Modulo** si vous souhaitez d'autres unités de notes.
- **96a** : sortie audio pour la position exacte dans l'échantillon, en 96e de notes.



Source monophonique recevant les messages MIDI d'un appareil MIDI externe (clavier, séquenceur, etc.) ou interne (un autre instrument de l'ensemble). L'ordre de sortie des ports (cf. ci-dessous) est fixé : de haut en bas. Ceci garantit que le type (p.ex. Control Change, changement de commande) et la source (p.ex. le contrôleur 7 sur le canal 1) du message soient toujours transmis avant la valeur du message.

St : port de sortie transmettant le type de message reçu.

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

Ch : numéro du canal MIDI (1-16).

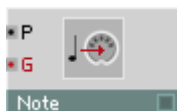
Nr : numéro d'une Note, d'un Control Change ou d'un Program Change (0-127).

Val : vélocité de la Note, pression de l'Aftertouch, ou valeur du Control Change ou du Pitchbend. Pour un appareil MIDI externe, les valeurs sont quantifiées sur 7 bits (14 bits pour le Pitchbend). Pour le MIDI interne, les valeurs sont les virgules flottantes en 32 bits. Toutes les valeurs sont mises à l'échelle en fonction des valeurs Min et Max (réglables dans les propriétés), qui sont par défaut 0 et 1. Un autre réglage typique est de 0 à 127, ce qui peut faciliter l'interprétation des valeurs dans certaines situations (Program Change par exemple).

MIDI Out

REAKTOR traite les données MIDI, mais il peut également les générer. Les modules MIDI Out les envoient à des appareils MIDI extérieurs. A chaque module MIDI In correspond un module MIDI Out. Certains des modules MIDI Out peuvent servir à des connexions internes ou OSC.

Note Pitch/Gate



MIDI Out

Convertit un signal monophonique ou polyphonique en notes MIDI. A l'entrée Gate **G**, chaque événement génère un message MIDI sur le port MIDI utilisé par REAKTOR pour l'émission. La valeur de l'événement définit l'intensité de l'attaque (Velocity). Un événement de valeur 1 génère une vélocité de 127. Un événement de valeur 0 génère un événement Note Off, donc coupe le son.

La hauteur de son (numéro de touche) des événements Note On et Note Off est la valeur actuelle de l'entrée pitch **P**, dans la plage réglée avec **Min** et **Max**. Un événement de valeur égale à **Min** provoque le passage du pitch de note MIDI à 0, un événement de valeur égale à **Max** le fait passer à 127.

Pitchbend



MIDI Out

Convertit un signal monophonique ou polyphonique en pitchbend MIDI. Chaque événement génère un message MIDI sur le port MIDI utilisé par REAKTOR pour l'émission. La plage du signal d'entrée est définie avec **Min** et **Max**. A la sortie, la résolution est de 16 384 pas. Un événement de valeur égale à **Min** fait passer la valeur pitchbend MIDI à -8192, un événement de valeur égale à **Max** à +8191.



Convertit un signal monophonique d'événement en événement de contrôleur MIDI. Chaque événement génère un message MIDI sur le port MIDI utilisé par REAKTOR pour l'émission. Réglez le numéro du contrôleur MIDI dans les propriétés, dans **Controller No.** La plage du signal d'entrée est définie avec **Min** et **Max**. A la sortie, la résolution est de 128 pas. Un événement de valeur égale à **Min** provoque le passage du contrôleur MIDI à 0, un événement de valeur égale à **Max** le fait passer à 127.

Ch. Aftertouch

MIDI Out

Convertit un signal monophonique d'événement en événement MIDI Channel Aftertouch. Chaque événement génère un message MIDI sur le port MIDI utilisé par REAKTOR pour l'émission. La plage du signal d'entrée est définie avec **Min** et **Max**. A la sortie, la résolution est de 128 pas. Un événement de valeur égale à **Min** règle l'aftertouch de canal MIDI sur 0, un événement de valeur égale à **Max** le règle sur 127.

Poly Aftertouch

MIDI Out

Convertit les événements Aftertouch (**AT**) en événements MIDI Poly Aftertouch. La valeur existant à l'entrée pitch (**P**) est convertie en les numéros de note. Les valeurs comprises entre **Min** et **Max** correspondent à des numéros de note entre 0 et 127. Les valeurs Aftertouch comprises entre 0 et 1 sont converties en valeurs comprises entre 0 et 127.

- **P** : entrée pitch de hauteur de son convertie en un numéro de note. Les valeurs comprises entre **Min** et **Max** donnent des numéros de note compris entre 0 et 127.
- **AT** : entrée du signal Aftertouch. Les valeurs comprises entre 0 et 1 sont converties en valeurs Aftertouch MIDI comprises entre 0 et 127.



Convertit un signal monophonique d'événement en événement MIDI Poly Aftertouch. Chaque événement génère un message MIDI sur le port MIDI utilisé par REAKTOR pour l'émission. Réglez le numéro de la touche à laquelle la pression définie est destinée dans les propriétés, sous **Note No.**. La plage du signal d'entrée est définie avec **Min** et **Max**. A la sortie, la résolution est de 128 pas. Un événement de valeur égale à **Min** règle l'Aftertouch MIDI à 0, un événement de valeur égale à **Max** le fait passer à 127.

Remarque : seuls de très rares appareils MIDI sont capables de traiter des Aftertouch polyphoniques.

Program Change



Convertit un signal d'événement monophonique en événements MIDI Program Change (changement de programme). Chaque événement génère un message MIDI sur le port MIDI utilisé par REAKTOR pour l'émission. La plage du signal d'entrée est définie avec **Min** et **Max**. A la sortie, la résolution est de 128 pas. Un événement de valeur égale à **Min** règle Program Change sur 0, un événement de valeur égale à **Max** sur 127.

Start/Stop



Les événements d'entrée génèrent des événements Start/Continue/Stop à la sortie MIDI.

- **G** : un événement positif émet un ordre Start ou Continue, un événement négatif ou nul émet un ordre Stop.
- **Rst** : lorsqu'un événement positif a été reçu par cette entrée **Rst**, l'événement positif suivant parvenant à l'entrée **G**(ate) déclenche un message Start (à 0).

1/96 Clock

MIDI Out

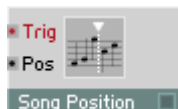


Les événements d'entrée génèrent des événements 1/96e d'horloge à la sortie MIDI.

- **In** : un événement à valeur positive génère un événement d'horloge MIDI.

Song Pos

MIDI Out



La valeur existant à l'entrée **Pos** est envoyée, avec l'événement, à l'entrée **Trig**, représentant la Song Position (position du morceau).

- **Trg** : chaque événement à valeur positive génère un événement Song Position MIDI.
- **Pos** : La valeur existant à cette entrée (sous forme de multiple de 96e de notes) est regroupée avec un événement Trigger puis émise sous forme de Song Position MIDI.

Channel Message

MIDI Out



Source monophonique envoyant des messages MIDI vers un appareil MIDI externe (clavier, séquenceur, etc.) ou interne (un autre instrument de l'ensemble). Les messages MIDI sont envoyés à chaque fois qu'un événement arrive à l'entrée St. Ainsi, la valeur et la destination désirées du message doivent être correctement définies, avec les valeurs appropriées aux autres entrées, avant que l'événement n'arrive à l'entrée St. Toutes les entrées reçoivent des signaux monophoniques uniquement.

St : entrée événements définissant le type de message à envoyer. Un message est transmis à chaque fois qu'un événement arrive à cette entrée.

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

Ch : entrée audio pour le numéro de canal MIDI (1-16). Les valeurs fractionnaires sont arrondies à l'entier le plus proche (la valeur de 0,5 est arrondie à 1).

Nr : entrée audio pour le numéro de Note, Control Change ou Program Change (0-127).

Val : entrée audio pour la vélocité de la Note, la pression de l'Aftertouch, ou la valeur du Control Change ou du Pitchbend.

Math

REAKTOR contient des modules permettant les opérations mathématiques courantes, comme l'addition, la soustraction, la multiplication et la division, mais également certaines moins familières, comme arc, tangente et l'inverse de la racine. Vous disposez aussi de modules permettant le calcul exponentiel et logarithmique, qui vous permettent de piloter les entrées de modules à échelles différentes. Certains modules par exemples ont des entrées de fréquences linéaires (mesurées en hertz), alors que les entrées de fréquences d'autres modules sont exponentielles (donc mesurées en demi-tons et accordées aux notes MIDI). Il existe donc des modules assurant la conversion entre ces deux formats.

Tous les modules de cette section sont hybrides. Ils sont donc utilisables à la fois comme modules audio et d'événements, en fonction du câblage de leurs entrées. Beaucoup de ces modules (par ex. Add et Mult) présentent également une gestion dynamique des entrées de modules, signalisée par trois petits points situés dans la moitié inférieure du module. Lorsque, touche **Ctrl** enfoncée, vous tirez un câble vers un emplacement libre de la zone de port d'entrée (côté gauche du module) d'un module de ce type, un nouveau port d'entrée se crée automatiquement.

Constant

Math



Source monophonique d'une constante. Réglez la valeur souhaitée dans le dialogue Properties. Le module génère un événement unique, et ceci pour son initialisation, lorsqu'il est activé.

Add

Math



Ajoute deux ou plusieurs signaux audio ou d'événements. Le signal de sortie est la somme des signaux d'entrée (**Out = In1 + In2 + In3 etc.**).

Vous pouvez utiliser ce module comme table de mixage multi-canaux, tous ceux-ci étant réglés sur 0 dB.

Subtract

Math



Soustrait deux signaux audio ou d'événements. Le signal de sortie est le résultat de la soustraction du deuxième signal d'entrée au premier (**Out = In1 - In2**).

Invert, -X

Math



Inverse un signal audio ou d'événement. Le signal de sortie est l'inverse du signal d'entrée (**Out = -In**).

L'inversion simple d'un signal n'est généralement pas audible, même lorsqu'elle est combinée avec un signal non inversé. L'inversion de signaux de commande a par contre un effet très net.

Multiply

Math



Multiplie deux ou plusieurs signaux audio ou d'événements. Le signal de sortie est le produit des signaux d'entrée (**Out = In1 * In2 * In3 etc**).

L'application caractéristique : un amplificateur commandé par le signal (correspond à VCA pour les synthétiseurs analogiques), avec un signal audio établi à une entrée et un facteur d'amplification établi à l'autre.

Si l'une des entrées a un signal nul, le signal de sortie est toujours nul.

Cette fonction permet également de calculer le carré dans les cas où le même signal est affecté aux deux entrées (**Out = In · In = In²**).

Lorsque des signaux différents sont affectés aux entrées, le signal de sortie en est la modulation en anneaux.



Multiplicateur-additionneur combiné : la valeur de sortie est le résultat de $(A * B) + C$.

Reciprocal 1/x

Math

Le signal de sortie est l'inverse du signal d'entrée.

Prudence lorsque les valeurs d'entrée sont faibles (proches de 0), car la valeur de sortie sera très élevée. Lorsque le signal d'entrée est exactement égal à 0, le signal de sortie est également nul.

En règle générale, cette fonction n'est pas utile pour les signaux audio. Ce module est plutôt prévu pour les signaux de commande (qui n'approchent pas le 0).

Divide x/y

Math

Le signal de sortie est le quotient du signal x, existant à l'entrée supérieure, et du signal y, de l'entrée inférieure.

Prudence lorsque la valeur de l'entrée inférieure est faible (proche de 0), car la valeur de sortie sera très élevée. Lorsque le signal d'entrée est exactement égal à 0, le signal de sortie est également nul.

En règle générale, cette fonction n'est pas utile pour les signaux audio.

Dans la mesure du possible, utilisez un multiplicateur plutôt qu'un diviseur pour les signaux audio, car ceci réduit considérablement la contrainte à laquelle le processeur est soumis. Par exemple, plutôt que de diviser par une constante ou un signal d'événement, inversez le signal d'événement (1/x), puis multipliez le résultat par le signal audio.

Modulo x % y

Math



Modulo et Div. servent à calculer le nombre entier résultant de la division des deux valeurs d'entrée, et le reste de l'opération.

- **A** : entrée hybride destinée au signal **A** divisé par le signal **B**.
- **B** : entrée hybride destinée au signal **B** utilisé pour diviser le signal **A**. Normalement, **B** est un entier, mais ce n'est pas une obligation.
- **Div** : sortie hybride destinée à l'entier qui résulte de la division de **A** par **B**. Il s'agit de la valeur entière maximum inférieure ou égale à A/B .
- **Mod** : sortie hybride destinée au modulo de **A** et **B**. Il s'agit du reste de la division de **A** par **B**. Plage : $[0 \dots B]$.

Rectifier

Math



Redresseur du signal d'entrée, qui inverse les valeurs négatives pour les rendre positives.

- **In** : entrée hybride destinée au signal à redresser. Les valeurs négatives deviennent positives, la magnitude demeure inchangée.
- **Out** : sortie hybride destinée au signal redressé.

Rect./Sign

Math



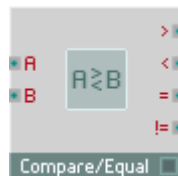
Redresseur du signal d'entrée qui inverse les valeurs négatives pour les rendre positives. Vous disposez d'une fonction de signe du signal d'entrée, à la sortie **Sign**.

- **In** : entrée hybride destinée au signal à redresser. Les valeurs négatives deviennent positives, la magnitude demeure inchangée.
- **Sign** : sortie hybride destinée au signe du signal d'entrée (-1 ou +1).
- **Out** : sortie hybride destinée au signal redressé.



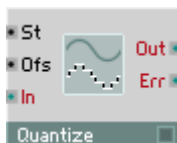
Fonction logique de comparaison. Compare les valeurs d'entrée entre elles et règle la sortie sur la valeur logique résultant de cette comparaison.

- **A** : entrée destinée à la première des deux valeurs à comparer.
- **B** : entrée destinée à la deuxième valeur à comparer.
- **>**: sortie destinée au résultat de la comparaison « **A** supérieure à **B** » (0=FAUX, 1=VRAI)
- **<=**: sortie destinée au résultat de la comparaison « **A** inférieure à **B** » (0 = FAUX, 1 =VRAI).



Fonction logique de comparaison. Compare les valeurs d'entrée entre elles et règle les quatre sorties sur la valeur logique résultant de cette comparaison.

- **A** : entrée destinée à la première des deux valeurs à comparer.
- **B** : entrée destinée à la deuxième valeur à comparer.
- **>**: sortie destinée au résultat de la comparaison « **A** supérieure à **B** » (0=FAUX, 1=VRAI)
- **<**: sortie destinée au résultat de la comparaison « **A** inférieure à **B** » (0 = FAUX, 1 =VRAI).
- **=**: sortie destinée au résultat de la comparaison « **A** égale à **B** » (0 = FAUX, 1 =VRAI).
- **!=**: sortie destinée au résultat de la comparaison « **A** différente de **B** » (0 = FAUX, 1 =VRAI).



Diviseur à facteur réglable. Le signal d'entrée est émis après avoir été arrondi au facteur de quantification immédiatement supérieur.

- **St** : entrée hybride destinée au contrôle du pas de quantification. Lorsque **St** = 0, le signal n'est pas quantifié.
- **In** : entrée hybride destinée au signal à quantifier.
- **Out** : sortie hybride destinée au signal quantifié.
- **Err** : sortie hybride destinée à l'erreur de quantification apparue lorsque le signal est arrondi : $\text{Err} = \text{Out} - \text{In}$.

Expon. (A)



Fonction exponentielle permettant de convertir des valeurs de niveau, logarithmiques et en dB, en valeurs linéaires d'amplitude.

- **Lvl** : entrée hybride destinée aux valeurs de niveau logarithmiques à convertir en valeurs linéaires d'amplitude. Plage caractéristique : [50 ... 10].
- **A** : sortie hybride destinée au signal linéaire de commande de l'amplitude.

Expon. (F)



Fonction exponentielle permettant de convertir des valeurs pitch logarithmiques, en demi-tons, en valeurs linéaires de fréquences, en Hz.

- **P** : entrée hybride destinée aux valeurs logarithmiques de pitch, en demi-tons, à convertir en valeurs linéaires de fréquences, en Hz. Plage caractéristique : [0 ... 127].

- **F** : sortie hybride destinée au signal linéaire de commande de la fréquence.

Log (A)

Math



Fonction logarithmique servant à convertir les valeurs linéaires d'amplitude en valeurs logarithmiques de niveaux, en dB. Elle peut être utilisée pour piloter les entrées logarithmiques de temps d'enveloppes, par exemple.

- **A** : entrée hybride destinée aux valeurs linéaires d'amplitude à convertir en valeurs de niveau logarithmiques, en dB. Plage caractéristique : [0 ... 1000].
- **Lvl** : sortie hybride de niveau, en dB. Plage caractéristique : [-60 ... 0].

Log (F)

Math



Fonction logarithmique de conversion de valeurs linéaires de fréquence, en Hz, en valeurs pitch logarithmiques, en demi-tons.

- **F** : entrée hybride destinée aux valeurs linéaires de fréquences, en Hz, à convertir en valeurs logarithmiques de pitch, en demi-tons. Plage caractéristique : [0 ... 5000].
- **P** : sortie hybride destinée au pitch, en demi-tons. Plage caractéristique : [0 ... 100].

Power x y

Math



Le module Power calcule X puissance Y (indiqué généralement par X^Y ou X^Y).

- **X** : entrée hybride destinée à la base.
- **Y** : entrée hybride destinée à l'exposant.
- **X^Y** : entrée hybride destinée au résultat de l'opération.

Square Root

Math

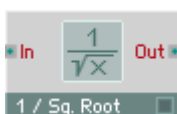


Calcule la racine carrée de la valeur à l'entrée.

- **In** : entrée hybride destinée à l'argument de la fonction de racine. Si la valeur y est négative, la valeur de sortie est égale à 0.
- **Out** : sortie hybride destinée à la racine carrée de la valeur d'entrée.

1 / Square Root

Math



Ce module calcule la valeur inverse de la racine de la valeur établie à l'entrée. Ce module est plus efficace que la combinaison des deux modules **Square Root** et **Reciprocal 1/x**. Si les valeurs d'entrée sont négatives, ce module émet la valeur 0.

- **In** : entrée hybride destinée à l'argument.
- **Out** : sortie hybride destinée au résultat.

Sine

Math



Le module Sine calcule la fonction trigonométrique Sinus. Tout comme la sortie, l'entrée dispose d'une plage de -1 à +1. Pour un calcul à partir d'une valeur d'entrée en degrés, commencez par la diviser par 360. Lorsque la valeur d'entrée est une mesure d'arc (radians), divisez-la par 2*Pi (environ 6,283). La plage de sortie est comprise entre 1 (valeur d'entrée 0,25) et -1 (valeur d'entrée 0,75).

- **In** : entrée hybride destinée à l'argument.
- **Out** : sortie hybride destinée au résultat.



Le sinus et le cosinus de chaque événement d'entrée sont calculés. Une valeur d'entrée de 1,0 correspond à une période complète des fonctions sinus et cosinus (donc 360 degrés).

- **In** : entrée hybride destinée à l'argument (l'angle) de la fonction sinus. Une valeur de 1,0 correspond à une période de la fonction sinus (360 degrés). Plage caractéristique : [-1 ... 1].
- **Sin** : sortie hybride destinée au sinus de la valeur d'entrée.
- **Cos** : sortie hybride destinée au cosinus de la valeur d'entrée.



Le module ArcSin calcule la fonction inverse du sinus (arcsinus de x est la valeur comprise entre 0 et 1 dont le sinus est x). Dans la mesure où la valeur de sortie de la fonction sinus est comprise entre -1 et 1, la fonction arcsinus s'applique elle aussi uniquement pour des valeurs d'entrée de cette plage. Si l'argument est inférieur à -1, le module émet la valeur - 0,25, et s'il est supérieur à 1, la valeur 0,25.

- **In** : entrée hybride destinée à l'argument.
- **Out** : sortie hybride destinée à l'arcsinus de la valeur d'entrée.



Le module ArcCos calcule la fonction inverse du cosinus (arccosinus de x est la valeur comprise entre 0 et 1 dont le cosinus est x). Dans la mesure où la valeur de sortie de la fonction cosinus est comprise entre -1 et 1, la fonction arccosinus est elle aussi valable uniquement pour des valeurs d'entrée de cette plage. Si l'argument est inférieur à -1, le module émet la valeur -0,5, et s'il est supérieur à 1, la valeur 0.

- **In** : entrée hybride destinée à l'argument.
- **Out** : sortie hybride destinée à l'arccosinus de la valeur d'entrée.



Le module ArcTan calcule la fonction tangente inversée (la tangente est le sinus divisé par le cosinus). La valeur de sortie du module ArcTan est comprise entre -0,25 et 0,25, ce qui correspond à une plage de -90 à 90 degrés.

- **In** : entrée hybride destinée à l'argument.
- **Out** : sortie hybride destinée à l'arctangente de la valeur d'entrée.

Signal Path

Les modules de cette catégorie permettent de router des données audio et d'événement dans la structure REAKTOR avec une grande souplesse. Ce sont entre autres des tables de mixage, routeurs pour entrées et sorties, connexions pilotables (relays), crossfaders et panoramiseurs.

Selector/Scanner



Signal Path

Selector/Scanner. Les entrées sont parcourues en fonction de la valeur existant à l'entrée **Pos**. Si cette valeur **Pos** est entière, vous obtiendrez un signal d'entrée unique, sinon ce sera le mélange de deux entrées. Le signal de sortie est le fondu-enchaîné des deux entrées dont l'index est le plus proche de la valeur existant à l'entrée **Pos**.

Lorsque le mode **Wrap** est activé, dans **Properties**, **Pos** fonctionne comme une boucle, et **Max +1** devient 0, **Max +2** devient 1, etc.

Le Selector/Scanner permet de créer des effets intéressants en alimentant les entrées avec des signaux provenant de délais multi-bascule, et lorsque la position d'échantillonnage est commandée par le Ramp Oscillator.

Le module dispose d'une gestion dynamique des ports d'entrée. Définissez le nombre d'entrées en utilisant **Min Num Port Groups** de la page **Function** des propriétés.

- **Pos** : entrée de commande audio destinée à sélectionner l'entrée/ les entrées à balayer. **Pos** = 0 sélectionne **In0**, **Pos** = 1 **In1**, **Pos** = 0,5 génère un mix de **In0** et **In1**. Plage de valeurs caractéristique : [0 ... Max]
- **In0...Max**: entrées destinées aux signaux audio à balayer.
- **Out** : sortie audio destinée au signal balayé.

Relay 1,2

Signal Path



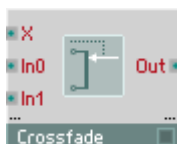
Relais. L'entrée supérieure est connectée à la sortie lorsque $Ctl > 0$. Sinon, c'est l'entrée inférieure qui y est connectée. S'il n'existe pas d'entrée inférieure, un signal nul est généré à la sortie.

Le module peut présenter un ou deux ports d'entrée. Définissez le nombre d'entrées en utilisant **Min Num Port Groups** de la page **Function** des propriétés.

- **Ctl** : entrée de commande destinée à la sélection d'une entrée de signal.
- **In** : entrée de signal (jusqu'à deux).
- **Out** : sortie destinée au signal sélectionné.

Crossfade

Signal Path



Module destiné au fondu-enchaîné (crossfade) de deux signaux. Le signal de sortie est un mix pondéré des deux signaux d'entrée (interpolation linéaire) **In0** et **In1**. Réglez la proportion des deux signaux d'entrée dans le signal de sortie via le port **X**.

Le module dispose d'une gestion dynamique des ports d'entrée. Définissez le nombre des paires d'entrée (**In0** et **In1**) et des ports de sortie en utilisant **Min Num Port Groups** de la page **Function** des propriétés.

- **X** : entrée hybride de commande destinée à la proportion du mix. Valeur comprise entre 0 (**Out** = **In1**) et 1 (**Out** = **In2**).
- **In1, In2** : entrées hybrides destinées aux signaux à mixer.
- **Out** : sortie destinée au signal mixé ($Out = (1 - X) In1 + X In2$).



Distributeur/Panoramiseur. Le signal d'entrée est connecté à une ou plusieurs sorties sélectionnées via l'entrée **Pos**, ou panoramisé entre celles-ci. Si la valeur **Pos** est entière, seul le signal d'une entrée est pris en compte, sinon le panning a lieu entre deux entrées.

Lorsque le mode **Wrap** de Propriétés est activé, **Pos** fonctionne comme boucle, et Max +1 devient 0, Max +2 devient 1, etc.

Le module dispose d'une gestion dynamique des ports de sortie. Définissez le nombre des ports de sortie en utilisant **Min Num Port Groups** de la page **Function** des propriétés.

- **Pos** : entrée hybride destinée à sélectionner la/les sortie(s) à connecter à l'entrée.
- **In** : entrée hybride de signal.
- **Out** : sortie(s) destinée(s) au signal d'entrée.

Stereo Pan



Réglage du panorama. En modifiant le niveau de signal des deux sorties, vous placez le signal d'entrée dans l'espace stéréo. La somme des valeurs existant aux sorties **L** et **R** est toujours exactement le double de la valeur d'entrée, c-à-d. que la valeur d'entrée est répartie entre les deux sorties dans une proportion variable.

Le module dispose d'une gestion dynamique des ports. Définissez le nombre des entrées et des paires de sortie (**L** et **R**) en utilisant **Min Num Port Groups** de la page **Function** des propriétés.

- **Pan** : entrée de commande servant à définir la position stéréo (-1 = gauche, 0 = médiane, 1 = droite).
- **In** : entrée hybride destinée au signal à positionner dans l'espace stéréo.
- **L** : sortie hybride destiné au signal gauche.
- **R** : sortie hybride destiné au signal droit.



Amplificateur/table de mixage pour un nombre variable de signaux audio. Les signaux d'entrée sont amplifiés ou diminués de la valeur existant au port **Lvl** correspondant (en dB) puis ajoutés à la sortie.

Le module dispose d'une gestion dynamique des ports d'entrée. Définissez le nombre des paires d'entrées (**Lvl** et **In**) en utilisant **Min Num Port Groups** de la page **Function** des propriétés.

- **Lvl** : entrée de commande logarithmique de l'amplification (gain) en dB.
- **In** : entrée de signal.
- **Out** : sortie destinée au signal mixé
($\text{Out} = \text{In1} \cdot 10^{\text{Lvl1}/20} + \text{In2} \cdot 10^{\text{Lvl2}/20}$ etc.).

Stereo Amp/Mixer**Signal Path**

Réglage du panorama et amplification pour un nombre variable de signaux audio. Les signaux d'entrée sont amplifiés ou diminués de la valeur existant aux ports **Lvl** (en dB). En modifiant le niveau de signal des deux sorties, vous positionnez le signal d'entrée dans l'espace stéréo.

Le module dispose d'une gestion dynamique des ports d'entrée. Définissez le nombre des paires d'entrées (**Lvl**, **Pan** et **In**) en utilisant **Min Num Port Groups** de la page **Function** des propriétés.

- **Lvl** : entrée de commande logarithmique de l'amplification (gain), en dB. Une valeur inférieure à 0 signifie que le signal de sortie est inférieur au signal d'entrée.
- **Pan** : entrée de commande servant à définir la position stéréo (-1 = gauche, 0 = médiane, 1 = droite).
- **In** : entrée audio destiné au signal audio à amplifier et à positionner dans l'espace stéréo.
- **L** : sortie destinée au signal audio gauche amplifié.
- **R** : sortie destinée au signal audio droit amplifié.

Oscillator

Dans REAKTOR, le terme Oscillator (oscillateur) décrit un grand nombre de générateurs de signaux. En réalité, toute procédure de génération de sons n'utilisant aucun échantillon est nommée Oscillator. Ceci inclut les générateurs à forme d'onde classique (Sine, Pulse, Sawtooth, etc.) tout comme Impulse, Step, Noise et Audio-Table.

Tous les oscillateurs de REAKTOR fonctionnent à n'importe quelle fréquence comprise entre 0 Hz (immobilisation) et la limite définie par le taux d'échantillonnage. Ceci permet de les utiliser tous comme LFOs, mais aussi pour générer des formes d'ondes sonores. Si vous utilisez un oscillateur comme LFO pour moduler une entrée d'un autre module qui n'accepte que les événements (par ex. P, par opposition à F), il est indispensable d'insérer un convertisseur A à E (perm).

Sawtooth



Oscillator

Oscillateur pour forme d'onde en dents de scie, à entrée logarithmique de commande de la hauteur de son (pitch) et modulation linéaire de l'amplitude.

- **P** : entrée logarithmique d'événement destinée à commander la hauteur de son (fréquence d'oscillateur), par demi-ton (69 = 440 Hz).
- **A** : entrée de commande audio destinée à l'amplitude. Le signal de sortie varie entre **+A** et **-A**.
- **Out** : sortie audio destinée à la forme d'onde en dents de scie.

Saw FM



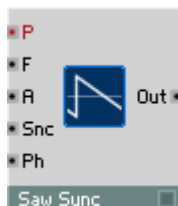
Oscillator

Oscillateur pour forme d'onde en dents de scie, à entrée logarithmique de commande de la hauteur de son (pitch), modulation linéaire de fréquence (FM) et de l'amplitude.

- **P** : entrée logarithmique d'événement destinée à commander la hauteur de son (fréquence d'oscillateur), par demi-ton (69 = 440 Hz).

- **F** : entrée de commande audio destinée à la modulation linéaire de fréquence, en Hz. La fréquence de l'oscillateur est définie à partir de **P** et **F**.
- **A** : entrée de commande audio destinée à l'amplitude. Le signal de sortie varie entre **+A** et **-A**.
- **Out** : sortie audio destinée à la forme d'onde en dents de scie.

Saw Sync



Oscillator

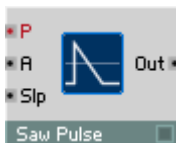
Oscillateur pour forme d'onde en dents de scie, à entrée de synchronisation, entrée logarithmique de commande de la hauteur de son (pitch), modulation linéaire de fréquence (FM) et de l'amplitude.

Dès que le signal de synchronisation dépasse le point zéro dans le sens positif (rampe ascendante), la phase de l'oscillateur se remet à zéro pour démarrer la forme d'onde à son origine.

- **P** : entrée logarithmique d'événement destinée à commander la hauteur de son (fréquence d'oscillateur), par demi-ton ($69 = 440$ Hz).
- **F** : entrée de commande audio destinée à la modulation linéaire de fréquence, en Hz. La fréquence de l'oscillateur est définie à partir de **P** et **F**.
- **A** : entrée de commande audio destinée à l'amplitude. Le signal de sortie varie entre **+A** et **-A**.
- **Snc** : entrée de commande audio destinée au signal de synchronisation. Lorsque l'onde traverse l'axe horizontal en croissant, ceci redémarre l'oscillateur à zéro.
- **Ph** : entrée permettant de définir la phase (position sur la forme d'onde) à laquelle l'oscillateur est ramené par la synchronisation.
- **Out** : sortie audio destinée à la forme d'onde en dents de scie.

Saw Pulse

Oscillator



Oscillateur pour forme d'onde en dents de scie/en pics, à entrée logarithmique de commande de la hauteur de son (pitch) et modulation linéaire de l'amplitude. La pente de la rampe est modifiable, et vous pouvez par conséquent choisir, pour la forme d'onde, entre dents de scie classiques et des pics plus courts.

- **P** : entrée logarithmique d'événement destinée à commander la hauteur de son (fréquence d'oscillateur), par demi-ton (69 = 440 Hz).
- **A** : entrée de commande audio destinée à l'amplitude. Le signal de sortie varie entre **+A** et **-A**.
- **Slp** : entrée de commande audio destinée à la rampe (slope) de la forme d'onde. La valeur 0 signifie une forme en dents de scie classique, une valeur supérieure diminue la rampe jusqu'à obtenir des pics courts (aiguilles).
- **Out** : sortie audio destinée à la forme d'onde en dents de scie/pics.

Bi-Saw

Oscillator



Oscillateur pour forme d'onde bipolaire en dents de scie et à phase nulle, à entrée logarithmique de commande de la hauteur de son (pitch), modulation de largeur d'impulsion (PWM) et modulation linéaire de l'amplitude.

- **P** : entrée logarithmique d'événement destinée à commander la hauteur de son (fréquence d'oscillateur), par demi-ton (69 = 440 Hz).
- **A** : entrée de commande audio destinée à l'amplitude. Le signal de sortie varie entre **+A** et **-A**.
- **W** : entrée de commande audio destinée à moduler la largeur d'impulsion (PWM). Plage de valeurs : entre 0 et 6 environ. **W** = 0 représente une forme d'onde en dents de scie normale (Saw Wave), une valeur supérieure de **W** signifie des impulsions plus courtes et une phase nulle plus longue.
- **Out** : sortie audio destinée à la forme d'onde bipolaire en dents de scie.

Triangle

Oscillator



Oscillateur pour forme d'onde triangulaire, à entrée logarithmique de commande de la hauteur de son (pitch) et modulation linéaire de l'amplitude.

- **P** : entrée logarithmique d'événement destinée à commander la hauteur de son (fréquence d'oscillateur), par demi-ton ($69 = 440$ Hz).
- **A** : entrée de commande audio destinée à l'amplitude. Le signal de sortie varie entre $+A$ et $-A$.
- **Out** : sortie audio destinée à la forme d'onde triangulaire.

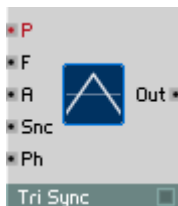
Tri FM

Oscillator



Oscillateur pour forme d'onde triangulaire symétrique, à entrée logarithmique de commande de la hauteur de son (pitch), modulation linéaire de fréquence (FM) et de l'amplitude.

- **P** : entrée logarithmique d'événement destinée à commander la hauteur de son (fréquence d'oscillateur), par demi-ton ($69 = 440$ Hz).
- **F** : entrée de commande audio destinée à la modulation linéaire de fréquence, en Hz. La fréquence de l'oscillateur est définie à partir de **P** et **F**.
- **A** : entrée de commande audio destinée à l'amplitude. Le signal de sortie varie entre $+A$ et $-A$.
- **Out** : sortie audio destinée à la forme d'onde triangulaire.



Oscillateur pour forme d'onde triangulaire symétrique, à entrée de synchronisation, entrée logarithmique de commande de la hauteur de son (pitch), modulation linéaire de fréquence (FM) et de l'amplitude.

Dès que le signal de synchronisation dépasse le point zéro dans le sens positif (rampe ascendante), la phase de l'oscillateur se remet à zéro.

- **P** : entrée logarithmique d'événement destinée à commander la hauteur de son (fréquence d'oscillateur), par demi-ton ($69 = 440$ Hz).
- **F** : entrée de commande audio destinée à la modulation linéaire de fréquence, en Hz. La fréquence de l'oscillateur est définie à partir de **P** et **F**.
- **A** : entrée de commande audio destinée à l'amplitude. Le signal de sortie varie entre **+A** et **-A**.
- **Snc** : entrée de commande audio destinée au signal de synchronisation. Lorsque l'onde traverse l'axe horizontal en croissant, ceci redémarre l'oscillateur à zéro.
- **Ph** : entrée permettant de définir la phase (position sur la forme d'onde) à laquelle l'oscillateur est ramené par la synchronisation.
- **Out** : sortie audio destinée à la forme d'onde triangulaire.



Oscillateur pour forme d'onde variable triangulaire ou parabolique, à entrée logarithmique de commande de la hauteur de son (pitch) et modulation linéaire de l'amplitude. La symétrie, réglable avec (W), vous permet d'utiliser des formes d'ondes symétriques, à peu d'harmoniques, jusqu'à des formes d'ondes très asymétriques, riches en harmoniques.

- **P** : entrée logarithmique d'événement destinée à commander la hauteur de son (fréquence d'oscillateur), par demi-ton ($69 = 440$ Hz).
- **A** : entrée de commande audio destinée à l'amplitude. *
Le signal de sortie varie entre **+A** et **-A**.
- **W** : entrée destinée à commander la symétrie de la forme d'onde :
-1 = dent de scie décroissante, 0 = triangle, +1 = dent de scie croissante.
Plage caractéristique : [0 ... 1]
- **Par** : sortie audio destinée à la forme d'onde parabolique.
- **Tri** : sortie audio destinée à la forme d'onde triangulaire.

Parabol

Oscillator



Oscillateur pour forme d'onde parabolique, à entrée logarithmique de commande de la hauteur de son (pitch) et modulation linéaire de l'amplitude. La forme d'onde est composée de deux moitiés résultant chacune d'une fonction parabolique. L'oscillateur produit un son similaire à une sinusoïde, avec une légère pointe d'harmoniques impaires, ce qui permet souvent de l'utiliser à la place d'un générateur de sinusoïdes tout en chargeant bien moins la CPU.

- **P** : entrée logarithmique d'événement destinée à commander la hauteur de son (fréquence d'oscillateur), par demi-ton ($69 = 440$ Hz).
- **A** : entrée de commande audio destinée à l'amplitude.
Le signal de sortie varie entre **+A** et **-A**.
- **Out** : sortie audio destinée à la forme d'onde parabolique.

Par FM

Oscillator



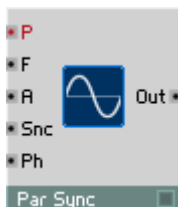
Oscillateur pour forme d'onde parabolique, à entrée logarithmique de commande de la hauteur de son (pitch), modulation linéaire de fréquence (FM) et de l'amplitude. La forme d'onde est composée de deux moitiés résultant chacune d'une fonction parabolique.

L'oscillateur produit un son similaire à une sinusoïde, avec une légère pointe d'harmoniques impaires, ce qui permet souvent de l'utiliser à la place d'un générateur de sinusoïdes tout en chargeant bien moins la CPU.

- **P** : entrée logarithmique d'événement destinée à commander la hauteur de son (fréquence d'oscillateur), par demi-ton ($69 = 440$ Hz).
- **F** : entrée de commande audio destinée à la modulation linéaire de fréquence, en Hz. La fréquence de l'oscillateur est définie à partir de **P** et **F**.
- **A** : entrée de commande audio destinée à l'amplitude. Le signal de sortie varie entre **+A** et **-A**.
- **Out** : sortie audio destinée à la forme d'onde parabolique.

Par Sync

Oscillator



Oscillateur pour forme d'onde parabolique, à entrée de synchronisation, entrée logarithmique de commande de la hauteur de son (pitch), modulation linéaire de fréquence (FM) et de l'amplitude. La forme d'onde est composée de deux moitiés résultant chacune d'une fonction parabolique. L'oscillateur produit un son similaire à une sinusoïde, avec une légère pointe d'harmoniques impaires, ce qui permet souvent de l'utiliser à la place d'un générateur de sinusoïdes tout en chargeant bien moins la CPU.

Dès que le signal de synchronisation dépasse le point zéro dans le sens positif (rampe ascendante), la phase de l'oscillateur se remet à zéro.

- **P** : entrée logarithmique d'événement destinée à commander la hauteur de son (fréquence d'oscillateur), par demi-ton ($69 = 440$ Hz).
- **F** : entrée de commande audio destinée à la modulation linéaire de fréquence, en Hz. La fréquence de l'oscillateur est définie à partir de **P** et **F**.
- **A** : entrée de commande audio destinée à l'amplitude. Le signal de sortie varie entre **+A** et **-A**.
- **Snc** : entrée de commande audio destinée au signal de synchronisation. Lorsque l'onde traverse l'axe horizontal en croissant, ceci redémarre l'oscillateur à zéro.

- **Ph** : entrée permettant de définir la phase (position sur la forme d'onde) à laquelle l'oscillateur est ramené par la synchronisation.
- **Out** : sortie audio destinée à la forme d'onde parabolique.

Par PWM

Oscillator



Oscillateur pour forme d'onde parabolique variable, à entrée logarithmique de commande de la hauteur de son (pitch) et modulation linéaire de l'amplitude. Le rapport entre la longueur des parties inférieure et supérieure de la courbe est modifiable, vous permettant de varier entre une onde parabolique symétrique et une parabole simple.

- **P** : entrée logarithmique d'événement destinée à commander la hauteur de son (fréquence d'oscillateur), par demi-ton ($69 = 440$ Hz).
- **A** : entrée destinée à commander l'amplitude. Le signal de sortie varie entre $+A$ et $-A$.
- **W** : entrée d'événement destinée à commander la symétrie de la forme d'onde. La valeur -1 signifie une parabole ouverte vers le bas, 0 une parabole symétrique normale, et $+1$ une parabole vers le haut.
- **Out** : sortie audio destinée à la forme d'onde parabolique variable.

Sine

Oscillator



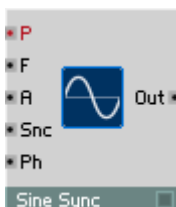
Oscillateur pour forme d'onde sinusoïdale, à entrée logarithmique de commande de la hauteur de son (pitch) et modulation linéaire de l'amplitude. Dans les cas où il est inutile que la sinusoïde soit parfaite, un oscillateur parabolique est suffisant et représente une charge de calcul moins importante.

- **P** : entrée logarithmique d'événement destinée à commander la hauteur de son (fréquence d'oscillateur), par demi-ton ($69 = 440$ Hz).
- **A** : entrée audio destinée à commander l'amplitude. Le signal de sortie varie entre $+A$ et $-A$.
- **Out** : sortie audio destinée à la forme d'onde sinusoïdale.



Oscillateur pour forme d'onde sinusoïdale, à entrée logarithmique de commande de la hauteur de son (pitch), modulation linéaire de fréquence (FM) et de l'amplitude. Dans les cas où il est inutile que la sinusoïde soit parfaite, un oscillateur parabolique est suffisant et représente une charge de calcul moins importante.

- **P** : entrée logarithmique d'événement destinée à commander la hauteur de son (fréquence d'oscillateur), par demi-ton ($69 = 440$ Hz).
- **F** : entrée destinée à la modulation linéaire de fréquence, en Hz. La fréquence de l'oscillateur est définie à partir de **P** et **F**.
- **A** : entrée audio destinée à commander l'amplitude. Le signal de sortie varie entre **+A** et **-A**.
- **Out** : sortie audio destinée à la forme d'onde sinusoïdale.



Oscillateur pour forme d'onde sinusoïdale, à entrée de synchronisation, entrée logarithmique de commande de la hauteur de son (pitch), modulation linéaire de fréquence (FM) et de l'amplitude.

Dès que le signal de synchronisation dépasse le point zéro dans le sens positif (rampe ascendante), la phase de l'oscillateur se remet à une position que vous définissez avec l'entrée de phase.

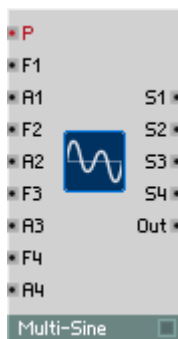
Dans les cas où il est inutile que la sinusoïde soit parfaite, un oscillateur parabolique est suffisant et représente une charge de calcul moins importante.

- **P** : entrée logarithmique d'événement destinée à commander la hauteur de son (fréquence d'oscillateur), par demi-ton ($69 = 440$ Hz).

- **F** : entrée audio destinée à la modulation linéaire de fréquence, en Hz. La fréquence de l'oscillateur est définie à partir de **P** et **F**.
- **A** : entrée audio destinée à commander l'amplitude. Le signal de sortie varie entre **+A** et **-A**.
- **Snc** : entrée audio destinée à la commande du signal de synchronisation. Lorsque l'onde traverse l'axe horizontal en croissant, ceci redémarre l'oscillateur à zéro.
- **Ph** : entrée permettant de définir la phase (position sur la forme d'onde) à laquelle l'oscillateur est ramené par la synchronisation. **Ph** = 0: Phase = 0° (milieu de la partie croissante), **Ph** = 0,5 : Phase = 180° (milieu de la partie décroissante), **Ph** = 1 : Phase = 360° (comme 0°).
- **Out** : sortie audio destinée à la forme d'onde sinusoïdale.

Multi-Sine

Oscillator



Oscillateur destiné à la synthèse additive. Une forme d'onde est composée par superposition de plusieurs sinusoïdes (les harmoniques). Vous pouvez régler séparément, pour chaque composant, l'amplitude (**A**) et le multiple de la fréquence (**F**).

La position de phase des sinusoïdes les unes par rapport aux autres est invariable. Ce qui signifie que la forme d'onde se répète exactement à la fréquence réglée par la valeur de l'entrée **P**.

En règle générale, les multiples de fréquence sont des nombres entiers (numéros d'harmoniques), sinon les sinusoïdes ne sont pas pures.

- **P** : entrée logarithmique d'événement destinée à commander la hauteur de son (fréquence d'oscillateur), par demi-ton (69 = 440 Hz).
- **F1** : entrée audio destinée au facteur de fréquence (numéro d'harmonique) de la première sinusoïde, qui est un multiple de la fréquence de base. Plage caractéristique : [1...20]

- **A1** : entrée linéaire audio destinée à commander l'amplitude de la première sinusoïde. Utilisable également pour trémolos et modulation en anneau. Plage caractéristique : [0 ... 1].
- **F2** : comme **F1**, mais pour la deuxième sinusoïde.
- **A2** : comme **A1**, mais pour la deuxième sinusoïde.
- **F3** : comme **F1**, mais pour la troisième sinusoïde.
- **A3** : comme **A1**, mais pour la troisième sinusoïde.
- **F4** : comme **F1**, mais pour la quatrième sinusoïde.
- **A4** : comme **A1**, mais pour la quatrième sinusoïde.
- **S1** : sortie destinée à la première sinusoïde.
- **S2** : sortie destinée à la deuxième sinusoïde.
- **S3** : sortie destinée à la troisième sinusoïde.
- **S4** : sortie destinée à la quatrième sinusoïde.
- **Out** : sortie audio destinée au signal généré en additionnant les sinusoïdes.

Pulse



Oscillator

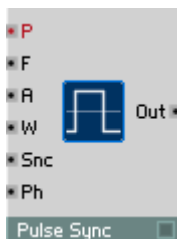
Oscillateur pour forme d'onde rectangulaire, à entrée logarithmique de commande de la hauteur de son (pitch), modulation de largeur d'impulsion (PWM) et modulation linéaire de l'amplitude.

- **P** : entrée logarithmique d'événement destinée à commander la hauteur de son (fréquence d'oscillateur), par demi-ton (69 = 440 Hz).
- **A** : entrée audio destinée à commander l'amplitude. Le signal de sortie varie entre **+A** et **-A**.
- **W** : entrée audio destinée à commander la largeur d'impulsion (PWM). Plage de valeurs -1 à 1. Forme d'onde symétrique (Square Wave) 50:50 avec **W** = 0, rapport haut-bas 33:66 avec -0,33, 66:33 avec 0,33, 75:25 avec 0,5, 90:10 avec 0,8. haut : bas = $(1 + W) : (1 - W)$.
- **Out** : sortie audio destinée à la forme d'onde rectangulaire.



Oscillateur pour forme d'onde rectangulaire, à entrée logarithmique de commande de la hauteur de son (pitch), modulation linéaire de la fréquence (FM), modulation de la largeur d'impulsion (PWM) et modulation linéaire de l'amplitude.

- **P** : entrée logarithmique d'événement destinée à commander la hauteur de son (fréquence d'oscillateur), par demi-ton ($69 = 440$ Hz).
- **F** : entrée audio destinée à la modulation linéaire de fréquence, en Hz. La fréquence de l'oscillateur est définie à partir de **P** et **F**.
- **A** : entrée audio destinée à commander l'amplitude. Le signal de sortie varie entre **+A** et **-A**.
- **W** : entrée audio destinée à moduler la largeur d'impulsion (PWM). Plage de valeurs -1 à 1. Forme d'onde symétrique (Square Wave) 50:50 avec **W** = 0, rapport haut-bas 33:66 avec -0,33, 66:33 avec 0.33, 75:25 avec 0,5, 90:10 avec 0,8. haut : bas = $(1 + W) : (1 - W)$.
- **Out** : sortie audio destinée à la forme d'onde rectangulaire.



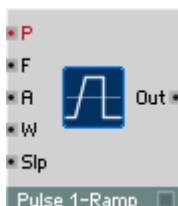
Oscillateur pour forme d'onde rectangulaire, à entrée de synchronisation, entrée logarithmique de commande de la hauteur de son (pitch), modulation linéaire de la fréquence (FM), modulation de la largeur d'impulsion (PWM) et modulation linéaire de l'amplitude.

Dès que le signal de synchronisation dépasse le point zéro dans le sens positif (rampe ascendante), la phase de l'oscillateur se remet à zéro.

- **P** : entrée logarithmique d'événement destinée à commander la hauteur de son (fréquence d'oscillateur), par demi-ton ($69 = 440$ Hz).
- **F** : entrée audio destinée à la modulation linéaire de fréquence, en Hz. La fréquence de l'oscillateur est définie à partir de **P** et **F**.
- **A** : entrée audio destinée à commander l'amplitude. Le signal de sortie varie entre **+A** et **-A**.
- **W** : entrée audio destinée à moduler la largeur d'impulsion (PWM). Plage de valeurs -1 à 1. Forme d'onde symétrique (Square Wave) 50:50 avec **W** = 0, rapport haut-bas 33:66 avec -0,33, 66:33 avec 0.33, 75:25 avec 0,5, 90:10 avec 0,8. haut : bas = $(1 + \mathbf{W}) : (1 - \mathbf{W})$.
- **Snc** : entrée audio destinée à commander le signal de synchronisation. Lorsque l'onde traverse l'axe horizontal en croissant, ceci redémarre l'oscillateur à zéro.
- **Ph** : entrée permettant de définir la phase (position sur la forme d'onde) à laquelle l'oscillateur est ramené par la synchronisation.
- **Out** : sortie audio destinée à la forme d'onde rectangulaire.

Pulse 1-ramp

Oscillator



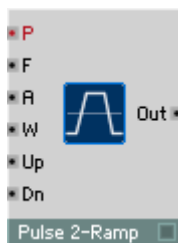
Oscillateur pour forme d'onde rectangulaire trapézoïdale, à entrée logarithmique de commande de la hauteur de son (pitch), modulation linéaire de la fréquence (FM), modulation de la largeur d'impulsion (PWM) et modulation linéaire de l'amplitude. La forme d'onde est un mélange de carré et de dent de scie : vous pouvez modifier la pente de la rampe ascendante et sa durée. La rampe descendante est verticale.

- **P** : entrée logarithmique d'événement destinée à commander la hauteur de son (fréquence d'oscillateur), par demi-ton ($69 = 440$ Hz).
- **F** : entrée audio destinée à la modulation linéaire de fréquence, en Hz. La fréquence de l'oscillateur est définie à partir de **P** et **F**.
- **A** : entrée audio destinée à commander l'amplitude. Le signal de sortie varie entre **+A** et **-A**.

- **W** :entrée audio destinée à moduler la largeur d'impulsion (PWM). Plage de valeurs : -1 à 1.
- **Slp** : entrée audio destinée à commander la pente de la rampe ascendante. Lorsque **Slp** = 0 ou que l'entrée n'est pas connectée, la forme d'onde ne s'élève pas, et la valeur à la sortie est 0, ce qui signifie que vous n'entendez rien. Plage de valeurs caractéristiques : 1 ... 20.
- **Out** : sortie audio destinée à la forme d'onde trapézoïdale.

Pulse 2-ramp

Oscillator



Oscillateur pour forme d'onde trapézoïdale, à entrée logarithmique de commande de la hauteur de son (pitch), modulation linéaire de la fréquence (FM), modulation de la largeur d'impulsion (PWM) et modulation linéaire de l'amplitude.

La forme d'onde est un mélange de carré, de dent de scie et de triangle : vous pouvez aplanir la forme d'onde rectangulaire, car la pente est réglable.

- **P** : entrée logarithmique d'événement destinée à commander la hauteur de son (fréquence d'oscillateur), par demi-ton (69 = 440 Hz).
- **F** : entrée audio destinée à la modulation linéaire de fréquence, en Hz. La fréquence de l'oscillateur est définie à partir de **P** et **F**.
- **A** : entrée audio destinée à commander l'amplitude. Le signal de sortie varie entre **+A** et **-A**.
- **W** : entrée destinée à moduler la largeur d'impulsion (PWM). Plage de valeurs : -1 à 1.
- **Up** : entrée audio destinée à commander la pente de la rampe ascendante.
- **Dn** : entrée audio destinée à commander la pente de la rampe descendante.
- **Out** : sortie audio destinée à la forme d'onde trapézoïdale.



Oscillateur destiné à la forme d'onde rectangulaire à phase nulle, à entrée logarithmique de commande de la hauteur de son (pitch), modulation de largeur d'impulsion (PWM) et modulation linéaire de l'amplitude.

- **P** : entrée logarithmique d'événement destinée à commander la hauteur de son (fréquence d'oscillateur), par demi-ton ($69 = 440$ Hz).
- **A** : entrée audio destinée à commander l'amplitude.
Le signal de sortie varie entre **+A** et **-A**.
- **W** : entrée audio destinée à moduler la largeur d'impulsion (PWM).
Plage de valeurs -0 à 1 . Avec **W** = 0 , vous obtenez une forme d'onde rectangulaire symétrique normale (Square Wave) 50:50, avec une valeur supérieure de **W**, les impulsions sont plus courtes et les phases nulles prolongées.
- **Out** : sortie audio destinée à la forme d'onde rectangulaire bipolaire.



Oscillateur pour forme d'onde d'impulsion à entrée logarithmique de commande de la hauteur de son (pitch) et modulation linéaire de l'amplitude.

- **P** : entrée logarithmique d'événement destinée à commander la hauteur de son (fréquence d'oscillateur), par demi-ton ($69 = 440$ Hz).
- **A** : entrée audio destinée à commander l'amplitude. Le signal de sortie varie entre **+A** et **-A**.
- **Out** : sortie audio destinée à la forme d'onde d'impulsion.

Impulse FM

Oscillator

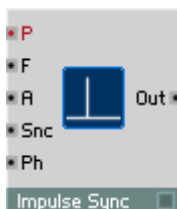


Oscillateur pour forme d'onde d'impulsion, à entrée logarithmique de commande de la hauteur de son (pitch), modulation linéaire de fréquence (FM) et de l'amplitude.

- **P** : entrée logarithmique d'événement destinée à commander la hauteur de son (fréquence d'oscillateur), par demi-ton ($69 = 440$ Hz).
- **F** : entrée audio destinée à la modulation linéaire de fréquence, en Hz. La fréquence de l'oscillateur est définie à partir de **P** et **F**.
- **A** : entrée audio destinée à commander l'amplitude.
- **Out** : sortie audio destinée à la forme d'onde d'impulsion.

Impulse Sync

Oscillator



Oscillateur pour forme d'onde d'impulsion, à entrée de synchronisation, entrée logarithmique de commande de la hauteur de son (pitch), modulation linéaire de fréquence (FM) et de l'amplitude.

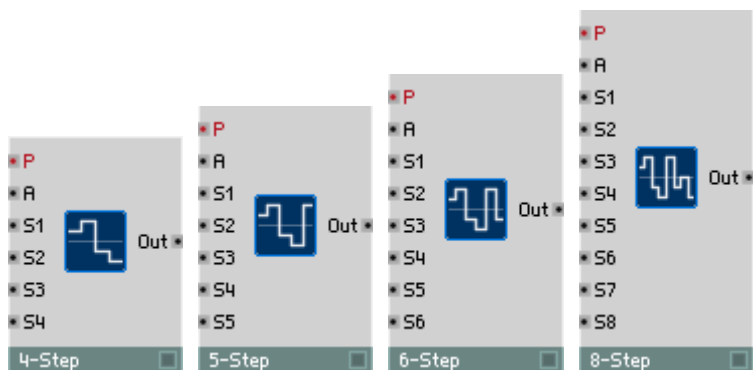
Dès que le signal de synchronisation dépasse le point zéro dans le sens positif (rampe ascendante), la phase de l'oscillateur se remet à zéro.

- **P** : entrée logarithmique d'événement destinée à commander la hauteur de son (fréquence d'oscillateur), par demi-ton ($69 = 440$ Hz).
- **F** : entrée audio destinée à commander la modulation linéaire de fréquence, en Hz. La fréquence de l'oscillateur est définie à partir de **P** et **F**.
- **A** : entrée audio destinée à commander l'amplitude. Le signal de sortie varie entre **+A** et **-A**.
- **Snc** : entrée audio destinée à commander le signal de synchronisation. Lorsque l'onde traverse l'axe horizontal en croissant, ceci redémarre l'oscillateur à zéro.

- **Ph** : entrée permettant de définir la phase (position sur la forme d'onde) à laquelle l'oscillateur est ramené par la synchronisation.
- **Out** : sortie audio destinée à la forme d'onde d'impulsion.

Multi-Step

Oscillator



4-Step

Oscillator

Oscillateur pour forme d'onde à 4 pas, à entrée logarithmique de commande de la hauteur de son (pitch) et modulation linéaire de l'amplitude. Vous pouvez définir la hauteur de chaque pas séparément.

- **P** : entrée logarithmique d'événement destinée à commander la hauteur de son (fréquence d'oscillateur), par demi-ton ($69 = 440$ Hz).
- **A** : entrée destinée à commander l'amplitude. Le signal de sortie varie entre **+A** et **-A**.
- **S1** : entrée audio destinée à commander la hauteur du premier pas.
- **S2** : entrée audio destinée à commander la hauteur du deuxième pas.
- **S3** : entrée audio destinée à commander la hauteur du troisième pas.
- **S4** : entrée audio destinée à commander la hauteur du quatrième pas.
- **Out** : sortie audio destinée à la forme d'onde 4 pas.

5-Step

Oscillator

Identique à l'oscillateur 4 pas, mais à 5 pas.

6-Step

Oscillator

Identique à l'oscillateur 4 pas, mais à 6 pas.

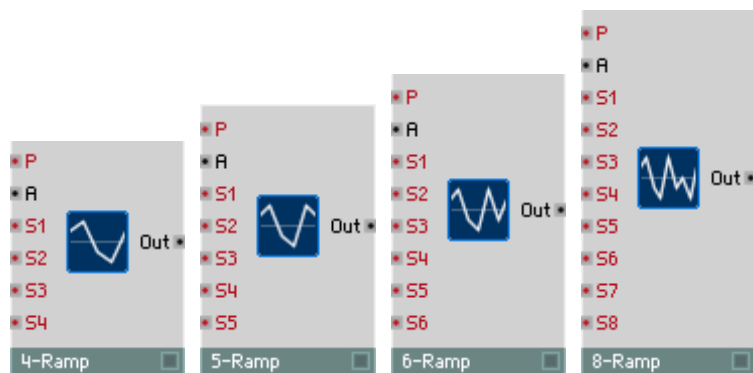
8-Step

Oscillator

Identique à l'oscillateur 4 pas, mais à 8 pas.

Multi-Ramp

Oscillator



4-Ramp

Oscillator

Oscillateur pour forme d'onde à 4 rampes, à entrée logarithmique de commande de la hauteur de son (pitch) et modulation linéaire de l'amplitude. Vous pouvez régler séparément la hauteur de différents points d'arrêt reliés par les rampes (breakpoints).

- **P** : entrée logarithmique d'événement destinée à commander la hauteur de son (fréquence d'oscillateur), par demi-ton ($69 = 440$ Hz).
- **A** : entrée audio destinée à commander l'amplitude. Le signal de sortie varie entre **+A** et **-A**.
- **S1** : entrée d'événement destinée à commander la hauteur du premier point d'arrêt.

- **S2** : entrée d'événement destinée à commander la hauteur du deuxième point d'arrêt.
- **S3** : entrée d'événement destinée à commander la hauteur du troisième point d'arrêt.
- **S4** : entrée d'événement destinée à commander la hauteur du quatrième point d'arrêt.
- **Out** : sortie audio destinée à la forme d'onde 4 rampes.

5-Ramp

Oscillator

Identique à l'oscillateur 4 rampes, mais à 5 rampes.

6-Ramp

Oscillator

Identique à l'oscillateur 4 rampes, mais à 6 rampes.

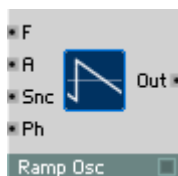
8-Ramp

Oscillator

Identique à l'oscillateur 4 rampes, mais à 8 rampes.

Ramp

Oscillator



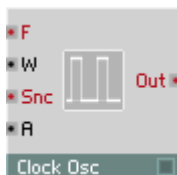
Oscillateur destiné à générer des formes d'onde à rampes, utilisées généralement comme signaux de commande lorsque vous choisissez comme oscillateur de forme d'onde un module Audio Table. Le signal augmente de 0 à A avant de revenir d'un coup à 0.

- **F** : entrée audio destinée à commander la fréquence de l'oscillateur, en Hz. Pour régler la hauteur de son en demi-tons, utilisez un module Event Expon. (F).
- **A** : entrée de commande de l'amplitude du signal de rampe.
Valeur caractéristique : 1.

- **Snc** : entrée audio destinée à commander le signal de synchronisation. Lorsque l'onde traverse l'axe horizontal en croissant, ceci redémarre l'oscillateur à **Ph**.
- **Ph** : entrée audio destinée à la phase de synchronisation. Lorsque l'oscillateur est synchronisé, sa sortie revient à cette valeur multipliée par **A**. Plage caractéristique : 0...1.
- **Out** : sortie audio destinée au signal de rampe. Plage de valeurs : 0...A.

Clock Oscillator

Oscillator

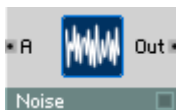


Source indépendante d'impulsion d'horloge. Un oscillateur interne (monophonique) génère des impulsions Clock On/Off régulières sous forme d'événements, afin par ex. de piloter un séquenceur. Vous pouvez régler la valeur des événements On dans les propriétés du module.

- **F** : entrée d'événement destinée à régler la fréquence des impulsions d'horloge, en Hz. Calculez la valeur en Hz, pour pouvoir définir le tempo en BPM (battements par minute), de la manière suivante : $\text{clocks-per-beat} \times \text{BPM}/60$. Pour des impulsions d'horloge à 16^{es} de note à 4 temps par mesure (donc quatre 16^{es} par battement) la valeur est $F = 4/60 \times \text{BPM}$ ou $0,0667 \times \text{BPM}$.
- **W** : entrée d'événement destinée à commander la largeur d'impulsion, c'est-à-dire le rapport entre la durée de la phase On et de la phase Off. Plage de valeurs : -1 à 1. La forme du signal est symétrique (durée égale de On et Off) lorsque **W** = 0 ; $\text{Off:On} = (1 + W) / (1 - W)$.
- **Snc** : entrée d'événement destinée à synchroniser la forme d'onde. Un événement positif synchronise la source du signal d'horloge.
- **A** : entrée d'amplitude. Appliquez-y une valeur supérieure à 0 pour permettre la production d'événements autres que nuls. Valeur caractéristique : 1.
- **Out** : sortie d'événement destiné au signal d'horloge. La valeur varie entre la valeur On et 0.

Noise

Oscillator



Générateur de bruit blanc, qui est un signal aléatoire qui comprend en proportions égales toutes les fréquences possibles. Le signal est composé uniquement de deux valeurs, $A/2$ et $-A/2$, qui se succèdent dans un ordre aléatoire.

- **A** : entrée audio destinée à commander l'amplitude du signal de sortie.
- **Out** : sortie audio destinée au signal du bruit.

Random

Oscillator



Générateur de valeurs aléatoires. Crée une forme d'onde à pas, leur hauteur étant une valeur aléatoire de la plage donnée. Son fonctionnement est comparable à celui d'un générateur de bruit à répartition uniforme suivie d'un circuit d'échantillonnage et de maintien (Sample & Hold) synchronisé à fréquence régulière.

- **P** : entrée logarithmique de commande d'événement pour la hauteur de pas (fréquence Sample & Hold), par demi-ton ($69 = 440$ Hz).
- **A** : entrée audio destinée à commander l'amplitude. Le signal audio de sortie est une valeur aléatoire comprise entre $+A$ et $-A$.
- **Out** : sortie audio destinée au signal de valeur aléatoire.

Geiger

Oscillator



Génère des événements à intervalles aléatoires, de la même manière qu'un compteur Geiger détecte les particules radioactives. Vous pouvez régler la fréquence moyenne des événements à l'entrée **P**. **Rnd** pilote le « caractère aléatoire ».

- **P** : entrée logarithmique de commande d'événement pour la fréquence moyenne (ou la densité) des événements générés au hasard.

- **Rnd** : entrée d'événement commandant l'«aléa » ou la régularité de la répartition des événements dans le temps : 0 = complètement aléatoire, 1 = complètement périodique.
- **Clk** : sortie audio destinée aux clics générés de manière aléatoire.
- **Out** : sortie audio destinée aux événements générés de manière aléatoire. Vous pouvez les utiliser pour, par exemple, déclencher une enveloppe (entrée **G**).

Echantillonneur

Un module qui génère un signal audio de manière autonome sans pour autant être un oscillateur est un échantillonneur. REAKTOR contient de simples lecteurs d'échantillons, mais également des processeurs d'échantillonnage avancés permettant une synthèse granulaire, avec PitchShift, TimeShift et BeatSlicer. Certains modules vous permettent même de sélectionner un échantillon unique à l'aide d'un numéro, via l'entrée Sel.

Page Properties - Fonction

- **Touche de forme d'onde** : cliquez sur la touche portant le symbole de forme d'onde pour ouvrir le Sample Map Editor.
- **Embed Samples In Ensemble** active l'option d'enregistrement d'échantillons avec l'ensemble.
- **No Stereo** inhibe la reproduction stéréo des échantillons. Activez cette fonction pour décharger le processeur.
- Le menu déroulant **Quality** vous permet de commander la qualité de reproduction des échantillons sur trois niveaux (**Poor**, **Good** et **Excellent**). Une grande qualité est irrémédiablement liée à une contrainte plus importante du processeur. Le terme de « Qualité » représente ici l'absence de parasites. Il est évident que ces bruits peuvent également participer à améliorer la qualité d'un échantillon, au sens musical.
- **Oscil. Mode** : activée, cette fonction commute le module en mode oscillateur.

En mode oscillateur, les échantillonneurs attendent des échantillons contenant des *formes d'onde* ou des *jeux d'ondes*. La forme d'onde est la représentation d'une période d'oscillation, donc dans son principe un échantillon très court. Une reproduction répétée de l'échantillon permet de reproduire l'oscillation. Le module échantillonneur devient ainsi un oscillateur numérique. En mode oscillateur, les modules échantillonneurs interprètent les valeurs existant à l'entrée **P** exactement comme le feraient les oscillateurs de REAKTOR, c'est-à-dire comme des numéros de notes MIDI, et ils génèrent une oscillation périodique ayant la hauteur de son correspondante..

Les modules **Sampler** et **Sampler FM** considèrent, en mode oscillateur, l'échantillon entier comme une période. Pour générer un ton d'une hauteur de son de 440 Hertz, l'échantillon entier est parcouru 440 fois en une seconde, indépendamment de sa durée.

Sampler Loop considère, en mode oscillateur, la longueur de boucle comme la période. La longueur de boucle est lue dans le fichier audio, mais vous pouvez la modifier via l'entrée **LL** du module. Un échantillon est ainsi susceptible de contenir plusieurs, et en réalité un nombre quelconque, de formes d'onde ; celles-ci forment alors un jeu d'ondes, un alignement de formes d'onde. Si par exemple une forme d'onde contient 100 valeurs d'échantillonnage, un échantillon gros de 10 000 valeurs d'échantillonnage contient 100 de ces formes d'onde. La première forme d'onde couvre les valeurs de 0 à 99, la deuxième celles de 100 à 199, etc.. Lorsque la longueur de boucle définit la période d'une forme d'onde, la position de la boucle définit logiquement la sélection de la forme d'onde dans le WaveSet.

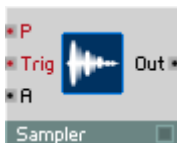
Sampler Loop dispose de l'entrée **LS** pour commander l'origine de la boucle. En mode oscillateur, les valeurs existant à cette entrée sont quantifiées en terme de limites de forme d'onde, garantissant ainsi une transition coulée entre les formes d'onde. Vous pouvez ainsi commander facilement la position dans le WaveSet par le biais de l'enveloppe, de la vélocité, etc. Il est clair que de tels échantillons WaveSet doivent être construits ou générés depuis un échantillon. La bibliothèque de REAKTOR est pleine de nombreux WaveSets des plus intéressants. **Sampler Loop** intègre la synthèse WaveSet dans la gamme des formes de synthèse de REAKTOR, ce qui la met pour la première fois à disposition en combinaison avec la modulation de fréquence.

Page Properties - Appearance

- **Picture** : cette option, activée, fait s'afficher une représentation de la forme d'onde du module échantillonneur dans le panneau.
- **Size X/Size Y** : sert à régler la taille de l'affichage de la forme d'onde du module échantillonneur, en pixel.
- **Scroll Bar** : activez cette option pour afficher une barre de défilement en dessous de l'affichage de la forme d'onde, à des fins de navigation. Cette barre vous permet de vous déplacer dans cette forme d'onde (en la tirant par le milieu) et d'y effectuer des zooms avant et arrière (en la tirant par une de ses extrémités vers la droite ou la gauche).

Sampler

Sampler



Lecteur destiné à la reproduction polyphonique et transposée d'un échantillon ou d'un ensemble d'échantillons (Sample-Map).

Pour gérer les échantillons, utilisez le **Sample Map Editor** (Voir *Sample Map Editor* à la page 167).

Lorsque la boucle est activée, l'échantillon est répété en continu et redémarré par des événements positifs atteignant l'entrée Trigger. Avec la boucle désactivée, et lorsqu'un événement positif atteint l'entrée Trigger, l'échantillon est reproduit une seule fois du début à la fin ou, si le sens de reproduction **Backward** est sélectionné, de la fin au début.

Lorsque **Waveform-Mode** est activé, le taux de reproduction est adapté à la longueur de l'échantillon actuel, afin qu'une hauteur de son appropriée soit obtenue en mode oscillateur de forme d'onde.

- **P** : entrée logarithmique de commande destinée à la hauteur de reproduction (Pitch) et à la sélection d'un échantillon dans le Sample-Map chargé. Lorsque **P** est égal à **Root-Key** de l'échantillon actuel, l'échantillon est reproduit à sa hauteur de son d'origine.
- **Trig** : un événement positif qui atteint cette entrée déclenche (trigger) une nouvelle lecture au début de l'échantillon.
- **A** : entrée destinée à commander l'amplitude de sortie.
- **Out** : sortie du lecteur d'échantillons.

Sampler FM

Sampler



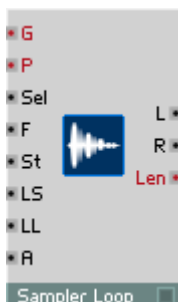
Lecteur destiné à la reproduction polyphonique et transposée d'un échantillon ou d'un ensemble d'échantillons (Sample-Map). L'entrée **F** et l'entrée de commande du point de démarrage (**St**) servent à piloter la reproduction de l'échantillon.

Pour gérer les échantillons, utilisez le **Sample Map Editor** (Voir *Sample Map Editor* à la page 167).

Lorsque la boucle est activée, l'échantillon entier est reproduit en continu ; un événement positif atteignant l'entrée Trigger provoque le retour au point de démarrage défini via l'entrée **St**. Avec la boucle désactivée, et lorsqu'un événement positif atteint l'entrée Trigger, l'échantillon est reproduit une seule fois du point d'origine à la fin ou, si le sens de reproduction **Backward** est sélectionné, dans le sens inverse.

Lorsque **Waveform-Mode** est activé, le taux de reproduction est adapté à la longueur de l'échantillon actuel, afin qu'une hauteur de son appropriée soit obtenue en mode oscillateur de forme d'onde.

- **P** : entrée logarithmique destinée à commander la hauteur de reproduction (Pitch) et à la sélection d'un échantillon dans le Sample-Map chargé. Lorsque **P** est égal à **Root-Key** de l'échantillon actuel, l'échantillon est reproduit à sa hauteur de son d'origine.
- **F** : entrée linéaire destinée à la modulation de la hauteur de reproduction. Il en résulte une modulation de fréquence similaire à celle ayant lieu dans les modules oscillateur de REAKTOR. Des valeurs importantes et négatives provoquent la reproduction en arrière.
- **St** : entrée destinée à commander le point du démarrage suivant ; défini en millisecondes à partir du début de l'échantillon.
- **Trig** : un événement positif qui atteint cette entrée déclenche (trigger) une nouvelle lecture au début de l'échantillon.
- **A** : entrée destinée à commander l'amplitude de sortie.
- **Out** : sortie du lecteur d'échantillons.
- **Lng** : sortie d'événement polyphonique destinée à la longueur de l'échantillon actuel, en millisecondes.



Lecteur universel permettant la reproduction polyphonique et transposée d'échantillons mono et stéréo, de Sample-Maps et de WaveSets.

Pour gérer les échantillons, utilisez le **Sample Map Editor** (Voir *Sample Map Editor* à la page 167).

Lorsque l'événement **Gate** est positif, la reproduction de l'échantillon commence au point de démarrage (défini via l'entrée **St**). Avec la boucle désactivée, l'échantillon est reproduit une seule fois du point d'origine à la fin ou, si le sens de reproduction **Backward** est sélectionné, de la fin au début. Lorsque la boucle est activée, l'échantillon est reproduit à l'infini dans la plage de la boucle dès que la reproduction l'a atteinte. La plage de boucle est définie par des données du fichier audio depuis lequel l'échantillon a été chargé. Si le fichier audio ne contient aucune donnée de boucle, le début de l'échantillon est considéré comme le début de la boucle, et sa longueur comme celle de la boucle. Les données de boucle du fichier audio sont des réglages par défaut (réglages d'origine) qui s'appliquent toujours si l'entrée Loop Start (**LS**) ou l'entrée Loop Length (**LL**) n'est pas connectée à d'autres modules.

Si l'option **Loop in Release** est désactivée, la reproduction de la boucle, en cas d'événement **Gate** négatif ou nul, est interrompue ; l'échantillon passe à la fin ou revient au début, selon le sens de reproduction, puis devient muet. Si l'option **Loop in Release** est activée, ou si la reproduction de la boucle est inactivée, les événements **Gate** négatifs ou nuls n'ont aucun effet.

Si l'option **No Stereo** est activée (via le dialogue Properties), les échantillons stéréo sont traités comme des échantillons mono, et donc le signal émis aux sorties **L** et **R** est identique. **Sampler Loop** utilise alors uniquement le canal gauche des échantillons stéréo. Cette option a été intégrée au logiciel car la reproduction mono représente une contrainte réduite du processeur.

- **G** : un événement positif atteignant cette entrée déclenche la lecture à la position définie à l'entrée **St**. En cas de valeurs négatives, la repro-

duction de la boucle est interrompue sauf si l'option **Loop in Release** est activée.

- **P** : entrée logarithmique destinée à commander la hauteur de reproduction (Pitch). Lorsque **P** est égal à **Root-Key** de l'échantillon actuel, l'échantillon est reproduit à sa hauteur de son d'origine. **P** détermine, si l'entrée **Sel** n'est pas connectée, la sélection des échantillons dans le Sample-Map. En mode **Oscillator, Sampler Loop** fonctionne comme un oscillateur numérique ; **P** détermine alors, comme pour les modules oscillateurs de REAKTOR, la hauteur de son de l'oscillation générée.
- **Sel** : entrée destinée à commander la sélection (Select) d'un échantillon dans le Sample-Map. Si cette entrée n'est pas connectée, les valeurs utilisées sont celles existant à l'entrée **P**.
- **F** : entrée linéaire destinée à la modulation de la hauteur de reproduction. Il en résulte une modulation de fréquence similaire à celle ayant lieu dans les modules oscillateur de REAKTOR. Des valeurs importantes et négatives provoquent la reproduction en arrière.
- **St** : entrée destinée à commander le point du démarrage suivant, défini en millisecondes à partir du début de l'échantillon.
- **LS** : entrée destinée à commander le point d'origine de la boucle, défini en millisecondes à partir du début de l'échantillon. Si cette entrée n'est pas connectée, les données de boucle du fichier audio s'appliquent. Si aucune donnée de boucle n'existe dans le fichier audio, la valeur par défaut est : **LS** = 0. Les modifications ayant lieu à cette entrée s'appliquent lors du redéclenchement d'échantillons ou lorsqu'une limite de boucle est atteinte.
- **LL** : entrée de commande destinée à la longueur de boucle, en millisecondes. Si cette entrée n'est pas connectée, les données de boucle du fichier audio s'appliquent. Si aucune donnée de boucle n'existe dans le fichier audio, la valeur par défaut est : **LL** = longueur de l'échantillon. Les modifications ayant lieu à cette entrée s'appliquent lors du redéclenchement d'échantillons ou lorsqu'une limite de boucle est atteinte.
- **A** : entrée de commande destinée à l'amplitude de sortie.
- **L** : sortie audio destinée au canal gauche du lecteur d'échantillons.
- **R** : sortie audio destinée au canal droit du lecteur d'échantillons. Lors du traitement d'échantillons mono ou lorsque l'option **No Stereo** est activée, le même signal existe aux sorties **L** et **R**.
- **Lng** : sortie d'événement polyphonique destinée à la longueur de l'échantillon actuel, en millisecondes.



Re-synthétiseur en temps réel pour la reproduction polyphonique, transposée d'échantillons mono et stéréo et de Sample Maps. **Resynth** permet le contrôle en temps réel et séparé de la hauteur de son et de la vitesse de reproduction, tout en permettant des manipulations importantes des échantillons.

Les échantillonneurs « normaux » que sont les échantillonneurs matériels connus ou les modules **Sampler**, **Sampler FM** et **Sampler Loop** de REAKTOR présentent un *pointeur* par voix, qui indique la position actuelle dans l'échantillon. La valeur de sortie de l'échantillonneur est toujours l'amplitude de l'échantillon à la position du pointeur. Le pointeur se déplace plus ou moins rapidement dans l'échantillon, générant aux sorties une amplitude relative au temps, donc une oscillation.

La vitesse de déplacement du pointeur détermine la vitesse de reproduction de l'échantillon (par exemple le tempo d'une boucle de battement). Elle définit également la hauteur de son perçue : plus le signal enregistré dans l'échantillon est échantillonné lentement, plus longue sera la période des oscillations générées à la sortie, donc la hauteur de son diminue. Si le pointeur s'immobilise, l'amplitude de sortie ne se modifie plus. Aucun signal audible n'est généré.

Comme les échantillonneurs classiques, **Resynth** utilise un pointeur indiquant la position actuelle dans l'échantillon. Le signal existant aux sorties n'est pourtant pas simplement l'amplitude de l'échantillon à la position du pointeur. En réalité, le signal de sortie est généré par un synthétiseur interne à ce module, qui *re-synthétise* le signal existant à la position du pointeur. La hauteur de son de ce signal créé par le synthétiseur est indépendante de

la vitesse de déplacement du pointeur. Même si celui-ci est immobilisé, le synthétiseur continue à générer un son. Alors que la hauteur de son du signal sortant est définie de manière classique via l'entrée **P**, l'entrée **Sp** définit la vitesse de déplacement du pointeur.

Evidemment, le signal ralenti ou « gelé » ne correspond pas toujours à ce que vous avez pu imaginer : quel son produit le choc d'un marteau sur un clou ralenti à l'infini ? L'algorithme de re-synthèse de **Resynth** a été élaboré pour vous permettre de traiter une gamme étendue de sons de manière sensible et subtile, ou radicale, musicalement parlant. Vous pouvez l'ajuster à l'aide des paramètres **Gr** (Granularity) et **Sm** (Smoothness), qui permettent une adaptation manuelle à l'échantillon et de produire des effets radicalement étranges. Le dialogue Properties vous permet –pour chacun des échantillons d'un Sample Map– d'activer l'option **Signal-Informed Granulation**. Lorsque cette option est activée, l'algorithme de re-synthèse de **Resynth** prend en compte les informations concernant l'échantillon contenues dans l'analyse de celui-ci qui a eu lieu lors du tout premier chargement d'échantillons. Il réagit ainsi aux propriétés du matériau utilisé. Lorsque cette option est désactivée, les résultats ont généralement une sonorité très « électronique ».

Pour gérer les échantillons, utilisez le **Sample Map Editor** (Voir *Sample Map Editor* à la page 167).

Lorsque l'événement **Gate** est positif, la reproduction de l'échantillon commence au point de démarrage (défini via l'entrée **St**). Avec la boucle désactivée, l'échantillon est reproduit une seule fois du début à la fin ou, si le sens de reproduction **Backward** est sélectionné, du point d'origine au début. Lorsque la boucle est activée, l'échantillon est reproduit à l'infini dans la plage de la boucle dès que la reproduction l'a atteinte. La plage de boucle est définie par des données du fichier audio depuis lequel l'échantillon a été chargé. Si le fichier audio ne contient aucune donnée de boucle, le début de l'échantillon est considéré comme le début de la boucle, et sa longueur comme celle de la boucle. Les données de boucle du fichier audio sont des réglages par défaut (réglages d'origine) qui s'appliquent toujours si l'entrée Loop Start (**LS**) ou l'entrée Loop Length (**LL**) ne sont pas connectées à d'autres modules.

Si l'option **Loop in Release** est désactivée, la reproduction de la boucle, en cas d'événement **Gate** négatif ou nul, est interrompue ; l'échantillon passe à la fin ou revient au début, selon le sens de reproduction, puis devient muet. Si l'option **Loop in Release** est activée, ou si la reproduction de la boucle est inactivée, les événements **Gate** négatifs ou nuls n'ont aucun effet.

Si l'option **No Stereo** est activée (via le dialogue Properties), les échantillons stéréo sont traités comme des échantillons mono, et donc le signal émis

aux sorties **L** et **R** est identique. **Resynth** utilise alors uniquement le canal gauche des échantillons stéréo. Cette option a été intégrée au logiciel car la reproduction mono représente une contrainte réduite du processeur.

Toutes les entrées de **Resynth**, à l'exception de **Gate**, sont des entrées audio. Les valeurs existantes sont reprises lors du (re-) déclenchement des échantillons (donc en présence d'événements **Gate** positifs). Les modifications des valeurs s'appliquent pendant la reproduction de l'échantillon, dans l'intervalle **Granularity** (réglable à l'entrée **Gr**).

- **G** : un événement positif atteignant cette entrée déclenche la lecture à la position définie à l'entrée **St**.

En cas de valeurs négatives, la reproduction de la boucle est interrompue sauf si l'option **Loop in Release** est activée.

- **P** : entrée logarithmique audio destinée à commander la hauteur de reproduction (Pitch). Lorsque **P** est égal à **Root-Key** de l'échantillon actuel, l'échantillon est reproduit à sa hauteur de son d'origine. **P** détermine, si l'entrée **Sel** n'est pas connectée, la sélection des échantillons dans le Sample-Map.
- **Sel** : entrée audio destinée à commander la sélection (Select) d'un échantillon dans le Sample-Map. Si cette entrée n'est pas connectée, les valeurs utilisées sont celles existant à l'entrée **P**.
- **St** : entrée audio destinée à commander le point du démarrage suivant ; défini en millisecondes à partir du début de l'échantillon.
- **LS** : entrée audio destinée à commander le point d'origine de la boucle, défini en millisecondes à partir du début de l'échantillon. Si cette entrée n'est pas connectée, les données de boucle du fichier audio s'appliquent. Si aucune donnée de boucle n'existe dans le fichier audio, la valeur par défaut est : **LS** = 0.
- **LL** : entrée audio destinée à commander la longueur de boucle, en millisecondes. Si cette entrée n'est pas connectée, les données de boucle du fichier audio s'appliquent. Si aucune donnée de boucle n'existe dans le fichier audio, la valeur par défaut est : **LL** = longueur de l'échantillon. Lorsque **LL** = 0, le déplacement est suspendu dans l'échantillon lorsque le point d'origine de la boucle, le son est gelé à cet endroit.
- **Sp** : entrée audio destinée à commander la vitesse de lecture (Speed), indépendante de la hauteur de son. Les valeurs en présence sont les coefficients : avec **Sp** = 1, la reproduction a lieu à la vitesse d'origine, **Sp** = 2 correspond à une vitesse double, **Sp** = 0 signifie l'immobilisation. Si cette entrée n'est pas connectée, la vitesse d'origine transposée est

la valeur par défaut, ce qui fait que la vitesse diminue avec la hauteur du son, comme avec les échantillonneurs conventionnels.

- **Gr** : entrée audio destinée à commander la granularité du processus de re-synthèse, en millisecondes. Ce paramètre permet de définir la taille des particules de son utilisées dans la re-synthèse. Lorsque l'option **Signal-Informed Granulation** est activée, les valeurs existantes à l'entrée sont les valeurs par défaut, et les valeurs utilisées réellement sont adaptées au matériel.
- **SO** : entrée audio destinée à moduler la position de l'échantillon (Sample Offset), en millisecondes. Utilisez cette entrée pour moduler la position, dans l'échantillon, indépendamment de la hauteur de son, via un générateur de bruit.
- **Sm** : entrée audio destinée à commander la *Smoothness* (uniformité) du processus de re-synthèse. Cette fonction modifie la forme des particules sonores. En général, des valeurs élevées conduisent à un son rugueux.
- **Pan** : entrée audio destinée à commander la position dans l'espace stéréo (-1 = à gauche, 0 = au milieu, 1 = à droite).
- **A** : entrée audio destinée à commander l'amplitude de sortie.
- **L** : sortie audio destinée au canal gauche du re-synthétiseur.
- **R** : sortie audio destinée au canal droit du re-synthétiseur. Lors du traitement d'échantillons mono ou lorsque l'option **No Stereo** est activée, le même signal existe aux sorties **L** et **R**.
- **Lng** : sortie d'événement polyphonique destinée à la longueur de l'échantillon actuel, en millisecondes.
- **Pos** : sortie d'événement polyphonique destiné à la position actuelle dans l'échantillon, en millisecondes.



Re-synthétiseur en temps réel pour la reproduction polyphonique d'échantillons mono et stéréo, de Sample Maps et de WaveSets. **Pitch Former** est un synthétiseur WaveSet dans lequel vous pouvez charger non seulement des WaveSets, mais également des échantillons quelconques. **Pitch Former** élimine l'évolution de la hauteur de son d'un échantillon et lui en imprime une nouvelle selon vos souhaits. **Pitch Former** vous permet, en plus d'un contrôle en temps réel indépendant, d'effectuer une transposition spectrale, donc une transposition du timbre indépendante de la hauteur de son.

Les échantillonneurs « normaux » que sont les échantillonneurs matériels connus ou les modules **Sampler**, **Sampler FM** et **Sampler Loop** de REAKTOR présentent un *pointeur* par voix, qui indique la position actuelle dans l'échantillon. La valeur de sortie de l'échantillonneur est toujours l'amplitude de l'échantillon à la position du pointeur. Le pointeur se déplace plus ou moins rapidement dans l'échantillon, générant aux sorties une amplitude relative au temps, donc une oscillation.

La vitesse de déplacement du pointeur détermine la vitesse de reproduction de l'échantillon (par exemple le tempo d'une boucle de battement). Elle définit également la hauteur de son perçue : plus le signal enregistré dans l'échantillon est échantillonné lentement, plus longue sera la période des oscillations générées à la sortie, donc la hauteur de son diminue. Si le pointeur s'immobilise, l'amplitude de sortie ne se modifie plus. Aucun signal audible n'est produit.

Comme les échantillonneurs classiques, **Pitch Former** utilise un pointeur indiquant la position actuelle dans l'échantillon. Le signal existant aux sorties

n'est pourtant pas simplement l'amplitude de l'échantillon à la position du pointeur. En réalité, le signal de sortie est généré par un synthétiseur interne à ce module, qui *re-synthétise* le signal existant à la position du pointeur. La hauteur de son de ce signal créé par le synthétiseur est indépendante de la vitesse de déplacement du pointeur. Même si celui-ci est immobilisé, le synthétiseur continue à générer un son. Alors que la hauteur de son du signal sortant est définie de manière classique via l'entrée **P**, l'entrée **Sp** définit la vitesse de déplacement du pointeur.

Alors que les échantillonneurs conventionnels et le module **Resynth** de REAKTOR permettent une modification *relative* de la hauteur de son via la *transposition* d'un échantillon, **Pitch Former** impose à l'échantillon une hauteur de son *absolue et définie*. **Pitch Former** peut ainsi être utilisé comme un oscillateur de REAKTOR. Selon le matériau sonore traité et les réglages utilisés, le résultat produit est plus ou moins teinté « électronique ». **Pitch Former** génère également des signaux à une certaine hauteur de son lorsque l'échantillon traité n'a pas de hauteur de son claire (par exemple des enregistrements de cymbales ou de gongs). Plus la hauteur de son du matériau à traiter est clairement définissable et moins la hauteur originale diffère de celle créée, moins **Pitch Former** ne génère de modification.

Dans les échantillonneurs conventionnels et le module **Resynth** de REAKTOR, la transposition de la hauteur de son est conduite conjointement avec celle de *toutes* les plages spectrales. Ceci est souvent considéré comme une limite, car les plages spectrales que l'auditeur souhaite ne pas voir modifiées sont concernées elles aussi par la transposition. L'effet nasillard résultant du désaccord d'enregistrement de voix est la conséquence de la transposition des formants, dont la position est le plus souvent indépendante de la hauteur de son lorsque le récitant est « humain ». **Pitch Former** sépare, dans une certaine mesure, hauteur de son et position de formant. L'entrée Formant Shift (**FS**) sert à modifier la position du formant indépendamment de la hauteur de son. L'oreille utilise toutes les plages spectrales pour identifier la hauteur du son, vous pouvez donc, en manipulant ce paramètre, générer des inversions intéressantes de hauteurs de son et de timbres, en particulier lorsque les sons sont très bas. Des effets sonore similaires sont souvent créés par les connaisseurs de synthétiseurs via la *synchronisation d'oscillateurs*.

Pour gérer les échantillons, utilisez le **Sample Map Editor** (Voir *Sample Map Editor* à la page 167).

Lorsque l'événement **Gate** est positif, la reproduction de l'échantillon commence au point de démarrage (défini via l'entrée **St**).

Avec la boucle désactivée, l'échantillon est reproduit une seule fois du point d'origine à la fin ou, si le sens de reproduction **Backward** est sélectionné, du point d'origine au début. Lorsque la boucle est activée, l'échantillon est reproduit à l'infini dans la plage de la boucle dès que la reproduction l'a atteinte. La plage de boucle est définie par des données du fichier audio depuis lequel l'échantillon a été chargé. Si le fichier audio ne contient aucune donnée de boucle, le début de l'échantillon est considéré comme le début de la boucle, et sa longueur comme celle de la boucle. Les données de boucle du fichier audio sont des *réglages par défaut* (réglages d'origine) qui s'appliquent toujours si l'entrée Loop Start (**LS**) ou l'entrée Loop Length (**LL**) ne sont pas connectées à d'autres modules.

Si l'option **Loop in Release** est désactivée, la reproduction de la boucle, en cas d'événement **Gate** négatif ou nul, est interrompue ; l'échantillon passe à la fin ou revient au début, selon le sens de reproduction, puis devient muet. Si l'option **Loop in Release** est activée, ou si la reproduction de la boucle est inactivée, les événements **Gate** négatifs ou nuls n'ont aucun effet.

Si l'option **No Stereo** est activée (via le dialogue Properties), les échantillons stéréo sont traités comme des échantillons mono, et donc le signal émis aux sorties **L** et **R** est identique. **Pitch Former** utilise alors uniquement le canal gauche des échantillons stéréo. Cette option a été intégrée au logiciel car la reproduction mono représente une contrainte réduite du processeur.

Toutes les entrées de **Pitch Former**, à l'exception de **Gate**, sont des entrées audio. Les valeurs existantes sont reprises lors du (re-) déclenchement des échantillons (donc en présence d'événements **Gate** positifs). Les modifications des valeurs s'appliquent pendant la reproduction de l'échantillon, dans l'intervalle de la période de base du pitch (réglable via l'entrée **P**).

- **G** : un événement positif atteignant cette entrée déclenche la lecture à la position définie à l'entrée **St**.

En cas de valeurs négatives, la reproduction de la boucle est interrompue sauf si l'option **Loop in Release** est activée.

- **P** : entrée logarithmique audio destinée à commander la hauteur de reproduction (Pitch). **P** détermine, si l'entrée **Sel** n'est pas connectée, la sélection des échantillons dans le Sample-Map.
- **Sel** : entrée audio destinée à commander la sélection (Select) d'un échantillon dans le Sample-Map. Si cette entrée n'est pas connectée, les valeurs utilisées sont celles existant à l'entrée **P**.
- **St** : entrée audio destinée à commander le point du démarrage suivant ; défini en millisecondes à partir du début de l'échantillon.

- **LS** : entrée audio destinée à commander le point d'origine de la boucle, défini en millisecondes à partir du début de l'échantillon. Si cette entrée n'est pas connectée, les données de boucle du fichier audio s'appliquent. Si aucune donnée de boucle n'existe dans le fichier audio, la valeur par défaut est : **LS** = 0.
- **LL** : entrée audio destinée à commander la longueur de boucle, en millisecondes. Si cette entrée n'est pas connectée, les données de boucle du fichier audio s'appliquent. Si aucune donnée de boucle n'existe dans le fichier audio, la valeur par défaut est : **LL** = longueur de l'échantillon. Lorsque **LL** = 0, le déplacement est suspendu dans l'échantillon lorsque le point d'origine de la boucle, le son est gelé à cet endroit.
- **Sp** : entrée audio destinée à commander la vitesse de lecture (Speed), indépendante de la hauteur de son. Les valeurs en présence sont les coefficients : avec **Sp** = 1, la reproduction a lieu à la vitesse d'origine, **Sp** = 2 correspond à une vitesse double, **Sp** = 0 signifie l'immobilisation. Si cette entrée n'est pas connectée, la vitesse d'origine transposée est la valeur par défaut, ce qui fait que la vitesse diminue avec la hauteur du son, comme avec les échantillonneurs conventionnels.
- **FS** : entrée audio destinée à transposer la position du formant par demi-tons, indépendamment de la hauteur de son.
- **SO** : entrée audio destinée à moduler la position de l'échantillon (Sample Offset), en millisecondes. Utilisez cette entrée pour moduler la position, dans l'échantillon, indépendamment de la hauteur de son, via un générateur de bruit.
- **Sm** : entrée audio destinée à commander la *Smoothness* (uniformité) du processus de re-synthèse. Cette fonction modifie la forme des particules sonores. En général, des valeurs élevées conduisent à un son rugueux.
- **Pan** : entrée audio destinée à commander la position dans l'espace stéréo (-1 = à gauche, 0 = au milieu, 1 = à droite).
- **A** : entrée audio destinée à commander l'amplitude de sortie.
- **L** : sortie audio destinée au canal gauche du re-synthétiseur.
- **R** : sortie audio destinée au canal droit du re-synthétiseur. Lors du traitement d'échantillons mono ou lorsque l'option **No Stereo** est activée, le même signal existe aux sorties **L** et **R**.
- **Lng** : sortie d'événement polyphonique destinée à la longueur de l'échantillon actuel, en millisecondes.
- **Pos** : sortie d'événement polyphonique destiné à la position actuelle dans l'échantillon, en millisecondes.



Synthétiseur granulaire stéréo multi-échantillons à commande indépendante de la hauteur de son **P**, du portamento **PS** (Pitch-Slide), de la sélection d'échantillon **Sel**, de la position d'échantillon **Pos** et de la longueur **Len** de chaque grain. Vous pilotez l'enveloppe de chaque grain via l'attaque **Att** et le decay **Dec**.

Vous pouvez régler le différentiel **dt** représentant le temps s'écoulant jusqu'au démarrage du grain suivant. Vous pouvez définir dans Properties le nombre maximum de grains à reproduire simultanément.

Une série d'entrées Jitter servent à définir la plage de valeurs appliquée à l'entrée correspondante.

Pour gérer les échantillons, utilisez le **Sample Map Editor** (Voir *Sample Map Editor* à la page 167).

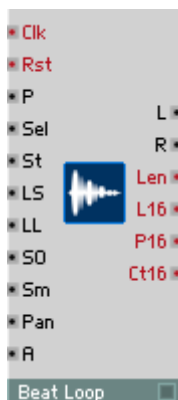
- **Trig** : entrée d'événement destinée au signal Gate. (G) > 0 déclenche immédiatement le grain suivant.
- **P** : entrée audio destinée au contrôle logarithmique de la hauteur de son (par demi-ton). La hauteur de son est indépendante de la vitesse de reproduction. L'échantillon est reproduit à hauteur de son originale lorsque P=Root Key.
Plage caractéristique : [0...127], par défaut : 60.

- **D/F** : entrée audio destinée à commander le sens de reproduction lorsque **P** est connectée. Sinon, elle sert à commander la fréquence. L'échantillon est reproduit à la hauteur de son originale lorsque $F=1$, et en sens inverse lorsque $F=-1$. Plage caractéristique : [-4...4], par défaut : 1.
- **PJ** : entrée audio destinée aux variations aléatoires de la hauteur de son (Pitch Jitter), par demi-ton. Plage caractéristique : [0...3].
- **PS** : entrée audio destinée à la commande logarithmique du décalage de la hauteur de son (Pitch Shift) du grain actuel, par demi-ton. Plage caractéristique : [-3...3].
- **Sel** : entrée audio destinée à la sélection de multi-échantillons (par demi-ton). Lorsque cette entrée n'est pas connectée, les valeurs de l'entrée (**P**) sont utilisées. Plage caractéristique : [0...127].
- **Pos** : entrée audio destinée au réglage de la position dans le fichier audio, en ms. Plage [0...<longueur d'échantillon en ms>].
- **PsJ** : entrée audio destinée à commander les variations aléatoires de position (Position Jitter), en ms. Plage caractéristique : [0...<longueur d'échantillon, en ms>]
- **Len** : entrée audio destinée au réglage de la longueur de grains, en ms. Plage caractéristique : [10...100]. Par défaut : 20 ms.
- **LnJ** : entrée audio destinée à commander la variation aléatoire de longueur (Length Jitter), en ms. Plage caractéristique : [10...100]. Par défaut : 0 ms.
- **Att** : entrée audio destinée au réglage de la durée de l'attaque. Plage caractéristique : [0...1]. Par défaut : 0.2.
- **Dec** : entrée audio destinée au réglage de la durée du decay. Plage caractéristique : [0...1]. Par défaut : 0.2.
- **Dist** : entrée audio destinée au réglage du différentiel de temps s'écoulant jusqu'au démarrage du grain suivant, en ms. Plage caractéristique : [5...100]. Par défaut : 20.
- **DisJ** : entrée audio destinée à commander la modification aléatoire du différentiel de temps (Delta time Jitter). Plage caractéristique : [10...100]. Par défaut : 20 ms.
- **Pan** : entrée audio destinée au réglage de la position dans l'espace stéréo. Plage caractéristique : [-1(à gauche)...1(à droite)].
- **PnJ** : entrée audio destinée à commander la variation aléatoire de la position stéréo (Pan Jitter). Plage caractéristique : [0...1].

- **A** : entrée audio destinée à commander l'amplitude. Plage caractéristique : [0...1]. Par défaut : 1.
- **L** : sortie polyphonique destinée au canal stéréo gauche. Plage caractéristique : [-1...1].
- **R** : sortie polyphonique destinée au canal stéréo droit. Plage caractéristique : [-1...1].
- **Lng** : sortie d'événement destinée à la longueur de l'échantillon, en ms. Plage caractéristique : [0...<longueur d'échantillon en ms>].
- **GTr** : sortie d'événement destinée au déclenchement d'un grain. Emet 1 lorsqu'un grain démarre, 0, lorsqu'il s'interrompt.

Beat Loop

Sampler



Re-synthétiseur en temps réel destiné à la reproduction synchronisée d'échantillons Beatloop. Réglez la transposition des Beatloops via l'entrée **P**, indépendamment de la vitesse de reproduction. **Beat Loop** se cale sur une source d'horloge à 96e de note connectée via l'entrée **C**, on utilise en général le module **1/96 Clock**. Ceci vous permet de synchroniser facilement tous les **Beat Loops** de REAKTOR, indépendamment du tempo propre des échantillons. **Beat Loop** vous permet également de connecter simplement les modules séquenceurs internes à REAKTOR et l'horloge MIDI à des échantillons rythmiques.

Pour gérer les échantillons, utilisez le **Sample Map Editor** (Voir *Sample Map Editor* à la page 167).

Beat Loop requiert des échantillons très exactement découpés, de 2, 4, 8, 16, ou 32, etc. battements. Le tempo du Beatloop utilisé doit être compris entre 87 et 174 BPM. Lorsque les échantillons utilisés remplissent ces conditions,

Beat Loop est en mesure de reproduire les échantillons avec une bonne qualité, même si la vitesse de reproduction est modifiée. En activant l'option **Pitched Sound**, accessible via le dialogue Properties et relative à l'échantillon, vous pouvez éviter des déformations éventuelles des hauteurs de son des lignes de basses ou autres, mais la précision rythmique en souffre.

L'entrée Loop Start (**LS**) et l'entrée Loop Length (**LL**) permettent de régler la plage de boucle de l'échantillon. Si **LS** et **LL** ne sont pas connectées, l'échantillon complet est reproduit dans la boucle. Les événements positifs **Rst** poursuivent la reproduction de l'échantillon depuis le point d'origine (réglable via l'entrée **St**).

Si l'option **No Stereo** est activée (via le dialogue Properties), les échantillons stéréo sont traités comme des échantillons mono, et donc le signal émis aux sorties **L** et **R** est identique. **Beat Loop** utilise alors uniquement le canal gauche des échantillons stéréo. Cette option a été intégrée au logiciel car la reproduction mono représente une contrainte réduite du processeur.

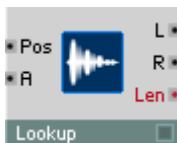
Toutes les entrées de **Beat Loop**, à l'exception de **C** et de **Rst**, sont des entrées audio. Les modifications des valeurs s'appliquent pendant la reproduction de l'échantillon, dans l'intervalle de valeurs de 16es de notes.

- **C** : un événement positif sur cette entrée commute le module **Beat Loop** d'un 96e de note vers l'avant.
- **Rst** : un événement positif sur cette entrée ramène la reproduction à la position réglée à l'entrée **St**.
- **P** : entrée logarithmique audio de commande destinée à la hauteur de reproduction (Pitch). **P** définit aussi, si l'entrée **Sel** n'est pas connectée, quels échantillons du Sample Map vous sélectionnez.
- **Sel** : entrée audio de commande destinée à sélectionner (Select) un échantillon du Sample Map. Si cette entrée n'est pas connectée, les valeurs utilisées sont celles existant à l'entrée **P**.
- **St** : entrée audio de commande destinée au point du démarrage suivant ; défini en seizièmes de notes à partir du début de l'échantillon.
- **LS** : entrée audio de commande destinée au point d'origine de la boucle en seizièmes de notes. Si cette entrée n'est pas connectée, la valeur par défaut est : **LS** = 0.
- **LL** : entrée audio de commande destinée à la longueur de la boucle, en seizièmes de note. Si cette entrée n'est pas connectée, la valeur par défaut est : **LL** = longueur de l'échantillon.

- **SO** : entrée audio de commande destinée à moduler la position de l'échantillon (Sample Offset), en seizièmes de note. Cette entrée sert à obtenir des sauts dans l'échantillon, que vous pouvez ensuite commander via un module séquenceur.
- **Sm** : entrée audio de commande destinée à la *Smoothness* (l'uniformité) du processus de re-synthèse. Cette fonction modifie la forme des particules sonores. Des valeurs très faibles provoquent en règle générale des « claquements » à intervalles de seizième de note.
- **Pan** : entrée audio de commande destinée à la position dans l'espace stéréo (-1 = à gauche, 0 = au milieu, 1 = à droite).
- **A** : entrée audio de commande destinée à l'amplitude de sortie.
- **L** : sortie audio destinée au canal gauche du re-synthétiseur.
- **R** : sortie audio destinée au canal droit du re-synthétiseur. Lors du traitement d'échantillons mono ou lorsque l'option **No Stereo** est activée, le même signal existe aux sorties **L** et **R**.
- **Lng** : sortie d'événement polyphonique destinée à la longueur de l'échantillon actuel, en millisecondes.
- **L16** : sortie d'événement polyphonique destinée à la longueur de l'échantillon actuel, en seizièmes de note.
- **P16** : sortie d'événement polyphonique destinée à la position actuelle dans l'échantillon, en seizièmes de note.
- **Ct16** : sortie d'événement polyphonique destinée à un compteur (Count) des seizièmes de note écoulés depuis Start / Reset.

Sample Lookup

Sampler



Ce module met à votre disposition un échantillon sous forme de table de valeurs de fonction (look-up table). L'entrée **Pos** sert à définir une position dans l'échantillon, en millisecondes. La valeur de l'échantillon à cette position est appliquée aux sorties.

Chargez les fichiers son via le point **Load Audio...** du menu contextuel.

Pour gérer les échantillons, utilisez le **Sample Map Editor** (Voir *Sample Map Editor* à la page 167).

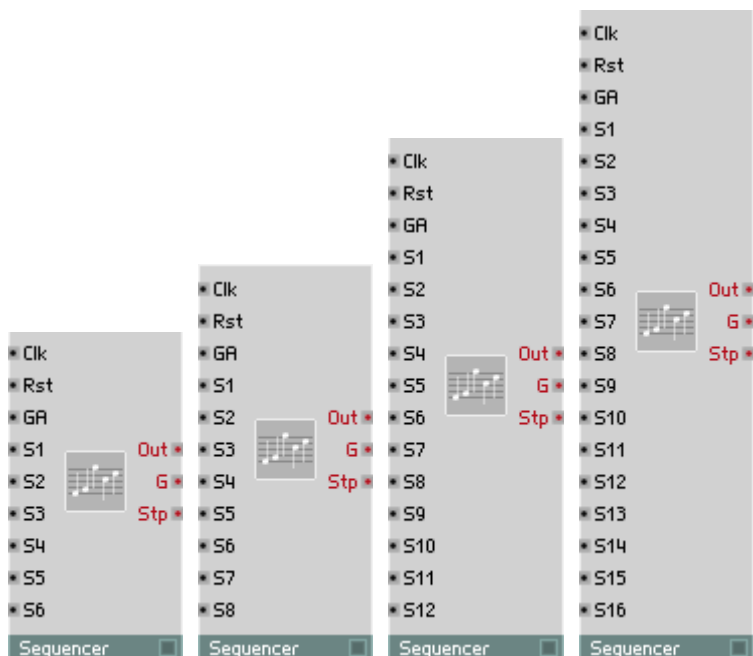
- **Pos** : entrée audio destinée à la position actuelle dans l'échantillon, en millisecondes.
- **A** : entrée audio destinée à la modulation de l'amplitude.
- **L** : sortie audio destinée au canal gauche de l'échantillon.
- **R** : sortie audio destinée au canal droit de l'échantillon. Lors du traitement d'échantillons mono ou lorsque l'option **No Stereo** est activée, le même signal existe aux sorties **L** et **R**.
- **Lng** : sortie d'événement polyphonique destinée à la longueur de l'échantillon actuel, en millisecondes.

Séquenceur

REAKTOR dispose de séquenceurs pas à pas à sortie Gate, de quatre tailles différentes, et d'un module permettant, via une valeur de positionnement, de sélectionner une entrée pour le traitement du signal.

Sequencer

Sequencer



6-Step

Sequencer

Séquenceur à 6 pas. Vous pouvez régler la valeur de sortie de chaque pas (par ex. pour la hauteur de son) séparément. A chaque pas, un signal Gate est également émis à la sortie, d'amplitude égale à la valeur d'entrée actuelle.

- **C** : entrée audio destinée au signal d'horloge (Clock). Lorsque l'onde traverse l'axe horizontal en croissant, le pas suivant s'enclenche. Dans la plupart des cas, on y connecte un oscillateur d'impulsion ou le module MIDI Sync.

- **Rst** : entrée audio destinée au signal d'initialisation (Reset). Lorsque l'onde traverse l'axe horizontal en croissant, le séquenceur revient au premier pas. Dans la plupart des cas, on y connecte une touche ou un module MIDI Start.
- **GA** : entrée audio destinée à commander l'amplitude du signal de sortie Gate. Lorsque la valeur est égale à 0, ou que l'entrée n'est pas connectée, aucun signal n'apparaît à la sortie Gate.
- **S1** : entrée audio destinée à commander la valeur du premier pas.
- **S2** : entrée audio destinée à commander la valeur du deuxième pas.
- **S3** : entrée audio destinée à commander la valeur du troisième pas.
- **S4** : entrée audio destinée à commander la valeur du quatrième pas.
- **S5** : entrée audio destinée à commander la valeur du cinquième pas.
- **S6** : entrée audio destinée à commander la valeur du sixième pas.
- **Out** : sortie de signal d'événement destiné aux pas du séquenceur.
- **G** : sortie de signal d'événement destinée au signal Gate.
- **St** : sortie de signal d'événement destiné au pas actuel.

8-Step

Sequencer

Identique au séquenceur **6-Step** , mais à 8 pas.

12-Step

Sequencer

Identique au séquenceur **6-Step** , mais à 12 pas.

16-Step

Sequencer

Identique au séquenceur **6-Step** , mais à 16 pas.



Commutateur de sélection de valeurs, par ex. pour une utilisation comme séquenceur. Un événement, à l'entrée **Pos**, adresse l'une des 16 entrées avec sa valeur. La valeur actuelle de l'entrée correspondante est transmise à la sortie.

Lorsque l'entrée **Pos** est alimentée par un signal qui croit par paliers, la sortie **Out** émet une séquence des différentes valeurs à destination des entrées **0...15**. Les valeurs de l'entrée **Pos** sont « plissées » dans la plage numérique [0 ... (**Len** - 1)], vous permettant ainsi de générer des séquences de longueur variable (**Len**).

Vous pouvez également piloter l'entrée **Pos** avec un élément de commande, un module **Beat Loop**, une source d'événements aléatoires ou l'horloge maître (module **Song Pos**).

Multiplex 16 est le module que vous connaissez déjà sous le nom **Demux** ou **Select 16**.

- **Pos** : entrée d'événement destinée à commander la sélection. Chaque événement présent à cette entrée génère un événement aux sorties. Pour pouvoir utiliser **Multiplex 16** comme séquenceur, entrez comme valeur de pilotage le nombre de 16es de note écoulés depuis Start/Reset.

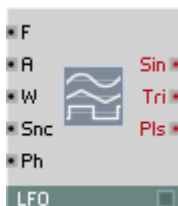
- **Len** :entrée destinée à définir la longueur de la séquence.
Plage caractéristique : [1 ... 16].
- **0-15**: entrée audio destinée à commander la valeur du pas correspondant.
- **Stp** : numéro du pas actuel du séquenceur. La valeur existant à cette sortie se calcule avec la formule « **Pos** modulo **Len** ».
Plage caractéristique : [0...**Len**].
- **Bar** : nombre actuel de mesures. La valeur existant à cette sortie est le résultat d'« Intégrale(**Pos** / **Len**) ».
- **Out** : valeur de sortie du pas actuel du séquenceur.

LFO, Enveloppe

Cette catégorie présente un LFO entièrement modulable, avec plusieurs formes d'onde, un générateur de hasard et des générateurs d'enveloppe de tous types, y compris des générateurs de rampe temps/niveau de trois tailles.

Une option sert à représenter les courbes correspondant à toutes les enveloppes dans un graphique. Vous l'activez via **Visible** de la page **Appearance** des propriétés du module. Réglez la taille de la représentation dans les propriétés, avec **Size X** et **Size Y**. L'axe du temps de la courbe n'est pas mis à l'échelle.

LFO



LFO, Envelope

Low Frequency Oscillator (oscillateur basses fréquences) à sorties pour ondes rectangulaires, triangulaires et sinusoïdales. Il est en règle générale utilisé comme source de signaux de modulation (pour vibrato, trémolo, etc.). Le signal de sortie est un flux d'événements, dont le taux prend la valeur réglée, dans le menu **Settings**, pour **Control Rate**.

Dans la mesure où le LFO fonctionne au Control Rate, il est extrêmement efficace et soumet le processeur à une contrainte bien moindre qu'un oscillateur audio assurant les mêmes fonctions.

- **F** : entrée d'événement destinée à commander la fréquence de l'oscillateur, en Hz. Calculez la valeur en Hz, pour pouvoir définir le tempo en BPM (battements par minute), de la manière suivante : *oscillations par battement* x BPM/60. Pour 3 oscillations par mesure, à 4 temps par mesure ($\frac{3}{4}$ d'oscillation par battement), on obtient, par ex. **F** = $(\frac{3}{4})/60$ x BPM ou 0,0125 x BPM.
- **A** : entrée destinée à commander l'amplitude. Le signal de sortie varie entre **+A** et **-A**.
- **W** : entrée d'événement destinée à commander la largeur d'impulsion, c'est-à-dire le rapport entre la durée de la phase On et de la phase Off. Les autres formes d'onde se décalent de la même manière. La plage de valeurs est de -1 à 1. Le signal est symétrique lorsque **W**=0.

- **Snc** : entrée d'événement destinée à synchroniser la forme d'onde du LFO. Un événement positif synchronise l'oscillateur en réglant la phase sur la valeur existant à l'entrée **Ph**.
- **Ph** : entrée permettant de définir la phase (position sur la forme d'onde) à laquelle l'oscillateur est ramené par la synchronisation. **Ph** = 0: Phase = 0° (milieu de la partie croissante), **Ph** = 0,5 : Phase = 180° (milieu de la partie décroissante), **Ph** = 1 : Phase = 360° (comme 0°).
- **Par** : sortie d'événement destiné au signal à forme d'onde sinusoïdale.
- **Tri** : sortie d'événement destiné au signal à forme d'onde triangulaire.
- **Pls** : sortie d'événement destiné au signal à forme d'onde rectangulaire.

Slow Random

LFO, Envelope

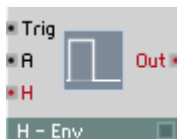


Low Frequency Oscillator (oscillateur basses fréquences) émettant une forme d'onde à pas aléatoire.

- **F** : entrée de commande destinée à la fréquence de pas, en Hz.
- **A** : entrée de commande destinée à l'amplitude. Le signal de sortie varie entre **+A** et **-A**.
- **Out** : sortie d'événement destinée au signal de valeur aléatoire.

H-Env

LFO, Envelope



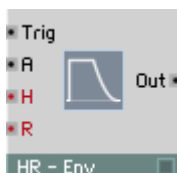
Envelope à fonction Hold. Lorsque l'enveloppe est déclenchée, la valeur de sortie saute à la valeur actuelle de l'entrée d'amplitude et y demeure jusqu'à la fin du temps de maintien (Hold). La sortie revient ensuite à 0. Vous pouvez déclencher à nouveau l'enveloppe à tout moment, même pendant le maintien.

- **T** : entrée audio destinée au déclenchement (Trigger), lorsque la valeur passe de 0 à une valeur positive.

- **A** : entrée audio de commande destinée à la valeur de sortie pendant le maintien.
- **H** : entrée logarithmique d'événement destinée à la commande du temps de maintien (Hold Time).
0 = 1 ms, 20 = 10 ms, 40 = 100 ms (donc, en dB_{1ms}).
- **Out** : sortie audio destinée au signal de l'enveloppe.

HR-Env

LFO, Envelope



Générateur d'enveloppes à fonction maintien-relâchement (Hold-Release). Lorsque l'enveloppe est déclenchée, la valeur de sortie saute à la valeur actuelle de l'entrée d'amplitude et y demeure jusqu'à la fin du temps de maintien (Hold). Le niveau de la sortie décline ensuite de manière exponentielle jusqu'à 0, en fonction de la valeur Release.

- **T** : entrée audio destinée au déclenchement (Trigger), lorsque la valeur passe de 0 à une valeur positive.
- **A** : entrée audio destinée à la valeur de sortie pendant le maintien et de relâchement.
- **H** : entrée logarithmique d'événement destinée à la commande du temps de maintien (Hold Time).
0 = 1 ms, 20 = 10 ms, 40 = 100 ms (donc, en dB_{1ms}).
- **R** : entrée logarithmique d'événement destinée à la commande du temps de relâchement (Release).
0 = 1 ms, 20 = 10 ms, 40 = 100 ms (donc, en dB_{1ms}).
- **Out** : sortie audio destinée au signal de l'enveloppe.



Générateur d'enveloppes à fonction decay. Lorsque l'enveloppe est déclenchée, la valeur de sortie saute à la valeur actuelle de l'entrée d'amplitude, puis diminue ensuite de manière exponentielle jusqu'à 0 en fonction de la valeur Decay.

- **T** : entrée audio destinée au déclenchement (Trigger), lorsque la valeur passe de 0 à une valeur positive.
- **A** : entrée audio destinée à la valeur de sortie.
- **D**: entrée logarithmique d'événement destinée à la commande du temps decay. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (donc, en $\text{dB}_{1\text{ms}}$).
- **Out** : sortie audio destinée au signal de l'enveloppe.



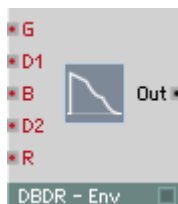
Générateur d'enveloppes à fonction decay et relâchement. Lorsque l'enveloppe est déclenchée via un événement Gate, la valeur de sortie saute à la l'amplitude de cet événement, puis diminue ensuite de manière exponentielle jusqu'à 0 en fonction du temps decay. Un événement Gate d'amplitude 0 (avec Note Off) règle la durée de l'atténuation sur le temps du relâchement.

- **G** : entrée d'événement destinée au signal Gate qui déclenche l'enveloppe (Trigger). L'amplitude de ce signal détermine la première valeur de sortie de la courbe.
- **D**: entrée logarithmique d'événement destinée à la commande du temps decay. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (donc, en $\text{dB}_{1\text{ms}}$).
- **R** : entrée logarithmique d'événement destinée à la commande du temps de relâchement (Release).
0 = 1 ms, 20 = 10 ms, 40 = 100 ms (donc, en $\text{dB}_{1\text{ms}}$).
- **Out** : sortie audio destinée au signal de l'enveloppe.



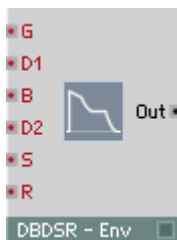
Générateur d'enveloppes à fonction decay, entretien et relâchement. Lorsque l'enveloppe est déclenchée via un événement Gate, la valeur de sortie saute à la l'amplitude de cet événement, diminue ensuite de manière exponentielle, selon le temps decay, jusqu'au niveau d'entretien (sustain, multiplié par l'amplitude), et s'y maintient. La réception d'un événement Gate d'amplitude nulle (Note Off) provoque la diminution exponentielle de la valeur de sortie à 0, en fonction du temps Release.

- **G** : entrée d'événement destinée au signal Gate qui déclenche l'enveloppe (Trigger). L'amplitude de ce signal détermine la première valeur de sortie de la courbe.
- **D**: entrée logarithmique d'événement destinée à la commande du temps decay. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (donc, en $\text{dB}_{1\text{ms}}$).
- **S**: entrée d'événement destinée au niveau d'entretien. La plage de valeurs caractéristique est de 0 (atténuation jusqu'à 0) à 1 (maintien au niveau d'origine), mais l'entretien peut prendre une valeur supérieure à 1, voire même inférieure à 0.
- **R** : entrée logarithmique d'événement destinée à la commande du temps de relâchement (Release). 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (donc, en $\text{dB}_{1\text{ms}}$).
- **Out** : sortie audio destinée au signal de l'enveloppe.



Générateur d'enveloppes à fonction decay, point d'arrêt et relâchement. Lorsque l'enveloppe est déclenchée via un événement Gate, la valeur de sortie saute à l'amplitude de cet événement, puis diminue ensuite de manière exponentielle pendant le temps Decay 1 jusqu'au niveau du point d'arrêt (multiplié par l'amplitude), avant de diminuer ensuite pendant le temps Decay 2. La réception d'un événement Gate d'amplitude nulle (Note Off) provoque la diminution exponentielle de la valeur de sortie à 0, en fonction du temps Release.

- **G** : entrée d'événement destinée au signal Gate qui déclenche l'enveloppe (Trigger). L'amplitude de ce signal détermine la première valeur de sortie de la courbe.
- **D1**: entrée logarithmique d'événement destinée à commander le premier temps decay. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (donc, en $\text{dB}_{1\text{ms}}$).
- **B** : entrée d'événement destinée au niveau du point d'arrêt. Plage de valeurs entre 0 (ne jamais passer à decay 2) et 1 (passer immédiatement à decay 2).
- **D2**: entrée logarithmique d'événement destinée à commander le deuxième temps decay. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (donc, en $\text{dB}_{1\text{ms}}$).
- **R** : entrée logarithmique d'événement destinée à commander le temps de relâchement (Release). 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (donc, en $\text{dB}_{1\text{ms}}$).
- **Out** : sortie audio destinée au signal de l'enveloppe.



Générateur d'enveloppes à fonction decay, point d'arrêt, decay, entretien et relâchement. Lorsque l'enveloppe est déclenchée via un événement Gate, la valeur de sortie saute à la l'amplitude de cet événement, puis diminue ensuite de manière exponentielle pendant le temps Decay 1 jusqu'au niveau du point d'arrêt (multiplié par l'amplitude), avant de diminuer ensuite pendant le temps Decay 2 jusqu'au niveau d'entretien (multiplié par l'amplitude). Elle se maintient alors à cette valeur. La réception d'un événement Gate d'amplitude nulle (Note Off) provoque la diminution exponentielle de la valeur de sortie à 0, en fonction du temps Release.

- **G** : entrée d'événement destinée au signal Gate qui déclenche l'enveloppe (Trigger). L'amplitude de ce signal détermine la première valeur de sortie de la courbe.
- **D1**: entrée logarithmique d'événement destinée à commander le premier temps decay. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (donc, en $\text{dB}_{1\text{ms}}$).
- **B** : entrée d'événement destinée au niveau du point d'arrêt. Plage de valeurs entre 0 (ne jamais passer à decay 2) et 1 (passer immédiatement à decay 2).
- **D2**: entrée logarithmique d'événement destinée à commander le deuxième temps decay. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (donc, en $\text{dB}_{1\text{ms}}$).
- **S**: entrée d'événement destinée au niveau d'entretien. La plage de valeurs caractéristique est de 0 (atténuation jusqu'à 0) à 1 (maintien au niveau d'origine), mais l'entretien peut prendre une valeur supérieure à 1, voire même inférieure à 0.
- **R** : entrée logarithmique d'événement destinée à commander le temps de relâchement (Release).
0 = 1 ms, 20 = 10 ms, 40 = 100 ms (donc, en $\text{dB}_{1\text{ms}}$).
- **Out** : sortie audio destinée au signal de l'enveloppe.

AD-Env

LFO, Envelope

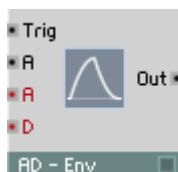


Générateur d'enveloppes à fonction attaque et decay. Lorsque l'enveloppe est déclenchée avec une rampe positive à l'entrée Trigger, la valeur de sortie augmente de manière linéaire pendant la durée de l'attaque jusqu'à l'amplitude de l'entrée A. Le niveau de sortie décline ensuite de manière exponentielle jusqu'à 0, en fonction du temps Decay.

- **T** : entrée audio destinée au déclenchement (Trigger), lorsque la valeur passe de 0 à une valeur positive.
- **A** : entrée audio destinée à la valeur maximum.
- **A** : entrée logarithmique d'événement destinée à commander le temps d'attaque. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (donc, en $\text{dB}_{1\text{ms}}$).
- **D** : entrée logarithmique d'événement destinée à commander le temps decay. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (donc, en $\text{dB}_{1\text{ms}}$).
- **Out** : sortie audio destinée au signal de l'enveloppe.

AR-Env

LFO, Envelope



Générateur d'enveloppes à fonction attaque-relâchement. Lorsque l'enveloppe est déclenchée par un événement Gate, la valeur de sortie augmente de manière linéaire pendant la durée de l'attaque jusqu'à l'amplitude de cet événement. La sortie est maintenue au niveau maximum, pour diminuer jusqu'à 0, suite à un événement Gate d'amplitude 0 (Note Off), de manière exponentielle et en fonction du temps Release.

- **G** : entrée d'événement destinée au signal Gate qui déclenche l'enveloppe (Trigger). L'amplitude de ce signal détermine la valeur de sortie maximum de la courbe.

- **A** : entrée logarithmique d'événement destinée à commander le temps d'attaque. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (donc, en $\text{dB}_{1\text{ms}}$).
- **R** : entrée logarithmique d'événement destinée à commander le temps de relâchement (Release). 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (donc, en $\text{dB}_{1\text{ms}}$).
- **Out** : sortie audio destinée au signal de l'enveloppe.

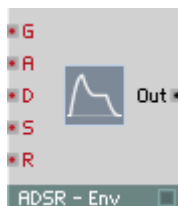
ADR-Env

LFO, Envelope



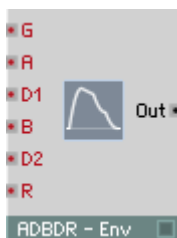
Générateur d'enveloppes à fonction attaque-decay-relâchement. Lorsque l'enveloppe est déclenchée par un événement Gate, la valeur de sortie augmente de manière linéaire pendant la durée de l'attaque jusqu'à l'amplitude de cet événement, puis diminue de manière linéaire jusqu'à 0 pendant le temps decay. Un événement Gate d'amplitude 0 (Note Off) provoque la diminution exponentielle du niveau à 0, pendant le temps Release.

- **G** : entrée d'événement destinée au signal Gate qui déclenche l'enveloppe (Trigger). L'amplitude de ce signal détermine la valeur de sortie maximum de la courbe.
- **A** : entrée logarithmique d'événement destinée à commander le temps d'attaque. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (donc, en $\text{dB}_{1\text{ms}}$).
- **D** : entrée logarithmique d'événement destinée à commander le temps decay. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (donc, en $\text{dB}_{1\text{ms}}$).
- **R** : entrée logarithmique d'événement destinée à commander le temps de relâchement (Release). 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (donc, en $\text{dB}_{1\text{ms}}$).
- **Out** : sortie audio destinée au signal de l'enveloppe.



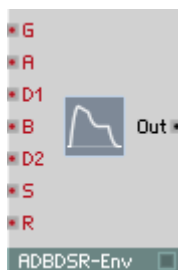
Générateur d'enveloppes à fonction classique attaque, decay, entretien et relâchement. Lorsque l'enveloppe est déclenchée par un événement Gate, la valeur de sortie augmente de manière linéaire pendant la durée de l'attaque jusqu'à l'amplitude de cet événement, puis diminue de manière exponentielle jusqu'au niveau d'entretien (multiplié par l'amplitude) pendant le temps decay. La sortie est maintenue au niveau d'entretien, pour diminuer jusqu'à 0, suite à un événement Gate d'amplitude 0 (Note Off), de manière exponentielle et en fonction du temps Release.

- **G** : entrée d'événement destinée au signal Gate qui déclenche l'enveloppe (Trigger). L'amplitude de ce signal détermine la valeur de sortie maximum de la courbe.
- **A** : entrée logarithmique d'événement destinée à commander le temps d'attaque. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (donc, en $\text{dB}_{1\text{ms}}$).
- **D** : entrée logarithmique d'événement destinée à commander le temps decay. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (donc, en $\text{dB}_{1\text{ms}}$).
- **S** : entrée d'événement destinée au niveau d'entretien. La plage de valeurs caractéristique est de 0 (atténuation jusqu'à 0) à 1 (maintien en fin de la phase d'attaque), mais l'entretien peut prendre une valeur supérieure à 1, voire même inférieure à 0.
- **R** : entrée logarithmique d'événement destinée à commander le temps de relâchement (Release). 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (donc, en $\text{dB}_{1\text{ms}}$).
- **Out** : sortie audio destinée au signal de l'enveloppe.



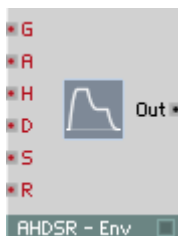
Générateur d'enveloppes à fonction attaque-decay-point d'arrêt-decay-relâchement. Lorsque l'enveloppe est déclenchée via un événement Gate, la valeur de sortie augmente de manière linéaire jusqu'à l'amplitude de cet événement pendant le temps d'attaque, puis diminue ensuite de manière exponentielle pendant le temps Decay 1 jusqu'au niveau du point d'arrêt (multiplié par l'amplitude), et ensuite pendant le temps Decay 2 jusqu'à 0. La réception d'un événement Gate d'amplitude nulle (avec Note Off) provoque la diminution exponentielle de la valeur de sortie à 0, en fonction de la valeur Release.

- **G** : entrée d'événement destinée au signal Gate qui déclenche l'enveloppe (Trigger). L'amplitude de ce signal détermine la valeur de sortie maximum de la courbe.
- **A** : entrée logarithmique d'événement destinée à commander le temps d'attaque. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (donc, en $\text{dB}_{1\text{ms}}$).
- **D1**: entrée logarithmique d'événement destinée à commander le premier temps decay. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (donc, en $\text{dB}_{1\text{ms}}$).
- **B** : entrée d'événement destinée au niveau du point d'arrêt. Plage de valeurs entre 0 (ne jamais passer à decay 2) et 1 (passer immédiatement à decay 2).
- **D2**: entrée logarithmique d'événement destinée à commander le deuxième temps decay. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (donc, en $\text{dB}_{1\text{ms}}$).
- **R** : entrée logarithmique d'événement destinée à commander le temps de relâchement (Release). 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (donc, en $\text{dB}_{1\text{ms}}$).
- **Out** : sortie audio destinée au signal de l'enveloppe.



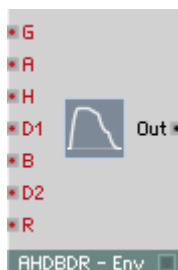
Générateur d'enveloppes à fonction étendue attaque, decay, point d'arrêt, decay, entretien et relâchement. Lorsque l'enveloppe est déclenchée par un événement Gate, la valeur de sortie augmente de manière linéaire pendant la durée de l'attaque jusqu'à l'amplitude de cet événement, puis diminue de manière linéaire jusqu'au niveau du point d'arrêt pendant le premier decay. Ensuite, la courbe se poursuit de manière exponentielle jusqu'au niveau d'entretien, pendant la valeur decay (les niveaux sont fonction de l'amplitude). La sortie est maintenue au niveau d'entretien, pour diminuer jusqu'à 0, suite à un événement Gate d'amplitude 0 (Note Off), de manière exponentielle et en fonction du temps Release.

- **G** : entrée d'événement destinée au signal Gate qui déclenche l'enveloppe (Trigger). L'amplitude de ce signal détermine la valeur de sortie maximum de la courbe.
- **A** : entrée logarithmique d'événement destinée à commander le temps d'attaque. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (donc, en $\text{dB}_{1\text{ms}}$).
- **D1**: entrée logarithmique d'événement destinée à commander le premier temps decay. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (donc, en $\text{dB}_{1\text{ms}}$).
- **B** : entrée d'événement destinée au niveau du point d'arrêt. Plage de valeurs caractéristiques : 0 à 1.
- **D2**: entrée logarithmique d'événement destinée à commander le deuxième temps decay. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (donc, en $\text{dB}_{1\text{ms}}$).
- **S**: entrée d'événement destinée au niveau d'entretien. Plage de valeurs caractéristiques : 0 à 1.
- **R** : entrée logarithmique d'événement destinée à commander le temps de relâchement (Release).
0 = 1 ms, 20 = 10 ms, 40 = 100 ms (donc, en $\text{dB}_{1\text{ms}}$).
- **Out** : sortie audio destinée au signal de l'enveloppe.



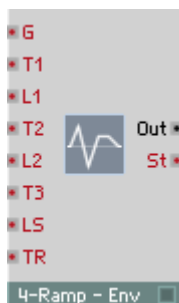
Générateur d'enveloppes à fonction attaque, maintien, decay, entretien et relâchement. Lorsque l'enveloppe est déclenchée par un événement Gate, la valeur de sortie augmente de manière linéaire pendant la durée de l'attaque jusqu'à l'amplitude de cet événement, et y demeure jusqu'à expiration du maintien, pour ensuite diminuer de manière exponentielle jusqu'au niveau d'entretien (multiplié par l'amplitude) pendant le decay. La sortie est maintenue au niveau d'entretien, pour diminuer jusqu'à 0, suite à un événement Gate d'amplitude 0 (Note Off), de manière exponentielle et en fonction du temps Release.

- **G** : entrée d'événement destinée au signal Gate qui déclenche l'enveloppe (Trigger). L'amplitude de ce signal détermine la valeur de sortie maximum de la courbe.
- **A** : entrée logarithmique d'événement destinée à commander le temps d'attaque. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (donc, en $\text{dB}_{1\text{ms}}$).
- **H** : entrée logarithmique d'événement destinée à commander le temps de maintien (Hold Time). 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (donc, en $\text{dB}_{1\text{ms}}$).
- **D** : entrée logarithmique d'événement destinée à commander le temps decay. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (donc, en $\text{dB}_{1\text{ms}}$).
- **S** : entrée d'événement destinée au niveau d'entretien. La plage de valeurs caractéristique est de 0 (atténuation jusqu'à 0) à 1 (maintien en fin de la phase d'attaque), mais l'entretien peut prendre une valeur supérieure à 1, voire même inférieure à 0.
- **R** : entrée logarithmique d'événement destinée à commander le temps de relâchement (Release). 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (donc, en $\text{dB}_{1\text{ms}}$).
- **Out** : sortie audio destinée au signal de l'enveloppe.



Générateur d'enveloppes à fonction attaque-decay-point d'arrêt-decay-relâchement. Lorsque l'enveloppe est déclenchée via un événement Gate, la valeur de sortie augmente jusqu'à l'amplitude de cet événement de manière linéaire pendant le temps d'attaque, y demeure jusqu'à expiration du temps de maintien, puis diminue ensuite de manière exponentielle pendant le temps Decay 1 jusqu'au niveau du point d'arrêt (multiplié par l'amplitude), et diminue jusqu'à 0 pendant le temps Decay 2. La réception d'un événement Gate d'amplitude nulle (Note Off) provoque la diminution exponentielle de la valeur de sortie à 0, en fonction du temps Release.

- **G** : entrée d'événement destinée au signal Gate qui déclenche l'enveloppe (Trigger). L'amplitude de ce signal détermine la valeur de sortie maximum de la courbe.
- **A** : entrée logarithmique d'événement destinée à commander le temps d'attaque. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (donc, en $\text{dB}_{1\text{ms}}$).
- **H** : entrée logarithmique d'événement destinée à commander le temps de maintien (Hold Time).
0 = 1 ms, 20 = 10 ms, 40 = 100 ms (donc, en $\text{dB}_{1\text{ms}}$).
- **D1**: entrée logarithmique d'événement destinée à commander le premier temps decay. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (donc, en $\text{dB}_{1\text{ms}}$).
- **B** : entrée d'événement destinée au niveau du point d'arrêt. Plage de valeurs entre 0 (ne jamais passer à decay 2) et 1 (passer immédiatement à decay 2).
- **D2**: entrée logarithmique d'événement destinée à commander le deuxième temps decay. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (donc, en $\text{dB}_{1\text{ms}}$).
- **R** : entrée logarithmique d'événement destinée à commander le temps de relâchement (Release).
0 = 1 ms, 20 = 10 ms, 40 = 100 ms (donc, en $\text{dB}_{1\text{ms}}$).
- **Out** : sortie audio destinée au signal de l'enveloppe.



Générateur d'enveloppes à quatre phases et transitions linéaires.

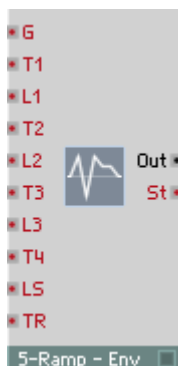
Lorsqu'une enveloppe est déclenchée par un événement Gate, la valeur de sortie augmente jusqu'au premier niveau, de manière linéaire, au cours du premier temps, puis jusqu'au deuxième niveau au cours du deuxième temps, etc. Le troisième niveau est le niveau d'entretien auquel la sortie se maintient jusqu'à réception d'un événement Gate à amplitude 0 (Note Off), à la suite duquel le niveau de la sortie diminue jusqu'à 0 au cours du dernier temps.

- **G** : entrée d'événement destinée au signal Gate qui déclenche l'enveloppe (Trigger). L'amplitude du signal Gate définit, en fonction de toutes les valeurs de niveau, lesquels sont véritablement atteints.
- **T1** : entrée logarithmique d'événement destinée à commander la durée de la première phase.
0 = 1 ms, 20 = 10 ms, 40 = 100 ms (donc, en $\text{dB}_{1\text{ms}}$).
- **L1** : entrée destinée à commander le niveau atteint à la fin de la première phase.
- **T2** : entrée logarithmique d'événement destinée à commander la durée de la deuxième phase.
0 = 1 ms, 20 = 10 ms, 40 = 100 ms (donc, en $\text{dB}_{1\text{ms}}$).
- **L2** : entrée destinée à commander le niveau atteint à la fin de la deuxième phase.
- **T3** : entrée logarithmique d'événement destinée à commander la durée de la troisième phase.
0 = 1 ms, 20 = 10 ms, 40 = 100 ms (donc, en $\text{dB}_{1\text{ms}}$).
- **L3** : entrée destinée à commander le niveau atteint à la fin de la troisième phase. Il s'agit du niveau d'entretien.

- **TR** : entrée logarithmique d'événement destinée à commander la durée de la dernière phase, la phase de relâchement. 0 = 1 ms, 20 = 10 ms, 40 = 100 ms (donc, en $\text{dB}_{1\text{ms}}$).
- **St** : sortie d'événement destinée au numéro de la phase dans laquelle l'enveloppe se trouve actuellement (1, 2 ...). Cette valeur est 0 entre la fin de la phase Release et le début d'un nouvel événement Gate On. Les connexions appropriées permettent d'utiliser cette valeur pour connecter d'autres modules d'enveloppes en série, ou de faire se redéclencher l'enveloppe automatiquement.
- **Out** : sortie audio destinée au signal de l'enveloppe.

5-Ramp

LFO, Envelope

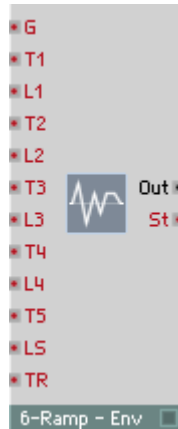


Générateur d'enveloppes à cinq phases et transitions linéaires.

Lorsqu'une enveloppe est déclenchée par un événement Gate, la valeur de sortie augmente jusqu'au premier niveau, de manière linéaire, au cours du premier temps, puis jusqu'au deuxième niveau au cours du deuxième temps, etc. Le quatrième niveau est le niveau d'entretien auquel la sortie se maintient jusqu'à réception d'un événement Gate d'amplitude 0 (Note Off), après lequel le niveau de la sortie diminue jusqu'à 0 au cours du dernier temps.

- **G** : entrée d'événement destinée au signal Gate qui déclenche l'enveloppe (Trigger). L'amplitude du signal Gate définit, en fonction de toutes les valeurs de niveau, lesquels sont véritablement atteints.
- **T1** : entrée logarithmique d'événement destinée à commander la durée de la première phase.
0 = 1 ms, 20 = 10 ms, 40 = 100 ms (donc, en $\text{dB}_{1\text{ms}}$).
- **L1** : entrée destinée à commander le niveau atteint à la fin de la première phase.

- **T2** : entrée logarithmique d'événement destinée à commander la durée de la deuxième phase.
0 = 1 ms, 20 = 10 ms, 40 = 100 ms (donc, en $\text{dB}_{1\text{ms}}$).
- **L2** : entrée destinée à commander le niveau atteint à la fin de la deuxième phase.
- **T3** : entrée logarithmique d'événement destinée à commander la durée de la troisième phase.
0 = 1 ms, 20 = 10 ms, 40 = 100 ms (donc, en $\text{dB}_{1\text{ms}}$).
- **L3** : entrée destinée à commander le niveau atteint à la fin de la troisième phase.
- **T4** : entrée logarithmique d'événement destinée à commander la durée de la quatrième phase.
0 = 1 ms, 20 = 10 ms, 40 = 100 ms (donc, en $\text{dB}_{1\text{ms}}$).
- **LS** : entrée destinée commander le niveau atteint à la fin de la quatrième phase. Il s'agit du niveau d'entretien.
- **TR** : entrée logarithmique d'événement destinée à commander la durée de la dernière phase, la phase de relâchement.
0 = 1 ms, 20 = 10 ms, 40 = 100 ms (donc, en $\text{dB}_{1\text{ms}}$).
- **St** : sortie d'événement destinée au numéro de la phase dans laquelle l'enveloppe se trouve actuellement (1, 2 ...). Cette valeur est 0 entre la fin de la phase Release et le début d'un nouvel événement Gate On. Les connexions appropriées permettent d'utiliser cette valeur pour connecter d'autres modules d'enveloppes en série, ou de faire se redéclencher l'enveloppe automatiquement.
- **Out** : sortie audio destinée au signal de l'enveloppe.



Générateur d'enveloppes à six phases et transitions linéaires.

Lorsqu'une enveloppe est déclenchée par un événement Gate, la valeur de sortie augmente jusqu'au premier niveau, de manière linéaire, au cours du premier temps, puis jusqu'au deuxième niveau au cours du deuxième temps, etc. Le cinquième niveau est le niveau d'entretien auquel la sortie se maintient jusqu'à réception d'un événement Gate à amplitude 0 (Note Off), à la suite duquel le niveau de la sortie diminue jusqu'à 0 au cours du dernier temps.

- **G** : entrée d'événement destinée au signal Gate qui déclenche l'enveloppe (Trigger). L'amplitude du signal Gate définit, en fonction de toutes les valeurs de niveau, lesquels sont véritablement atteints.
- **T1** : entrée logarithmique d'événement destinée à commander la durée de la première phase.
0 = 1 ms, 20 = 10 ms, 40 = 100 ms (donc, en $\text{dB}_{1\text{ms}}$).
- **L1** : entrée destinée à commander le niveau atteint à la fin de la première phase.
- **T2** : entrée logarithmique d'événement destinée à commander la durée de la deuxième phase.
0 = 1 ms, 20 = 10 ms, 40 = 100 ms (donc, en $\text{dB}_{1\text{ms}}$).
- **L2** : entrée destinée à commander le niveau atteint à la fin de la deuxième phase.
- **T3** : entrée logarithmique d'événement destinée à commander la durée de la troisième phase.
0 = 1 ms, 20 = 10 ms, 40 = 100 ms (donc, en $\text{dB}_{1\text{ms}}$).

- **L3** : entrée destinée à commander le niveau atteint à la fin de la troisième phase.
- **T4** : entrée logarithmique d'événement destinée à commander la durée de la quatrième phase.
0 = 1 ms, 20 = 10 ms, 40 = 100 ms (donc, en dB_{1ms}).
- **L4** : entrée destinée à commander le niveau atteint à la fin de la quatrième phase.
- **T5** : entrée logarithmique d'événement destinée à commander la durée de la cinquième phase.
0 = 1 ms, 20 = 10 ms, 40 = 100 ms (donc, en dB_{1ms}).
- **LS** : entrée destinée à commander le niveau atteint à la fin de la cinquième phase. Il s'agit du niveau d'entretien.
- **TR** : entrée logarithmique d'événement destinée à commander la durée de la dernière phase, la phase de relâchement.
0 = 1 ms, 20 = 10 ms, 40 = 100 ms (donc, en dB_{1ms}).
- **St** : sortie d'événement destinée au numéro de la phase dans laquelle l'enveloppe se trouve actuellement (1, 2 ...). Cette valeur est 0 entre la fin de la phase Release et le début d'un nouvel événement Gate On. Les connexions appropriées permettent d'utiliser cette valeur pour connecter d'autres modules d'enveloppes en série, ou de faire se redéclencher l'enveloppe automatiquement.
- **Out** : sortie audio destinée au signal de l'enveloppe.

Filtres

REAKTOR présente un nombre très important de modules de filtrage. Nous en présentons ici 22 types différents ! Cette présentation traite non seulement des modules standard comme passe-bas, passe-haut et passe-bande, mais aussi des émulations des filtres classiques de synthétiseurs que sont Sequential et Moog. Un filtre passe-tout existe également, destiné à générer l'écho et à diffuser les signaux, ainsi que les filtres Integrator et Differentiator.

Tous les filtres de REAKTOR fonctionnent à n'importe quelle fréquence comprise entre 0 Hz (signal constant) et la limite définie par le taux d'échantillonnage. Ils sont ainsi tous appropriés de la même manière au traitement des signaux audio et au lissage des signaux de commande (par ex. Portamento). Lorsque le filtre utilisé pour traiter le signal d'entrée d'un port n'accepte que les événements (par ex. P par opposition à F), il est nécessaire d'intégrer un module convertisseur A à E (perm).

Une option permet à tous les filtres et égaliseurs de représenter l'évolution des fréquences dans un graphique. Vous l'activez via **Visible** de la page **Appearance** des propriétés du module. Réglez la taille de la représentation dans les propriétés, avec **Size X** et **Size Y**. L'axe des fréquences a une échelle logarithmique de 10 Hz à 20 kHz.

HP/LP 1-Pole

Filter



Filtre 1 pôle à sortie passe-haut et passe-bas (raideur de flanc 6 dB/octave) et réglage logarithmique de la fréquence limite.

- **P** : entrée logarithmique d'événement destinée à commander la fréquence limite (fréquence de coupure), par demi-ton ($69 = 440$ Hz).
- **In** : entrée audio destinée au signal à filtrer.
- **HP** : sortie audio destinée au signal du filtre passe-haut (High Pass Filter).
- **LP** : sortie audio destinée au signal du filtre passe-bas (Low Pass Filter).



Filtre 1 pôle à sortie passe-haut et passe-bas (raideur de flanc 6 dB/octave) et réglage logarithmique et linéaire de la fréquence limite.

- **P** : entrée logarithmique d'événement destinée à commander la fréquence limite (fréquence de coupure), par demi-ton (69 = 440 Hz).
- **F** : entrée audio destinée à la commande linéaire de la fréquence limite, en Hz. La fréquence limite est définie à partir de **P** et **F**.
- **In** : entrée audio destinée au signal à filtrer.
- **HP** : sortie audio destinée au signal du filtre passe-haut (High Pass Filter).
- **LP** : sortie audio destinée au signal du filtre passe-bas (Low Pass Filter).

Allpass 1-Pole**Filter**

Filtre passe-tout de premier rang. Ce type de filtre a une raideur de flanc réduite, mais le décalage de phase entre entrée et sortie augmente de 0 degré à basses fréquences à -180 degrés à hautes fréquences. Lorsque la fréquence de coupure pilotée par l'entrée P est atteinte, la phase est décalée de -90 degrés.

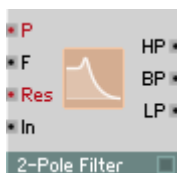
- **P** : entrée logarithmique destinée à commander la fréquence de coupure à laquelle le décalage de phase est de -90 degrés. Echelle : 1 demi-ton par unité, 43 = G1 = 98 Hz, 84 = C5 = 1047 Hz.
- **In** : entrée destinée au signal à filtrer par le filtre passe-tout (phase décalée).
- **Out** : sortie destinée au signal filtré par le filtre passe-tout (phase décalée).



Filtre 2 pôles à sortie passe-haut et passe-bas (raideur de flanc 12 dB/octave), sortie passe-bande (haut et bas, 6 dB/octave), résonance variable de filtre et réglage logarithmique de la fréquence limite. Lorsque la résonance est très importante, vous pouvez aussi utiliser ce filtre comme oscillateur (auto-oscillation).

Le gain de bande passante est toujours 1 (0 dB), alors qu'il augmente, pour la fréquence limite, avec la résonance. Attention : lorsque **Res** a une valeur proche de 1, les amplitudes sont très élevées.

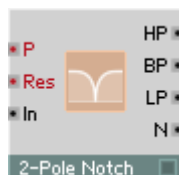
- **P** : entrée logarithmique d'événement destinée à commander la fréquence limite (fréquence de coupure), par demi-ton ($69 = 440$ Hz).
- **Res** : entrée d'événement destinée à commander la résonance/l'atténuation du filtre. Plage de valeurs 0 (atténuation maximum, résonance nulle) à 1 (aucune atténuation, résonance maximum, auto-oscillation). Avec une résonance élevée, le facteur $Q = \text{moyenne de bande [Hz]} / \text{bande passante [Hz]} \cong 1 / (2 - 2 \text{ Res})$.
- **In** : entrée audio destinée au signal à filtrer.
- **HP** : sortie audio destinée au signal du filtre passe-haut (High Pass Filter).
- **BP** : sortie audio destinée au signal du filtre passe-bande (Band Pass Filter).
- **LP** : sortie audio destinée au signal du filtre passe-bas (Low Pass Filter).



Filtre 2 pôles à sortie passe-haut et passe-bas (raideur de flanc 12 dB/octave), sortie passe-bande (haut et bas, 6 dB/octave), résonance variable de filtre et réglage logarithmique et linéaire de la fréquence limite. Lorsque la résonance est très importante, vous pouvez aussi utiliser ce filtre comme oscillateur (auto-oscillation).

Le gain de bande passante est toujours 1 (0 dB), alors qu'il augmente, pour la fréquence limite, avec la résonance. Attention : les amplitudes sont très élevées lorsque **Res** approche la valeur 1.

- **P** : entrée logarithmique d'événement destinée à commander la fréquence limite (fréquence de coupure), par demi-ton ($69 = 440$ Hz).
- **F** : entrée audio destinée à la commande linéaire de la fréquence limite, en Hz. La fréquence limite est définie à partir de **P** et **F**.
- **Res** : entrée d'événement destinée à commander la résonance/l'atténuation du filtre. Plage de valeurs 0 (atténuation maximum, résonance nulle) à 1 (aucune atténuation, résonance maximum, auto-oscillation). Avec une résonance élevée, le facteur $Q = \text{moyenne de bande [Hz]} / \text{bande passante [Hz]} \cong 1 / (2 - 2 \text{ Res})$.
- **In** : entrée audio destinée au signal à filtrer.
- **HP** : sortie audio destinée au signal du filtre passe-haut (High Pass Filter).
- **BP** : sortie audio destinée au signal du filtre passe-bande (Band Pass Filter).
- **LP** : sortie audio destinée au signal du filtre passe-bas (Low Pass Filter).



Filtre 2 pôles à sortie de rejet de bande, sortie passe-haut et passe-bas (raideur de flanc 12 dB/octave), sortie passe-bande (haut et bas, 6 dB/octave), résonance variable de filtre et réglage logarithmique de la fréquence limite.

Le gain de bande passante est toujours 1 (0 dB), alors qu'il diminue, pour la bande passante, lorsque la résonance augmente.

La fréquence réglée est entièrement inhibée à la sortie de rejet de bande.

- **P** : entrée logarithmique d'événement destinée à commander la fréquence limite (fréquence de coupure), par demi-ton ($69 = 440$ Hz).
- **Res** : entrée d'événement destinée à commander la résonance/l'atténuation du filtre. Plage de valeurs 0 (atténuation maximum, résonance nulle) à 1 (aucune atténuation, résonance maximum, auto-oscillation). Avec une résonance élevée, le facteur $Q = \text{moyenne de bande [Hz]} / \text{bande passante [Hz]} \cong 1 / (2 - 2 \text{ Res})$.
- **In** : entrée audio destinée au signal à filtrer.
- **HP** : sortie audio destinée au signal du filtre passe-haut (High Pass Filter).
- **BP** : sortie audio destinée au signal du filtre passe-bande (Band Pass Filter).
- **LP** : sortie audio destinée au signal du filtre passe-bas (Low Pass Filter).
- **N** : sortie audio destinée au signal de rejet de bande (Band-Reject-/Notch-Filter).

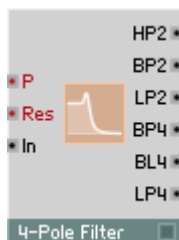


Filtre 2 pôles à sortie de rejet de bande, sortie passe-haut et passe-bas (raideur de flanc 12 dB/octave), sortie passe-bande (haut et bas, 6 dB/octave), résonance variable de filtre et réglage logarithmique et linéaire de la fréquence limite.

Le gain de bande passante est toujours 1 (0 dB), alors qu'il diminue, pour la bande passante, lorsque la résonance augmente.

La fréquence réglée est entièrement inhibée à la sortie de rejet de bande.

- **P** : entrée logarithmique d'événement destinée à commander la fréquence limite (fréquence de coupure), par demi-ton (69 = 440 Hz).
- **F** : entrée audio destinée à la commande linéaire de la fréquence limite, en Hz. La fréquence est définie à partir de **P** et **F**.
- **Res** : entrée d'événement destinée à commander la résonance/l'atténuation du filtre. Plage de valeurs 0 (atténuation maximum, résonance nulle) à 1 (aucune atténuation, résonance maximum, auto-oscillation). Avec une résonance élevée, le facteur $Q = \text{moyenne de bande [Hz]} / \text{bande passante [Hz]} \cong 1 / (2 - 2 \text{ Res})$.
- **In** : entrée audio destinée au signal à filtrer.
- **HP** : sortie audio destinée au signal du filtre passe-haut (High Pass Filter).
- **BP** : sortie audio destinée au signal du filtre passe-bande (Band Pass Filter).
- **LP** : sortie audio destinée au signal du filtre passe-bas (Low Pass Filter).
- **N** : sortie audio destinée au signal de rejet de bande (Band-Reject-/Notch-Filter).



Filtre 4 pôles à sorties passe-bas (24 dB/octave), passe-bande (12/12 dB/octave) et passe-bande/passe-bas (6/18 dB/octave), sorties 2 pôles passe-haut, passe-bas (12 dB/octave) et passe-bande (6/6 dB/octave), résonance variable de filtre et réglage logarithmique de la fréquence limite.

Le gain de bande passante est toujours 1 (0 dB), alors qu'il augmente, pour la fréquence limite, avec la résonance. Attention : les amplitudes sont très élevées lorsque **Res** approche la valeur 1.

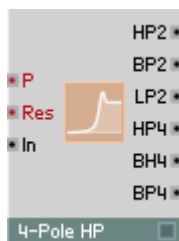
- **P** : entrée logarithmique d'événement destinée à commander la fréquence limite (fréquence de coupure), par demi-ton ($69 = 440$ Hz).
- **Res** : entrée d'événement destinée à commander la résonance/l'atténuation du filtre. Plage de valeurs 0 (atténuation maximum, résonance nulle) à 1 (aucune atténuation, résonance maximum, auto-oscillation).
- **In** : entrée audio destinée au signal à filtrer.
- **HP2** : sortie audio destinée au signal du filtre passe-haut à 2 pôles.
- **BP2** : sortie audio destinée au signal du filtre passe-bande à 2 pôles.
- **LP2** : sortie audio destinée au signal du filtre passe-bas à 2 pôles.
- **BP4** : sortie audio destinée au signal du filtre passe-bande à 4 pôles.
- **BL4** : sortie audio destinée au signal passe-bande/passe-bas.
- **LP4** : sortie audio destinée au signal du filtre passe-bas à 4 pôles.



Filtre 4 pôles à sorties passe-haut (24 dB/octave), passe-bande (12/12 dB/octave) et passe-bande/passe-haut (18/6 dB/octave), sorties 2 pôles passe-haut, passe-bas (12 dB/octave) et passe-bande (6/6 dB/octave), résonance variable de filtre et réglage logarithmique et linéaire de la fréquence limite.

Le gain de bande passante est toujours 1 (0 dB), alors qu'il augmente, pour la fréquence limite, avec la résonance. Attention : les amplitudes sont très élevées lorsque **Res** approche la valeur 1.

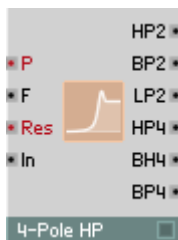
- **P** : entrée logarithmique d'événement destinée à commander la fréquence limite (fréquence de coupure), par demi-ton ($69 = 440$ Hz).
- **F** : entrée audio destinée à la commande linéaire de la fréquence limite, en Hz. La fréquence limite est définie à partir de **P** et **F**.
- **Res** : entrée d'événement destinée à commander la résonance/l'atténuation du filtre. Plage de valeurs 0 (atténuation maximum, résonance nulle) à 1 (aucune atténuation, résonance maximum, auto-oscillation).
- **In** : entrée audio destinée au signal à filtrer.
- **HP2** : sortie audio destinée au signal du filtre passe-haut à 2 pôles.
- **BP2** : sortie audio destinée au signal du filtre passe-bande à 2 pôles.
- **LP2** : sortie audio destinée au signal du filtre passe-bas à 2 pôles.
- **HP4** : sortie audio destinée au signal du filtre passe-haut à 4 pôles.
- **BH4** : sortie audio destinée au signal passe-bande/passe-haut.
- **BP4** : sortie audio destinée au signal du filtre passe-bande à 4 pôles.



Filtre 4 pôles à sorties passe-bas (24 dB/octave), passe-bande (12/12 dB/octave) et passe-bande/passe-bas (6/18 dB/octave), sorties 2 pôles passe-haut, passe-bas (12 dB/octave) et passe-bande (6/6 dB/octave), résonance variable de filtre et réglage logarithmique de la fréquence limite.

Le gain de bande passante est toujours 1 (0 dB), alors qu'il augmente, pour la fréquence limite, avec la résonance. Attention : les amplitudes sont très élevées lorsque **Res** approche la valeur 1.

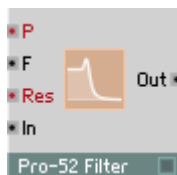
- **P** : entrée logarithmique d'événement destinée à commander la fréquence limite (fréquence de coupure), par demi-ton ($69 = 440$ Hz).
- **Res** : entrée d'événement destinée à commander la résonance/l'atténuation du filtre. Plage de valeurs 0 (atténuation maximum, résonance nulle) à 1 (aucune atténuation, résonance maximum, auto-oscillation).
- **In** : entrée audio destinée au signal à filtrer.
- **HP2** : sortie audio destinée au signal du filtre passe-haut à 2 pôles.
- **BP2** : sortie audio destinée au signal du filtre passe-bande à 2 pôles.
- **LP2** : sortie audio destinée au signal du filtre passe-bas à 2 pôles.
- **BP4** : sortie audio destinée au signal du filtre passe-bande à 4 pôles.
- **BL4** : sortie audio destinée au signal passe-bande/passe-bas.
- **LP4** : sortie audio destinée au signal du filtre passe-bas à 4 pôles.



Filtre 4 pôles à sorties passe-haut (24 dB/octave), passe-bande (12/12 dB/octave) et passe-bande/passe-haut (18/6 dB/octave), sorties 2 pôles passe-haut, passe-bas (12 dB/octave) et passe-bande (6/6 dB/octave), résonance variable de filtre et réglage logarithmique et linéaire de la fréquence limite.

Le gain de bande passante est toujours 1 (0 dB), alors qu'il augmente, pour la fréquence limite, avec la résonance. Attention : les amplitudes sont très élevées lorsque **Res** approche la valeur 1.

- **P** : entrée logarithmique d'événement destinée à commander la fréquence limite (fréquence de coupure), par demi-ton (69 = 440 Hz).
- **F** : entrée audio destinée à la commande linéaire de la fréquence limite, en Hz. La fréquence limite est définie à partir de **P** et **F**.
- **Res** : entrée d'événement destinée à commander la résonance/l'atténuation du filtre. Plage de valeurs 0 (atténuation maximum, résonance nulle) à 1 (aucune atténuation, résonance maximum, auto-oscillation).
- **In** : entrée audio destinée au signal à filtrer.
- **HP2** : sortie audio destinée au signal du filtre passe-haut à 2 pôles.
- **BP2** : sortie audio destinée au signal du filtre passe-bande à 2 pôles.
- **LP2** : sortie audio destinée au signal du filtre passe-bas à 2 pôles.
- **HP4** : sortie audio destinée au signal du filtre passe-haut à 4 pôles.
- **BH4** : sortie audio destinée au signal passe-bande/passe-haut.
- **BP4** : sortie audio destinée au signal du filtre passe-bande à 4 pôles.



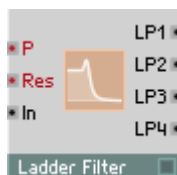
Filtre du synthétiseur analogique Pro 52. Il s'agit d'un filtre passe-bas à 4 pôles (24 dB/octave) à résonance variable de filtre, réglage logarithmique et linéaire de fréquence limite.

Il passe en mode auto-oscillation lorsque **Res** est proche de 1. L'auto-oscillation a une amplitude proche de 1.

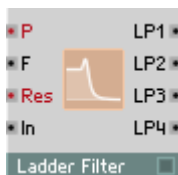
- **P** : entrée logarithmique d'événement destinée à commander la fréquence limite (fréquence de coupure), par demi-ton ($69 = 440$ Hz).
- **F** : entrée audio destinée à la commande linéaire de la fréquence limite, en Hz. La fréquence limite est définie à partir de **P** et **F**.
- **Res** : entrée d'événement destinée à commander la résonance/l'atténuation du filtre. Plage de valeurs 0 (atténuation maximum, résonance nulle) à 1 (aucune atténuation, résonance maximum, auto-oscillation).
- **In** : entrée audio destinée au signal à filtrer.
- **Out** : sortie audio destinée au signal du filtre passe-bas à 4 pôles.

Ladder Filter

Filter



Absolument identique à **Ladder Filter FM**, mais sans l'entrée destinée à la commande de la modulation de fréquence. Consultez la description du module suivant.

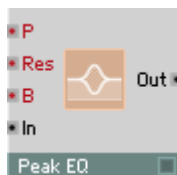


Filtre reposant sur le circuit en échelle, classique et breveté, de Bob Moog. Il s'agit d'un filtre 4 pôles à différentes sorties passe-bas : 24 dB/octave, 18 dB/octave, 12 dB/octave, 6 dB/octave. Il dispose également d'une résonance réglable et d'un réglage logarithmique et linéaire de la fréquence limite.

Une option permet de simuler la caractéristique de saturation du circuit analogique, en activant **Distortion**, dans les propriétés. Avec **Distortion** activée, le filtre passe en auto-oscillation lorsque **Res** est égale ou supérieure à 1. L'amplitude de l'auto-oscillation est environ égale à 1 lorsque **Res** est immédiatement supérieure à 1, mais peut aussi prendre des valeurs bien supérieures si **Res** prend des valeurs plus élevées.

Ce filtre dispose également d'options de réglage qualitatif de la simulation : **Standard**, **High** et **Excellent**. Cet effet est particulièrement perceptible lorsque **Distortion** est active. Un meilleur réglage de la qualité implique évidemment une contrainte plus importante de la CPU.

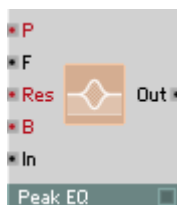
- **P** : entrée logarithmique d'événement destinée à commander la fréquence limite (fréquence de coupure), par demi-ton (69 = 440 Hz).
- **F** : entrée audio destinée à la commande linéaire de la fréquence limite, en Hz. La fréquence limite est définie à partir de **P** et **F**.
- **Res** : entrée d'événement destinée à commander la résonance/l'atténuation du filtre. Plage de valeurs 0 (atténuation maximum, résonance nulle) à 1 (aucune atténuation, résonance maximum, auto-oscillation). Des valeurs supérieures à 1 sont possibles en mode Distorsion.
- **In** : entrée audio destinée au signal à filtrer.
- **LP1** : sortie audio destinée au signal du filtre passe-bas à 6 dB/octave.
- **LP2** : sortie audio destinée au signal du filtre passe-bas à 12 dB/octave.
- **LP3** : sortie audio destinée au signal du filtre passe-bas à 18 dB/octave.
- **LP4** : sortie audio destinée au signal du filtre passe-bas à 24 dB/octave.



Egaliseur paramétrique à accentuation/atténuation et bande passante réglables, et fréquence à réglage logarithmique.

Peak EQ vous permet d'amplifier ou d'atténuer une certaine fréquence et une plage plus au moins importante autour de celle-ci. Les fréquences plus lointaines ne sont pas concernées.

- **P** : entrée logarithmique d'événement destinée à commander la fréquence, par demi-ton (69 = 440 Hz).
- **Res** : entrée d'événement destinée à commander la résonance (facteur Q) du filtre. Plage de valeurs de 0 (résonance minimum, bande passante maximum) à 1 (résonance maximum, bande passante minimale). Avec une résonance élevée, le facteur $Q = \text{moyenne de bande [Hz]} / \text{bande passante [Hz]} \approx 1 / (2 - 2 \text{ Res})$.
- **B** : entrée d'événement destinée à commander l'accentuation/l'atténuation (Boost/Cut), en dB. **B** = 0 signifie que le signal n'est pas modifié.
- **In** : entrée audio destinée au signal d'entrée de l'égaliseur.
- **Out** : sortie audio destinée au signal de sortie de l'égaliseur.



Egaliseur paramétrique à accentuation/atténuation et bande passante réglables, et fréquence à réglage logarithmique et linéaire.

Peak EQ vous permet d'amplifier ou d'atténuer une certaine fréquence et une plage plus ou moins importante autour de celle-ci. Les fréquences plus lointaines ne sont pas concernées.

- **P** : entrée logarithmique d'événement destinée à commander la fréquence, par demi-ton ($69 = 440$ Hz).
- **F** : entrée audio destinée à la commande linéaire de la fréquence, en Hz. La fréquence est définie à partir de **P** et **F**.
- **Res** : entrée d'événement destinée à commander la résonance (facteur Q) du filtre. Plage de valeurs de 0 (résonance minimum, bande passante maximum) à 1 (résonance maximum, bande passante minimale). Avec une résonance élevée, le facteur $Q = \text{moyenne de bande [Hz]} / \text{bande passante [Hz]} \approx 1 / (2 - 2 \text{ Res})$.
- **B** : entrée d'événement destinée à commander l'accentuation/l'atténuation (Boost/Cut), en dB. **B** = 0 signifie que le signal n'est pas modifié.
- **In** : entrée audio destinée au signal d'entrée de l'égaliseur.
- **Out** : sortie audio destinée au signal de sortie de l'égaliseur.

High Shelf EQ

Filter



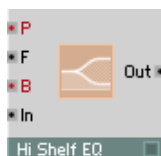
Egaliseur paramétrique high-shelving, à accentuation/atténuation et bande passante réglables, et fréquence à réglage logarithmique.

Les fréquences situées au-dessus de la fréquence de coupure sont accentuées ou diminuées de la valeur réglée, les fréquences inférieures ne sont pas concernées.

- **P** : entrée logarithmique d'événement destinée à commander la fréquence de coupure, par demi-ton ($69 = 440$ Hz).
- **B** : entrée d'événement destinée à commander l'accentuation/l'atténuation (Boost/Cut), en dB. **B** = 0 signifie que le signal n'est pas modifié.
- **In** : entrée audio destinée au signal d'entrée de l'égaliseur.
- **Out** : sortie audio destinée au signal de sortie de l'égaliseur.

High Shelf EQ FM

Filter



Egaliseur paramétrique high-shelving, à accentuation/atténuation et fréquence à réglage logarithmique et linéaire.

Les fréquences situées au-dessus de la fréquence de coupure sont accentuées ou diminuées de la valeur réglée, les fréquences inférieures ne sont pas concernées.

- **P** : entrée logarithmique d'événement destinée à commander la fréquence de coupure, par demi-ton ($69 = 440$ Hz).
- **F** : entrée audio destinée à la commande linéaire de la fréquence de coupure, en Hz. La fréquence est définie à partir de **P** et **F**.
- **B** : entrée d'événement destinée à commander l'accentuation/l'atténuation (Boost/Cut), en dB. **B** = 0 signifie que le signal n'est pas modifié.
- **In** : entrée audio destinée au signal d'entrée de l'égaliseur.
- **Out** : sortie audio destinée au signal de sortie de l'égaliseur.

Low Shelf EQ

Filter



Egaliseur paramétrique low-shelving, à accentuation/atténuation réglables, et fréquence à réglage logarithmique.

Les fréquences situées en dessous de la fréquence de coupure sont accentuées ou diminuées de la valeur réglée, les fréquences supérieures ne sont pas concernées.

- **P** : entrée logarithmique d'événement destinée à commander la fréquence de coupure, par demi-ton ($69 = 440$ Hz).
- **B** : entrée d'événement destinée à commander l'accentuation/l'atténuation (Boost/Cut), en dB. **B** = 0 signifie que le signal n'est pas modifié.
- **In** : entrée audio destinée au signal d'entrée de l'égaliseur.
- **Out** : sortie audio destinée au signal de sortie de l'égaliseur.

Low Shelf EQ FM

Filter



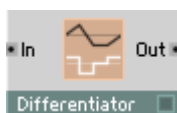
Egaliseur paramétrique low-shelving, à accentuation/atténuation réglables et fréquence à réglage logarithmique et linéaire.

Les fréquences situées en dessous de la fréquence de coupure sont accentuées ou diminuées de la valeur réglée, les fréquences supérieures ne sont pas concernées.

- **P** : entrée logarithmique d'événement destinée à commander la fréquence de coupure, par demi-ton ($69 = 440$ Hz).
- **F** : entrée audio destinée à la commande linéaire de la fréquence de coupure, en Hz. La fréquence est définie à partir de **P** et **F**.
- **B** : entrée d'événement destinée à commander l'accentuation/l'atténuation (Boost/Cut), en dB. **B** = 0 signifie que le signal n'est pas modifié.
- **In** : entrée audio destinée au signal d'entrée de l'égaliseur.
- **Out** : sortie audio destinée au signal de sortie de l'égaliseur.

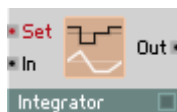
Differentiator

Filter



Le Differentiator définit, à la sortie, la pente du signal d'entrée, en « unités par milliseconde ». Il agit donc comme un filtre passe-haut dont l'amplification augmente proportionnellement à la fréquence. A 159 Hz, l'amplification est égale à 1.

- **In** : entrée destinée au signal à différencier.
- **Out** : sortie destinée au signal différencié.



L'Integrator définit un signal de sortie dont la pente est fonction de l'amplitude du signal d'entrée, en « unités par milliseconde ». Il agit donc comme un filtre passe-bas dont l'amplification diminue proportionnellement à la fréquence. A 159 Hz, l'amplification est égale à 1. Un événement apparaissant à l'entrée commute d'un coup la sortie à la valeur de l'événement.

- **Set** : un événement apparaissant à cette entrée commute la sortie de l'Integrator à la valeur de l'événement.
- **In** : entrée destinée au signal à intégrer. Elle pilote la croissance de la sortie en « unités par milliseconde ».
- **Out** : sortie destinée au signal intégré.

Delay

REAKTOR dispose de six types de délais qui produisent des effets de décalage. Ce sont deux délais Tap, deux délais granulaires, un diffuseur (pour effets d'écho), et le délai Unit, qui permet des feedbacks audio et un modelage physique.

Single Delay



Delay

Élément de décalage destiné aux signaux audio. Le signal d'entrée apparaît à la sortie avec un décalage fonction du temps réglé. Ce décalage se pilote via l'entrée **Dly**.

La limite supérieure de ce décalage est définie dans le dialogue de propriétés du module, dans le champ **Max Delay Buffer** (1 seconde en général), et influence la consommation de mémoire. La valeur maximum possible est fonction de la RAM disponible de l'ordinateur. Avec un taux d'échantillonnage de 44,1 kHz, il vous faut 172 ko de RAM par seconde de longueur de tampon et par voix, et donc, pour une minute, 10 Mo. Lorsque le module est utilisé comme délai d'événement (port **In** rouge, indiquant que le module se trouve en mode de traitement d'événement), vous pouvez définir **Max Count Of Buffered Events** (nombre maxi. d'événements stockés dans le tampon) au même endroit.

Ce module remplace plusieurs modules des versions précédentes de REAKTOR:

- Il se comporte comme le **Static Delay** de REAKTOR 3 si vous connectez un signal audio à l'entrée **In** et un signal d'événement à l'entrée **Dly**. Si le décalage ne correspond pas à un nombre entier d'échantillons, le signal de sortie est défini par interpolation. Ceci peut provoquer de légères modifications de la sonorité. Sélectionnez la méthode d'interpolation dans le dialogue des propriétés. Une interpolation linéaire peut provoquer une faible réduction de la teneur en aigus de la sonorité.
- Il se comporte comme le **Modulation Delay** de REAKTOR 3 si vous connectez un signal audio à l'entrée **In** et un signal audio à l'entrée **Dly**. Vous pouvez moduler le décalage via un signal audio à l'entrée **Dly**. Si le décalage ne correspond pas à un nombre entier d'échantillons, le signal de sortie est défini par interpolation. Ceci peut provoquer de légères modifications de la sonorité. Sélectionnez la méthode d'interpolation dans le dialogue des propriétés.

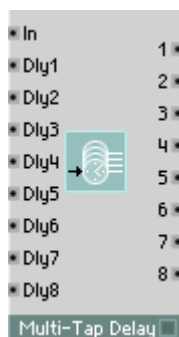
- Il se comporte comme l'**Event Delay** de REAKTOR 3 si vous connectez un signal d'événement à l'entrée **In**.

Ports

- **Dly** : entrée hybride destinée à la durée du décalage (Delay Time), en millisecondes (ms).
- **In** : entrée hybride destinée au signal audio à décaler.
- **Out** : sortie destinée au signal décalé.

Multi-Tap Delay

Delay



Ligne de délais multi-tap pour les signaux audio. Lorsque la durée du décalage correspond à un nombre non entier d'échantillons, une interpolation a lieu. La méthode d'interpolation se sélectionne dans le dialogue des propriétés.

En règle générale, la sortie est connectée à une table de mixage ou à un scanner.

- **In** : entrée audio destinée au signal audio à décaler.
- **Dly1...8** : entrées audio destinées à commander la durée du décalage, en millisecondes. Plage de valeurs caractéristiques : [0...1000].
- **1...8**: sorties audio destinées au signal d'entrée décalé. **1** est décalé de la durée **Dly1**, **2** de la durée **Dly2** etc.



Le diffuseur est un filtre passe-tout qui présente un élément de décalage (Delay-Line) à feedback. Il provoque la diffusion du signal d'entrée, sans pour autant accentuer des fréquences particulières. Son application caractéristique est la création d'effets d'écho (Reverb) bien que ceci ait lieu souvent en connectant plusieurs de ces modules à la suite et en leur donnant des valeurs différentes de décalage.

Pilotez le décalage à l'entrée **Dly**, via un signal audio. Si le décalage ne correspond pas à un nombre entier d'échantillons, le signal de sortie est défini par interpolation. Ceci peut provoquer de légères modifications de la sonorité. Sélectionnez la méthode d'interpolation dans le dialogue des propriétés. Réglez le taux de feedback interne par le biais de l'entrée **Dffs**.

Si vous réglez une valeur nulle de délai, le diffuseur fonctionne comme un filtre passe-tout à 1 pôle. Vous pouvez ainsi créer un phaseur en connectant plusieurs diffuseurs en série, et en modulant le paramètre **Dffs** en conséquence.

La limite supérieure de ce décalage est définie dans le dialogue de propriétés du module (200 ms en général), et influence la consommation de mémoire.

- **Dly** : entrée hybride destinée à la durée du décalage (Delay Time), en millisecondes.
- **Dffs**: entrée d'événement destinée à commander le coefficient de diffusion. Plage de valeurs : -1...1. **Dffs** = 0 : le module est uniquement un module de décalage, **Dffs** = 1 : sortie = entrée, **Dffs** = -1 : sortie = -entrée. Les valeurs les plus utiles, pour **Dffs**, se situent aux alentours de 0,5.
- **In** : entrée destinée au signal audio à diffuser.
- **Out** : sortie destinée au signal audio diffusé.



Élément de décalage et décaleur de hauteur de son (Pitch Shifter) destiné aux signaux audio. Le signal d'entrée apparaît aux sorties avec un décalage correspondant au temps réglé à l'entrée **Dly**, transposé de la valeur en demi-tons appliquée à l'entrée **P**.

Le signal d'entrée est « saucissonné » en particules de son dont la taille est définie via l'entrée **Granularity (Gr)**. L'entrée **Smoothness (Sm)** sert à commander la « rugosité » du son produit. La position des particules sonores dans l'espace stéréo est définie via l'entrée **Pan**.

Vous pouvez modifier le délai, dans **Grain Delay**, sans aucune influence sur la hauteur de son. Vous obtiendrez des effets intéressants en combinant cet élément avec un générateur de hasard et un générateur de bruit.

La qualité de la reproduction et la limite supérieure du décalage se règlent via le dialogue Properties. Le délai maximum effectivement disponible est susceptible de varier de 50 % par rapport à la valeur réglée.

- **P** : entrée de commande audio logarithmique destinée à la transposition, en demi-tons (Pitch).
- **Dly** : entrée de commande audio destinée au décalage (Delay Time), en millisecondes.
- **Gr** : entrée audio de commande destinée à la granularité du processus de re-synthèse, en millisecondes. Ce paramètre permet de définir la taille des particules de son utilisées dans la re-synthèse.
- **Sm** : entrée audio de commande destinée à la *Smoothness* (l'uniformité) du processus de re-synthèse. Cette fonction modifie la forme des particules sonores. En général, des valeurs élevées conduisent à un son rugueux.
- **Pan** : entrée audio de commande destinée à la position dans l'espace stéréo (-1 = à gauche, 0 = au milieu, 1 = à droite).

- **A** : entrée audio de commande destinée à l'amplitude de sortie.
- **In** : entrée destinée au signal audio à décaler.
- **L** : sortie audio destinée au canal gauche du délai.
- **R** : sortie audio destinée au canal droit du délai.
- **Dly** : sortie d'événement polyphonique à laquelle le décalage actuel est établi. Cette sortie produit un événement à chaque fois qu'une particule sonore est générée.

Grain Cloud Delay

Delay



Le module Grain Cloud Delay est très proche du module Grain Cloud de la section Echantillonneur. La différence réside dans le fait que le module Grain Cloud fonctionne avec des échantillons stockés dans la RAM, alors que le module Delay correspondant traite le tampon audio qui alimente le port d'entrée du module (labellisé **In**), modifié en permanence.

- **Trig** : entrée d'événement servant à déclencher le grain suivant. Les valeurs > 0 lancent le grain suivant immédiatement. Une valeur $= -1$ inhibe le grain suivant (voir l'entrée **Dist**).

- **Frz** : entrée d'événement servant à geler le tampon audio. Les valeurs > 0 gèlent le tampon.
- **P** : entrée audio destinée au contrôle logarithmique de la hauteur de son (par demi-ton). La hauteur de son est indépendante de la vitesse de reproduction. Plage caractéristique : [-20...20].
- **D/F** : entrée audio destinée à commander le sens de reproduction lorsque **P** est connectée. Sinon, elle sert à commander la fréquence. Le tampon audio est reproduit avec la hauteur de son originale lorsque $F=1$, et en sens inverse lorsque $F=-1$. Plage caractéristique: [-4...4], par défaut: 1.
- **PJ** : entrée audio destinée aux variations aléatoires de la hauteur de son (Pitch Jitter), par demi-ton. Plage caractéristique : [0...3].
- **PS** : entrée audio destinée à la commande logarithmique du décalage de la hauteur de son (Pitch Shift) du grain actuel, par demi-ton. Plage caractéristique : [-3...3].
- **Dly** : entrée audio destinée à commander la durée du décalage, en ms. Plage autorisée : [0...longueur tampon].
- **DIJ** : entrée audio destinée aux variations aléatoires du décalage. Plage autorisée : [0...longueur tampon].
- **Len** : entrée audio destinée à régler la longueur de grain, en ms. Plage caractéristique : [10...100]. Par défaut : 30 ms.
- **LnJ** : entrée audio destinée à commander la variation aléatoire de longueur (Length Jitter), en ms. Plage caractéristique : [10...100]. Par défaut : 0 ms.
- **Att** : entrée audio destinée à régler la durée de l'attaque. Plage caractéristique : [0...1]. Par défaut : 0.2.
- **Dec** : entrée audio destinée à régler la durée du decay. Plage caractéristique : [0...1]. Par défaut : 0.2.
- **Dist** : entrée audio destinée à régler le différentiel de temps s'écoulant jusqu'au démarrage du grain suivant, en ms. Plage caractéristique : [5...100]. Par défaut : 20.
- **DisJ** : entrée audio destinée à commander la modification aléatoire du différentiel de temps (Delta time Jitter). Plage caractéristique : [10...100]. Par défaut : 20 ms.
- **Pan** : entrée audio destinée à régler la position dans l'espace stéréo. Plage caractéristique : [-1(à gauche)...1(à droite)].

- **PnJ** : entrée audio destinée à commander la variation aléatoire de la position stéréo (Pan Jitter). Plage caractéristique : [0...1].
- **A** : entrée audio destinée à commander l'amplitude. Plage caractéristique: [0...1]. Par défaut : 1.
- **In** : entrée destinée au signal audio à décaler.
- **L** : sortie polyphonique destinée au canal stéréo gauche. Plage caractéristique : [-1...1].
- **R** : sortie polyphonique destinée au canal stéréo droit. Plage caractéristique : [-1...1].
- **Dly** : sortie polyphonique d'événement destinée à la durée du décalage à chaque démarrage de grain.
- **GTr** : sortie d'événement destinée au déclenchement d'un grain. Emet 1 lorsqu'un grain démarre, 0, lorsqu'il s'interrompt.

Unit Delay



Delay

Décale un signal audio de la durée d'un échantillon audio (1/taux d'échantillonnage). Un module Unit Delay est intégré dans toute structure utilisant un type de feedback, dans la boucle de feedback. Si vous n'utilisez pas de manière explicite un module Unit-Delay, REAKTOR intègre de lui-même un délai de ce type à un endroit quelconque de la boucle, visualisé par un trait bleu vertical apparaissant sur un des ports du module.

- **In** : entrée destinée au signal à décaler d'un échantillon audio.
- **Out** : sortie destinée au signal décalé.

Audio Modifier

Ces modules de traitement audio génèrent différents types de distorsion (shaping, clipping et effet de miroir, par exemple). Vous disposez également des modules Slew Limiter, Peak Detector, Sample et Hold, et du module Frequency Divider (diviseur de fréquence).

Saturator



Audio Modifier

Module de distorsion à courbe caractéristique arrondie permettant une transition douce vers la saturation. La valeur de sortie est limitée à ± 2 (lorsque les valeurs d'entrée sont supérieures à ± 4). Les valeurs très faibles ne sont pas modifiées.

- **In** : entrée audio destinée au signal à limiter.
- **Out** : sortie audio destinée au signal limité.

Saturator 2



Audio Modifier

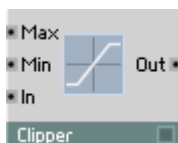
Saturator 2 est un saturateur asymétrique et parabolique de quatrième rang, avec lequel vous accédez à la courbe de distorsion.

- **LA** : asymétrie de niveau. Avec $LA = 0$, les niveaux de saturation des signaux positifs et négatifs sont identiques. Avec $LA > 0$, le niveau positif est réduit. Avec $LA = 1$ il est nul. $LA < 0$ signifie une réduction du niveau négatif. Avec $LA = -1$, il est nul.
- **KH** : Knee Hardness. Avec $KH = 0$, la saturation augmente aussi progressivement que possible. La plage de valeurs complète est utilisée, de 0 au niveau de saturation, pour créer la courbe arrondie. Lorsque KH augmente, la plage de la courbe est réduite à $(1 - KH)$ du niveau de saturation. Avec $KH = 1$, le signal se coude brutalement au niveau de saturation (clipped).

- **KHA** : Knee Hardness Asymmetry. Avec $KHA = 0$, KH est identique pour les signaux positifs et négatifs. Avec $KHA > 0$, KH est diminué pour les signaux positifs. Avec $KHA = 1$, KH est $=0$. Avec $KHA < 0$, KH est réduit pour les signaux négatifs. Avec $KHA = -1$, KH est $=0$.
- **Offs** : Cette entrée ajoute un offset au signal d'entrée et le déplace en fonction de la courbe de saturation. Lorsque le signal est nul, l'offset est entièrement compensé à la sortie.
- **In** : entrée destinée au signal à distordre.
- **Out** : sortie destinée au signal distordu.

Clipper

Audio Modifier



Module de distorsion à clipping brutal et limitations inférieure et supérieure réglables. Si le signal d'entrée dépasse la valeur maximum, il est limité à celle-ci (clipping). Si le signal n'atteint pas la valeur minimum, il est augmenté à celle-ci. Les signaux de valeur intermédiaire sortent sans être modifiés.

- **Max** : entrée audio destinée à commander la valeur maximum du signal.
- **Min** : entrée audio destinée à commander la valeur minimum du signal.
- **In** : entrée audio destinée au signal à limiter.
- **Out** : sortie audio destinée au signal limité.

Mod. Clipper

Audio Modifier



Module de distorsion à clipping brutal et limitations modulables. Lorsque la valeur absolue du signal d'entrée dépasse la valeur réglée, elle est limitée à celle-ci. Les signaux dont la valeur absolue est inférieure ne sont pas modifiés sortent sans modification.

- **M** : entrée audio destinée à commander la valeur absolue maximum du signal.
- **In** : entrée audio destinée au signal à limiter.
- **Out** : sortie audio destinée au signal limité.

Mirror 1 Level



Audio Modifier

Miroir de valeur de signal à taux de réflexion réglable. Les valeurs de signaux supérieures au taux de réflexion sont « réfléchies » avec cette valeur, et sont alors réduites. Les valeurs inférieures sont émises sans être modifiées.

- **Max** : entrée audio destinée à commander le taux de réflexion.
- **In** : entrée audio destinée au signal à modifier.
- **Out** : sortie audio destinée au signal modifié.

Mirror 2 Levels



Audio Modifier

Miroir double de valeur de signal à taux de réflexion réglable. Les valeurs de signal supérieures au taux supérieur de réflexion sont « réfléchies » vers le bas. Les valeurs inférieures au taux de réflexion inférieur sont réfléchies vers le haut. Les valeurs intermédiaires sont émises sans être modifiées.

- **Max** : entrée audio destinée à commander le taux supérieur de réflexion.
- **Min** : entrée audio destinée à commander le taux inférieur de réflexion.
- **In** : entrée audio destinée au signal à modifier.
- **Out** : sortie audio destinée au signal modifié.

Chopper

Audio Modifier

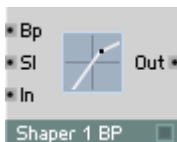


Modulateur Chopper, qui commute l'amplification du signal d'entrée entre une variable et un. Lorsque le signal de modulation est positif, la valeur d'entrée est multipliée par **X**, lorsqu'il est négatif, la valeur d'entrée est émise sans modification.

- **M** : entrée audio destinée au signal de modulation (seul le signe est significatif).
- **X** : entrée audio destinée à commander le facteur d'amplification, avec **M** positif.
- **In** : entrée audio destinée au signal à moduler.
- **Out** : sortie audio destinée au signal modulé.

Shaper 1 BP

Audio Modifier

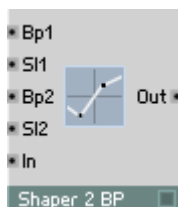


Formant de signal à courbe caractéristique partiellement linéaire et un point d'arrêt. La pente de la courbe après le point d'arrêt est réglable. Les valeurs de signal d'entrée inférieures au point d'arrêt sont émises sans modification.

- **Bp** : entrée audio destinée à commander la valeur du point d'arrêt.
- **Sl** : entrée audio destinée à commander la pente de la partie supérieure de la courbe (1 = aucune modification du signal).
- **In** : entrée audio destinée au signal à former.
- **Out** : sortie audio destinée au signal formé.

Shaper 2 BP

Audio Modifier

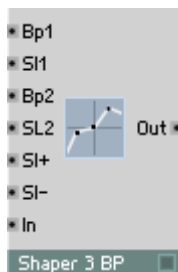


Formant de signal à courbe caractéristique partiellement linéaire et deux points d'arrêt. La croissance de la courbe est réglable après le point d'arrêt supérieur et avant le point d'arrêt inférieur. Les valeurs de signal d'entrée comprises entre les points d'arrêt sont émises sans modification.

- **Bp1** : entrée audio destinée à commander la valeur du point d'arrêt supérieur.
- **SI1** : entrée audio destinée à commander la pente de la partie supérieure de la courbe (1 = aucune modification du signal).
- **Bp2** : entrée audio destinée à commander la valeur du point d'arrêt inférieur.
- **SI2** : entrée audio destinée à commander la pente de la partie inférieure de la courbe (1 = aucune modification du signal).
- **In** : entrée audio destinée au signal à former.
- **Out** : sortie audio destinée au signal formé.

Shaper 3 BP

Audio Modifier



Formant de signal à courbe caractéristique partiellement linéaire et trois points d'arrêt. La croissance de la courbe est réglable après le point d'arrêt supérieur, avant le point d'arrêt inférieur et entre les points d'arrêt et le point zéro.

- **Bp1** : entrée audio destinée à commander la valeur du point d'arrêt supérieur.

- **SI1** : entrée audio destinée à commander la pente de la partie supérieure de la courbe.
- **Bp2** : entrée audio destinée à commander la valeur du point d'arrêt inférieur.
- **SI2** : entrée audio destinée à commander la pente de la partie inférieure de la courbe.
- **SI+** : entrée audio destinée à commander la pente de la partie de la courbe située entre le zéro et le point d'arrêt supérieur (1 = aucune modification du signal).
- **SI-** : entrée audio destinée à commander la pente de la partie de la courbe située entre le point d'arrêt inférieur et le zéro (1 = aucune modification du signal).
- **In** : entrée audio destinée au signal à former.
- **Out** : sortie audio destinée au signal formé.

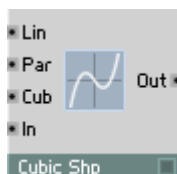
Shaper Parabolic

Audio Modifier



Formant de signal à courbe caractéristique parabolique (polynôme du 2e degré). Vous pouvez régler la composante linéaire et la composante carrée du signal séparément (**Out** = **Lin In** + **Par In**²).

- **Lin** : entrée audio destinée à commander la composante linéaire, non distordue, du signal.
- **Par** : entrée audio destinée à commander la composante carrée, distordue, du signal.
- **In** : entrée audio destinée au signal à former.
- **Out** : sortie audio destinée au signal formé.

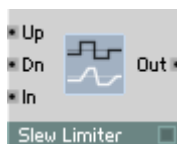


Formant à courbe caractéristique cubique et parabolique (polynôme du 3e degré). Vous pouvez régler les composantes linéaire, carrée et cubique séparément (**Out** = **Lin In** + **Par In**² + **Cub In**³).

- **Lin** : entrée audio destinée à commander la composante linéaire, non distordue, du signal.
- **Par** : entrée audio destinée à commander la composante carrée, distordue, du signal.
- **Cub** : entrée audio destinée à commander la composante cubique, distordue, du signal.
- **In** : entrée audio destinée au signal à former.
- **Out** : sortie audio destinée au signal formé.

Slew Limiter

Audio Modifier



Limiteur à taux maximum (Slew) réglable séparément dans les directions positive et négative. Le signal de sortie suit le signal d'entrée, lorsque la modification du signal et les sauts sont rapides, la sortie ne suit qu'avec une certaine rampe jusqu'à ce qu'elle ait atteint le niveau du signal d'entrée.

- **Up** : entrée audio destinée à commander le taux maximum (en 1/sec) lorsque les signaux augmentent.
- **Dn** : entrée audio destinée à commander le taux maximum (en 1/sec) lorsque les signaux diminuent.
- **In** : entrée audio destinée au signal à limiter.
- **Out** : sortie audio destinée au signal limité.

Peak Detector

Audio Modifier



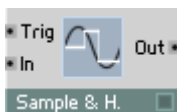
Redresseur d'amplitude maximum à lissage. L'attaque a une durée nulle, le release est réglable.

Ce détecteur d'amplitude maximum fait que la valeur de sortie suit l'enveloppe d'amplitude du signal d'entrée; utilisez ce module comme suiveur d'enveloppe.

- **In** : entrée audio destinée au signal à redresser.
- **Rel** : entrée destinée à commander le temps de release.
- **Out** : sortie audio destinée au signal redressé.

Sample & Hold

Audio Modifier

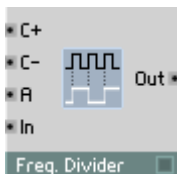


Élément d'échantillonnage et de maintien à entrée d'horloge. Lorsque le signal d'horloge est supérieur à 0, la valeur actuelle d'entrée s'applique aussi à la sortie, et est maintenue jusqu'à la nouvelle impulsion d'horloge. L'onde à la sortie a donc une forme à pas.

- **C** : entrée audio destinée au signal d'horloge (Clock). L'entrée est échantillonnée avec un flanc croissant.
- **In** : entrée audio destinée au signal à échantillonner.
- **Out** : sortie audio destinée au signal échantillonné.

Frequency Divider

Audio Modifier



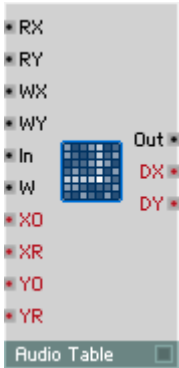
Diviseur de fréquence (sous-oscillateur rectangulaire) à temps réglable pour les deux demi-ondes. Une onde de forme rectangulaire est générée en comptant les

passages à zéro, dont la fréquence est une fraction de celle du signal d'entrée. Le rapport de ces fréquences est réglable ($f_{out} = 2 f_{in} / (C++C-)$). Une forme d'onde asymétrique en résulte, lorsque **C+** et **C-** ont des valeurs différentes.

- **C+** : entrée audio destinée à commander les passages à zéro du signal d'entrée pendant la phase montante du signal de sortie.
- **C-** : entrée audio destinée à commander les passages à zéro du signal d'entrée pendant la phase descendante du signal de sortie.
- **A** : entrée destinée à commander l'amplitude de sortie.
- **In** : entrée audio destinée au signal à diviser.
- **Out** : sortie audio destinée au signal à fréquence divisée.

Audio Table

Audio Modifier



Contient une table de valeurs. Vous pouvez lire cette table comme un signal audio, y enregistrer des signaux audio, en afficher le contenu et l'éditer sous forme graphique. Son utilisation peut être mono-dimensionnelle (une ligne de valeurs adressables par X) ou bi-dimensionnelle (une matrice de lignes et de colonnes de valeurs, adressables par X et Y).

La valeur de sortie se lit à la position résultant de la combinaison des entrées **RX** et **RY**. Un signal établi à l'entrée **In** est enregistré dans des cellules individuelles de la table en fonction des positions d'écriture correspondant aux entrées **WX** et **WY**.

X représente la position horizontale de gauche à droite, Y la position verticale de haut en bas. Le décompte commence toujours avec la valeur 0 pour le premier élément.

L'affichage du module est en mesure d'afficher toutes les données ou celles d'une zone définie. De nombreuses options des propriétés permettent à l'utilisateur de personnaliser le comportement de ce module.

Vous trouverez une description complète des propriétés, des menus et des raccourcis clavier à la Section *Les modules Table*, à partir de la page - 175.

- **RX** : entrée audio destinée à la position X de la cellule de la table dont les données sont lues.
- **RY** : entrée audio destinée à la position Y de la cellule de la table dont les données sont lues. Cette entrée est utilisée en mode 2D, ou pour adresser le numéro de ligne lorsque plusieurs lignes existent.
- **WX** : entrée audio destinée à la position X de la cellule de la table dans laquelle les données sont écrites.
- **WY** : entrée audio destinée à la position Y de la cellule de la table dans laquelle les données sont écrites.
- **W** : entrée audio destinée à déclencher une écriture à cette position.
- **In** : entrée audio destinée au signal à écrire dans la table. Lorsque la valeur **W** est supérieure à 0, la valeur existant sur **In** est écrite dans la table, à la position définie par **WX** et **WY**.
- **XO** : entrée d'événement destinée au déplacement horizontal de la zone de données affichée. **XO** commande la position des données apparaissant à l'affichage (conformément à l'orientation de la vue). La valeur **XO** est définie en unités (Units) dans les propriétés.
- **XR** : entrée d'événement destinée à la zone d'affichage horizontale des données. **XR** définit combien d'unités de données sont contenues dans l'affichage, permettant ainsi un agrandissement et un rétrécissement de la section (zoom).
- **YO** : entrée d'événement destinée au déplacement vertical de la zone de données affichée. **YO** commande la position des données apparaissant à l'affichage (conformément à l'orientation de la vue). La valeur **YO** est précisée en unités (Units) dans les propriétés.
- **YR** : entrée d'événement destinée à la zone d'affichage verticale des données en mode 2D. **YR** définit combien d'unités de données sont contenues dans l'affichage, permettant ainsi un agrandissement et un rétrécissement de la section (zoom).
- **Out** : sortie audio de la cellule définie par les entrées **RX**, **RY** et **R**.
- **DX** : sortie d'événement destinée à la longueur des lignes horizontales de la table, en unités.

- **DY** : sortie d'événement destinée à la hauteur des colonnes de la table, en unités.

Event Processing

Vous pouvez utiliser les modules Event pour compter des opérations logiques, pour répartir et relier des signaux d'événements, pour mesurer une durée entre deux événements (Timer). Certains de ces modules servent à former et à randomiser les signaux d'événements. Et, pour finir, vous disposez également d'un module de table à équipement complet, l'équivalent de la table audio de la section Oscillateur.

Accumulator

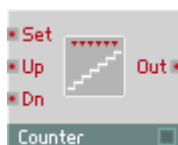


Event Processing

Accumulateur des valeurs d'événements (somme). La valeur de chaque événement apparaissant à l'entrée est ajoutée à la somme intrinsèque du module. La nouvelle valeur de cette somme est émise sous forme d'événement à la sortie.

- **In** : entrée destinée aux événements à accumuler.
- **Set** : entrée d'événement destinée à définir (remettre à zéro) la somme intrinsèque. La valeur d'un événement à cette entrée définit la nouvelle valeur de la somme.
- **Out** : sortie d'événement destinée à la valeur de la somme.

Counter



Event Processing

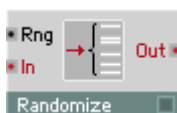
Compteur piloté par les événements. Un événement positif à l'entrée appropriée fait augmenter ou diminuer d'un la valeur de sortie.

- **Up** : entrée destinée à ajouter.
Seuls les événements à valeur positive (et pas nulle) entrent ici en

ligne de compte et augmentent la valeur de sortie d'un.

- **Dwn** : entrée destinée à soustraire.
Seuls les événements à valeur positive (et pas nulle) entrent ici en ligne de compte et diminuent la valeur de sortie d'un.
- **Set** : entrée d'événement destinée à définir (remettre à zéro) la valeur du compteur. La valeur d'un événement à cette entrée définit la nouvelle valeur du compteur.
- **Out** : sortie d'événement destinée à la valeur du compteur.

Randomizer



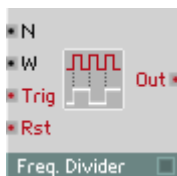
Event Processing

Modificateur aléatoire d'événement à plage de diffusion réglable.

Les événements arrivants sont émis avec une modification de valeur aléatoire, dont l'étendue est réglable (**Out** = **In** + X, avec $-\text{Rng} \leq X \leq \text{Rng}$).

- **Rng** : entrée de signal audio destinée à commander l'étendue de la modification aléatoire de valeur.
- **In** : entrée d'événement destinée au signal à modifier.
- **Out** : sortie d'événement destinée au signal modifié.

Frequency Divider



Event Processing

Diviseur de fréquence destiné aux signaux d'événement. Le signal de sortie revient à zéro après un certain nombre d'événements d'entrée, relatif à la longueur de l'impulsion **W**.

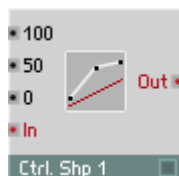
- **N** : entrée de commande du facteur de division ($N < 2$: aucune division, $2 \dots 2,99$: facteur = 2, $3 \dots 3,99$: facteur = 3, etc.).
- **W** : entrée de commande destinée à la longueur d'impulsion du signal de sortie. Plage de valeurs caractéristiques : $0 \dots 1$ (0% ... 100%). **W** = 0 : le signal Gate établi sur **Out** revient, lors du prochain événement sur **In**, à zéro ; **W** = 0.5 : le signal Gate, sur **Out**, revient à zéro après la moitié de

la période ; **W** = 1 : le signal Gate, sur **Out**, est actif en permanence.

- **In** : entrée destinée au signal d'événement dont la fréquence est partagée (par ex. horloge MIDI). Seuls les événements à valeur positive influencent ce module.
- **Rst** : entrée d'événement de remise à zéro (Reset) du compteur intrinsèque, permettant de forcer un démarrage synchronisé (par ex. piloté par un signal MIDI Start). Remet également la valeur de **Out** à zéro. Les événements de valeur inférieure ou égale à zéro sont ignorés, seuls les événements positifs provoquent un reset.
- **Out** : sortie d'événement destinée au signal à fréquence divisée.

Ctrl. Shaper 1 BP

Event Processing

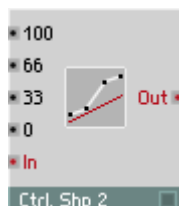


Formant de signal de commande à courbe partiellement linéaire et un point d'arrêt. Interpolation linéaire entre les valeurs réglées.

- **100** : valeur de sortie avec **In** = 1 (100%).
- **50** : valeur de sortie destinée au point d'arrêt avec **In** = 0,5 (50%).
- **0** : valeur de sortie avec **In** = 0 (0%).
- **In** : entrée d'événement destinée au signal à former.
- **Out** : sortie d'événement destinée au signal formé.

Ctrl. Shaper 2 BP

Event Processing

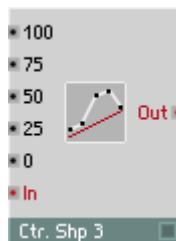


Formant de signal de commande à courbe caractéristique partiellement linéaire et deux points d'arrêt fixes. Interpolation linéaire entre les valeurs réglées.

- **100** : valeur de sortie avec **In** = 1 (100%).
- **66** : valeur de sortie destinée au point d'arrêt supérieur avec **In** = 0,66 (66%).
- **33** : valeur de sortie destinée au point d'arrêt inférieur avec **In** = 0,33 (33%).
- **0** : valeur de sortie avec **In** = 0 (0%).
- **In** : entrée d'événement destinée au signal à former.
- **Out** : sortie d'événement destinée au signal formé.

Ctrl. Shaper 3 BP

Event Processing



Formant de signal de commande à courbe caractéristique partiellement linéaire et trois points d'arrêt fixes. Interpolation linéaire entre les valeurs réglées.

- **100** : valeur de sortie avec **In** = 1 (100%).
- **75** : valeur de sortie destinée au troisième point d'arrêt avec **In** = 0,75 (75%).
- **50** : valeur de sortie destinée au deuxième point d'arrêt avec **In** = 0,5 (50%).
- **25** : valeur de sortie destinée au premier point d'arrêt avec **In** = 0,25 (25%).
- **0** : valeur de sortie avec **In** = 0 (0%).
- **In** : entrée d'événement destinée au signal à former.
- **Out** : sortie d'événement destinée au signal formé.

Logic AND

Event Processing



Porte logique pour signaux d'événements. La valeur de sortie est la liaison logique ET des deux entrées, donc elle est égale à 1 lorsque les deux entrées sont positives, et nulle sinon. Les entrées considèrent les valeurs nulles et négatives comme relevant de l'état logique « faux » (0), les valeurs positives sont logiquement « vraies » (1).

La sortie **Not** est la négation logique de la sortie **Out**.

Logic OR

Event Processing



Porte logique pour signaux d'événements. La valeur de sortie est la liaison logique OU des deux entrées, donc elle est égale à 1 lorsqu'une ou les deux entrées sont positives, et nulle sinon.

Les entrées considèrent les valeurs nulles et négatives comme relevant de l'état logique « faux » (0), les valeurs positives sont logiquement « vraies » (1).

La sortie **Not** est la négation logique de la sortie **Out**.

Logic EXOR

Event Processing



Porte logique pour signaux d'événements. La valeur de sortie est la liaison logique EXOR (OU exclusif) des deux entrées, donc elle est égale à 1 lorsqu'une entrée est positive, mais pas les deux, et nulle sinon. En réalité, l'une des entrées inverse l'autre.

Les entrées considèrent les valeurs nulles et négatives comme relevant de l'état logique « faux » (0), les valeurs positives sont logiquement « vraies » (1).

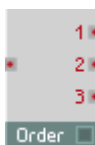
La sortie **Not** est la négation logique de la sortie **Out**.



Porte logique pour signaux d'événements. La valeur de sortie est la négation logique du signal d'entrée, donc elle est égale à 0 lorsque l'entrée a une valeur positive, et sinon égale à 1.

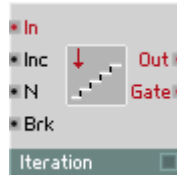
L'entrée considère les valeurs nulles et négatives comme relevant de l'état logique « faux » (0), les valeurs positives sont logiquement « vraies » (1).

La sortie **Not** est la négation logique de la sortie **Out**, et a donc logiquement la même valeur que l'entrée.



Suite d'événements. Un événement arrivant à l'entrée est transmis sans modification à toutes les sorties, mais dans un ordre établi : d'abord à la sortie **1**, puis à la sortie **2**, et enfin à **3**. L'événement transite par TOUS les modules connectés à la sortie **1** avant de parvenir au premier module connecté à **2**, etc. Voir également la Section *Signaux d'événement*, à partir de la page - 129.

- **In** : entrée destinée aux événements devant être transmis dans un ordre déterminé.
- **1** : un événement est envoyé tout d'abord à cette sortie.
- **2** : un événement est transmis en deuxième à cette sortie.
- **3** : un événement est transmis en troisième à cette sortie.



Un événement arrivant à l'entrée **Trg** est transmis à la sortie et déclenche une série de **N** événements de sortie supplémentaires. Tout événement en aval possède la valeur de son prédécesseur, incrémentée de la valeur **Inc**. Tous les événements sont ainsi traités avant le calcul de l'échantillon audio suivant. Vous pouvez utiliser ce module pour toutes les opérations qui requièrent un calcul récurrent d'événements. Il vous aide à éviter les boucles d'événements, qui sont susceptibles de rendre REAKTOR instable.

- **Trg** : entrée de déclenchement. Un événement à cette entrée déclenche N+1 événements à la sortie.
- **Inc** : augmentation de la valeur de tous les événements suivants.
- **N** : nombre des événements de sortie supplémentaires. La valeur doit être supérieure ou égale à un entier (les décimales sont supprimées).

Separator

Event Processing



Séparateur d'événements. Les événements arrivant sont comparés à la valeur seuil, puis envoyés vers l'une ou l'autre des sorties.

Le séparateur permet par ex. de transformer un signal Gate en signal de déclenchement en éliminant par filtrage les événements nuls (**Thld** = 0, **Hi** = sortie de déclenchement).

- **Thld** : entrée audio de commande destinée à la valeur du seuil (threshold).
- **In** : entrée destinée aux événements à répartir sur les deux sorties.
- **Hi** : sortie destinée aux événements supérieurs à la valeur du seuil.
- **Lo** : sortie destinée aux événements inférieurs ou égaux à la valeur de seuil.



Modificateur de valeur des événements. Les événements arrivant à l'entrée **In** voient l'entrée **Val** leur attribuer une nouvelle valeur, et sont ensuite émis avec celle-ci par la sortie **Out**.

Vous pouvez aussi utiliser ce module comme un module Sample&Hold piloté par les événements.

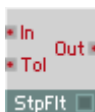
- **In** : entrée destinée aux événements dont la valeur est modifiée.
- **Val** : entrée destinée à déterminer la nouvelle valeur affectée aux événements.
- **Out** : sortie destinée aux événements à valeur modifiée.



Module de fusion des signaux d'événements. Lorsque plus d'un câble sont connectés à l'entrée, c'est toujours le dernier événement reçu qui forme la valeur de sortie, quel que soit le câble qui l'a convoyé.

Le module dispose d'une gestion dynamique des ports d'entrée. Définissez le nombre d'entrées en utilisant **Min Num Port Groups** de la page **Function** des propriétés.

Contrairement à la procédure des versions précédentes de REAKTOR, les événements suivants possédant la même valeur ne sont pas éliminés par filtrage. Si vous le souhaitez, utilisez pour ce faire le module **Step Filter**.



La transmission d'un événement est garantie lorsque la valeur d'entrée est supérieure/inférieure à la valeur d'entrée principale, +/- la tolérance.

- **In** : entrée destinée aux événements à filtrer.

- **Tol** : entrée destinée au degré de tolérance.
- **Out** : sortie des événements.

Router M->1



Event Processing

Routeur entre plusieurs entrées et une sortie. Les événements de l'entrée sélectionnée sont transmis, les autres éliminés par filtrage. Sélectionnez les entrées via l'entrée **Pos**.

Lorsque le mode **Wrap** est activé, dans Propriétés, **Pos** fonctionne comme une boucle, et Max +1 devient 0, Max +2 devient 1, etc.

Le module dispose d'une gestion dynamique des ports d'entrée. Définissez le nombre d'entrées en utilisant **Min Num Port Groups** de la page **Function** des propriétés.

- **Pos** : entrée destinée à l'entrée à traverser.
- **In** : entrée de signal.
- **Out** : sortie destinée à l'entrée de signal sélectionnée.

Router 1,2



Event Processing

Commutateur On/Off ou permutateur de signaux d'événements. La dernière valeur connectée est maintenue à la sortie.

Le module dispose d'une gestion dynamique des ports d'entrée. Sélectionnez 1 ou 2, pour le nombre d'entrées, en utilisant **Min Num Port Groups** de la page **Function** des propriétés.

- **Ctrl** : entrée hybride destinée à la commande du commutateur. Une valeur supérieure à 0 dirige tous les événements de l'entrée vers la sortie.
- **In** : entrée d'événement destinée au signal à connecter.
- **Out** : sortie d'événement destinée au signal connecté. Tant que **Ctrl** est inférieure à 0, la dernière valeur transmise est maintenue à la sortie.

Router 1->M



Event Processing

Router d'une entrée vers des sorties multiples. Les événements apparaissant à l'entrée sont transmis vers la sortie sélectionnée. Sélectionnez la sortie via l'entrée **Pos**. Lorsque le mode **Wrap** est activé, dans Propriétés, **Pos** fonctionne comme une boucle, et Max +1 devient 0, Max +2 devient 1, etc.

Le module dispose d'une gestion dynamique des ports de sortie. Définissez le nombre de sorties en utilisant **Min Num Port Groups** de la page **Function** des propriétés.

- **Pos** : entrée destinée à sélectionner la sortie qui accueillera le signal d'entrée.
- **In** : entrée de signal.
- **Out** : sortie destinée au signal d'entrée.

Timer



Event Processing

Module de mesure du temps et de la fréquence.

Il mesure le temps écoulé entre les deux derniers événements reçus et l'émet. Il calcule également la fréquence dont la période d'oscillation correspond à l'intervalle de temps mesuré, et l'émet également.

- **In** : entrée polyphonique destinée aux événements dont l'éloignement dans le temps sera mesuré.
- **F** : sortie polyphonique destinée à la fréquence d'événements, en Hz.
- **T** : sortie polyphonique d'événements destinée à l'intervalle entre les événements, en millisecondes.

Hold



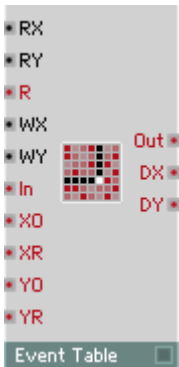
Event Processing

Enveloppe de maintien des événements. Lorsque ce module est déclenché par un événement positif à l'entrée **T**, la valeur de l'événement est émise à la

sortie et y est maintenue jusqu'à expiration du temps de maintien ; la sortie revient ensuite à zéro. Il est également possible de déclencher le module une nouvelle fois pendant la phase de maintien (retrigger).

- **G** : entrée d'événement destinée à déclencher (trigger) l'enveloppe de maintien. Seuls les événements à valeur positive influencent ce module.
- **H** : entrée destinée à commander le temps de maintien (hold), en millisecondes.
- **Out** : sortie d'événements destinée au signal de l'enveloppe.

Event Table



Event Processing

Contient une table de valeurs. Vous pouvez lire les valeurs d'événements dans cette table, les y enregistrer, en afficher le contenu et l'éditer sous forme graphique. Son utilisation peut être mono-dimensionnelle (une ligne de valeurs adressables par X) ou bi-dimensionnelle (une matrice de lignes et de colonnes de valeurs, adressables par X et Y).

La valeur de sortie se lit à la position résultant de la combinaison des entrées **RX** et **RY**. Les valeurs établies à l'entrée **W** sont enregistrées dans des cellules individuelles de la table en fonction des positions d'écriture correspondant aux entrées **WX** et **WY**.

X représente la position horizontale de gauche à droite, Y la position verticale de haut en bas. Le décompte commence toujours avec la valeur 0 pour le premier élément.

L'affichage du module est en mesure d'afficher toutes les données ou celles d'une zone définie. De nombreuses options des propriétés permettent à l'utilisateur de personnaliser le comportement de ce module.

Vous trouverez une description complète des propriétés, des menus et des raccourcis clavier à la Section *Les modules Table*, à partir de la page - 175.

- **RX** : entrée audio destinée à la position X de la cellule de la table dont les données sont lues.
- **RY** : entrée audio destinée à la position Y de la cellule de la table dont les données sont lues. Cette entrée est utilisée en mode 2D, ou pour adresser le numéro de ligne lorsque plusieurs lignes existent.
- **R** : entrée d'événement destinée à déclencher une procédure de lecture à la position définie par **RX** et **RY**. Chacun de ces événements déclenche un événement à la sortie **Out**.
- **WX** : entrée audio destinée à la position X de la cellule de la table dans laquelle les données sont écrites.
- **WY** : entrée audio destinée à la position Y de la cellule de la table dans laquelle les données sont écrites.
- **In** : entrée d'événement destinée à l'écriture dans la table, à la position définie par **WX** et **WY**. La valeur de données inscrite dans la table est celle de l'événement qui parvient à l'entrée **In**.
- **XO** : entrée d'événement destinée au déplacement horizontal de la zone de données affichée. **XO** commande la position des données apparaissant à l'affichage (conformément à l'orientation de la vue). La valeur **XO** est précisée en unités (Units) dans les propriétés.
- **XR** : entrée d'événement destinée à la zone d'affichage horizontale des données. **XR** définit combien d'unités de données sont contenues dans l'affichage, permettant ainsi un agrandissement et un rétrécissement de la section (zoom).
- **YO** : entrée d'événement destinée au déplacement vertical de la zone de données affichée. **YO** commande la position des données apparaissant à l'affichage (conformément à l'orientation de la vue). La valeur **YO** est précisée en unités (Units) dans les propriétés.
- **YR** : entrée d'événement destinée à la zone d'affichage verticale des données en mode 2D. **YR** définit combien d'unités de données sont contenues dans l'affichage, permettant ainsi un agrandissement et un rétrécissement de la section (zoom).
- **Out** : sortie d'événement destinée aux données provenant de la cellule définie par les entrées **RX**, **RY** et **R**.
- **DX** : sortie d'événement destinée à la longueur des lignes horizontales de la table, en unités.

- **DY** : sortie d'événement destinée à la hauteur des colonnes de la table, en unités.

Auxiliary

Cette section contient les platines qui vous permettent d'enregistrer et de reproduire des fichiers audio. Elle présente également des modules de gestion de la polyphonie, de conversion des signaux audio en signaux d'événement, et de connexion à des fonctions générales de REAKTOR, comme Tuning ou Master-Tempo.

Tapedeck 1-Ch



Auxiliary

Enregistreur mono-canal destiné à l'enregistrement et à la reproduction des signaux audio. Vous pouvez reproduire les fichiers audio depuis la mémoire vive ou le disque dur, et les y écrire en fonction des réglages correspondants des Properties.

En mode Harddisk, les fichiers sont enregistrés dans le répertoire que vous avez indiqué à cet effet dans les Preferences de Reaktor. Reaktor effectue une conversion des taux d'échantillonnage en temps réel et écrit le fichier au taux d'échantillonnage utilisé par votre installation audio.

En mode Memory, vous pouvez afficher la forme d'onde du fichier audio chargé dans le panneau en cochant la case **Picture** de la page **Appearance**, dans Properties. La forme d'onde entière est échelonnée automatiquement à la taille de l'affichage du panneau, que vous définissez sous **Appearance**, avec les valeurs **Size**.

Mode Memory

Lorsque vous cochez la case **Keep audio only in memory**, dans les Propriétés, la platine passe en mode Memory, et la section Memory, contenant les boutons de commande **Save**, **Save as...** et **Reload** est activée.

Vous réglez la durée maximum de l'enregistrement via le dialogue Propriétés du module, sous **Max Recording Size (sec)** (en secondes).

La valeur maximum possible est fonction de la RAM disponible de l'ordinateur. Un taux d'échantillonnage de 44,1 kHz correspond à 86 ko de RAM par seconde de tampon, une minute correspond à 5 Mo. Si la mémoire destinée à la platine d'enregistrement est trop importante, une activité du disque dur peut être provoquée, pendant l'enregistrement, par des activités de mémoire virtuelle du système d'exploitation, ce qui peut entraver considérablement REAKTOR dans ses calculs audio.

La touche **Select File...** des propriétés permet d'importer des fichiers audio pour les reproduire. Chargez à nouveau un fichier déjà importé avec **Reload**, par ex. s'il a été modifié dans un éditeur d'échantillons.

Lors de l'importation d'un fichier audio, la longueur du tampon audio est adaptée aux données. Pour procéder ultérieurement à un enregistrement plus long, vous devez d'abord entrer une valeur supérieure, dans le dialogue Propriétés de la platine, sous **Max Recording Size (sec)**. Le fichier audio chargé se règle automatiquement au taux d'échantillonnage actuel de Reaktor.

Note : pensez que Reaktor, en mode Memory, modifie le taux d'échantillonnage de tous les fichiers audio qui se trouvent dans la platine pour les régler sur la nouvelle valeur de celui-ci. Nous recommandons d'éviter de modifier le taux d'échantillonnage lorsque des fichiers audio se trouvent dans la mémoire vive utilisée par les platines d'enregistrement. Si le même fichier audio se trouve sur le disque dur, vous pouvez le charger à nouveau, après avoir modifié le taux, à l'aide de **Reload**.

Exportez un enregistrement en utilisant **Save**, dans les propriétés. Si un fichier portant le même nom existe déjà sur le disque dur, le logiciel vous demande si vous souhaitez l'écraser. **Save as...** vous permet d'enregistrer le fichier sous un autre nom.

Mode Harddisk

Cochez la case **Stream audio from/to harddisk**, dans Properties, pour faire commuter la platine d'enregistrement en mode Harddisk. Les fichiers audio sont alors écrits directement sur le disque dur et sont reproduits depuis celui-ci. En cliquant sur **Select File...**, vous sélectionnez un fichier audio pour sa reproduction. **New File** vous sert à créer un nouveau fichier sur le disque dur, nommé « untitled » (si un fichier portant ce nom existe déjà sur le disque dur, un chiffre est ajouté au nom). Vous pouvez éditer ce nom directement dans le champ correspondant de Properties.

Lorsque **Value** est activé, dans le dialogue Properties, une case permettant d'afficher le nom apparaît dans le panneau. Il vous permet également d'importer ou d'exporter des fichiers, avec le menu contextuel.

- **R** : entrée d'événement destinée à lancer/arrêter l'enregistrement (Record), par ex. via un signal Gate. L'enregistrement démarre lorsque l'événement positif, s'arrête lorsqu'il est inférieur ou égal à 0 ou que la durée maximum de l'enregistrement a été atteinte.
- **P** : entrée d'événement destinée à lancer/arrêter la reproduction (Play), par ex. via un signal Gate. La reproduction démarre lorsque l'événement est positif, s'arrête lorsqu'il est inférieur ou égal à 0.
- **Lp** : entrée d'événement destinée à lancer/arrêter la reproduction à l'infini (en boucle, Loop Play). Lorsque l'événement reçu est positif, la reproduction, une fois terminée, recommence depuis le début, une boucle infinie se crée ainsi.
- **In** : entrée audio monophonique destinée au signal à enregistrer. Lorsque le signal dépasse la plage de valeurs de ± 1 , des distorsions clipping apparaissent.
- **Pse** : entrée de commande du mode Pause. Une valeur supérieure à 0 active le mode Pause. Toute autre valeur provoque la poursuite de la reproduction. Plage caractéristique : [0 ... 1].
- **Pos** : entrée destinée à la position de reproduction, en ms. Plage caractéristique : [0 ... 20000].
- **Spd** : vitesse de reproduction de l'enregistrement variable. Les valeurs doivent être positives. La valeur 1 signifie vitesse d'origine. Plage caractéristique : [0.8 ... 1.2].
- **Out** : sortie audio destinée au signal de reproduction (Playback) ou au signal en cours d'enregistrement (Record).
- **Rec** : 1 = la platine enregistre, sinon 0. Plage caractéristique : [0 ... 1].

- **Play** : 1 = la platine reproduit, sinon 0. Plage caractéristique : [0... 1].
- **Wrp** : un événement de valeur 1 est émis chaque fois que la reproduction en boucle atteint le point de démarrage.
- **Pos** : 1=la platine est en mode Pause, sinon 0.
Plage caractéristique :[0...1].
- **Time** : position actuelle de reproduction/d'enregistrement, en ms [0 ... <longueur d'échantillon en ms>].
- **Lng** : longueur de l'enregistrement, en ms.
Plage caractéristique : [0 ... <longueur de l'enregistrement, en ms>].

Tapedeck 2-Ch

Auxiliary



Similaire à **Tapedeck 1-Ch**, mais avec deux canaux d'enregistrement et de reproduction de signaux audio.

Des fichiers audio sont importés et exportés en stéréo.

- **In L** : entrée audio monophonique destinée au signal à enregistrer sur le canal gauche. Lorsque le signal dépasse la plage de valeurs de ± 1 , des distorsions clipping apparaissent.
- **In R** : entrée audio monophonique destinée au signal à enregistrer sur le canal droit. Lorsque le signal dépasse la plage de valeurs de ± 1 , des distorsions clipping apparaissent.
- **L** : sortie audio destinée au signal de reproduction (Playback) ou au signal d'enregistrement (Record Monitor) du canal gauche.
- **R** : sortie audio destinée au signal de reproduction (Playback) ou au signal en cours d'enregistrement (Record Monitor) du canal droit.

Audio Voice Combiner



Auxiliary

Audio Voice Combiner (mélangeur de voix). Convertit un signal audio polyphonique en un signal monophonique en mélangeant toutes les voix par addition.

- **In** : entrée audio polyphonique destinée au signal à mélanger.
- **Out** : sortie audio monophonique destinée au signal mélangé.

Event V.C. All



Auxiliary

Event Voice Combiner (mélangeur de voix). Envoie tous les événements d'un signal polyphonique à une sortie monophonique.

- **In** : entrée d'événement polyphonique destinée au signal à mélanger.
- **Out** : sortie d'événement monophonique destinée au signal mélangé.

Event V.C. Max



Auxiliary

Event Voice Combiner (mélangeur de voix) à sélection de valeur maximum. La voix active à valeur maximum d'événement est déterminée, et cette valeur est émise en monophonie. Un signal Gate polyphonique est nécessaire pour détecter les voix actives.

- **In** : entrée d'événement polyphonique destinée au signal dont la valeur maximum sera émise.
- **G** : entrée d'événement polyphonique destinée au signal Gate de l'instrument polyphonique.
- **Out** : sortie d'événement monophonique destinée au signal de valeur maximum.

Event V.C. Min



Auxiliary

Event Voice Combiner (mélangeur de voix) à sélection de valeur minimum. La voix active à valeur minimum d'événement est déterminée, et cette valeur

est émise en monophonie. Un signal Gate polyphonique est nécessaire pour détecter les voix actives.

- **In** : entrée d'événement polyphonique destinée au signal dont la valeur minimum sera émise.
- **G** : entrée d'événement polyphonique destinée au signal Gate de l'instrument polyphonique.
- **Out** : sortie d'événement monophonique destinée au signal de valeur minimum.

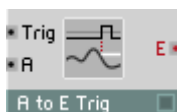
A to E



Auxiliary

Convertisseur audio à événement. Le signal audio est échantillonné au Control Rate réglé dans le menu Settings, les valeurs obtenues sont émises sous forme d'événements.

A to E (Trig)



Auxiliary

Convertisseur audio à événement à déclenchement (Trigger). Lorsque le signal de déclenchement passe de 0 à une valeur positive (pente croissante), le signal audio est échantillonné puis émis sous forme d'événement.

- **T** : entrée audio destinée à déclencher l'échantillonnage (Trigger). Déclenchement sur pente croissante.
- **In** : entrée audio destinée au signal à échantillonner et à émettre sous forme d'événements.
- **Out** : sortie d'événement destinée au signal échantillonné.

A to E (Perm)



Auxiliary

Convertisseur audio à événement, à échantillonnage régulier (permanent) via horloge interne. Le signal audio est échantillonné à la fréquence réglée et émis sous forme d'événements.

Si vous utilisez ce module à fréquence élevée (supérieure à 1000 Hz), la contrainte à laquelle est soumise le processeur peut augmenter considérablement. En principe, $F = 200$ Hz suffit.

- **F** : entrée de signe audio destinée à commander la fréquence d'échantillonnage, en Hz.
- **In** : entrée audio destinée au signal à échantillonner et à émettre sous forme d'événements.
- **Out** : sortie d'événement destinée au signal échantillonné.

A to Gate



Auxiliary

Convertisseur audio à événement de porte. Lorsque le signal d'impulsion d'échantillonnage passe de 0 à une valeur positive (pente croissante), le signal Gate est activé à l'amplitude actuelle de l'entrée d'amplitude. En cas de pente décroissante (signal de déclenchement inférieur ou égal à 0), le signal est désactivé.

- **T** : entrée de signe audio destinée à commander la porte.
- **A** : entrée du signal audio destinée à l'amplitude des événements Gate.
- **Out** : sortie d'événement destinée au signal Gate.

To Voice



Auxiliary

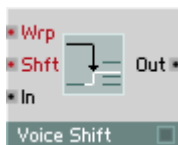
Des signaux d'entrée monophoniques sont affectés à une voix d'un signal de sortie polyphonique. Le numéro de la voix est déterminé par la valeur actuelle à l'entrée **V**. Les signaux sont rejetés lorsque la valeur **V** ne correspond pas à un numéro correct de voix.

- **V** : entrée monophonique destinée à déterminer le numéro de la voix qui accueillera le signal. Plage caractéristique : [1 ... 10].
- **In** : entrée hybride monophonique destinée au signaux à envoyer à une des voix polyphoniques.
- **Out** : sortie polyphonique hybride. Les signaux d'entrée sont transmis vers une voix du signal polyphonique, avec une valeur inchangée.



Une voix du signal d'entrée polyphonique est sélectionnée via la valeur existant à l'entrée **V**. Seuls les signaux de la voix sélectionnée sont transmis à la sortie monophonique. Aucun signal des autres voix n'est transmis.

- **V** : entrée monophonique destinée à déterminer le numéro de la voix dont les événements sont transmis. Plage caractéristique : [1 ... 10].
- **In** : entrée hybride. Une voix de ce signal polyphonique est transmise à la sortie.
- **Out** : sortie monophonique hybride.
Les signaux d'entrée appartenant à une voix sont émise, sans modification de valeur, à la sortie monophonique.



Le module Voice Shift permet de réarranger les valeurs de l'entrée polyphonique afin que les valeurs de sortie soient réassignées sur les différentes voix comme désiré. Par exemple, vous pourriez utiliser Voice Shift pour réassigner les voix 1, 2, 3, 4 aux voix de sortie 2, 3, 4, 1 ou 3, 4, 2, 1, etc. Si plusieurs voix d'entrée sont décalées vers la même voix de sortie, elles sont additionnées. Par exemple, si les voix d'entrée 1 et 2 sont envoyées sur la voix de sortie 3, le signal sortant de cette voix 3 sera la somme des signaux des voix d'entrée 1 et 2.

Wrp :valeur d'évènement monophonique qui "enroule" le décalage de voix sur lui-même : si l'enroulement est désactivé ($Wrp \leq 0$), les voix décalées vers des numéros de voix invalides sont supprimées. S'il est activé ($Wrp > 0$), les voix décalées vers des numéros de voix invalides sont "enroulées" pour retomber sur des numéros de voix valides via le modulo mathématique. Par exemple, un décalage de +1 dans un instrument à 3 voix entraîne la voix 3 à être réassignée à la voix 1 (soit 4 modulo 3). L'enroulement est désactivé par défaut.

Sh :valeur d'évènement polyphonique qui contrôle le décalage de voix. Les valeurs positives décalent les voix vers le haut (p.ex. $Sh = 1$ décale la voix

1 vers la voix 2) et les valeurs négatives les décalent vers le bas. Comme Sh est une entrée polyphonique, chaque voix peut être décalée d'une valeur particulière. La valeur par défaut de Sh est 1.

In :entrée du signal polyphonique dont vous voulez décaler les voix.

Out :sortie pour le signal polyphonique avec les voix décalées.

Audio Smoother



Auxiliary

Lisseur de signaux d'événements monophoniques à sortie audio. En règle générale, on installe un Smoother en aval d'un atténuateur ou d'une touche pour adoucir les transitions.

Les sauts du signal d'événement arrivant sont lissés en une rampe. Réglez la durée de transition dans le dialogue Properties, sous **Transition Time**, en millisecondes. Après ce temps de transition, la sortie atteint la même valeur que l'entrée tant qu'un nouveau saut ne s'est pas produit à l'entrée. Plus **Transition Time** est élevé, plus le lissage est important.

- **In** : entrée d'événement mono destinée au signal d'événement à lisser.
- **Out** : sortie audio mono destinée au signal lissé.

Event Smoother



Auxiliary

Lisseur à sortie d'événement, pour signaux monophoniques. En règle générale, on installe un Smoother en aval d'un atténuateur ou d'une touche pour adoucir les transitions.

Les sauts du signal d'événement arrivant sont lissés en une rampe. Réglez la durée de transition dans le dialogue Properties, sous **Transition Time**, en millisecondes. Après ce temps de transition, la sortie atteint la même valeur que l'entrée tant qu'un nouveau saut ne s'est pas produit à l'entrée. Plus **Transition Time** est élevé, plus le lissage est important.

Pendant la durée de la transition, les événements sont émis au taux réglé dans le menu **Settings**, sous **Control Rate**, en Hz. Plus le taux est élevé, plus le lissage a une résolution élevée.

- **In** : entrée d'événement monophonique destinée au signal à lisser. Le signal audio est échantillonné uniquement au taux Control Rate.
- **Out** : sortie d'événement mono destinée au signal lissé.

Master Tune/Level

Auxiliary



Commande Master Level et Master Tuning.

- **Tun** : entrée destinée à la commande du Master Tuning. Echelle : 1 demi-ton par unité. A 0,0, la note a3 (la 3) est accordée à 440 Hz. Plage caractéristique : [-1 ... 1].
- **Lvl** : entrée destinée à la commande du Master Level aux convertisseurs de sortie. Echelle : 1 dB par unité. 0,0 = 0 dB. Plage caractéristique : [-60...0].
- **Tun** : sortie destinée au Master Tuning.
- **Lvl** : sortie destinée au Master Level.

Tempo Info

Auxiliary

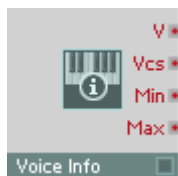


Source du réglage actuel du tempo, en battements par seconde. Pour obtenir la valeur BPM, multipliez celle-ci par 60.

- **Out** : sortie d'événement destinée au tempo actuel, en battements par seconde (Hz).

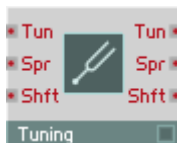
Voice Info

Auxiliary



Module Voice Info.

- **V** : entrée polyphonique destinée au numéro ID de chaque voix [1, 2, 3... Voices].
- **Vcs** : sortie destinée au nombre actuel des voix de l'instrument.
- **Min** : sortie de réglage **Min Unison Voices** de l'instrument. Ceci est le nombre minimum de voix affectées, en mode unisson, à une touche.
- **Max** : sortie de réglage **Max Unison Voices** de l'instrument. Ceci est le nombre maximum de voix affectées, en mode unisson, à une touche.



Module de commande et d'information concernant les réglages des paramètres Tuning de l'instrument.

- **Tun** : entrée destinée à commander le Tuning de l'instrument, en demi-tons.
- **Spr** : entrée destinée à commander l'importance de l'Unison Spread, en demi-tons.
- **Shft** : entrée destinée à commander la transposition des notes MIDI entrantes, par demi-ton (note shift).
- **Tun** : sortie destinée au Tuning de l'instrument, en demi-tons.
- **Spr** : sortie destinée à l'importance de l'Unison Spread, en demi-tons.
- **Shft** : sortie destinée à la transposition des notes MIDI entrantes, par demi-ton (note shift).



Source des informations concernant le système : taux d'échantillonnage (échantillons/seconde), Controlrate (en Hz) et contrainte CPU (en %).

- **SR** : sortie destinée au taux actuel d'échantillonnage, en échantillons par seconde.
- **CR** : sortie destinée au ControlRate actuel, en Hz.
- **DClk** : sortie destinée au taux d'affichage actuel, en frames par seconde. La valeur est émise avant chaque mise à jour de l'affichage.
- **CPU** : sortie destinée à la contrainte actuelle de la CPU, en pourcentage.

Note Range Info

Auxiliary

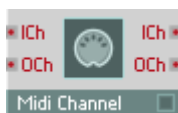


Module Note Range Info.

- **Upr** : entrée destinée à sélectionner la limite supérieure de réception des notes (upper range).
- **Lwr** : entrée destinée à sélectionner la limite inférieure de réception des notes (lower range).
- **Upr** : sortie destinée à sélectionner la limite supérieure de réception des notes (upper range).
- **Lwr** : sortie destinée à sélectionner la limite inférieure de réception des notes (lower range).

MIDI Channel Info

Auxiliary



Module Midi Channel Info.

- **ICh** : entrée destinée à sélectionner le canal d'entrée MIDI.
- **OCh** : entrée destinée à sélectionner le canal de sortie MIDI.
- **ICh** : sortie destinée au canal d'entrée MIDI actuel de l'instrument.
- **OCh** : sortie destinée au canal de sortie MIDI actuel de l'instrument.



Le module Snapshot vous permet de changer les snapshots et de morpher entre eux dans REAKTOR. Ce module contient un panneau de commande qui fonctionne exactement comme le module **List** (voir la section Module Panneau).

- **Snp** : entrée audio destinée à sélectionner le snapshot à appeler ou à mémoriser. Plage : 1 ... 128.
- **Bnk** : entrée audio destinée à sélectionner la banque de snapshots. Plage : 1 ... 16.
- **Recl** : une valeur positive appelle le snapshot sélectionné via les entrées Snp et Bnk.
- **St** : une valeur positive mémorise le snapshot à l'emplacement sélectionné via les entrées Snp et Bnk.
- **A** : une valeur positive utilise les valeurs existant aux entrées Snp et Bnk pour sélectionner un snapshot pour la position A du morph.
- **B** : une valeur positive utilise les valeurs existant aux entrées Snp et Bnk pour sélectionner un snapshot pour la position B du morph.
- **Mrph** : cette entrée commande la position de morph entre les snapshots sélectionnés pour A et B. Valeur $\leq 0,0$: A, valeur 0,0 -1,0 : morphe entre A et B. Valeur $\geq 1,0$: B.
- **Sw** : des valeurs négatives et nulles mettent les commutateurs en position donnée par snapshot A. Des valeurs positives mettent les commutateurs en position donnée par snapshot B.

- **MT** : entrée d'événement destinée au temps de morph, en ms.
- **Rnd** : une valeur positive déclenche la fonction aléatoire pour le snapshot sélectionné.
- **Amt** : entrée destinée à l'intensité aléatoire (randomization amount). Plage : 0.0 ... 1.0 (1.0 = 100%).
- **Mrg** : une valeur positive déclenche la fonction Random Merge (un snapshot est créé sur une base aléatoire, qui hérite de certaines propriétés des deux snapshots sélectionnés avec A et B).
- **Sn**p : sortie destinée au numéro du snapshot sélectionné actuellement. Une valeur est émise chaque fois qu'un snapshot est appelé ou mémorisé.
- **Bnk** : sortie destinée au numéro de la banque de snapshots sélectionnée actuellement. Une valeur est émise chaque fois qu'un snapshot est appelé ou mémorisé.
- **Recl** : la valeur 1,0 est émise chaque fois qu'un snapshot est appelé.
- **St** : la valeur 1,0 est émise chaque fois qu'un snapshot est enregistré.
- **A** : sortie destinée au numéro du snapshot de la position de morph A. Une valeur est émise lorsqu'un snapshot est appelé ou sélectionné pour la position A.
- **B** : sortie destinée au numéro du snapshot de la position de morph B. Une valeur est émise lorsqu'un snapshot est appelé ou sélectionné pour la position B.
- **Mrph** : sortie destinée à la position de morph. Valeur = 0,0 : A, valeur 0,0 -1,0 : morphe entre A et B. Valeur = 1,0 : B.
- **Sw** : sortie destinée à la position A/B des commutateurs et touches. Valeur = 0,0, lorsque les Switches/Buttons sont en position définie pas le snapshot A. Valeur = 1,0, lorsque les Switches/Buttons sont en position définie par snapshot B.
- **MT** : sortie destinée au temps de morph, en ms.
- **Rnd** : la valeur 1,0 est émise lors du déclenchement de la fonction aléatoire.
- **Amt** : sortie destinée à l'intensité aléatoire (randomization amount). Plage : 0.0 ... 1.0 (1.0 = 100%).
- **Mrg** : la valeur 1,0 est émise lors du déclenchement de la fonction Random Merge.

Set Random

Auxiliary



Définit le germe aléatoire et initialise ainsi le générateur de hasard, qui sera utilisé dans les tous les modules **Event Randomize**, **Slow Random** et **Geiger**. Seuls les modules situés dans un même instrument sont affectés. Chaque valeur génère une suite de chiffres aléatoires unique.

Unison Spread

Auxiliary



Source d'événement polyphonique pour une valeur invariable de laquelle les paramètres des voix unisson sont déviés.

En mode unisson, décalez légèrement les paramètres des différentes voix jouant la même note pour les différencier légèrement, ce qui rend le son plus large et pas simplement plus fort. Ceci est automatique pour la hauteur de son (Note Pitch), et est piloté via le paramètre **Unison Spread** des Propriétés de l'instrument. Vous pouvez également « écarter » d'autres paramètres en leur ajoutant une valeur du module Unison Spread.

La valeur existant à l'entrée du module détermine ces valeurs, et donc l'importance de l'« écartement ». Une valeur polyphonique est obtenue à la sortie, différente pour chacune des voix d'une note. La valeur se modifie uniquement lorsqu'une nouvelle note est jouée.

Snap Value

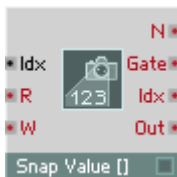
Auxiliary



Ce module enregistre la valeur d'entrée avec un snapshot et émet cette valeur lorsque le snapshot est appelé.

In: entrée destinée à la valeur à enregistrer avec un snapshot. Le signal d'entrée est prélevé au moment de l'enregistrement du snapshot. Si le module est connecté à une source d'événements, les événements d'entrée sont transmis à la sortie.

Out : sortie destinée à la valeur enregistrée dans le dernier snapshot appelé.



Enregistre/rappelle des tableaux de valeurs en virgule flottante (fractionnaires) dans/depuis le tampon d'édition et les snapshots.

Un Snap Value Array unique peut contenir entre 1 et 40 tableaux, chaque tableau pouvant contenir un nombre quelconque d'éléments, la seule limite étant la mémoire disponible. Tous les tableaux d'un module doivent contenir le même nombre d'éléments. Le nombre de tableaux ainsi que le nombre d'éléments de chaque tableau sont définis dans les propriétés du module.

La mémoire pour le module Snap Value Array est allouée dynamiquement. Ceci signifie que la création de snapshots entraîne l'allocation de mémoire supplémentaire, et leur suppression entraîne une libération de mémoire. Ceci permet de maintenir la mémoire utilisée aussi basse que possible.

Lorsque l'option Snap Isolate est activée dans les propriétés, seule la valeur la plus récente écrite dans chaque élément de tableau est enregistrée (et rappelée lors de l'initialisation de l'instrument), et aucune donnée n'est enregistrée ni rappelée lors des opérations sur les snapshots.

Une application typique de Snap Value Array est l'enregistrement de données de séquenceur dans un snapshot. Dans ce but, le module est souvent utilisé en conjonction avec des modules Event Tables ou Multi Display.

Les entrées de Snap Value Array acceptent des signaux monophoniques uniquement.

Idx : index de l'élément de tableau à appeler (à la fois pour les opérations d'écriture et de l'écriture). Les tableaux débutent à 1 : l'index du premier élément est 1 (et non 0). Les valeurs fractionnaires sont arrondies à l'entier le plus proche. Lorsque les valeurs de Idx sont en dehors de la plage de 1 à N, le comportement dépend de l'option Index Behaviour dans les propriétés.

R : lorsqu'un évènement de valeur quelconque arrive à l'entrée R (= Read, Lecture), l'élément de tableau spécifié par l'entrée Idx est lu et sa valeur est envoyée au port de sortie. Autrement dit, chaque évènement à l'entrée R propage un évènement unique vers le port de sortie du tableau. Plusieurs tableaux sont appelés en parallèle par le même index Idx.

Ainsi, des évènements arrivant à l'entrée R propagent des évènements en sortie à chaque port de sortie de tableau (de la valeur de l'élément appelé par Idx). Il est indispensable de fixer la valeur de Idx avant que des évènements n'arrivent à l'entrée R.

W : lorsqu'un évènement arrive à l'entrée W (= Write, Écriture), sa valeur est écrite dans l'élément [Idx] du tableau, écrasant toute données déjà écrite dans cet élément. Le Snap Value Array fournit une entrée W séparée pour chaque tableau qu'il contient (un tableau par défaut, mais vous pouvez en créer plus dans les propriétés). Chaque port peut être renommé comme vous le souhaitez. Lorsque l'option Events Thru est activée dans les propriétés, les évènements arrivant aux entrées W sont transmis aux sorties Out correspondantes.

N : la sortie N transmet le nombre d'éléments de chaque tableau.

Gate : Gate envoie un évènement de valeur 1 avant que les ports de sortie de tableau n'envoient des évènements, puis envoie un évènement de valeur 0 après.

Idx : sortie transmettant le numéro de l'élément en train d'être lu/écrit (en réponse aux évènements arrivant aux ports R et W, ou des opérations de snapshots comme le rappel et le fondu). En fonction des opérations de lecture, le numéro Idx est transmis avant que l'évènement de valeur ne soit transmis aux sorties Out.

Out : la valeur de l'élément de tableau actuellement appelé (tel que défini par Idx) est transmise ici en réponse aux évènements au port R et aux opérations de snapshots (rappel, fondu, etc.). Lorsque l'option Self-Iteration est activée dans les propriétés, tous les éléments de tableau sont envoyés en sortie en série (le premier élément, puis le second, puis le troisième, etc.) dès que l'une des opérations suivantes survient : initialisation, activation, rappel de snapshot, randomisation, random merge et fondu. Self-Iteration permet de mettre à jour un ensemble complet de données avec les opérations de snapshots.

Terminals

Les terminaux servent à introduire les signaux audio et d'événement dans les instruments de REAKTOR et leurs sous-structures macro, et de les en faire sortir.

In Port

Terminal



Terminal destiné aux signaux audio et d'événement pour créer des ports d'entrée pour les instruments et les macros. Pour générer automatiquement des ports, tirez un câble sur un instrument ou une macro dans la fenêtre Structure en maintenant la touche Ctrl enfoncée.

Out Port

Terminal



Terminal destiné aux signaux audio et d'événement pour créer des ports de sortie des instruments et macros. Pour générer automatiquement des ports, tirez un câble sur un instrument ou une macro dans la fenêtre Structure en maintenant la touche Ctrl enfoncée.

Send

Terminal



Terminal destiné aux signaux audio et d'événement pour créer une connexion sans câble à l'intérieur d'un instrument.

Chaque module Send ajouté génère une entrée dans la liste des sources de signal disponibles, dans Properties d'un module Receive.

Receive

Terminal



Terminal d'entrée destiné aux signaux audio et d'événement pour créer une connexion sans câble à l'intérieur d'un instrument.

Page Properties - Function

Chaque module Send ajouté à un même instrument génère une entrée dans la liste. Le nom de l'entrée est fonction de la vignette du module Send. Les touches **Up** et **Down** situées en dessus de la liste servent à déplacer une entrée de la liste vers le haut ou vers le bas.

La liste contient les colonnes suivantes, que vous pouvez ordonner en cliquant sur l'entrée correspondante de l'en-tête :

- **#** : indique la position du module Send. La valeur par défaut correspond à ce numéro.
- **Label** : le nom du module Send. Si vous renommez un module Send, la vignette est actualisée dans la liste.
- **State** : indique si une connexion est possible avec le module Send correspondant. Etablissez la connexion uniquement si ce champ indique **OK**.
- **Use** : cliquez dans ce champ pour établir une connexion avec le module Send correspondant. Une connexion établie est représentée par une croix.

Le réglage **Mouse Resolution** a de l'importance uniquement lorsque, pour l'élément de commande, le style **Spin** est sélectionné dans la page **Appearance**, qui vous permet de cliquer sur l'entrée de cet élément, et de déplacer la souris vers le haut ou vers le bas pour modifier celle-ci.

La valeur **Default** est utilisée dans tous les cas d'initialisation de l'élément de commande. Dans ce cas, l'entrée de la liste sélectionnée est celle dont **#** correspond à la valeur par défaut.

Page Properties - Appearance

La représentation du panneau de l'élément de commande se modifie en fonction du style sélectionné dans les propriétés, à la page **Appearance**. Les styles suivants sont disponibles :

- **Button** : chaque port d'entrée d'un module génère un bouton. Toutes les touches sont ordonnancées verticalement dans le panneau d'instrument, formant une barre. L'entrée sélectionnée est représentée de la couleur représentative de l'instrument.
- **Menu** : chaque port d'entrée du module génère une nouvelle ligne dans une liste déroulante.

- **Text Panel** : chaque port d'entrée du module génère une nouvelle entrée dans la liste, qui en affiche plusieurs simultanément. Si vous générez plus d'entrées que l'affichage, dont vous définissez la taille à l'aide des champs **Size X** et **Size Y**, à la page **Appearance** des propriétés, ne peut en contenir, il se munit de barres de déroulement.
- **Spin** : chaque port d'entrée du module génère une nouvelle entrée dans une liste. Les touches **+** et **-** situées à droite de l'affichage vous permettent de naviguer parmi les entrées.

Les champs **Size X** et **Size Y** servent à régler la taille de l'affichage de l'élément de commande dans le panneau.

IC Send

Terminal



Transmet des signaux évènements monophoniques à tout module capable de recevoir une transmission IC (Internal Connection). Ces modules incluent les modules IC Receive, mais aussi de nombreux éléments de panneau (p.ex. les potentiomètres et les commutateurs). Comme les connexions internes travaillent au niveau global (au niveau de l'ensemble), ce module peut être utilisé pour établir des connexions "sans fil" entre différents instruments de l'ensemble.

Le module IC Send dispose d'un affichage de panneau, ce qui permet de configurer les connexions depuis le panneau de l'instrument. Tous les modules de l'ensemble capables de recevoir les transmissions IC y apparaissent, sauf ceux dont l'option No Entry in IC Menu est activée (dans les propriétés). Les connexions peuvent également être établies via la boîte de dialogue Properties.

IC Receive

Terminal



Reçoit et génère des signaux évènements monophoniques aux modules connectés via le protocole IC (Internal Connection). Habituellement, le module IC Receive est utilisé en conjonction avec des modules IC Send, mais il peut être connecté à n'importe quel module capable de gérer les transmissions IC (comme les potentiomètres et les commutateurs).

Les connexions IC peuvent être établies dans les propriétés. Lorsque les connexions ont lieu avec des modules IC Send, elles peuvent aussi être établies via l'interface de panneau des IC Send.

OSC Send



Terminal

Les modules **OSC Send** et **OSC Receive** vous permettent de transmettre des messages OSC. Ces deux modules disposent d'une gestion dynamique des ports, donc vous pouvez ajouter de nouveaux ports d'entrée et de sortie en tirant un câble dans une zone vide jouxtant les ports existants en maintenant la touche **Ctrl** enfoncée.

Des connexions multiples au module **OSC Send** ont pour conséquence des messages OSC à arguments multiples. Par exemple, si vous connectez les sorties **MX** et **MY** du module **XY** à un module **OSC Send**, chaque message OSC contient à la fois la valeur **X** et la valeur **Y**. Il vous faut donc câbler plusieurs sorties dans le module **OSC Receive** pour qu'il soit en mesure de réceptionner les arguments supplémentaires des messages OSC. Jusqu'à 10 arguments sont ainsi possibles dans un message OSC.

Si vous avez créé plus d'un port In pour le module **OSC Send**, seul le premier d'entre eux déclenche les messages OSC.

Certains autres modules de REAKTOR permettent d'émettre et de recevoir des messages OSC, par exemple les éléments de commande. Les réglages d'émission et de réception sont les mêmes que ceux des modules OSC Terminal.

Le module dispose d'une gestion dynamique des ports d'entrée. Définissez le nombre d'entrées en utilisant **Min Num Port Groups** de la page **Function** des propriétés.

OSC Receive

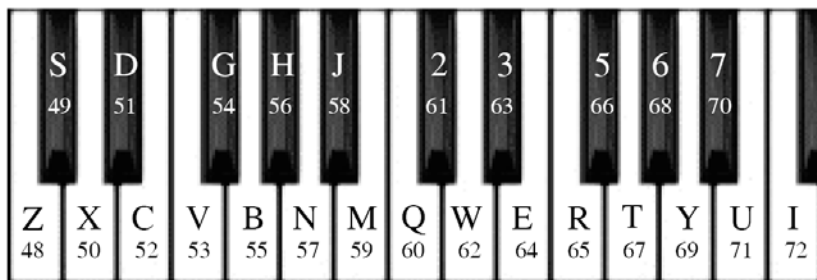


Terminal

Voir module OSC Send.

Annexe

Transposition des notes MIDI entrantes



- L'appui sur la touche **Majuscule** décale toutes les notes MIDI entrantes de deux octaves vers le haut (+24).
- L'appui sur Windows XP: *
Ctrl + Majuscule / OS X: **⌘ + Majuscule** décale toutes les notes MIDI entrantes de deux octaves vers le bas (-24).

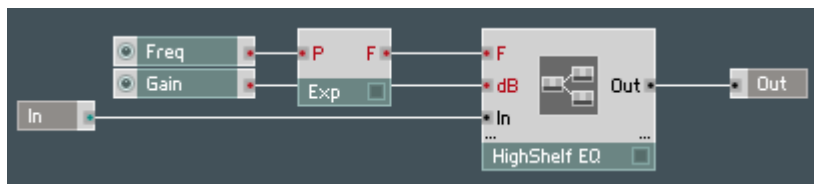
Premiers pas avec Reaktor Core

Qu'est-ce que Reaktor Core ?

Reaktor Core Reaktor Core est une nouvelle couche logicielle de fonctionnalités de Reaktor avec un nouvel ensemble de fonctions. Comme il y a déjà un ancien niveau de fonctionnalités, à partir de maintenant nous appellerons ces deux niveaux respectivement *niveau core* (*noyau* en anglais) et *niveau primaire*. De plus, lorsque nous parlerons de “structure de niveau primaire”, nous entendrons par là la structure interne d’un instrument ou d’une macro, mais pas celle d’un ensemble.

Les fonctions de Reaktor Core ne sont pas directement compatibles avec celles du niveau primaire de Reaktor, une interface est nécessaire entre les deux. Cette interface est constituée par les *cellules core*.

Les cellules *core* se situent dans les structures du niveau primaire, ressemblant et se comportant de façon similaire aux autres modules du niveau primaire. Voici un exemple de structure, utilisant une cellule *core HighShelf EQ*, qui est différente de la version habituelle du niveau primaire dans sa façon de répondre aux contrôles de fréquence et d’amplification :



À l'intérieur des cellules *core* se trouvent des structures Reaktor Core, qui constituent un moyen efficace d'implémenter des fonctionnalités personnalisées de traitement du signal (ou “DSP”) de bas niveau, et permettent aussi de construire des structures de traitement du signal à plus grande échelle à partir de ces mêmes fonctionnalités. Ces structures seront étudiées plus loin dans le texte.

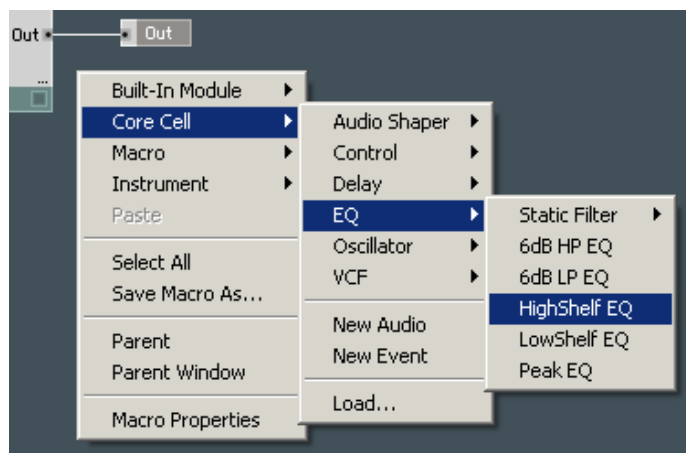
Bien que l'une des principales caractéristiques de Reaktor Core soit de pouvoir construire ces structures de DSP de bas niveau, elles ne se limitent pas à cela. Si vous n'êtes pas un expert en traitement du signal, nous vous avons facilité le travail en créant une librairie de modules que vous pouvez connecter dans les structures *core* de la même façon qu'au niveau primaire, et une librairie

de cellules *core* que vous pouvez immédiatement utiliser dans les structures du niveau primaire.

À dire vrai, chez Native Instruments, nous n'avons plus vraiment envie de créer des milliers de nouveaux modules de niveau primaire pour les nouvelles versions de Reaktor. Nous préférons les construire en utilisant notre nouvelle technologie Reaktor Core et les livrer sous forme de cellules *core*. Vous trouverez déjà un ensemble de nouveaux filtres, enveloppes, effets, etc., dans la librairie de cellules *core*.

Utilisation des cellules *core*

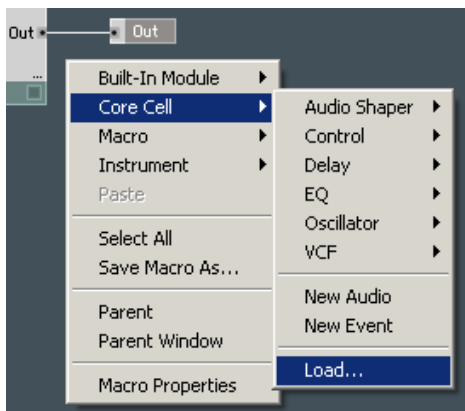
La librairie de cellules *core* est accessible depuis les structures du niveau primaire via un clic droit sur le fond de l'écran, puis en choisissant le sous-menu *Core Cell*:



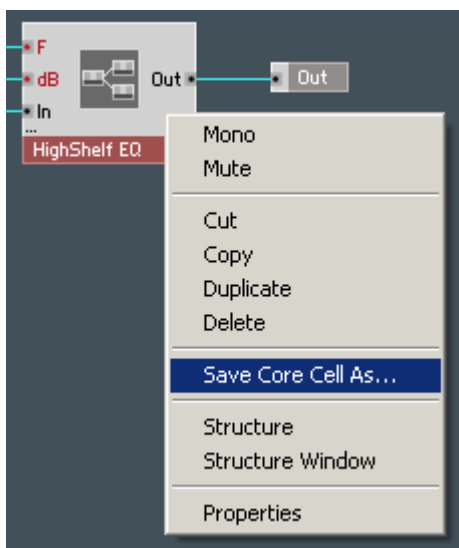
Comme vous pouvez le voir, différents types de cellules *core* peuvent être utilisés comme des modules classiques de niveau primaire.

Une limitation importante des cellules *core* est que vous ne pouvez pas les utiliser à l'intérieur des boucles d'événements. Toute boucle d'événement traversant une cellule *core* sera boquée par Reaktor.

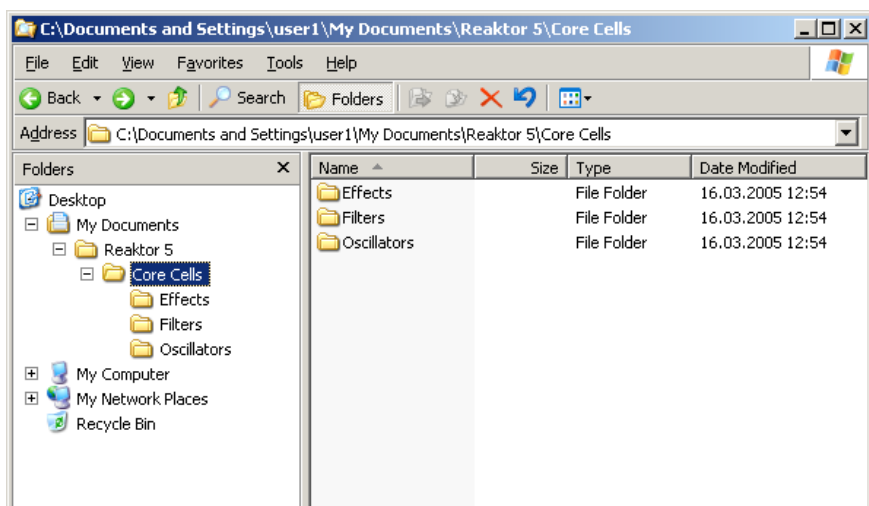
Vous avez aussi la possibilité d'insérer des cellules *core* qui ne sont pas dans la librairie. Pour ce faire, utilisez la commande *Load...* du même menu *Core Cell*:



Avant de pouvoir charger des fichiers de cellules core, il vous sauvegarder ces cellules core. Voici comment faire: effectuez un clic droit sur une cellule core et sélectionnez *Save Core Cell As*:

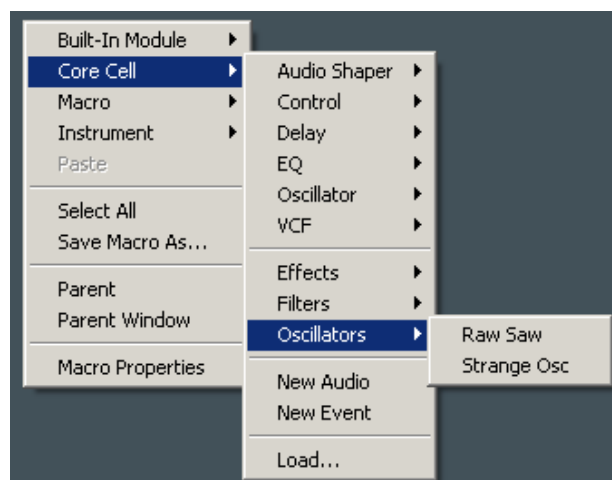


Vous pouvez également placer vos propres cellules core dans le menu pour éviter d'utiliser la commande *Load*. Pour ce faire, placez-les dans le sous-dossier "Core Cells" de votre dossier utilisateur. Encore mieux: mettez-les non pas directement dans le dossier "Core Cells" mais dans des sous-dossiers de votre crû. Voici un exemple de ce type d'organisation:



Dans cet exemple, “*My Documents\Reaktor 5*” est le dossier utilisateur. Sur votre ordinateur, il peut porter un autre nom, en fonction du choix que vous avez effectué lors de l’installation ou dans les préférences de Reaktor. Le dossier “*Core Cells*” se trouve dans ce dossier. S’il n’existe pas, créez-le manuellement.

Dans ce dossier, nous en observons trois autres: “*Effects*”, “*Filters*” et “*Oscillators*”. À l’intérieur de ces dossiers se trouvent les fichiers des cellules core qui seront affichées dans la partie utilisateur du menu *Core Cell*:



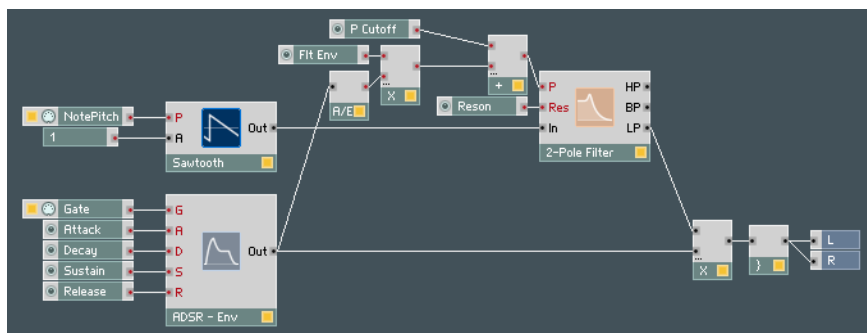
Le contenu du menu est scanné lors du démarrage de Reaktor. Si vous ajoutez de nouveaux fichiers dans ces dossiers, vous devez redémarrer Reaktor pour qu'il les prenne en compte.

Les dossiers vides ne sont pas affichés dans le menu, ils doivent contenir des fichiers pour être affichés.

Vous ne devez en aucun cas placer vos propres fichiers dans la librairie système. Le dossier de la librairie système peut être modifié voire complètement effacé lors des mises à jour, et vos fichiers seraient alors perdus! La librairie utilisateur (en anglais *user library*) est l'endroit indiqué pour placer tout fichier non inclus dans l'application elle-même.

Utilisation des cellules core dans un exemple réel

Nous allons prendre ici un instrument Reaktor construit uniquement avec des modules de niveau primaire et nous allons le modifier en y plaçant quelques cellules *core*. Dans le dossier “*Core Tutorial Examples*” du dossier d’installation de Reaktor, trouvez l’ensemble *One Osc.ens* et ouvrez-le. Cet ensemble est constitué d’un seul instrument ayant la structure interne suivante:



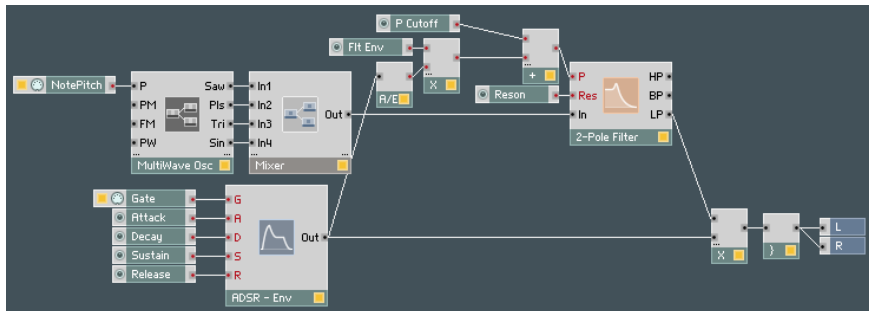
Comme vous pouvez le constater, il s'agit d'un synthétiseur soustractif très simple doté d'un oscillateur, d'un filtre et d'une enveloppe. Nous allons remplacer l'oscillateur par un autre, plus puissant. Faites un clic droit sur le fond et sélectionnez *Core Cell > Oscillator > MultiWave Osc*:



La caractéristique la plus importante de cet oscillateur est qu'il fournit simultanément plusieurs formes d'ondes analogues calées en phase. Nous souhaitons utiliser leur mélange en lieu et place d'un simple oscillateur en dents de scie. Fort heureusement, il existe déjà un mixeur accessible via *Insert Macro > Classic Modular > O2 Mixer Amp > Mixer – Simple – Mono*:



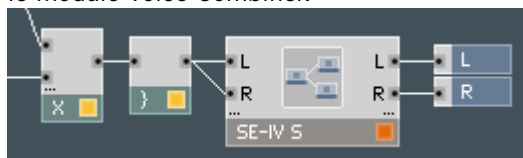
Connectons maintenant ensemble le mixeur et l'oscillateur et remplaçons l'oscillateur en dents de scie par leur combinaison:



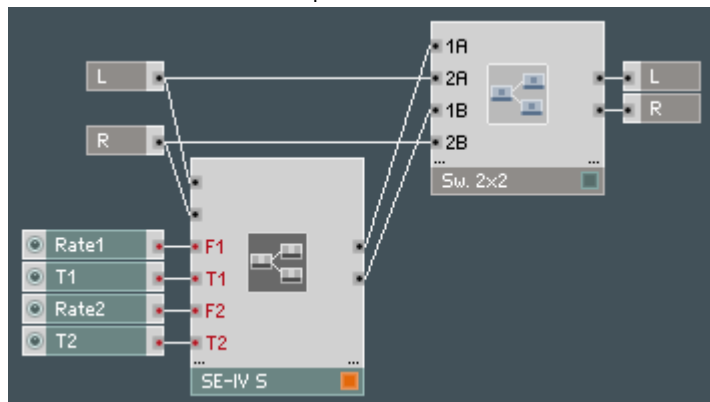
Passez en vue *panel*. Vous pouvez maintenant utiliser les quatre potentiomètres du mixeur pour modifier le mélange des formes d'onde.

Modifions encore l'instrument: ajoutons-lui un effet de chorus basé sur Reaktor Core. Nous écrivons "basé sur Reaktor Core" parce que, même si le chorus lui-même est construit en tant que cellule *core*, la partie contenant les contrôles du panneau de ce chorus est encore construite avec les fonctions du niveau primaire. Ceci est dû au fait que les structures Reaktor Core ne peuvent pas définir leurs propres panneaux de contrôles: ces panneaux doivent être construits au niveau primaire.

Sélectionnez *Insert Macro > RX Based > Chorus > SE IV S* et insérez-le après le module *Voice Combiner*:

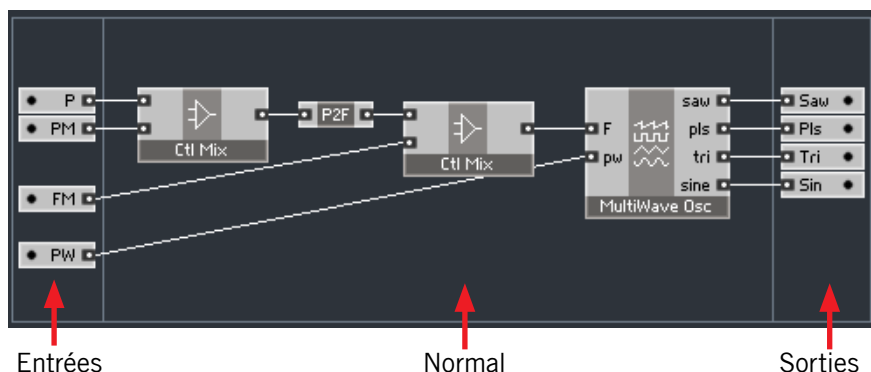


Si vous jetez un œil à l'intérieur du chorus, vous verrez la cellule *core* du chorus et les contrôles du panneau:



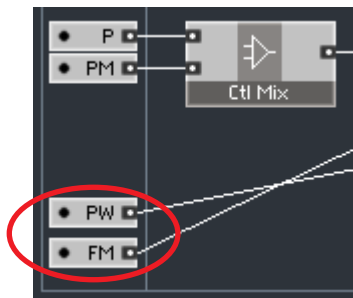
Édition basique des cellules core

Penchons-nous maintenant sur l'édition des cellules *core*. Nous allons commencer avec quelque chose de simple, comme la modification d'une cellule core existante pour l'adapter à vos besoins. Tout d'abord, double-cliquez sur le *MultiWave Osc* pour pénétrer à l'intérieur:

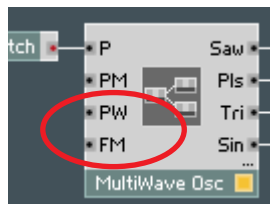


Ce que vous voyez maintenant est une structure Reaktor Core. Les trois zones séparées par des lignes verticales délimitent trois types de modules: les modules d'entrée (à gauche), les modules de sortie (à droite) et les modules normaux (au centre).

Les entrées et les sorties peuvent être déplacées uniquement verticalement, et leur ordre correspond à l'ordre dans lequel elles apparaissent à l'extérieur de la structure. Les modules normaux (au centre donc) peuvent être déplacés dans n'importe quelle direction. Par exemple, placez l'entrée *FM* sous l'entrée *PW*:



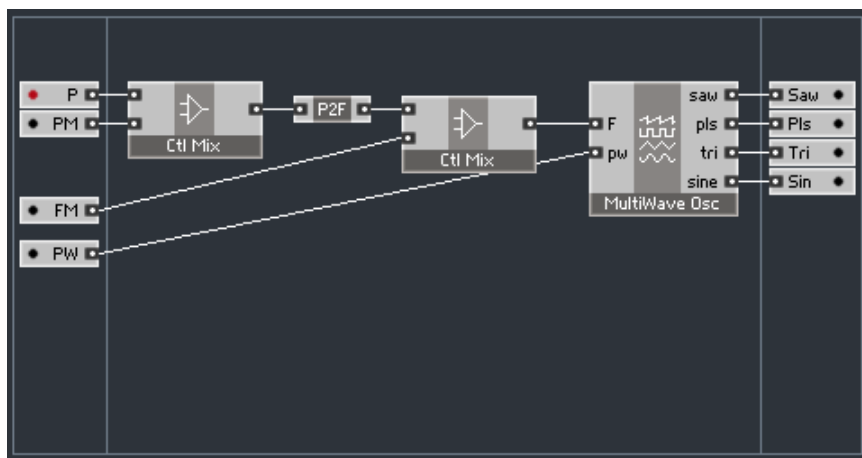
Vous pouvez maintenant double-cliquer sur le fond pour remonter à la structure extérieure du niveau primaire et observer la modification de l'ordre des ports d'entrée.



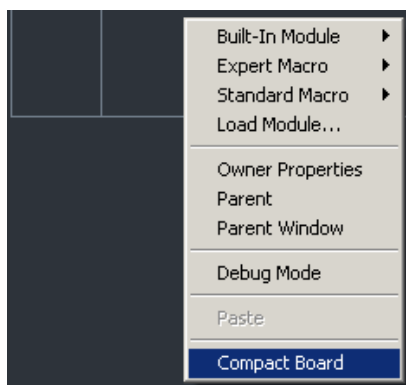
Revenons au niveau *core* et n'oublions pas de remettre les ports d'entrée dans l'ordre original:



Comme vous l'avez probablement déjà remarqué, si vous déplacez les modules, les trois parties de la structure *core* peuvent s'agrandir afin de toujours englober tous les modules. Cependant, elles ne se réduisent pas automatiquement, ce qui peut les conduire à devenir beaucoup, beaucoup trop larges:



Vous pouvez les réduire via un clic droit sur le fond et en sélectionnant la commande *Compact Board*:

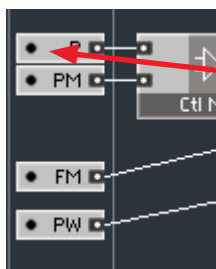


Maintenant que nous savons déplacer les composants d'une cellule *core*, et même réarranger l'ordre de ses ports d'entrée et de sortie, essayons-nous à quelques autres manipulations.


Pour une cellule *core* qui dispose de sorties audio, il est possible de commuter le type de ses entrées entre *audio* et *événement* (vous trouverez une explication détaillée de leur différence plus loin dans le texte). Dans l'exemple ci-dessus, nous avons utilisé un module *MultiWave Osc* qui dispose d'entrées et de sorties audio. Mais dans notre cas, nous n'avons pas vraiment besoin d'entrées audio, puisque le seul élément connecté à l'oscillateur est le potentiomètre de pitch (hauteur tonale). Ne serait-il pas plus économique (en ressources

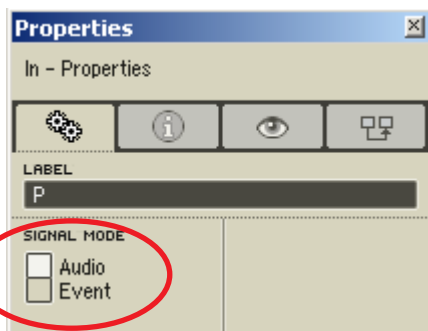
processeur) d'utiliser au moins partiellement des évènements ? La réponse est évidemment "Oui !", aussi allons-nous le faire de ce pas.

Nous vous suggérons de changer au moins les entrées *P* et *PM* pour les passer en mode évènement. Cela devrait permettre la plus grande économie en puissance processeur. Double-cliquez sur le module d'entrée *P* pour ouvrir sa fenêtre de propriétés:



Double-cliquez ici

Ouvrez la page de fonction de la fenêtre de propriétés (si nécessaire en cliquant sur le bouton ). Vous devriez maintenant voir s'afficher la propriété de mode de signal (Signal Mode):

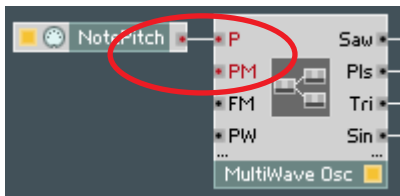


Réglez-la sur *event* (*évènement* en anglais). Remarquez comment le gros point à gauche du module d'entrée passe du noir au rouge pour indiquer que l'entrée est désormais en mode évènement (c'est plus visible lorsque le module n'est pas sélectionné, cliquez quelque part ailleurs pour le désélectionner):

Le bouton devient rouge 

Cliquez maintenant sur l'entrée *PM* et passez-la aussi en mode évènement. Si vous le souhaitez, vous pouvez aussi passer les deux entrées restantes en mode évènement. Après avoir fini, double-cliquez sur le fond de la structure

pour revenir au niveau primaire et observez la nouvelle couleur rouge des ports et l'utilisation diminuée du processeur.



Il est parfois absurde de commuter un port d'un type à l'autre. Par exemple, une entrée recevant véritablement un signal audio (contenant un vrai son, pas simplement un signal de contrôle tel qu'une enveloppe calée sur le son) ne doit pas être commutée en mode événement. Non seulement c'est absurde, mais de plus cela risque de ruiner le fonctionnement du module. Un autre cas absurde: lorsque vous avez une entrée de type événement vraiment sensible aux événements, p.ex. une entrée de contrôle sur une enveloppe (comme les entrées Gate des enveloppes du niveau primaire de Reaktor). Si vous essayez de les passer en mode audio, elles ne fonctionneront plus correctement.

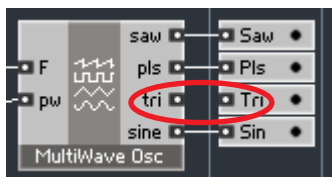
Parallèlement aux cas où il est évident que la commutation du type de port est absurde, il peut y avoir des cas dans lesquels elle est autorisée mais les modules refusent alors de fonctionner. Cependant, ces cas sont normalement très rares, ou bien c'est qu'une erreur s'est glissée dans l'implémentation du module. Généralement, la commutation du type de port fonctionne. Nous vous proposons la règle de commutation suivante:

Dans une cellule core bien conçue, une entrée de contrôle audio peut être commutée en événement sans aucun problème. Une entrée de type événement peut être commutée en audio seulement si elle n'a pas de fonction de commande.

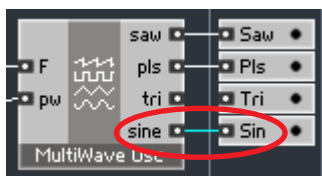
Pour économiser du processeur, vous pouvez aussi déconnecter les sorties audio dont vous n'avez pas besoin, désactivant ainsi les parties inutilisées de la structure Reaktor Core. Vous devez effectuer cette opération depuis l'intérieur de la structure – à l'extérieur, les connexions n'ont pas d'effet sur la désactivation des éléments de la structure *core*.

Dans notre exemple, décidons que nous n'avons pas besoin des quatre sorties mais seulement de la dent de scie et du train d'impulsions. Nous pouvons aller dans le *MultiWave Osc* et déconnecter les sorties inutiles. La déconnexion est une opération très simple: il suffit de cliquer sur le port d'entrée

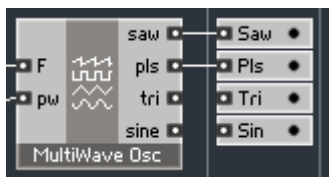
de la connexion, de déplacer la souris n'importe où sauf sur un port de sortie et de relâcher le bouton. Commençons avec la sortie "Tri" (pour "triangle"). Cliquez sur le port de la sortie "Tri" et déplacez la souris sur un endroit vide du fond de l'écran.



Il y a un autre moyen de supprimer une connexion. Cliquez sur la connexion entre la sortie "sine" du *MultiWave Osc* et la sortie "Sin" de la cellule *core*, de façon à la sélectionner (elle change de couleur):

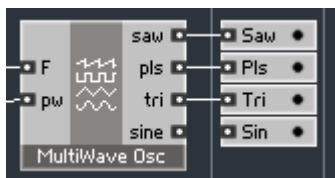


Vous pouvez maintenant appuyer sur la touche *Suppr* de votre clavier pour supprimer la connexion:



Après avoir supprimé les deux câbles, l'indicateur d'utilisation du processeur devrait descendre encore un peu.

Et si vous changez d'avis et décidez de les reconnecter ? C'est tout aussi simple: cliquez sur l'entrée ou la sortie que vous souhaitez reconnecter, déplacez la souris sur le port opposé de la connexion et relâchez le bouton. Par exemple, cliquez sur la sortie "tri" du *MultiWave Osc* et déplacez la souris jusqu'au module de sortie "Tri". La connexion est rétablie:



Bien sûr, les possibilités de modification des cellules core sont bien plus étendues que ce que nous venons de décrire. Vous en apprendrez beaucoup plus en continuant votre lecture.

Pénétrer dans l'environnement Reaktor Core

Cellules core évènement et audio

Les cellules *core* existent en deux parfums: Évènement et Audio (en anglais Event et Audio). Les cellules *core* événements ne peuvent recevoir en entrée que des signaux événements du niveau primaire, et ne produisent en sortie que des signaux événements du niveau primaire, en réponse aux premiers. Les cellules *core* audio peuvent recevoir en entrée des signaux événements et audio, mais ne produisent en sortie que des signaux audio:

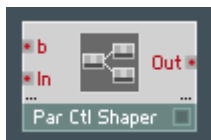
Parfum	Entrées	Sorties	Sources d'Horloge
Évènement	Évènements	Évènements	Désactivées
Audio	Évènements/ Audio	Audio	Activées

Ainsi, les cellules audio permettent d'implémenter oscillateurs, filtres, enveloppes et autres effets, tandis que les cellules événements ne servent qu'aux tâches de traitement des événements.

Les modules *HighShelf EQ* et *MultiWave Osc* que nous avons déjà vus sont des exemples de cellules core audio (elles ont des sorties audio).

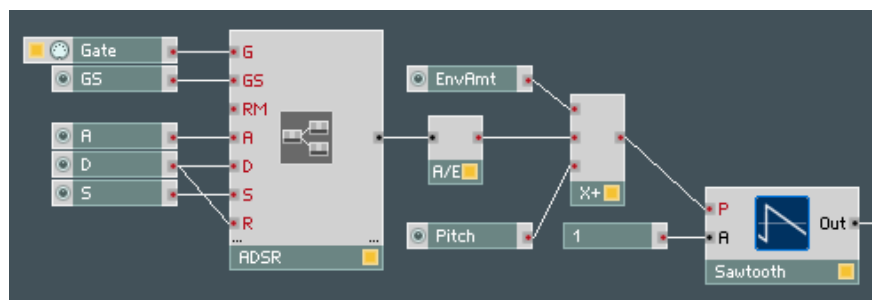


Et voici un exemple de cellule *core* évènement:



Ce module est un modelleur parabolique (en anglais “parabolic shaper”) pour les signaux de contrôle; il peut par exemple être utilisé pour le modelage de courbes de vélocité ou de signaux de LFO.

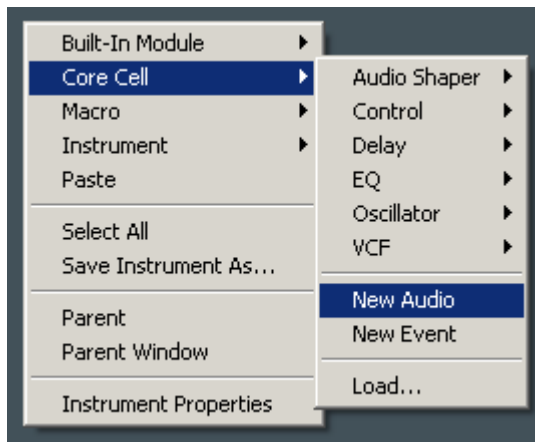
Nous avons écrit ci-dessus que les cellules *core* évènements ne servent qu’aux tâches de traitement des évènements. Les sources d’horloge de ces cellules étant désactivées dans ces cellules (cf. tableau ci-dessus), elles ne peuvent générer leurs propres évènements et ne peuvent donc servir à implémenter des modules comme des enveloppes ou des LFOs calés sur des évènements. Si vous souhaitez implémenter de tels modules, nous vous suggérons de prendre une cellule audio et de convertir sa sortie en évènements grâce à l’un des convertisseurs audio-évènement du niveau primaire:



La structure ci-dessus utilise d’abord une cellule *core* audio implémentant une enveloppe ADSR, puis convertit sa sortie sous forme d’évènements pour moduler un oscillateur.

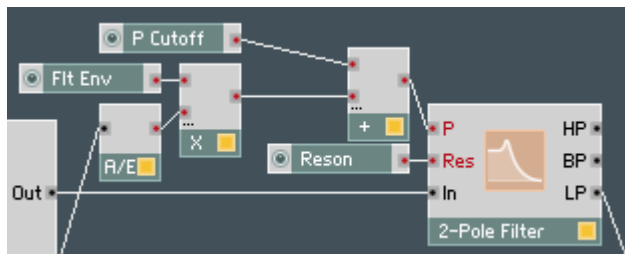
Création de votre première cellule core

Pour créer une nouvelle cellule core, faites un clic droit sur le fond de l'écran dans la structure du niveau primaire et sélectionnez *Core Cell > New Audio* ou (pour les cellules événements) *Core Cell > New Event*:



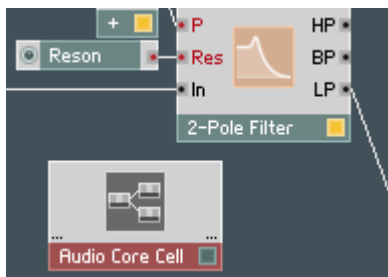
Nous allons construire une nouvelle cellule core à partir du modèle inclus dans le module *One Osc.ens* que vous avez déjà manipulé. Nous utiliserons la nouvelle version de cet ensemble (avec le nouvel oscillateur et le chorus) que nous avons construite précédemment, mais si vous ne l'avez pas sauvegardée, ne vous en faites pas, vous pouvez faire la même chose à partir du *One Osc.ens* original.

Comme vous pouvez le constater dans cet ensemble, nous modulons l'entrée P du filtre, qui n'accepte que des signaux événements. Nous n'utilisons pas la version FM du même filtre car d'une part elle se comporte moins bien aux fréquences de coupure élevées, et d'autre part l'échelle de modulation de l'entrée FM est linéaire, ce qui donne généralement de moins bons résultats musicaux pour une modulation par enveloppe (ce que l'on appelle habituellement – mais incorrectement – les “enveloppes lentes”):

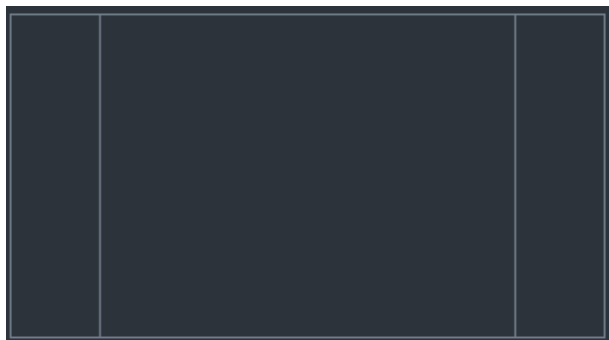


Comme nous voulons moduler une entrée évènement, nous avons besoin de convertir l'enveloppe en signal évènement, ce qui est accompli par le convertisseur A/E. La vitesse de contrôle en résultant est relativement basse. Bien entendu, nous aurions pu utiliser un convertisseur fonctionnant à une vitesse plus élevée (et consommant plus de puissance processeur), mais au lieu de cela nous allons remplacer ce filtre par un autre que nous construirons sous forme de cellule *core*. Nous aurions pu aussi en prendre un directement dans la librairie existante de cellules *core*, mais nous n'aurions pas le bonheur de fabriquer notre première structure Reaktor Core. Empruntons donc un chemin plus ardu.

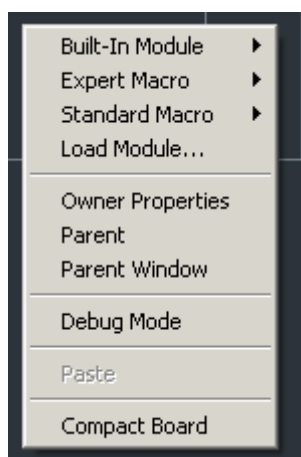
Commençons par créer une nouvelle cellule *core* audio en sélectionnant *Core Cell > New Audio*. Une cellule core audio vide apparaît:



Double-cliquez dessus pour faire apparaître sa structure Reaktor Core interne, qui est... vide. Vous vous en rappelez sûrement, les trois zones sont (de gauche à droite) les entrées, les modules normaux et les sorties:



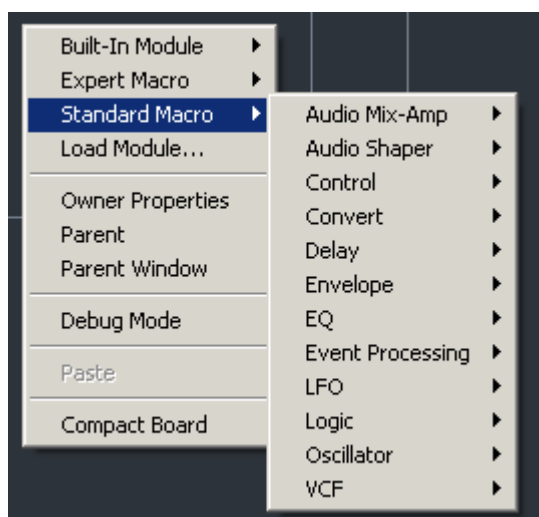
Attention, nous allons maintenant insérer notre premier module dans une structure core! Effectuez un clic droit dans la zone "normale" (au centre donc) pour appeler le menu de création de modules:



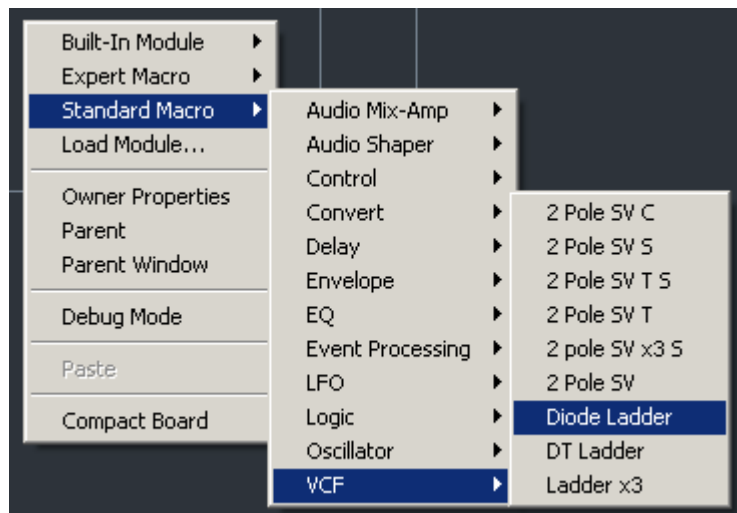
Le premier sous-menu est nommé *Built In Module* et permet d'accéder aux modules d'usine inclus dans Reaktor Core, qui sont généralement conçus pour effectuer des tâches de très bas niveau, et dont nous discuterons plus loin.

Le second sous-menu, appelé *Expert Macro*, contient les macros prévues pour être utilisées avec les modules de base pour les opérations de bas niveau. Donc oublions-les aussi pour l'instant.

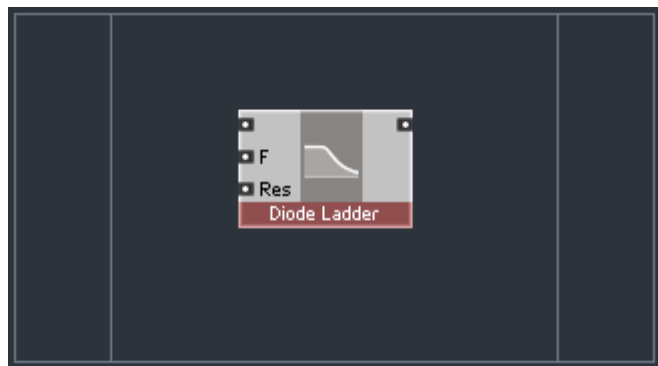
Le troisième sous-menu s'appelle *Standard Macro* et semble être celui qu'il nous faut:



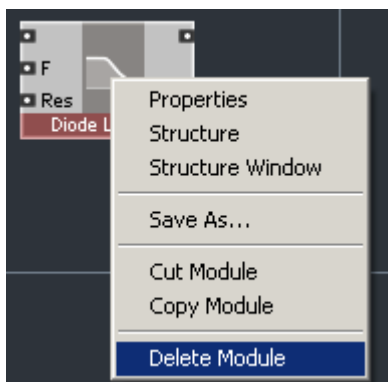
La section VCF pourrait bien être la bonne, jetons-y un œil:



Peut-être pourrions-nous utiliser le module *Diode Ladder* (Échelle de diodes) ?
Allons-y:

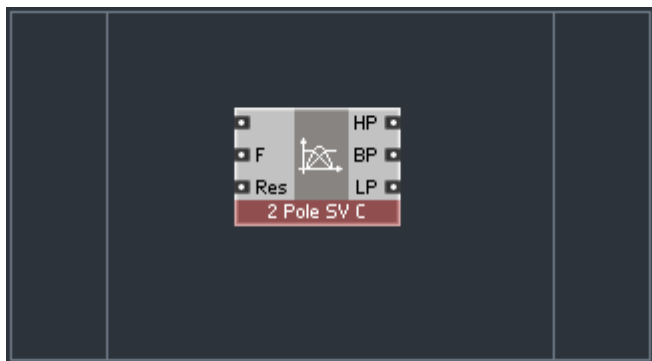


En fait, ce n'est peut-être pas la meilleure idée, car le son du *Diode Ladder* doit être vraiment différent de celui du module filtre de niveau primaire que nous voulons remplacer. L'échelle de diodes est un filtre avec au minimum 4 pôles (24 dB/octave) et celui que nous voulons remplacer a seulement 2 pôles (12 dB/octave). Supprimons donc ce *Diode Ladder*. Pour ce faire, vous avez deux options. La première est de faire un clic droit sur le module puis de choisir *Delete Module*:



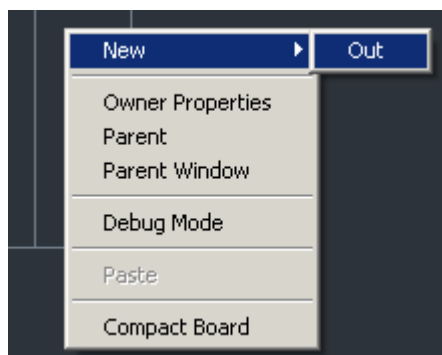
La seconde est de le sélectionner en cliquant dessus, puis d'appuyer simplement sur la touche *Suppr.*

Après avoir supprimé le *Diode Ladder*, insérons à la place un filtre *2 Pole SV C* depuis le même menu:



Celui-ci est bien un filtre à 2 pôles à variables d'état, semblable à celui que nous voulons remplacer (avec quelques différences tout de même, mais assez subtiles). L'important est que nous puissions moduler celui-ci avec de l'audio. Nous aurons bien sûr aussi besoin d'entrées et de sorties pour notre cellule *core*.

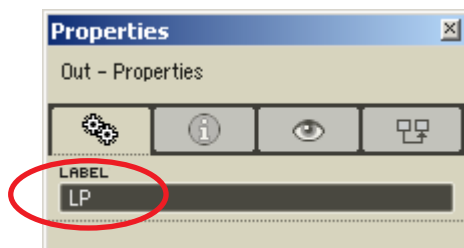
Pour être exact, nous aurons probablement besoin d'une seule sortie: le signal *LP* (passe-bas). Pour créer cette sortie, effectuez un clic droit dans la zone de sortie (à droite):



Dans cette zone, vous ne pouvez créer qu'un type de module, donc sélectionnez-le. La structure ressemble maintenant à ceci:



Double-cliquez sur le module de sortie pour ouvrir la fenêtre de propriétés (si elle n'est pas déjà ouverte). Saisissez "LP" dans le champ de nom (*Label* sur l'écran):



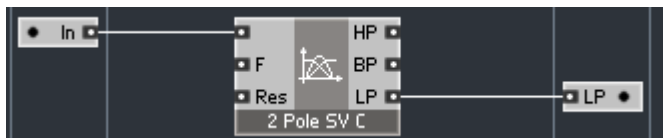
Connectez maintenant la sortie *LP* du filtre au module de sortie:




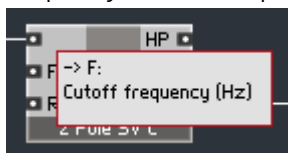
Passons aux entrées. La première entrée sera le signal audio d'entrée. Faites un clic droit sur le fond de l'écran dans la zone d'entrée et sélectionnez *New > In*:



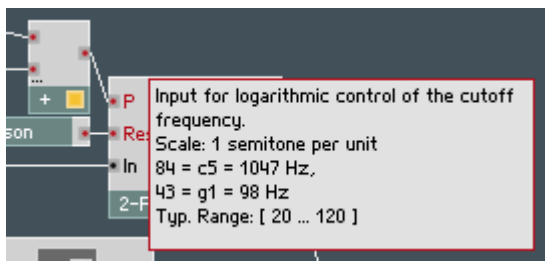
L'entrée est automatiquement créée avec le bon mode –comme vous l'indique le gros point noir, il s'agit d'une entrée audio. Vous pouvez la renommer "In" de la même façon que pour le module "LP", puis la connecter à la première entrée du module filtre:



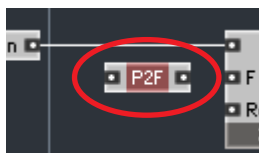
Pour la deuxième entrée du module filtre Reaktor Core, les choses se corsent. Comme vous pouvez le voir, cette entrée est nommée "F", pour "fréquence". Si vous maintenez le bouton de la souris enfoncé sur cette entrée (assurez-vous que le bouton  est activé), vous voyez apparaître le texte d'info "Cutoff frequency (Hz)" ("Fréquence de coupure (Hz)"):



Nous savons que la fréquence de coupure de notre module filtre du niveau primaire est contrôlée par une entrée appelée "P" et utilise une échelle de demi-tons, comme vous pouvez l'observer dans le texte d'info de l'entrée.



Visiblement, nous avons besoin de convertir les demi-tons en Hertz. Nous pouvons le faire au niveau primaire (avec le module *Expon. (F)*) ou bien à l'intérieur de notre structure Reaktor Core. Comme nous sommes en train d'apprendre à construire une structure Reaktor Core, choisissons la deuxième option. Effectuez un clic droit sur le fond de la zone normale (au cent...) et sélectionnez *Standard Macro > Convert > P2F*:



Comme son nom l'indique (et comme le texte d'info le confirme), ce module convertit une échelle "P" en échelle "F" –exactement ce dont nous avons besoin! Créons donc une deuxième entrée nommée "P" et connectons-la via le module *P2F*:



Tout cela devrait être suffisant. Mais, dans notre instrument, il reste le potentiomètre "P Cutoff", définissant la fréquence de coupure de base du filtre qui est ajoutée au signal de modulation venant de l'enveloppe. Ce signal est converti en signal événement au niveau primaire pour pouvoir alimenter l'entrée "P" du filtre. Cette conversion n'est évidemment plus nécessaire, nous pouvons retirer le module *A/E* et brancher directement le signal audio dans l'entrée audio "P" de notre nouveau filtre. Cette méthode est loin de nous déplaire, mais nous vous en suggérons une autre... pour la beauté du geste.

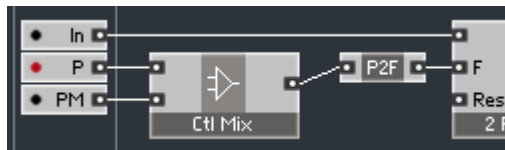
Nous allons partir d'une entrée "P" en mode événement et une entrée de modulation en mode audio. Si vous vous rappelez de notre discussion à propos des "enveloppes lentes", vous comprendrez pourquoi nous suggérons d'appeler cet autre module d'entrée "PM", et non "FM", et d'utiliser une modulation en demi-tons (le "P" vient de "Pitch", "hauteur tonale" en anglais). C'est exactement ce qui se passe dans l'instrument original: nous additionnions le signal d'enveloppe avec le signal "P Cutoff" et branchions la somme dans l'entrée "P".

Passons donc notre entrée "P" en mode événement (nous avons déjà décrit plus haut comment faire) et ajoutons une entrée "PM" en mode audio:

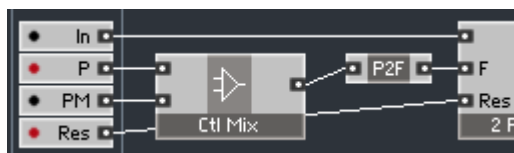


Comme un utilisateur du niveau primaire de Reaktor pourrait s'y attendre, nous devrions pouvoir additionner les deux signaux tout de suite. Mais dans

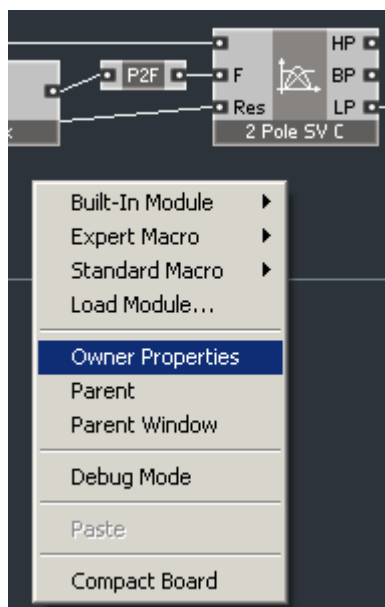
Reaktor Core, le module *Add* est considéré comme un module de bas niveau et suppose quelque connaissance des principes fondamentaux de fonctionnement du bas niveau de Reaktor Core. Ils ne sont pas si compliqués que cela, et nous les décrirons. Mais à ce stade, ils ne feraient que vous encombrer l'esprit. Utilisez plutôt un mixeur de signaux de contrôle, par exemple *Standard Macro > Control > Ctl Mix*:



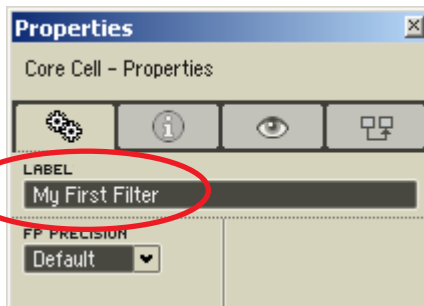
La dernière entrée dont nous avons besoin est la résonance. Sa forme audio ne nous est d'aucune utilité, utilisons donc un évènement:



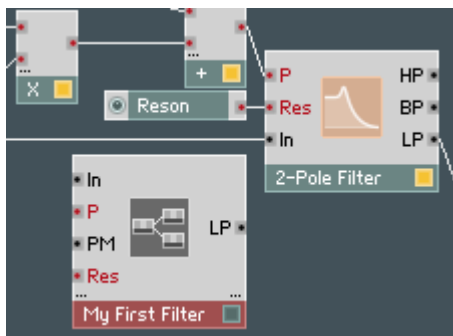
Il nous reste à donner un nom à notre cellule core. Pour accéder à ses propriétés, cliquez sur le fond si la fenêtre Propriétés est déjà ouverte, sinon faites un clic droit sur le fond et choisissez la commande Owner Properties:



Vous pouvez alors saisir un texte dans le champ de nom (*Label*):

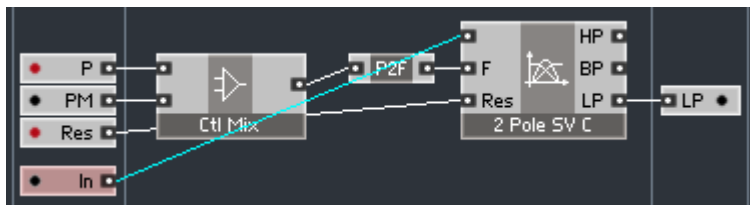


Double-cliquez sur le fond pour voir le résultat:



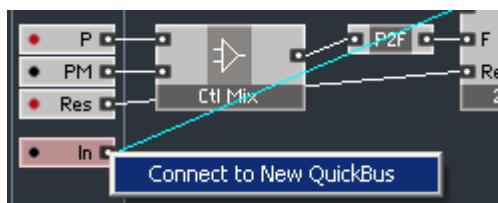
Le module a fière allure! On pourrait objecter que l'entrée du signal audio est en haut de la cellule *core*, alors qu'elle était en bas du module filtre du niveau primaire. Honnêtement, ce n'est pas un gros problème; si vous tenez absolument à arranger cela, ce n'est pas très compliqué, vous savez déjà comment faire. Mais faisons-le ensemble, nous en profiterons pour vous montrer en chemin une nouvelle fonction.

Retournons donc à l'intérieur de la structure *core* et commençons par déplacer l'entrée du signal audio tout en bas:

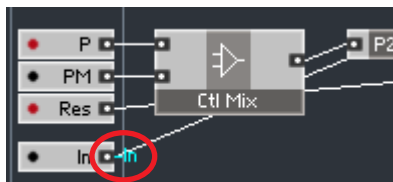


Cela suffit, si ce n'est que maintenant un câble diagonal traverse toute la structure, ce qui n'est pas très seyant. Voilà comment nous allons résoudre ce problème.

Effectuez un clic droit sur la sortie du module d'entrée *In* et sélectionnez la commande *Connect to New QuickBus*:



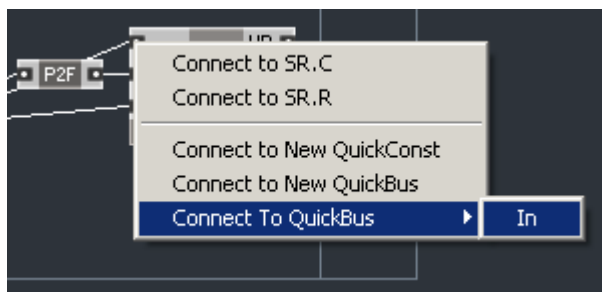
Vous devriez maintenant voir ceci:



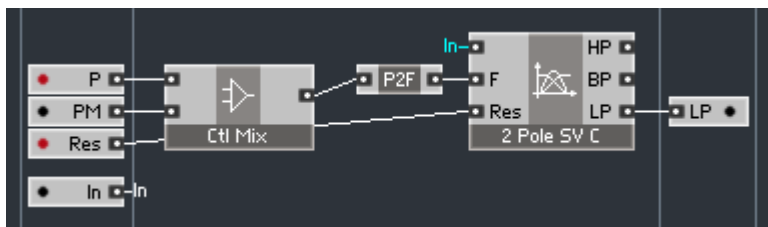
La fenêtre *Properties* devrait également afficher les propriétés du QuickBus que vous venez de créer. La propriété la plus utile du QuickBus est bien sûr de pouvoir le renommer (les autres propriétés sont assez avancées, laissons-les tranquilles pour l'instant). Vous pourrez rouvrir la fenêtre *Properties* plus tard en double-cliquant sur le QuickBus.

Bien que vous puissiez renommer ce QuickBus, nous pensons que son nom actuel est parfaitement approprié, puisqu'il correspond au nom de l'entrée qui lui est connectée. Les QuickBus sont internes à la structure en question, il n'y a donc pas de risque de collisions de noms si des QuickBus des structures voisines ou parentes portent le même nom.

Étape suivante: effectuer un clic droit sur l'entrée du haut du module filtre 2 *Pole SV C* et de sélectionner *Connect to QuickBus > In*:

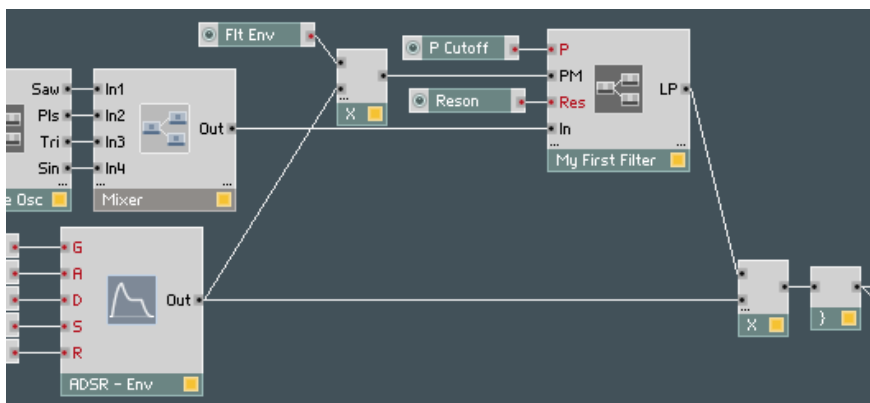


Dans le menu ci-dessus, *In* n'est rien d'autre que le nom du QuickBus auquel vous vous connectez. Ici, vous ne souhaitez pas créer de nouveau QuickBus, mais connecter l'entrée en question au QuickBus déjà existant. Voilà à quoi votre structure doit ressembler :



À la place du câble diagonal hideux, nous avons maintenant deux jolies références indiquant qu'elles sont connectées via un QuickBus nommé *In*.

Revenons au niveau primaire et modifions notre structure pour utiliser le filtre que nous venons de construire. Vous pouvez vous débarrasser des modules *Add* et *A/E*. Et voilà le résultat :



Notre nouvelle structure utilise un peu plus de puissance processeur, n'est-ce pas ? N'oubliez pas que ce filtre est modulé par un signal audio en échelle de demi-tons. S'il ne vous plaît pas, vous pouvez revenir à l'ancienne structure ou utiliser le module filtre *Multi 2 pole FM* du niveau primaire (les "enveloppes lentes", vous vous rappelez ?), mais nous espérons qu'il vous plaît. Dans le cas contraire, il y a plein d'autres filtres avec de nouvelles fonctions qui vous plairont peut-être plus. Et si vous n'aimez pas les nouveaux filtres Reaktor Core, il y a une foule d'autres modules Reaktor Core que vous pouvez essayer.

Les signaux audio et les signaux de contrôle

Avant de poursuivre, nous devons regarder d'un peu plus près une convention particulière utilisée dans les *Standard Macros* de la librairie Reaktor Core. Les modules qui s'y trouvent peuvent être catégorisés selon le type de signaux qu'ils traitent: audio, de contrôle, évènement et logique. Nous expliquerons les signaux évènements et logiques un peu plus tard, concentrons nous d'abord sur les deux premiers types de signaux.

Les signaux audio sont les signaux transportant de l'information audio (sic). Ils comprennent les signaux sortant des oscillateurs, filtres, amplificateurs, delays, etc. Ainsi, les modules comme les filtres, amplificateurs, saturations et autres delays doivent normalement recevoir en entrée un signal audio à traiter.

Les signaux de contrôle ne transportent pas d'audio, ils servent juste à contrôler certains modules. Par exemple, les sorties des enveloppes, des LFOs, les signaux de pitch (hauteur tonale) et de vélocité des notes du clavier ne transportent aucun son mais peuvent servir à contrôler la fréquence de coupure ou la résonance d'un filtre, la durée d'un délai ou quoi que ce soit d'autre. Ainsi, les ports d'entrée de fréquence de coupure ou de résonance d'un filtre ou de durée de délai sont supposés recevoir des signaux de contrôle.

Voilà un exemple de module filtre Reaktor Core que vous connaissez déjà:



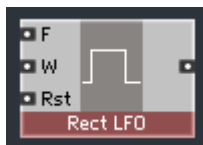
L'entrée supérieure du filtre est prévue pour recevoir le signal audio à filtrer et attend donc un signal de type audio. Les entrées *F* et *Res* sont visiblement de type contrôle. Les sorties du filtre produisent différentes sortes de signaux audio filtrés, elles sont donc toutes de type audio.

Un module oscillateur sinusoïdal, quant à lui, dispose d'une unique entrée de contrôle (pour la fréquence) et d'une unique sortie audio:



Si nous observons le module *Rect LFO*, il dispose de deux entrées de contrôle (pour la fréquence et la largeur de l'impulsion, la troisième entrée étant de type

évènement) ainsi que d'une sortie de contrôle (car le module sert à contrôler d'autres paramètres comme la fréquence de coupure d'un filtre, des niveaux d'amplification, etc.):



Certains types de traitements (et mixages) sont utilisés à la fois pour les signaux audio et pour les signaux de contrôle. Dans ce cas, vous trouverez pour ces macros une version dédiée aux signaux audio et une version dédiée aux signaux de contrôle. C'est le cas par exemple des mixeurs audio ou de contrôle, des amplificateurs audio ou de contrôle, etc. Généralement, il est déconseillé d'utiliser un module avec un signal pour lequel il n'est pas prévu, à moins que vous ne sachiez précisément ce que vous faites.

Ceci dit, il faut remarquer qu'il est assez souvent possible de réutiliser des signaux audio comme signaux de contrôle. L'exemple le plus typique est la modulation de la fréquence d'un oscillateur ou de la fréquence de coupure d'un filtre par un signal audio. Ceci fonctionne sans problème, à partir du moment où vous utilisez intentionnellement ce signal audio comme un signal de contrôle. Nous osons croire que le cas contraire, ie l'utilisation volontaire d'un signal de contrôle en tant que signal audio, est plutôt rare...

Cette séparation entre les signaux audio, de contrôle, évènement et logique ne doit pas être confondue avec la séparation audio/évènement du niveau primaire de Reaktor. La classification audio/évènement du niveau primaire spécifie une sorte de "vitesse de traitement" des signaux: les signaux audio sont traités bien plus "vite" (et nécessitent plus de puissance processeur). De plus, comme vous le savez probablement, les signaux évènements du niveau primaire ont des règles de propagation différentes de celles des signaux audio. Dans la terminologie de Reaktor Core, la différence entre les signaux audio, de contrôle, évènements et logiques est purement sémantique: elle définit l'utilisation du signal et non le type de traitement qu'il subit. La relation entre les catégories évènement/audio du niveau primaire et les catégories audio/contrôle/évènement/logique de Reaktor Core n'est pas biunivoque ("bijective" diraient les purs et durs), mais nous pouvons quand même tenter de la détailler:

- un signal audio du niveau primaire correspond normalement dans Reaktor Core soit à un signal audio (p.ex. la sortie d'un oscillateur ou d'un filtre), soit à un signal de contrôle (p.ex. la sortie d'une enveloppe).
- un signal événement du niveau primaire est généralement un signal de contrôle dans Reaktor Core. Comme exemple, on peut citer la sortie d'un LFO, d'un potentiomètre, d'une source MIDI de pitch ou de vélocité.
- parfois, un signal événement du niveau primaire correspond à un signal événement dans Reaktor Core. L'exemple le plus typique est un MIDI Gate (les signaux événements de Reaktor Core seront décrits plus loin, comme promis).
- parfois, un signal événement du niveau primaire *ressemble* à un signal logique de Reaktor Core, bien qu'ils ne soient pas entièrement compatibles et qu'une conversion explicite entre les deux soit nécessaire (nous traiterons aussi de ce sujet plus loin). On peut prendre comme exemple les signaux traités par *Logic AND* ou d'autres modules similaires du niveau primaire.

Il est important de comprendre que, lorsque vous sélectionnez un type de signal pour un port d'entrée d'une cellule *core*, vous choisissez en fait entre un signal événement et un signal audio du niveau primaire et non du niveau Reaktor *Core*! Les ports de la cellule *core* sont à la jonction des deux mondes, c'est la raison pour laquelle ils utilisent dans certains cas la terminologie du niveau primaire.

Nous allons en apprendre un peu plus sur ces distinctions en essayant de construire une émulation d'effet d'écho à bande. Nous commencerons par construire un simple écho numérique, puis nous l'améliorerons pour imiter certaines caractéristiques d'un écho à bande.

Créons d'abord une cellule *core* vide. Pénétrez-y et renommez-la "Echo".

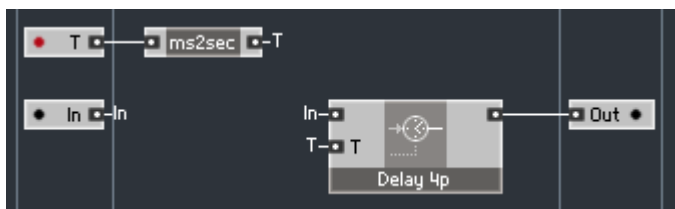
Le premier module que nous allons placer dans la structure est un module de délai. Prenons un délai à interpolation 4-points, car il est de meilleure qualité qu'un 2-points, et car un délai sans interpolation ne conviendrait pas à notre émulation de bande. Sélectionnons donc *Standard Macro > Delay > Delay 4p*:



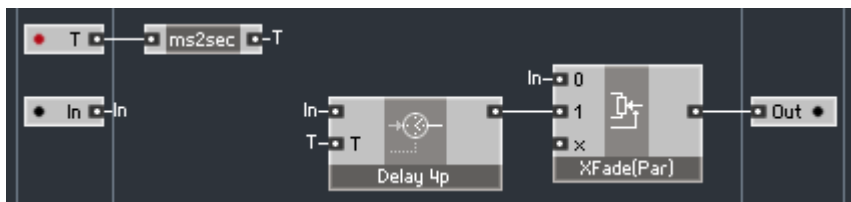
Nous avons évidemment besoin d'une entrée et d'une sortie audio pour notre cellule *core* de délai. Nous allons utiliser une connexion QuickBus pour l'entrée et une connexion normale pour la sortie:



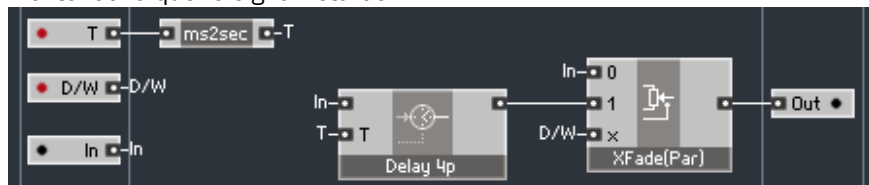
Il nous faut également une entrée événement pour contrôler la durée du délai. Il faut ici prendre en compte le fait que la durée de délai au niveau primaire est généralement exprimée en millisecondes, alors qu'au niveau Reaktor Core elle est exprimée en secondes. Pas de problème, un petit convertisseur est disponible, *Standard Macro > Convert > ms2sec*:



À ce point, nous n'avons qu'un écho simple, et il serait agréable de pouvoir entendre le signal original et pas uniquement le signal avec écho. Nous devons donc mixer le signal original avec le signal retardé. *Comme nous voulons mixer des signaux audio, nous avons besoin d'un mixeur audio* (si vous vous en rappelez, nous avons utilisé un mixeur de contrôle pour mixer des signaux de contrôle lorsque nous avons construit la cellule *core* de filtre). Encore mieux, nous pouvons utiliser un mixeur audio spécialement conçu pour mélanger deux signaux (bref un "crossfader"): *Standard Macro > Audio Mix-Amp > XFade(par)*:



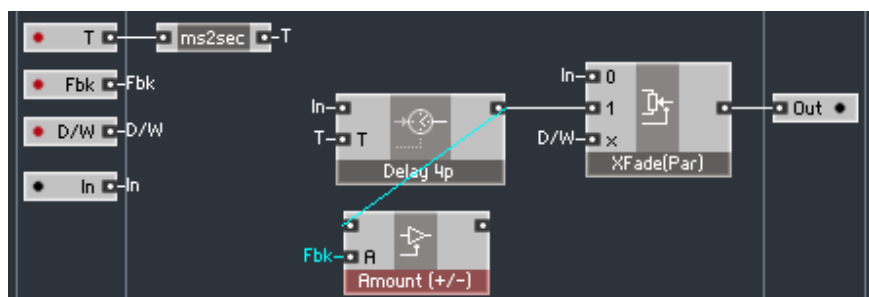
Le qualificatif “(par)” signifie “parabolique” – pour donner une transition plus naturelle que la transition linéaire. Nous allons connecter l’entrée de contrôle (“x”) du crossfader à une nouvelle entrée événement qui contrôle la balance entre les signaux original et retardé. Lorsque le signal de contrôle est égal à 0, nous n’entendons que le signal original, tandis que s’il est égal à 1 nous n’entendons que le signal retardé:



C’est bien mieux maintenant, puisque nous pouvons entendre le signal original et son écho. Mais ce n’est toujours qu’un écho. Pour avoir plusieurs échos, nous devons réinjecter une partie du signal retardé dans l’entrée du délai. Nous devons donc d’abord atténuer le signal retardé. D’après les mêmes considérations – un amplificateur audio pour un signal audio, prenons *Standard Macro* > *Audio Mix Amp* > *Amount*:

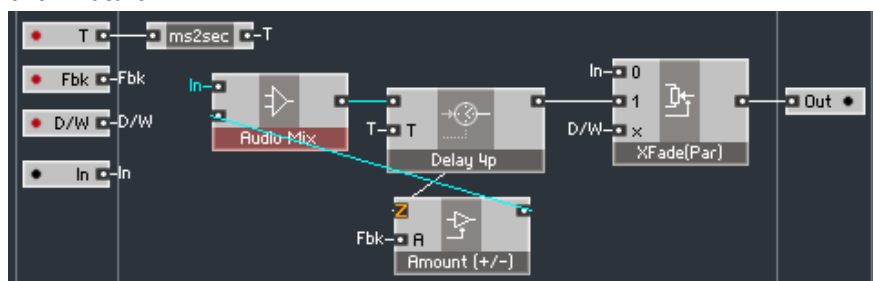


Nous utilisons l’ampli *Amount* car nous voulons contrôler la quantité de signal réinjectée. Cet amplificateur nous permet aussi d’inverser le signal en utilisant des réglages négatifs. En revanche, un ampli comme *Amp (dB)*, qui conviendrait mieux pour contrôler un volume sonore, ne serait pas très intéressant dans notre cas, car il ne permet pas d’inverser le signal. Connectons l’entrée de contrôle d’amplitude de l’amplificateur à une entrée événement que nous créons pour contrôler la quantité de réinjection:

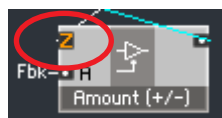


Les quantités raisonnables de réinjection se situent dans l'intervalle [-0.9..0.9]. Si vous souhaitez tester ce délai dès l'étape suivante, faites attention à la quantité de réinjection, car vous pouvez facilement atteindre des niveaux trop élevés (il n'y a pas encore de saturation dans notre circuit). Nous aurions pu équiper notre cellule core d'un limiteur de réinjection de sécurité, mais comme nous voulons ensuite y adjoindre une saturation, nous n'avons pas jugé cela nécessaire. En revanche, vous pourriez faire des essais avec des réinjections hautes et entendre la saturation du délai.

Nous devons maintenant mixer le signal de réinjection avec le signal original. Un mixeur audio (*Standard Macro > Audio Mix Amp > Audio Mix*) semble un choix naturel:



Vous vous demandez peut-être ce qui est arrivé à l'entrée supérieure du module *Amount* ci-dessus, celle qui affiche maintenant un gros "Z" orange:



En fait, selon la version du logiciel et les circonstances, ce "Z" peut apparaître à d'autres entrées de la structure, mais cela ne doit pas vous inquiéter plus que cela. Ce symbole indique qu'un délai numérique est apparu dans la structure à cet endroit. Il est utile à la conception de structures évoluées, pour lesquelles cette information peut être importante pour le concepteur.

Pour les structures simples comme celle qui nous occupe, nous pouvons oublier ce symbole. Sa présence montre juste qu'il existe un délai d'1 échantillon (environ 0,02 ms à 44,1 kHz, encore moins aux taux d'échantillonnage plus élevés) en ce point. Nous osons prétendre que vous ne remarquerez rien si le délai est décalé de 0,02 ms par rapport à la valeur que vous avez spécifiée...

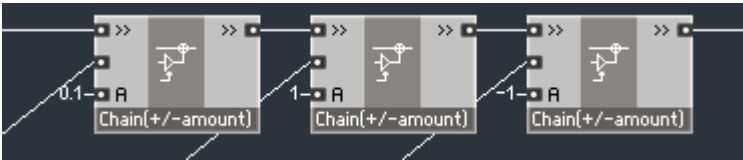
Revenons à notre structure. Elle peut maintenant produire une série d'échos

décroissants. Elle devrait être suffisante pour un écho numérique, mais nous voulons vous montrer une autre caractéristique de la librairie, une astuce qui vous permettra de réduire la taille de votre structure.

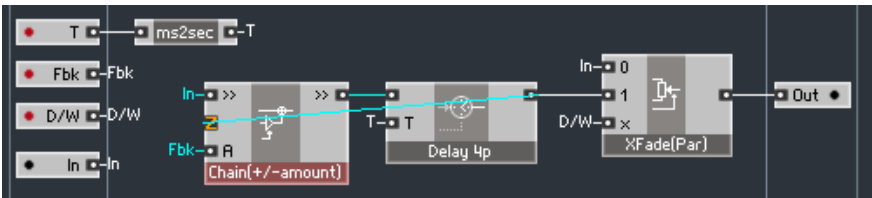
Parmi les amplificateurs audio, certains sont appelés “Chain”. Ces amplificateurs sont capables d’amplifier un signal donné et de le mélanger à un autre signal “enchaîné”. L’un de ces amplis, *Audio Mix Amp > Chain(amount)*, fonctionne comme *Amount* si ce n’est cette caractéristique “d’enchaînement”:



Le signal arrivant à la deuxième entrée de ce module est atténué selon le facteur donné par l’entrée “A” puis mélangé au signal arrivant dans l’entrée de chaîne (“>>”). Le signal arrivant dans cette entrée de chaîne n’est pas atténué. Ce type d’amplificateur peut servir à construire des chaînes de mixage, dans lesquelles les connexions des ports “>>” constituent une sorte de bus de mixage:



Dans le cas qui nous concerne, nous n’avons pas besoin de bus de mixage, mais nous pouvons utiliser ce module à la place de l’ensemble *Audio Mix* et *Amount*. Le signal réinjecté sera atténué de la quantité spécifiée par l’entrée *Fbk* et mixé avec le signal d’entrée exactement comme auparavant:

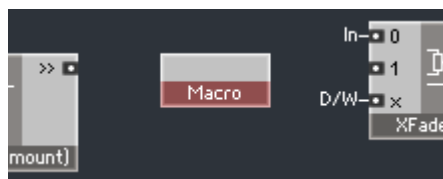


Félicitations! Vous avez construit un effet d’écho numérique simple. La prochaine étape sera d’y ajouter un effet de simulation de bande.

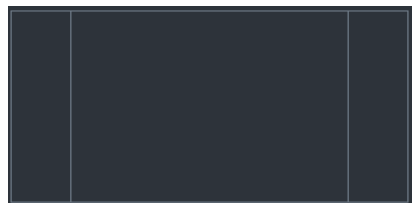
Création de vos premières macros Reaktor Core

Pour l'effet d'écho que nous venons de construire, nous avons utilisé une macro *Delay 4p* de la librairie qui nous offre un délai numérique de bonne qualité. Mais bonne qualité ou pas, il sonne toujours trop “numérique”. Nous pourrions lui donner un son plus chaud en lui ajoutant quelques caractéristiques typiques d'un écho à bande, comme la saturation et l'effet de pompage. C'est ce que nous allons réaliser maintenant.

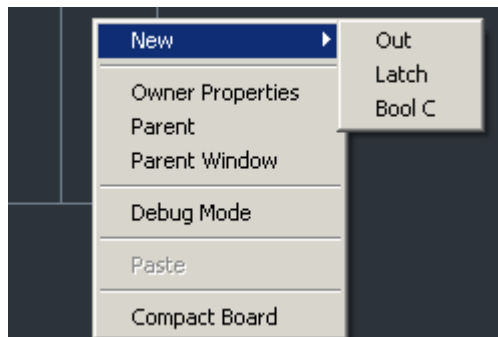
Commençons par supprimer la macro de délai de la structure, et créons une macro vide à la place via un clic droit sur le fond de l'écran et en sélectionnant *Built In Module > Macro*:



Double-cliquez sur cette macro pour pénétrer à l'intérieur. Vous voyez apparaître une structure vide similaire à celle de la cellule *core* précédente:

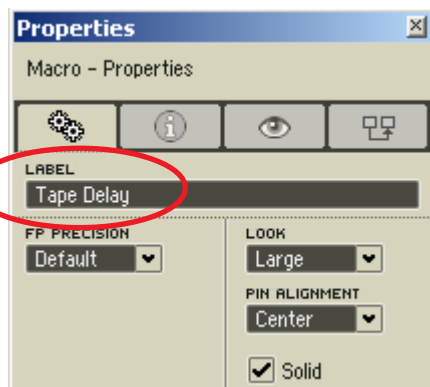


Elle fonctionne aussi de façon semblable, avec toutefois quelques différences importantes. Nous avons alors la structure d'une cellule Reaktor Core alors que nous avons maintenant la structure interne d'une macro Reaktor Core. En réalité, ces différences concernent l'ensemble des modules d'entrée et de sortie disponibles:



Les ports de type *Latch* et *Bool C* seront expliqués bien plus loin dans ce manuel; ils servent pour les opérations avancées. Pour l'instant, nous nous intéresserons uniquement à la sortie intitulée "Out" (ou "In" pour les entrées). C'est un type général de port, qui peut accepter des signaux audio, logiques, de contrôle et événements. En fait, le port n'a cure du type de signal qu'il reçoit: cette différenciation n'est utile que pour vous, en tant qu'utilisateur, parce qu'elle définit ce à quoi doit servir le signal. Pour Reaktor Core, tous les signaux sont identiques. La différence entre entrées/sorties audio et événement de la structure précédente n'existe plus, car il n'y a plus de niveau primaire à l'extérieur (contrairement à ce qui se passe pour une cellule *core*): nous sommes maintenant entièrement dans Reaktor Core.

La première chose que nous allons faire est de nommer la macro, de la même façon que pour la cellule *core*, via un clic droit sur le fond et en sélectionnant *Owner Properties*, où vous pouvez saisir le nom:



Les autres propriétés de la macro contrôlent divers aspects de l'apparence et du traitement du signal pour cette macro.

Même si vous êtes libre d'expérimenter différents réglages pour ces autres propriétés, nous vous conseillons fermement de laisser le paramètre *Solid* activé et de modifier le paramètre *FP Precision* avec modération. La signification de ces paramètres sera expliquée dans les sections avancées de ce manuel.

Créons ensuite un ensemble d'entrées/sorties pour notre macro *Tape Delay*:

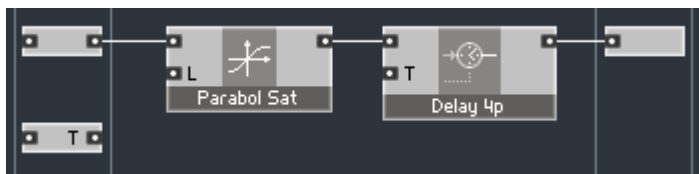


L'entrée supérieure recevra le signal audio, l'entrée inférieure le paramètre temporel. Vous avez peut-être remarqué la présence de ports supplémentaires sur la gauche des modules d'entrée, nous les expliquerons plus loin.

Au centre de notre macro, nous allons utiliser comme point de départ le même module *Delay 4p*:



Une simulation simple de l'effet de saturation peut être réalisée facilement, il suffit de connecter un module saturateur avant le délai. Le saturateur est une sorte de modelleur de signal et il travaille sur de *l'audio*, nous allons donc le chercher parmi les modelleurs audio (en anglais "audio shapers"). Sélectionnons *Standard Macro > Audio Shaper > Parabol Sat*:




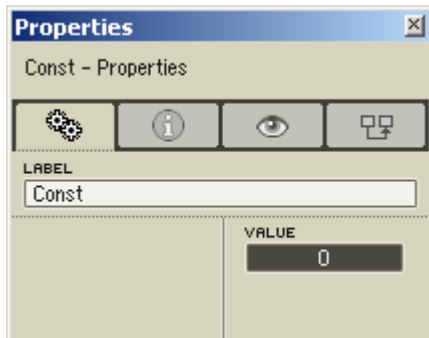
Le signal d'entrée est maintenant saturé sur un intervalle de -1 à $+1$. En fait, cet intervalle est contrôlé par l'entrée "L" du module saturateur; si celle-ci est déconnectée, la valeur est égale à 1 par défaut. Ceci peut vous paraître surprenant, puisque d'habitude une entrée déconnectée est considérée comme ne recevant aucun signal, autrement dit un signal nul. Ce n'est pas exactement le cas dans les structures Reaktor Core, les modules ont la possibilité de spécifier un traitement spécial pour leurs entrées déconnectées, comme pour ce saturateur dans lequel l'entrée "L" doit avoir 1 comme valeur par défaut.

Nous allons maintenant apprendre comment effectuer ce réglage, pour la valeur par défaut de l'entrée "T". Mettons que, si notre entrée "T" est déconnectée, nous voulions qu'elle soit traitée comme si la valeur d'entrée était de 0,25

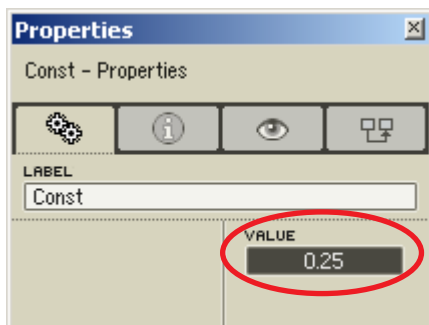
secondes. Rien de plus simple: faites un clic droit sur ce nouveau port sur la gauche du module d'entrée "T", et sélectionnez *Connect to New QuickConst*. Voilà ce que vous devriez voir:



De plus, la fenêtre de propriétés devrait afficher les propriétés de cette constante (si la fenêtre affiche une page différente, appuyez sur le bouton ):



Dans le champ de valeur ("Value"), saisissez la nouvelle valeur "0.25":



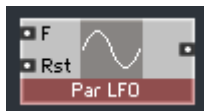
Voilà ce que vous devriez voir maintenant:



Expliquons ce que nous venons de faire. Le port sur la gauche des modules d'entrée spécifie un signal "par défaut", c'est-à-dire un signal utilisé si l'entrée n'est pas connectée (à l'extérieur de la macro). Dans notre cas, si l'entrée "T" de la macro *Tape Delay* n'est pas connectée à l'extérieur, elle se comportera comme si vous lui aviez connecté une constante de valeur 0,25.

Une connexion à une *QuickConst* n'est évidemment pas la seule possibilité pour un signal par défaut. Vous pouvez lui connecter n'importe quel autre module de la structure, y compris un autre module d'entrée.

Maintenant que nous avons une saturation et une valeur par défaut pour l'entrée "T", simulons l'effet de pompage de la bande. Une façon simple de le faire est de moduler la durée de délai par un LFO (pour "Low Frequency Oscillator", "Oscillateur Basse Fréquence" en anglais). Nous pourrions essayer différentes formes de LFOs pour obtenir un meilleur effet de pompage. Nous proposons pour l'instant d'en prendre un seul dans la librairie: *Standard Macro* > *LFO* > *Par LFO*:

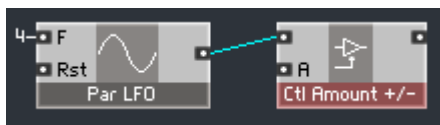


C'est un LFO parabolique produisant un signal qui ressemble par sa forme à une sinusoïde, mais qui nécessite moins de puissance processeur. Son entrée "F" doit recevoir le signal spécifiant sa vitesse d'oscillation (sa fréquence). Nous pouvons à nouveau utiliser une constante *QuickConst*. Une fréquence de 4 Hz semble raisonnable:

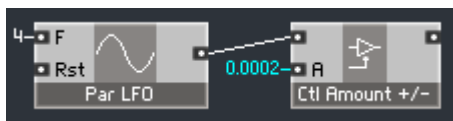


L'entrée "Rst" sert à redémarrer le LFO ("Rst" pour "Reset", "Redémarrer" en anglais), nous ne nous en servons pas pour l'instant.

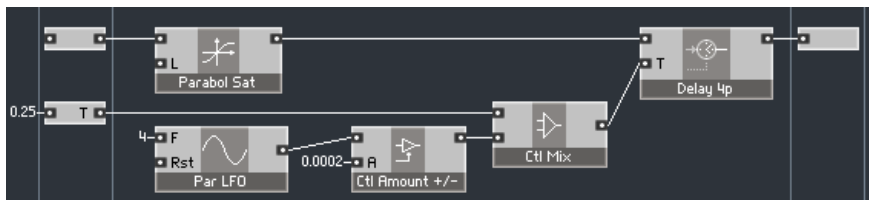
Nous devons maintenant spécifier une quantité de modulation en redimensionnant la sortie du LFO, car il génère pour l'instant un signal compris entre -1 et 1, ce qui est beaucoup trop. Ne perdant pas de vue que nous avons ici affaire à des signaux de contrôle, nous allons utiliser un module d'amplification de contrôle, similaire à l'amplificateur *Amount* que nous avons utilisé pour l'audio, *Standard Macro* > *Control* > *Ctl Amount +/-*:



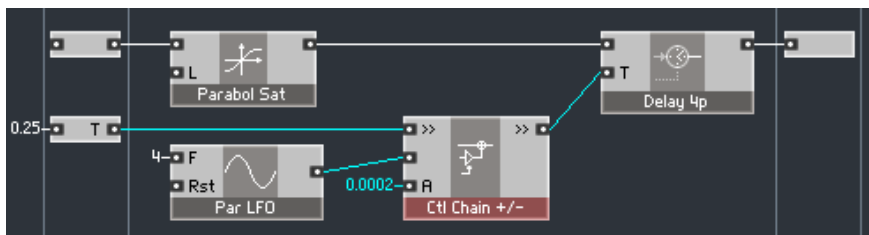
Une amplitude de modulation de 0,0002 devrait faire l'affaire, réduisons donc le signal de ce facteur:




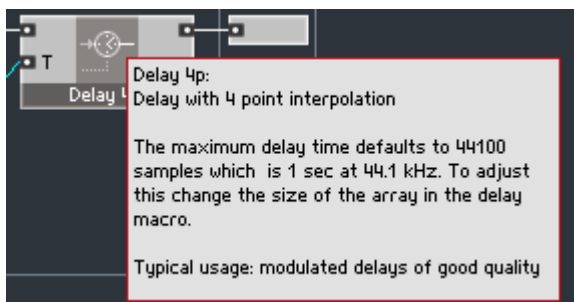
Enfin, nous pouvons mixer les deux signaux de contrôles (celui de l'entrée "T" et celui du module *Ctl Amount*) et les envoyer dans l'entrée "T" du module de délai. Le module *Ctl Mix* déjà familier peut être à nouveau utilisé:



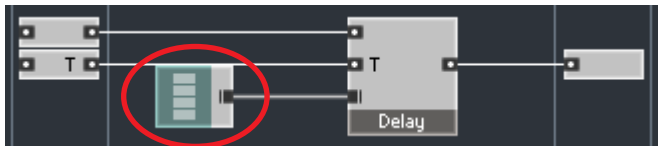
En fait, nous retrouvons la structure de mixeur "en chaîne" que nous avons pour les signaux audio. Nous pouvons donc remplacer les modules *Ctl Amount* et *Ctl Mix* comme nous l'avons fait pour le chemin du signal audio, avec cette fois-ci le module *Standard Macro > Control > Ctl Chain*:




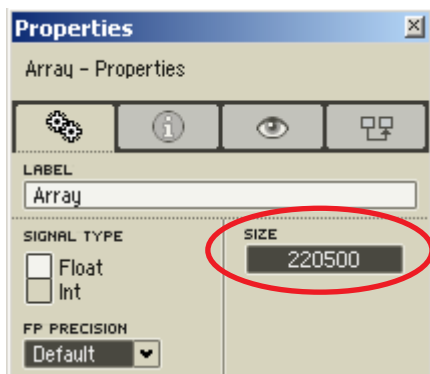
Nous pouvons apporter la touche finale à notre macro: changer la taille du tampon pour notre délai, qui définit la durée de délai maximale. Si vous maintenez le curseur sur la macro *Delay 4p* (et pour peu que le bouton  soit activé), vous pouvez lire dans le texte d'info que la taille de tampon (en anglais "buffer") par défaut correspond à 1 seconde à 44,1 kHz:



Passons-le à 5 secondes, soit $44100 \times 5 = 220500$ (au fait, puisque chaque échantillon occupe 4 octets, ceci fait en tout $220500 \times 4 = 882000$ octets, soit presque 1 Mo). Double-cliquez sur la macro *Delay 4p*:



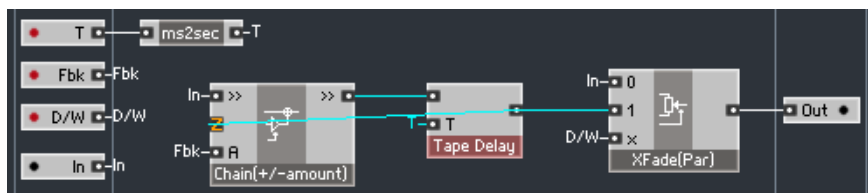
Le module sur la gauche est le module du tampon du délai. Double-cliquez dessus (ou faites un clic droit et sélectionnez *Show Properties*) pour modifier ses propriétés. Sélectionnez la page , dans laquelle vous devriez voir s'afficher la propriété *Size* (*Taille* en anglais...). Réglez-la sur 220500 échantillons:



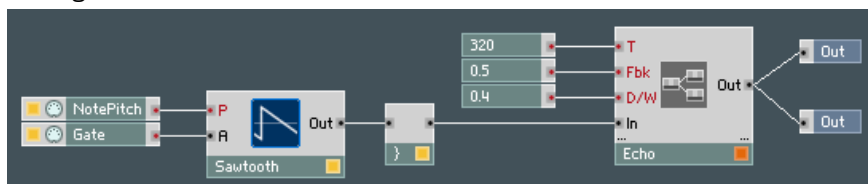
Comme nous l'avons vu, un tampon de 5 secondes d'audio prend environ 1 Mo de mémoire, donc soyez prudent lorsque vous modifiez les tampons de délai, tout particulièrement si les délais sont utilisés dans

des parties polyphoniques de la structure, dans lesquelles la taille du tampon sera multipliée par le nombre de voix !

Nous pouvons ressortir de la macro *Delay 4p* puis de la macro *Tape Delay* que nous venons de créer (double-cliquez sur le fond), pour nous occuper des connexions externes:



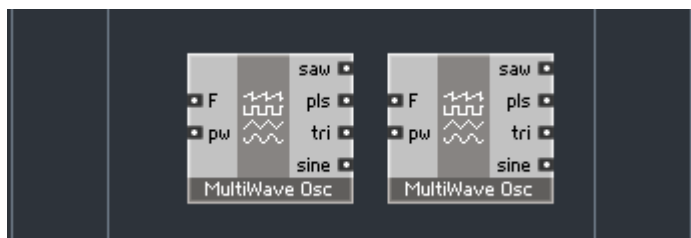
Si vous ne l'avez pas encore fait, nous vous suggérons d'essayer le module d'écho que nous avons construit. Voici une structure de test du niveau primaire de Reaktor, aussi simple que possible (veuillez noter que le module d'écho est réglé sur *mono*):



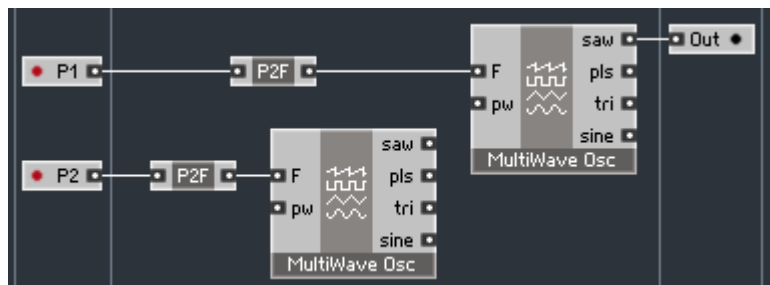
Vous pouvez l'améliorer de multiples façons, par exemple en insérant des potentiomètres contrôlant les différents paramètres de l'écho, en utilisant un véritable synthétiseur comme signal source, etc.

Utiliser l'audio comme signal de contrôle

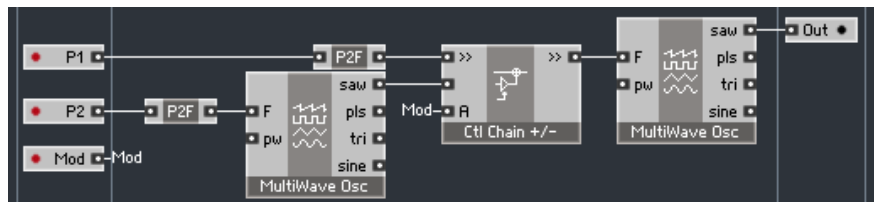
Nous avons mentionné plus haut qu'il était possible d'utiliser un signal audio en tant que signal de contrôle. Nous avons pensé qu'il serait utile d'en donner un exemple. Nous allons créer une cellule *core* implémentant une paire d'oscillateurs, l'un des deux étant modulé par l'autre. Prenons deux oscillateurs multi-ondes (en anglais "multiwave"):



Nous aurons besoin d'un contrôle de pitch (hauteur tonale) pour chacun des deux oscillateurs, et nous écouterons la sortie du deuxième oscillateur. Créons les entrées et sorties correspondantes:



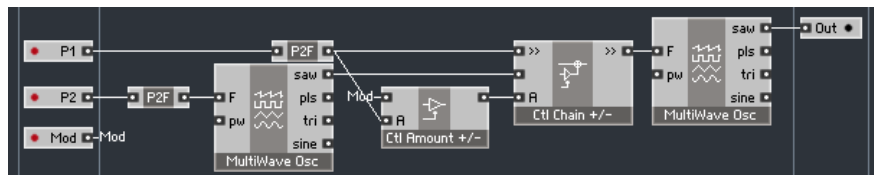
Nous voulons maintenant prendre la sortie de l'oscillateur de gauche et l'utiliser pour moduler la fréquence du second oscillateur (celui de droite, donc):



L'entrée *Mod* contrôle la quantité de modulation.

Veuillez noter que nous mixons le signal de modulation après le convertisseur *P2F*, afin que la modulation ait lieu sur l'échelle de fréquences (il serait aussi possible d'effectuer la modulation sur l'échelle de pitch).

En fait, il vaut mieux adapter la quantité de modulation en fonction de la fréquence de l'oscillation de base:



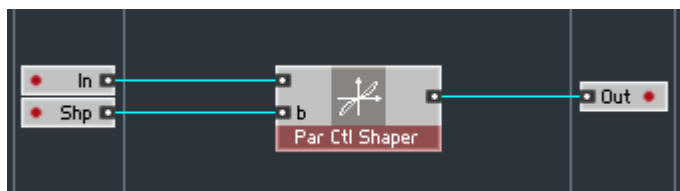
Maintenant, si vous analysez la structure ci-dessus du point de vue des signaux audio et de contrôle, vous remarquerez que tous les signaux de la structure, excepté les sorties des oscillateurs, sont des signaux de contrôle. Les sorties des deux oscillateurs sont évidemment des signaux audio. Cependant, nous utilisons de façon "détournée" la sortie de l'oscillateur de gauche, la considérant comme un signal de contrôle que nous envoyons dans le mixer *Ctl Chain*.

Les signaux évènements

Comme nous l'avons dit précédemment, il y a plusieurs significations à l'expression "signal évènement". Le concept des signaux évènements du niveau primaire de Reaktor devrait déjà vous être familier. Un signal évènement du niveau primaire peut avoir plusieurs utilisations. Une première consiste à l'utiliser comme signal de contrôle (p.ex. la sortie d'un LFO, d'un potentiomètre, etc.), simplement parce qu'il consomme moins de puissance processeur qu'un signal audio du niveau primaire. Dans ce cas, vous auriez pu utiliser un signal audio à la place et obtenir sensiblement le même effet. Une deuxième utilisation, dans laquelle le signal évènement ne peut être remplacé par de l'audio, se présente lorsque vous êtes intéressé(e) non seulement par les valeurs transportées par le signal, mais aussi par les *instants* auxquels chaque nouvelle valeur est délivrée par le câble, autrement dit *quand* l'évènement est envoyé. Comme exemple, citons un signal "gate" d'enveloppe du niveau primaire: l'enveloppe est lancée à l'instant où un évènement *arrive* à l'entrée de la porte (en anglais "gate").

Lorsque nous parlons de signaux audio, de contrôle, évènements et logiques dans Reaktor Core, nous ne parlons pas vraiment de différences techniques entre ces signaux (techniquement, ils sont tous équivalents dans Reaktor Core), mais plutôt des différentes façons d'utiliser un signal. Comme nous le voyons maintenant, un signal évènement du niveau primaire de Reaktor peut être utilisé comme signal de contrôle, évènement, voire logique (alors qu'un signal audio du niveau primaire de Reaktor peut être utilisé comme audio ou comme contrôle, comme vous devriez vous en rappeler...).

Nous avons déjà appris à envoyer des signaux évènements du niveau primaire dans Reaktor Core et à les utiliser comme signaux de contrôle. Les entrées en mode évènement de la cellule *core audio* réalisant un filtre - que nous avons construite auparavant - en est un bon exemple. Il y a également des cas dans lesquels vous utiliseriez plutôt une cellule *core évènement* pour traiter certains signaux évènements du niveau primaire, qui sont en fait des signaux de contrôle. Voici un exemple de cellule core évènement englobant une macro *core* de modelleur de type contrôle (*Ctl Shaper*):

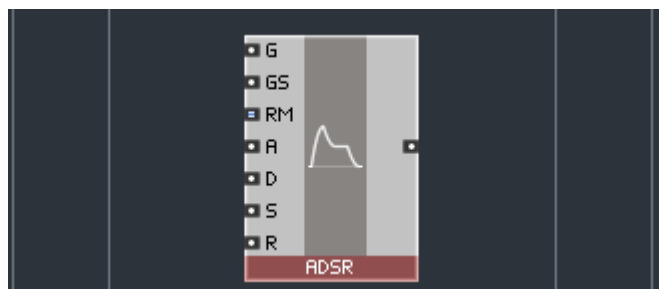


Ce modelleur de contrôle reçoit un signal de contrôle issu d'un signal événement du niveau primaire (p.ex. un signal de vélocité MIDI, ou un signal de LFO du niveau primaire), il le tord en fonction du paramètre "Shp" et envoie le résultat à la sortie.

Une restriction importante des cellules *core* événements (mentionnée plus haut) est que toutes leurs sources d'horloge sont désactivées. Ceci signifie que non seulement les oscillateurs et les filtres, mais aussi les enveloppes et les LFOs ne peuvent *pas* fonctionner dans les cellules *core* événements. Ces cellules sont vraiment conçues pour recevoir des événements du niveau primaire de Reaktor, les traiter et les renvoyer à l'extérieur, comme dans l'exemple ci-dessus.

Les signaux dérivés de signaux événements du niveau primaire peuvent aussi être utilisés comme véritables signaux événements dans les structures Reaktor Core. Nous allons maintenant regarder d'un peu plus près quelques cas simples d'utilisation d'événements dans Reaktor Core.

Le premier cas est l'utilisation d'une enveloppe dans une structure *core*. Comme nous venons de le voir, à cause de la restriction concernant les cellules *core* événements, nous pouvons uniquement utiliser une cellule *core* audio. Créons donc une nouvelle cellule core audio et utilisons *Standard Macro > Envelope > ADSR*:

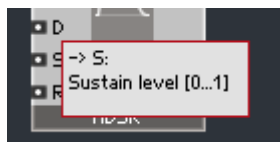


L'entrée supérieure de l'enveloppe est une entrée "gate" qui fonctionne sensiblement comme les entrées "gate" des enveloppes du niveau primaire: cette entrée allume ou éteint l'enveloppe en réponse à des *événements*. Pas de problème, créons une entrée événement dans notre cellule *core*:

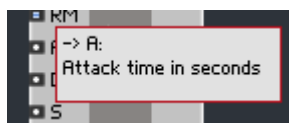


Cette entrée convertira les événements “gate” du niveau primaire en événements *core*.

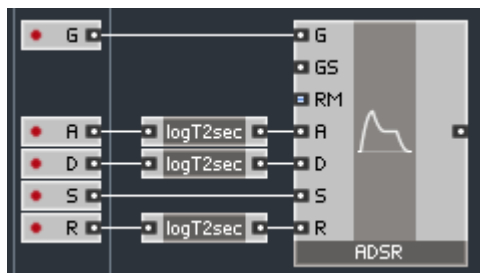
Intéressons-nous maintenant aux entrées A, D, S et R. L’entrée “S” (niveau de maintien, en anglais “sustain”) fonctionne comme au niveau primaire, ie elle attend un signal dans l’intervalle [0...1]:



Les entrées A, D et R sont quant à elles légèrement différentes en ce sens que, contrairement aux enveloppes du niveau primaire, elles attendent ici des temps spécifiés en secondes:

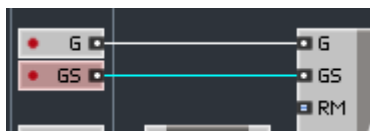


Ceci peut être géré par *Standard Macro > Convert > logT2sec*, qui convertit les temps des enveloppes du niveau primaire en secondes:

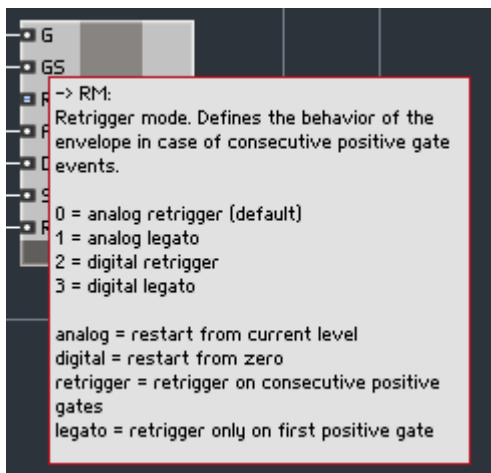


Bien que toutes les entrées de la structure ci-dessus soient en mode événement, du point de vue sémantique, la première entrée produit un signal événement alors que les autres produisent des signaux de contrôle.

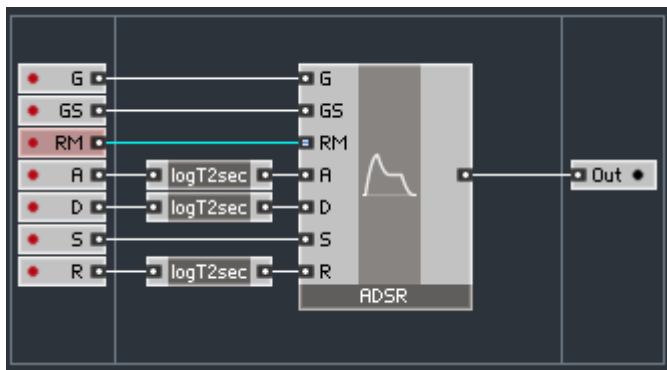
Notre enveloppe dispose de deux ports non encore connectés. Le port *GS* règle la sensibilité du seuil d’activation (“gate”). Réglé sur 0, l’enveloppe ignore complètement le niveau de l’entrée “gate” et elle est toujours à son amplitude maximale. Réglé sur 1, le niveau de seuil a son effet maximal, comme au niveau primaire de Reaktor. Nous pouvons contrôler cette sensibilité depuis l’extérieur via une nouvelle entrée:



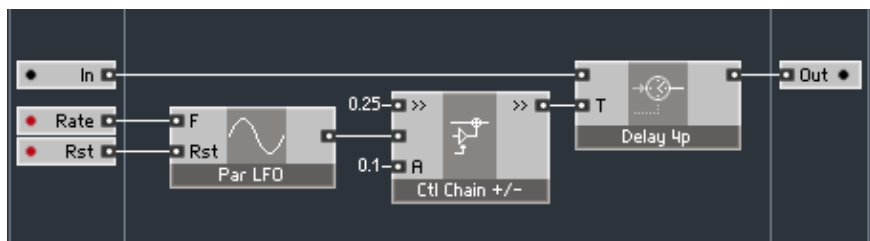
Le port *RM* spécifie le mode de redémarrage de l'enveloppe:



Ce port a un aspect différent car il attend des valeurs entières en entrée. Si nous considérons par exemple l'entrée pour le temps d'attaque, elle peut accepter un temps d'attaque d'1 s, 1,5 s ou 0,2 s. Au contraire, le port pour le mode de redémarrage ne s'attend pas à une valeur de 1,2 ou de 3,1. Ceci est indiqué par l'aspect différent du port, mais cela n'empêche pas de connecter à ce port un signal normal; utilisons donc une autre entrée évènement:



Si les valeurs venant de l'extérieur ne sont pas entières, elles sont arrondies à la valeur entière la plus proche. Par exemple, 1,2 sera arrondi à 1. Observons maintenant un autre exemple d'utilisation d'un signal évènement véritable:



La structure ci-dessus réalise une sorte de modulation de pitch (hauteur tonale). L'effet est produit par un délai dont la durée varie dans l'intervalle 250 ± 100 ms. La vitesse de cette variation est contrôlée par l'entrée *Rate*, qui contrôle en fait la vitesse du LFO (la valeur est en Hertz). Il s'agit donc d'un pur signal de contrôle. L'entrée *Rst* est un signal évènement, il peut être utilisé pour redémarrer le LFO. La valeur en entrée spécifie la phase de redémarrage, pour laquelle 0 signifie que le LFO redémarre au début du cycle, 0,5 au milieu du cycle et 1 à la fin. Vous pouvez faire des tests en connectant un bouton envoyant des valeurs spécifiques à cette entrée.

Les signaux logiques

Maintenant que nous en savons plus sur les signaux évènements et les signaux de contrôle, il est temps d'en apprendre un peu plus sur une autre manière d'utiliser les signaux dans Reaktor Core, via les *signaux logiques*. Voici un exemple d'un module qui traite les signaux logiques:



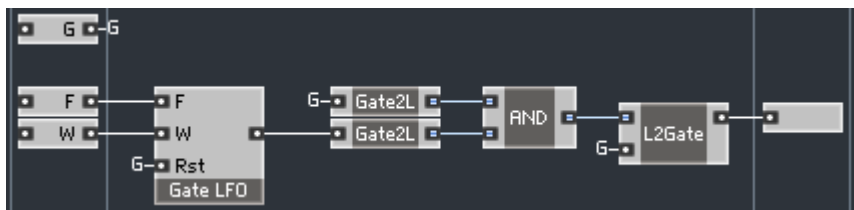
Comme vous pouvez le remarquer, les ports de ce module ont un type entier, comme l'entrée RM de l'enveloppe vue plus haut. Ceci est dû au fait que les signaux arrivant ne transportent généralement que des valeurs entières, et même "pire", ils ne transportent que des 0 et des 1.

Pour le signal logique, la valeur 1 correspond à l'état *true* (vrai en anglais), et la valeur 0 à l'état *false* (faux en anglais). La signification des expressions *true* et *false* doit être assignée par l'utilisateur. Vous pouvez décider qu'un signal logique détermine si une porte ("gate") particulière est ouverte ou non:



Ici, une macro *Gate2L* vérifie le signal d'entrée et produit une sortie *true* (ou 1) si la porte est ouverte et *false* (ou 0) si elle est fermée.

Vous pouvez utiliser les signaux logiques pour effectuer des “traitements logiques”. Par exemple, nous pourrions construire un processeur de porte logique qui appliquerait à une porte logique MIDI une autre porte, régulièrement cadencée, basée sur une horloge



Les modules *Gate2L*, *AND* et *L2Gate* sont des modules logiques et se trouvent dans le menu *Standard Macro > Logic*. Le *Gate LFO* est une macro que nous avons construite pour ce processeur. Elle génère un signal de porte, s'ouvrant et se fermant à intervalles réguliers.

La porte en entrée et la sortie du LFO sont connectées à des convertisseurs *Gate2L* qui convertissent les signaux de portes en signaux logiques, transformant les portes ouvertes en *true* et les portes fermées en *false*. Le module *AND* sort la valeur *true* si et seulement si les deux portes sont à l'état ouvert en même temps. En d'autres termes, la sortie du module *AND* est égale à *true* si et seulement si l'utilisateur maintient une touche enfoncée et si en même temps le LFO produit une porte ouverte. Ceci signifie que, tant que l'utilisateur maintient une touche enfoncée, il y aura alternativement des valeurs *true* et *false* en sortie du module *AND*, la vitesse d'alternance étant réglée par la fréquence du LFO. La sortie du module *AND* est reconvertie en signal de porte, l'amplitude du signal est prise du signal de porte en entrée, pour que le niveau du signal de porte reste inchangé.

Voici la structure de notre macro *Gate LFO*:



L'entrée F définit la vitesse des répétitions d'ouverture de la porte, et l'entrée W définit la durée d'ouverture des portes (à 0, elles restent ouvertes la moitié du cycle, à -1 elles restent ouvertes 0 % du temps et à 1, 100 % du temps). L'entrée Rst redémarre le LFO en réponse aux événements entrants (le LFO est donc redémarré à chaque fois qu'un événement arrive à l'entrée G principale).

Le module connecté à l'entrée Rst du *Rect LFO* est appelé *Value* et se trouve dans *Standard Macro > Event Processing*. Il assure que le LFO est redémarré avec une phase nulle en remplaçant toutes les valeurs des événements entrants par la valeur de l'entrée inférieure, qui est en l'occurrence 0. La sortie du LFO est convertie en signal de porte via un convertisseur *Ctl2Gate*, que l'on trouve aussi dans *Standard Macro > Event Processing*.

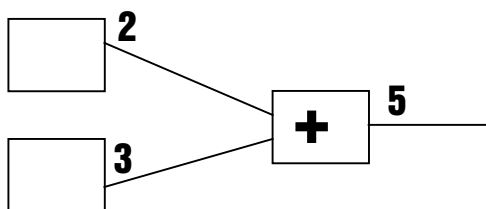
Comme nous l'avons vu, les LFOs ne fonctionnent pas dans les cellules *core* événements, donc si vous voulez essayer cette structure, vous devez utiliser une cellule *core* audio.

Fondements de Reaktor Core: le modèle du signal core

Les valeurs

La plupart des sorties des modules de Reaktor Core produisent des valeurs. “Produire” une valeur signifie qu’à tout moment une valeur est associée à la sortie en question. Cette valeur est disponible pour les modules dont les entrées sont connectées à cette sortie.

Dans l’exemple suivant, un module additionneur obtient les valeurs 2 et 3 de deux autres modules, dont les sorties sont connectées à ses entrées; l’additionneur produit la valeur 5 à sa sortie

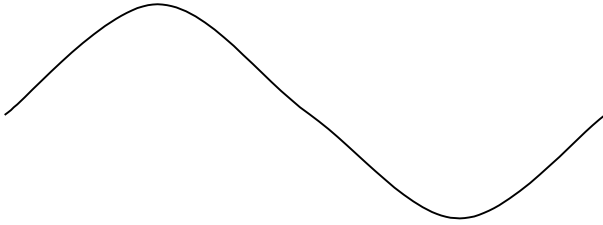


Si vous voulez faire une analogie avec le monde matériel, pensez les valeurs comme des niveaux de signaux (des tensions électriques), en particulier si vous avez affaire à des modules de grande échelle tels que les oscillateurs, les filtres, les enveloppes, etc. Cependant, les valeurs ne se limitent pas à cela. Après tout, ce ne sont que des valeurs, et elles peuvent être utilisées pour réaliser n’importe quel algorithme de traitement, pas seulement pour modeler un voltage.

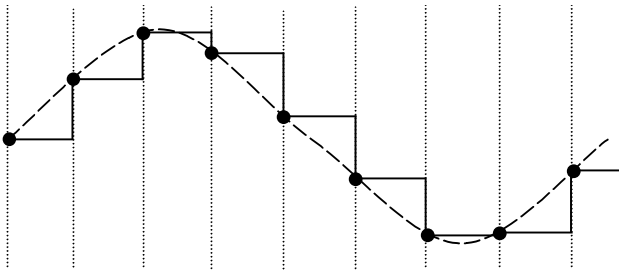
Les évènements

Dans le monde numérique, le temps n’est pas continu, il est discret, c’est-à-dire quantifié. L’un des exemples les plus connus est l’enregistrement numérique, qui n’enregistre pas l’information totale d’un signal audio évoluant continûment dans le temps, mais seulement son niveau à des instants précis régulièrement espacés. Le nombre de ces instants dans chaque seconde définit le fameux *taux d’échantillonnage*.

Voici la représentation d'un signal continu:

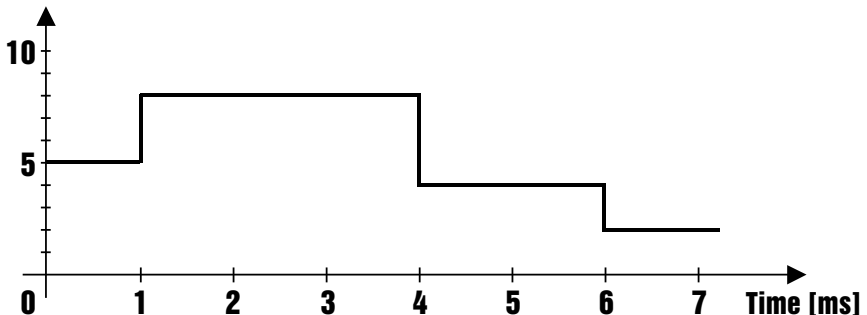


et sa représentation numérique (ou digitale):



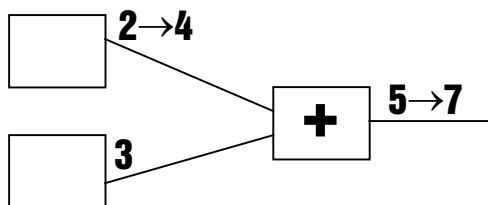
Ceci implique que, puisque nous sommes dans le monde numérique, les sorties de nos modules ne peuvent faire évoluer leurs valeurs de façon continue. D'un autre côté, nous n'avons pas besoin de forcer nos sorties à changer leur valeur "à des instants précis régulièrement espacés", autrement dit nous n'avons pas besoin de maintenir un taux d'échantillonnage particulier tout au long de nos structures. En outre, dans certaines parties de nos structures, nous n'avons pas même besoin de maintenir un quelconque taux d'échantillonnage, autrement dit nos changements n'ont pas besoin de survenir "à des instants précis régulièrement espacés".

Par exemple, au temps zéro, la sortie de notre additionneur a la valeur 5. Le premier changement pourrait survenir au temps 1 ms (une milliseconde). Le deuxième pourrait survenir à 4 ms. Le troisième à 6 ms:

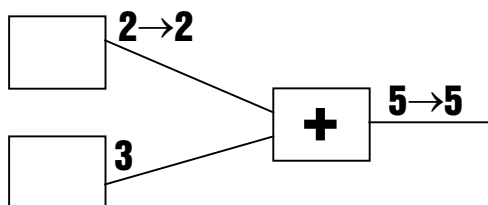


Sur la figure ci-dessus, nous voyons les changements de la sortie de notre additionneur survenant entre 0 et 7 ms. À l'instant où la sortie change de valeur, elle génère un évènement. Un *évènement* signifie que la sortie “rend compte” d'un changement de son état, en l'occurrence qu'elle change de valeur.

Dans l'exemple suivant, le module en haut à gauche a modifié sa valeur de 2 à 4, générant un évènement. En réponse, l'additionneur change aussi sa valeur de sortie et génère également un évènement à sa sortie.



Le module supérieur gauche aurait aussi pu générer un nouvel évènement avec la même valeur que l'ancienne. L'additionneur aurait quand même répondu en générant aussi un nouvel évènement, bien sûr sans changer non plus sa valeur de sortie.



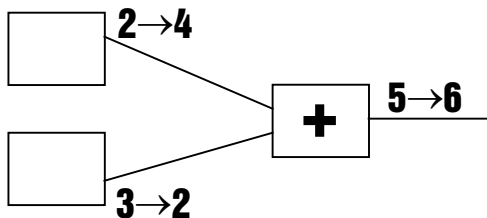
La nouvelle valeur apparaissant à la sortie ne doit pas forcément être différente de l'ancienne. Cependant, la seule manière pour une sortie de changer sa valeur est de générer un évènement.

Comme vous l'avez vu dans les exemples précédents, un évènement survenant à la sortie d'un module va être perçu par les modules connectés “en aval”, qui produiront en réponse d'autres évènements (comme l'additionneur qui produisait un évènement en réponse à l'évènement provenant du module en haut à gauche). Ces nouveaux évènements vont à leur tour être perçus par les modules connectés aux sorties correspondantes et propagés de proche en proche, jusqu'à ce que la propagation s'arrête, pour une des raisons dont nous discuterons plus loin dans ce texte.

Dans Reaktor Core, les évènements sont différents des évènements du niveau primaire de Reaktor. Ils suivent des règles différentes que nous détaillerons plus loin.

Évènements simultanés

Considérons la situation suivante: les deux modules de gauche des exemples précédents produisent *simultanément* un évènement.

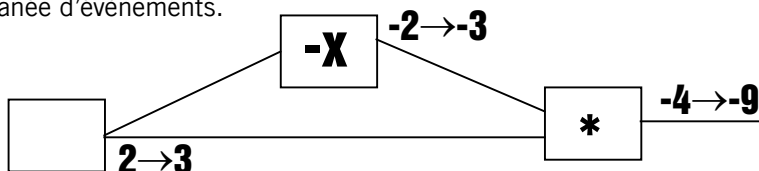


Ceci est l'une des caractéristiques essentielles de Reaktor Core: les évènements peuvent survenir simultanément en plusieurs endroits. Dans cette situation, les évènements produits simultanément par les deux modules de gauche arrivent aussi simultanément aux entrées de l'additionneur. En réponse, l'additionneur va produire exactement un évènement à sa sortie.

Ce comportement est différent de celui observé au niveau primaire de Reaktor, dans lequel les évènements ne peuvent pas survenir simultanément: dans une telle situation, l'additionneur (en mode évènement) produirait deux évènements en sortie.

Bien sûr, en réalité les évènements ne sont pas produits simultanément par les modules supérieur gauche et inférieur gauche, car les deux modules sont traités par le même processeur, et un processeur ne peut jamais traiter qu'un seul module à la fois. Mais l'important pour nous est que ces évènements sont *logiquement* simultanés, c'est-à-dire qu'ils sont traités comme simultanés par les modules qui les reçoivent.

Nous allons maintenant prendre un autre exemple d'une propagation simultanée d'évènements.



Dans la figure ci-dessus, le module à l'extrême gauche envoie un évènement, modifiant sa valeur de 2 à 3. L'évènement est envoyé simultanément à la fois aux modules inverseur (-x) et multiplicateur (*). En réponse à l'évènement entrant, l'inverseur produit une nouvelle valeur de sortie, -3. Il est important de remarquer que, bien que l'évènement en sortie de l'inverseur soit produit en réponse à l'arrivée de l'évènement venant du module tout à gauche, et donc qu'il doit survenir après ce dernier, les deux évènements restent logiquement simultanés. Ceci signifie qu'ils arrivent simultanément aux entrées du multiplicateur et que le multiplicateur, en réponse, ne produit qu'un seul évènement en sortie, avec une valeur de -9.

Dans le niveau primaire, vous auriez eu ici deux évènements à la sortie du module multiplicateur. Vous n'auriez pu déterminer si l'évènement produit à la sortie du module de gauche avait d'abord été envoyé à l'inverseur ou au multiplicateur (bien que cela soit sans importance pour cette structure particulière).

En général, vous pouvez utiliser la règle suivante pour déterminer si deux évènements particuliers sont simultanés ou non:

Tous les évènements issus d'un (envoyés en réponse à un) même évènement sont simultanés. Tous les évènements issus d'un nombre arbitraire d'évènements simultanés (survenant à différentes sorties mais que l'on sait simultanés) sont aussi simultanés.

Le dernier exemple montre l'utilisation de la simultanéité d'évènements. Dans ce cas, nous éliminons le traitement redondant du second évènement par le multiplicateur, qui aurait nécessité plus de temps processeur. Dans des structures plus grandes, en l'absence d'un tel concept de simultanéité, le nombre d'évènements peut croître de façon incontrôlée, à moins que le concepteur de la structure n'ait apporté un soin particulier à maintenir le nombre d'évènements doublons au plus bas.

Au-delà de l'économie en temps de processeur, ce concept mène à des différences importantes dans l'approche de la construction des structures, en particulier pour les structures réalisant des algorithmes de traitement du signal (DSP) de bas niveau. Vous comprendrez et percevrez mieux ces différences lorsque vous commencerez à construire vos propres structures.

L'ordre de traitement

Comme nous l'avons vu dans les exemples précédents, lorsqu'un module envoie un évènement, les modules "en aval" y répondent. On pourrait en conclure que, bien qu'ils produisent des évènements "logiquement simultanés", les modules sont en fait traités de façon non simultanée. On pourrait alors supposer que, pour une connexion, il soit raisonnable de traiter le module "en amont" avant le module "en aval". Si quelqu'un arrive à ces conclusions... il (ou elle) a absolument raison.

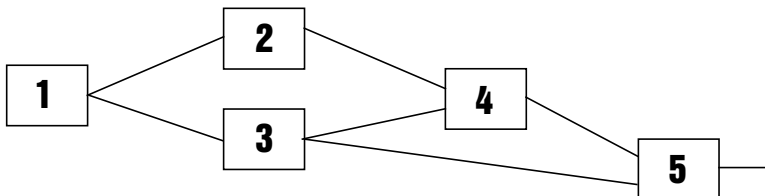
La règle générale de l'ordre de traitement des modules est la suivante:

Si deux modules connectés traitent des évènements simultanés, alors le module "en amont" est traité en premier. Si les évènements ne sont pas simultanés, l'ordre de traitement des modules correspond bien entendu à l'ordre des évènements traités.

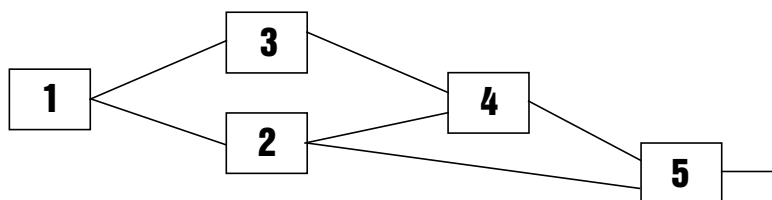
Nous déduisons de la règle ci-dessus qu'un chemin de connexion unidirectionnel (toujours vers l'amont ou toujours vers l'aval) d'un module au suivant induit un ordre de traitement déterminé de ces deux modules: le module en amont est traité en premier.

S'il n'y a pas de chemin unidirectionnel entre les deux modules, leur ordre de traitement est indéfini (pour des évènements simultanés). Ceci signifie que cet ordre est arbitraire et peut changer à tout moment. Le concepteur de la structure doit donc veiller à ce qu'une telle situation n'arrive que pour des modules dont l'ordre de traitement relatif est sans importance, ce qui est généralement le cas tant qu'aucune connexion OBC (cf. plus bas) n'entre en jeu. Here is an example, the digits showing the order of module processing:

Voici un exemple, les nombres indiquant l'ordre de traitement:

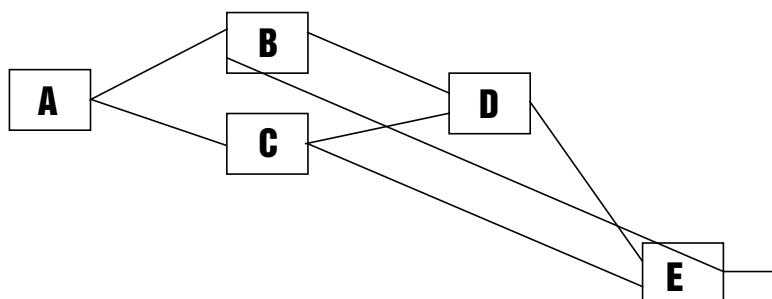


Pour la structure ci-dessus, il existe un autre ordre de traitement possible:



Il n'y a aucun moyen de dire lequel des deux sera choisi par l'application. Heureusement, tant que vous n'utilisez pas de connexions OBC (détaillées plus bas), l'ordre relatif de traitement des modules dans des cas comme celui-ci est tout-à-fait indifférent.

Ces règles permettant de déterminer l'ordre de traitement ne peuvent être appliquées s'il y a une réinjection dans les structures, car dans ce cas, pour n'importe quelle paire de modules dans la boucle de réinjection, nous ne pouvons plus dire lequel est "en amont" et lequel est "en aval". Le problème de la manipulation des réinjections, y compris pour l'ordre de traitement, sera abordé plus tard.



Pour la structure ci-dessus, il n'est pas possible de définir si le module B est en amont ou en aval du module D, et vice versa (!). Visiblement, il y a une connexion allant vers l'amont de D vers B, mais il y en a également une de B vers D (via E).

Retour aux cellules core évènements

Observons les cellules *core* évènements du point de vue du concept d'évènement de Reaktor Core que nous venons de décrire.

Comme vous vous en rappelez certainement, les cellules *core* évènements ont des entrées et des sorties évènements. Ces entrées et sorties sont des points

d'interface entre les niveaux primaire et *core* de Reaktor. Ils font fonction de convertisseurs entre les événements du niveau primaire et les événements *core*, et vice versa. Les règles de conversion sont les suivantes:

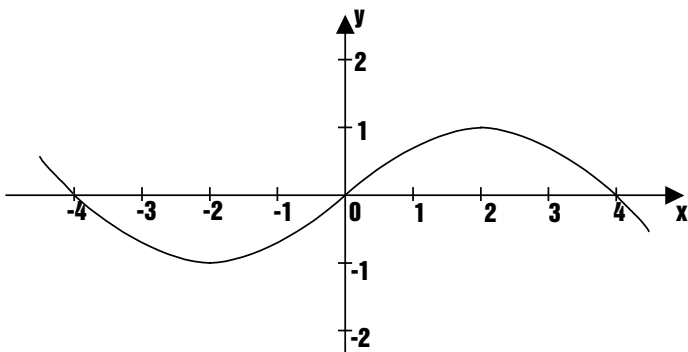
Les entrées événements envoient des événements *core* à l'intérieur de la structure en réponse aux événements du niveau primaire venant de l'extérieur. Comme les événements du niveau primaire extérieur ne peuvent arriver simultanément aux différentes entrées, les événements produits à l'intérieur ne sont pas non plus simultanés.

Les sorties événements envoient des événements de niveau primaire à l'extérieur de la structure en réponse aux événements *core* venant de l'intérieur. Bien que des événements *core* puissent être simultanés en différentes sorties, les événements du niveau primaire ne peuvent être envoyés simultanément. Ainsi, pour des événements *core* simultanés, les événements correspondants du niveau primaire sont envoyés les uns à la suite des autres, *les sorties supérieures envoyant toujours leurs événements avant les sorties inférieures*.

Nous allons maintenant tester ces règles en pratique.

Essayons de construire un module de traitement d'événements qui applique un modelage du signal selon la formule: $y = 0.25 * x * (4 - |x|)$

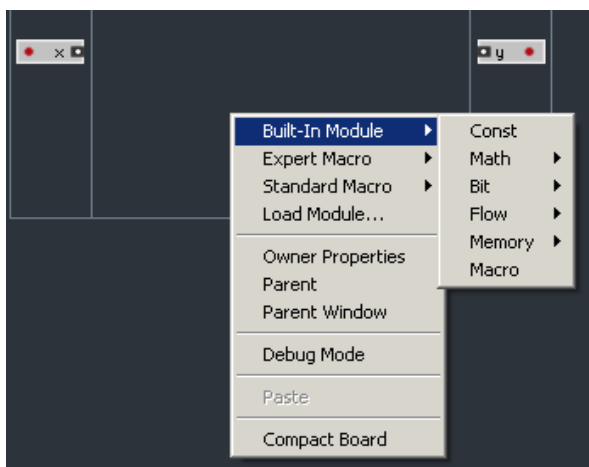
La représentation graphique de cette fonction ressemble à ceci:



Commençons par créer une nouvelle cellule *core* événement avec une entrée et une sortie, appelées respectivement "x" et "y".



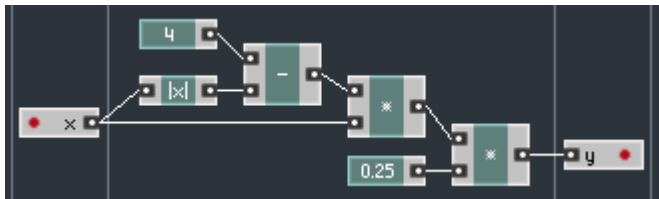
Créons ensuite la structure qui effectue la fonction. Nous devons créer les modules “|x|” (valeur absolue), “-” (soustraction) et deux “*” (multiplication) dans la zone des modules normaux (au centre). Ce ne sont pas des macros core mais bien des modules Reaktor Core préfabriqués. Pour les insérer dans les structures core, effectuez un clic droit sur le fond de la zone normale et sélectionnez le sous-menu *Built-In Module*:



Vous trouverez tous les modules dont nous avons besoin dans le sous-menu *Built In Module > Math*:

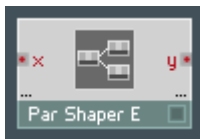


Nous avons besoin de deux valeurs constantes, 0,25 et 4. Nous pourrions utiliser des *QuickConsts*, tout comme nous l'avons fait dans les premiers chapitres, mais nous pouvons aussi insérer un module de constante réelle via *Built In Module > Const* (comme avec *QuickConst*, la valeur peut être précisée dans la fenêtre Properties):

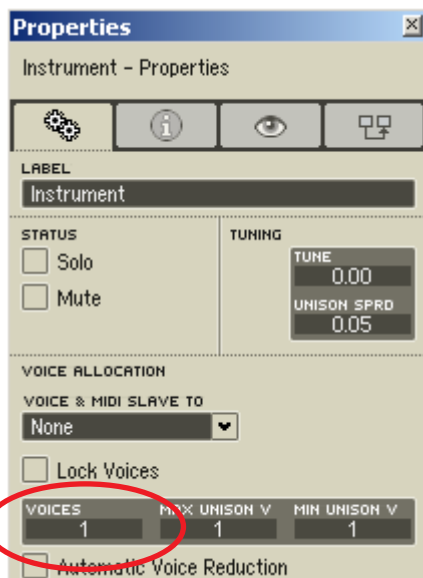


Bien sûr, dans notre cas, il n'y a pas d'intérêt particulier à utiliser des modules *Const* en lieu et place des *QuickConsts*, mais cela peut parfois se révéler utile (p.ex. si la même constante doit être connectée à plusieurs entrées, il peut être judicieux d'utiliser un module *Const*, parce qu'alors vous n'avez besoin que d'un seul module, et vous avez en outre un point unique pour contrôler la valeur de toutes les entrées).

Bref, la structure maintenant créée modèle bien le signal de la manière voulue. Nous pouvons nommer notre module et revenir au niveau primaire:



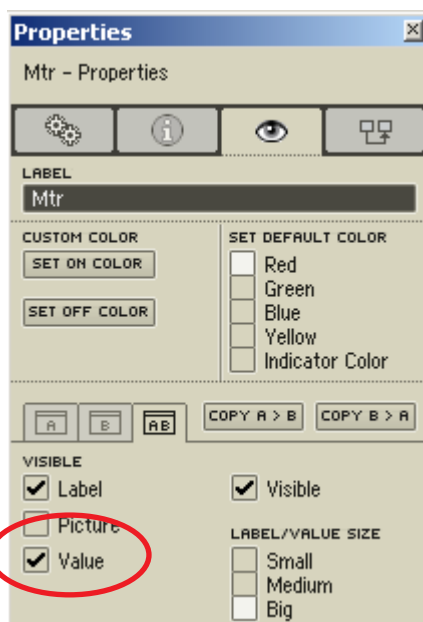
Maintenant, testons-le. Fixez le nombre de voix de l'instrument Reaktor à 1, pour qu'il soit plus facile d'utiliser un module Meter (*Indicateur* en anglais):



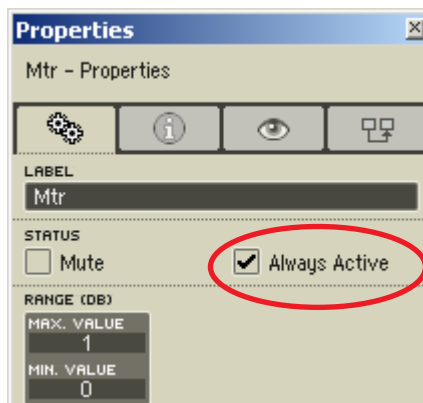
Créez un module *Knob* (*Potentiomètre* en anglais) et un module *Meter*, et connectez-les à l'entrée et à la sortie de votre module:



Réglez les propriétés du potentiomètre et de l'indicateur de niveau. N'oubliez pas de régler l'indicateur pour qu'il affiche sa valeur:



ni de cocher la case “Always Active” (“Toujours actif” en anglais):



Maintenant, tournez le potentiomètre et observez l'évolution de la valeur.



La structure de modelage d'évènements que nous venons de construire devrait fonctionner parfaitement pour le modelage de signaux de contrôle, mais elle comporte encore un défaut mineur dans sa façon de traiter les évènements. Nous reviendrons sur ce problème (et nous le résoudrons !) un peu plus tard.

Les structures avec état interne

Les signaux d'horloge

La manière dont une cellule Reaktor Core traite les évènements entrants est entièrement fonction du module en question. Normalement, un module traite la valeur entrante d'une certaine manière, mais il peut aussi tout-à-fait l'ignorer. Le cas le plus typique d'un tel traitement est celui des *entrées d'horloge*.

Un exemple de module avec une entrée d'horloge est le *Latch* (*Loquet* en anglais). Le *Latch* n'est pas un module d'usine, c'est une macro, mais c'est néanmoins un exemple parfait pour illustrer le principe d'horloge.

Le *Latch* a deux entrées: une pour la valeur et une pour l'horloge.



En réponse à un évènement entrant, l'entrée de valeur (l'entrée supérieure) enregistre la valeur entrante dans la mémoire interne du loquet, rien n'est envoyé en sortie. L'entrée d'horloge, en réponse à un évènement entrant, envoie vers la sortie la dernière valeur enregistrée (elle ouvre le loquet).

À moins que cela ne soit spécifié autrement, l'entrée d'horloge ignore complètement la valeur de l'évènement arrivant, elle répond uniquement à l'arrivée de l'évènement.

Comme nous discutons pour l'instant des signaux d'horloge et non des loquets, les exemples d'utilisation du module *Latch* seront détaillés plus loin.

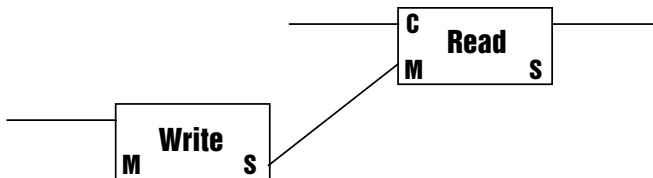
De même qu'il existe des modules avec des entrées d'horloge, il paraît clair que certains signaux de la structure ne transportent aucune *valeur* utilisée par (ou utile pour) le module. Certains signaux peuvent même être produits dans le seul but d'être utilisés comme des sources d'horloge. Appelons-les *signaux d'horloge*.

Un exemple de signal d'horloge est l'horloge du taux d'échantillonnage. Elle produit un évènement pour chaque nouvel échantillon à générer, donc à 44,1 kHz elle "bat" 44100 fois par seconde. La valeur de ce signal n'a pas de sens, elle n'est pas prévue pour servir en quoi que ce soit, et d'ailleurs (dans l'implémentation actuelle) elle est égale à zéro.

Les Object Bus Connections (OBC)

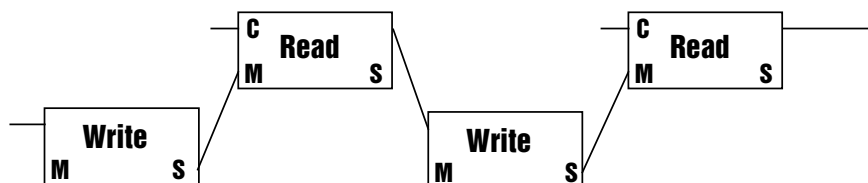
Les *Object Bus Connections* (OBC, *Connexions sur Bus Objet*) sont un type spécial de connexion entre les modules. Une connexion OBC entre deux modules "déclare" qu'ils partagent un même objet (ou état) interne. Le cas le plus typique est la paire de modules *Read* et *Write*, qui partagent une mémoire commune s'ils sont connectés via OBC.

La fonction du module *Write* est d'écrire une valeur entrante dans la mémoire partagée via OBC. La fonction du module *Read* est de lire la valeur dans la mémoire partagée via OBC en réponse au signal d'horloge entrant (entrée C). La valeur lue est envoyée à la sortie du module *Read*.



La structure ci-dessus effectue le traitement de la macro *Latch* (en fait c'est la structure interne de la macro *Latch*). Les broches *M* et *S* des modules *Read* et *Write* sont des broches de type *Latch OBC*. La broche *M* est l'entrée maîtresse de la connexion et *S* la sortie esclave de la connexion. L'entrée maîtresse du module *Read* est connectée à la sortie esclave du module *Write* (les deux autres broches maîtresse et esclave ne sont pas utilisées). Ainsi, dans cette structure, les modules *Write* et *Read* partagent la même mémoire.

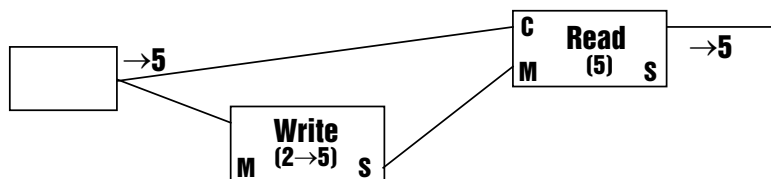
Dans la structure suivante, on trouve deux paires de modules *Write* et *Read*. Chaque paire dispose de sa propre mémoire. Notez que la connexion du milieu (de la sortie du *Read* à l'entrée du *Write*) n'est pas une connexion OBC.



On pourrait se demander quelle différence il peut bien y avoir entre être maître(sse) et esclave. Du point de vue de la possession de l'objet partagé (dans notre cas une mémoire), il n'y a aucune différence. Cependant, comme nous l'avons déjà vu dans les sections précédentes, une règle énonce que les modules "en amont" sont traités avant les modules "en aval" en ce qui concerne le traitement des "événements simultanés". De ce fait, dans les deux derniers exemples, les modules *Write* sont traités avant leurs *Read* esclaves.

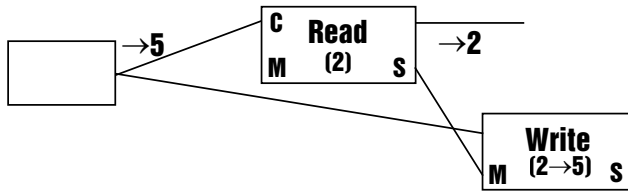
L'ordre relatif de traitement des modules connectés par OBC est défini par les mêmes règles que pour les autres modules: les modules "en amont" sont traités en premier.

En effet, considérons les deux cas sur l'exemple suivant. Dans les deux cas, l'état original de la mémoire est à 2 et le même événement de valeur 5 est envoyé aux modules *Read* et *Write*. Dans un cas, le module *Write* est le maître, dans l'autre c'est le module *Read*.



On a représenté ci-dessus la structure du premier cas. Le module de gauche envoie un événement de valeur 5 qui arrive d'abord au module *Write*, ce qui entraîne l'écriture de la nouvelle valeur 5 dans la mémoire partagée par la paire *Write* et *Read*. Maintenant, l'événement arrive au module *Read*, fonctionne comme un signal d'horloge et commande l'opération de lecture, qui à son tour lit la valeur de 5 récemment enregistrée et l'envoie à sa sortie. C'est la fonction de la macro *Latch* de la librairie de macros de Reaktor Core.

Considérons maintenant la structure du second cas:

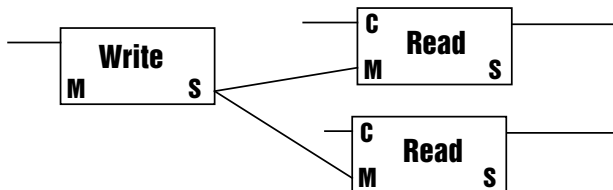


Nous avons ici la situation opposée. D'abord, l'évènement d'horloge arrive au module *Read*, envoyant la valeur de 2 à sa sortie. C'est seulement après cette lecture que l'évènement arrive à l'entrée du module *Write*, ce qui passe la valeur enregistrée à 5. Cette structure réalise en fait la fonction d'un bloc Z^{-1} (retard d'un échantillon), très souvent utilisé en théorie du traitement du signal numérique. En effet, la valeur de sortie est toujours en retard d'un échantillon par rapport à l'entrée.

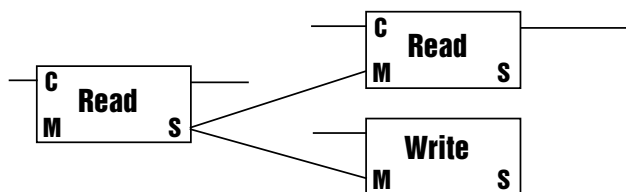
Comme nous l'avons mentionné, la structure ci-dessus réalise la fonction Z^{-1} . Mais avant de pouvoir réellement construire ou utiliser vous-même de telles structures, vous devez prendre connaissance de certains points importants. Nous vous conseillons donc de poursuivre votre lecture.

Si vous avez plus de deux modules connectés par des câbles OBC, tous ces modules partagent le même état interne. Il devient alors très important de savoir si un ordre particulier d'opérations de lecture et d'écriture doit être appliqué, et si oui, lequel.

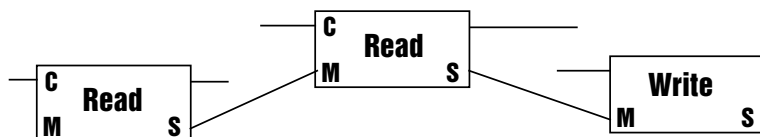
Par exemple, dans la structure suivante, l'ordre relatif de traitement des deux opérations de lecture est indéfini, mais elles arrivent toutes deux après l'opération d'écriture, il faut donc que cela vous convienne:



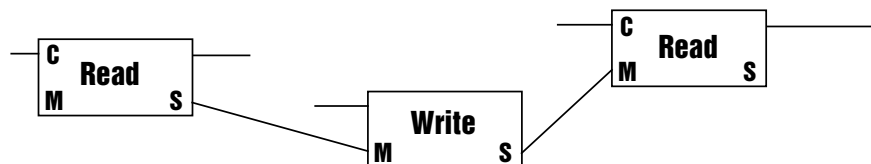
Dans la structure suivante, l'ordre relatif des opérations de première écriture et de seconde lecture est indéfini, donc cette structure est potentiellement dangereuse et doit être généralement évitée:



Une manière plus satisfaisante de réaliser la structure ci-dessus est la suivante:



Ou encore celle-là:



Même si vous pensez qu'à un certain endroit, l'ordre relatif des opérations de lecture et d'écriture n'est pas important, imposer un certain ordre de traitement n'est pas gênant et c'est un peu plus sûr.

L'ordre relatif des opérations d'écriture est important. L'ordre relatif des opérations de lecture importe peu, tant que leur ordre par rapport aux opérations d'écriture est bien défini.

Les connexions OBC ne sont pas compatibles avec les connexions normales. De plus, les connexions OBC correspondant à des types d'objets différents ne sont pas compatibles entre elles. Les broches de types incompatibles ne peuvent pas être connectées, p.ex. vous ne pouvez pas connecter une sortie de signal normal à une entrée OBC.

Initialisation

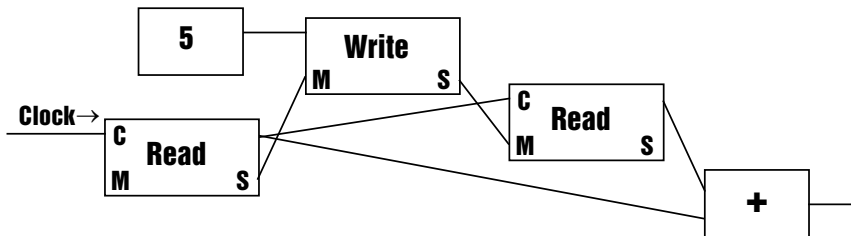
Comme nous commençons à travailler avec des objets ayant un état interne (dans le cas des *Read* et *Write*, la mémoire partagée), il devient nécessaire de comprendre ce qu'est *l'état initial* de la structure que vous avez construite. Par exemple, si nous voulons lire la valeur dans une mémoire (au moyen d'un module *Read*) avant que quoi que ce soit n'y ait été écrit, quelle sera la valeur lue ? Et si la valeur par défaut ne nous convient pas, comment pouvons-nous la modifier ?

Ces questions sont réglées par le mécanisme d'initialisation de Reaktor Core. L'initialisation des structures *core* est effectuée de la manière suivante:

- d'abord, tous les états sont initialisés à des valeurs par défaut, généralement des zéros. En particulier, les mémoires partagées et toutes les valeurs de sortie des modules sont réglées sur zéro, à moins que vous ne l'ayez spécifié autrement ;
- ensuite, un *évènement d'initialisation* est envoyé *simultanément* par toutes les *sources d'initialisation*. Les sources d'initialisation comprennent la plupart des modules qui n'ont pas d'entrée: les modules *Const* (y compris les *QuickConsts*), les entrées des cellules *core*, et certains autres. Les sources envoient généralement leur valeur initiale comme évènement d'initialisation, p.ex. les constantes envoient leurs valeurs, et les entrées des cellules *core* envoient les valeurs initiales reçues du niveau primaire, à l'extérieur.

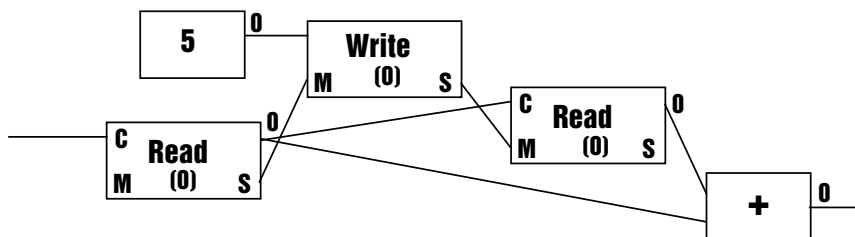
Si le module est une source d'évènement d'initialisation, vous en trouverez mention dans la section de référence des modules. Si le module n'est pas une source d'initialisation, il traite les évènements d'initialisation exactement comme n'importe quel autre évènement. *La plupart* des sources d'initialisation sont les modules qui n'ont pas d'entrée(s) et seulement ceux-ci.

Observons le fonctionnement de l'initialisation avec l'exemple suivant:

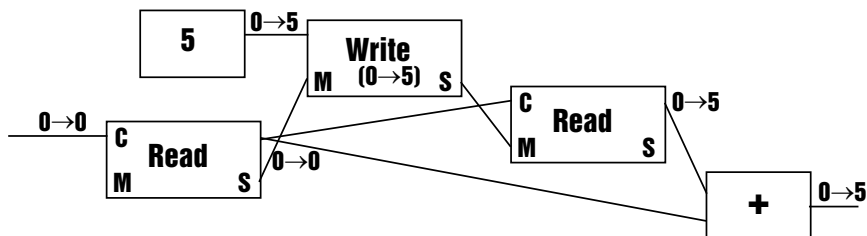


Ce n'est qu'une partie de la structure, le module *Read* sur la gauche est connecté à une source d'horloge (en anglais *clock*), qui envoie aussi un évènement d'initialisation (comme toute source d'horloge qui se respecte).

Initialement, tous les signaux de sortie ainsi que l'état interne de la chaîne *Read-Write-Read* sont fixés à zéro.

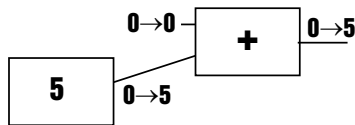


Puis un évènement d'initialisation est envoyé simultanément de la source d'horloge et de la constante 5.



Le module *Read* sur la gauche est traité avant le module *Write* et donc l'évènement d'horloge y arrive avant que la nouvelle valeur ne soit écrite dans la mémoire, donc la sortie de ce module *Read* est égale à zéro. Puis la nouvelle valeur est écrite dans la mémoire par le module *Write*. Maintenant, le second module *Read* est lancé, générant une valeur de 5 à sa sortie. Enfin, le module additionneur est traité et génère une somme de 5.

Comme vous vous en souvenez, les entrées déconnectées sont traitées dans Reaktor Core comme des valeurs nulles (à moins qu'elles ne soient spécifiées autrement par un module particulier). Cela signifie que ces entrées reçoivent aussi un évènement d'initialisation, exactement comme si un véritable module de constante nulle y était connecté.



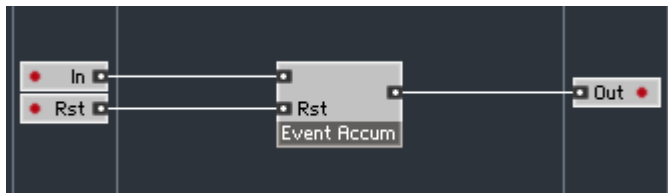
Ci-dessus, un additionneur avec une entrée déconnectée et une connectée à une constante reçoit deux évènements d'initialisation simultanés, l'un venant de la constante zéro "par défaut" et l'autre de la vraie connexion à une constante.

Il peut aussi y avoir une signification spéciale aux entrées déconnectées qui ne sont pas des entrées de signal (elles ne peuvent donc pas être connectées à une constante nulle). Par exemple, une entrée maîtresse déconnectée d'un module *Write* signifie que la chaîne de la mémoire partagée démarre à cet endroit et continue avec les modules connectés à la sortie esclave.

Construire un accumulateur d'évènements

Le module accumulateur d'évènements que nous voulons construire maintenant doit comporter deux entrées, l'une pour les valeurs d'évènements à accumuler, et l'autre pour remettre l'accumulateur à zéro. Il lui faut aussi une sortie qui envoie la somme totale des évènements accumulés.

Nous allons construire ce module sous la forme d'une macro *core*. Une telle macro sera facile à utiliser à l'intérieur d'une cellule *core* évènement:



L'intérieur de notre macro ressemble pour l'instant à ceci:

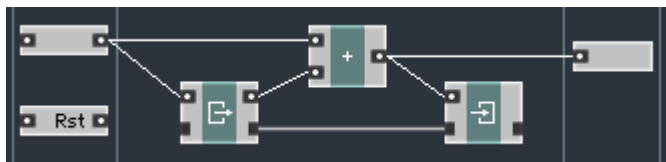


Le module accumulateur a besoin d'un état interne dans lequel il stockera la valeur somme des événements accumulés. Nous allons utiliser les modules *Read* et *Write* pour construire la boucle de l'accumulateur. Vous les trouverez dans le sous-menu *Built In Module > Memory*:



Le module que vous voyez à gauche (avec une flèche sortant du rectangle) est le module *Read*, et le module à droite (avec une flèche entrant dans le rectangle) est le module *Write*.

En réponse aux événements entrants, la boucle de l'accumulateur doit prendre l'ancienne valeur et lui ajouter la nouvelle. Nous allons donc utiliser le module *Read* pour récupérer l'ancien état, utiliser un additionneur pour y ajouter la nouvelle valeur et utiliser le module *Write* pour stocker la valeur en résultant.



Veuillez noter que l'horloge du module *Read* est pilotée par l'évènement entrant, et que le module *Write* (relié au *Read* par une connexion OBC) est bien sûr situé en aval du premier, puisque nous voulons écrire après avoir lu.

La structure ci-dessus accumule les valeurs entrantes et envoie en sortie le résultat de leur somme. Il ne manque plus que la fonction "reset" (le redémarrage) ainsi que la circuiterie pour définir l'état initial.

Penchons-nous d'abord sur la circuiterie de reset. Puisque nous sommes dans le monde de Reaktor Core, l'entrée "In" et la sortie "Rst" doivent envoyer un évènement simultanément. Si nous voulons que cette macro core soit utilisable dans des contextes divers et variés, nous devons en tenir compte. Supposons que les entrées "In" et "Rst" produisent simultanément un évènement. Que voulons-nous qu'il arrive dans ce cas ? Le reset doit-il logiquement survenir avant que l'évènement accumulé soit traité, ou après ? Ceci est très similaire à la différence entre les fonctions *Latch* et Z^{-1} , qui diffèrent uniquement par l'ordre relatif de traitement de l'entrée du signal et de l'entrée de l'horloge.

Nous suggérons l'approche du *Latch*, car ce module est très largement utilisé dans les structures Reaktor Core, et son comportement est donc plus intuitif. Dans un *Latch*, le signal d'horloge arrive logiquement après le signal de valeur. Dans notre cas, le signal de reset doit arriver logiquement après le signal accumulé (obligeant l'état et la sortie à prendre la valeur zéro).

Nous devons donc, d'une façon ou d'une autre, "écraser" la sortie de l'accumulateur avec une valeur initiale. Pour ce faire, il nous faudra utiliser un nouveau concept que nous allons détailler maintenant.

Le mélange d'évènements

Nous avons déjà abordé différentes manières de combiner deux signaux dans Reaktor Core, par exemple avec les opérations arithmétiques. Mais il nous manque toujours un moyen simple de mélanger deux signaux.

Mélanger n'est pas additionner. Mélanger signifie prendre la dernière valeur de tous les signaux, et non les additionner. Pour mélanger les signaux, vous devez utiliser le module *Merge* (*Mélanger* en anglais...). Observons son fonctionnement.

Imaginez un module *Merge* avec deux entrées. La valeur initiale de sortie (avant l'évènement d'initialisation) est bien sûr zéro, comme pour la plupart de modules:



Maintenant, un évènement de valeur 4 arrive à la deuxième entrée du module:



L'évènement traverse le module et apparaît à la sortie. Maintenant, la sortie du module mélangeur a pour valeur 4.

Puis un autre évènement, de valeur 5, arrive à la première entrée:



L'évènement traverse le module et apparaît à la sortie, qui passe donc à la valeur 5.

Maintenant, deux évènements avec les valeurs 2 et 8 arrivent simultanément aux deux entrées:



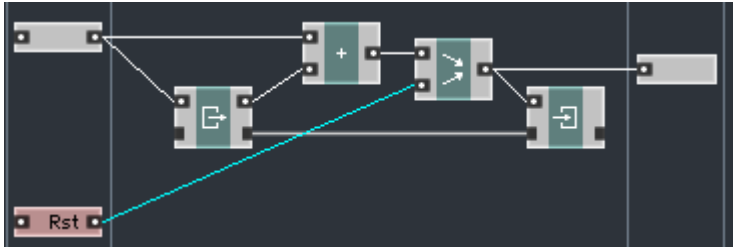
Là, nous devons appliquer au module *Merge* une règle spéciale:

Des évènements simultanés arrivant aux entrées d'un module *Merge* sont traités dans l'ordre de numérotation des entrées. Il n'y a toujours qu'un seul évènement généré en sortie, car une sortie Reaktor Core ne peut produire plusieurs évènements simultanés.

Dans le cas ci-dessus, l'évènement arrivant à la seconde entrée est traité après celui arrivant à la première, "écrasant" la valeur 2 par la valeur 8, qui est transmise (seule) à la sortie.

L'accumulateur d'évènements avec reset et initialisation

Pour réaliser la fonction “reset”, nous devons donc “écraser” la sortie de l'accumulateur par une certaine valeur initiale. Pour ce faire, nous pouvons utiliser le module *Merge*, que l'on trouve dans le sous-menu *Built In Module > Flow*. Un moyen simple est de connecter la seconde entrée du module mélangeur à l'entrée “Rst”.



L'évènement de reset est désormais directement envoyé au module *Merge*, écrasant la sortie de l'additionneur même si un évènement accumulé arrive au même moment. De là, l'évènement est envoyé en sortie ainsi que dans l'état interne de l'accumulateur.

Dans la structure ci-dessus, la valeur arrivant à l'entrée “Rst” est utilisée comme nouvelle valeur par l'accumulateur. Ce n'est peut-être pas une mauvaise idée, mais en tout cas il ne s'agit pas vraiment d'une fonction “reset” mais plutôt d'une fonction “set”, implémentée dans le module accumulateur standard de Reaktor. Si nous voulons une véritable fonction “reset”, nous devons écrire expressément un zéro dans l'état interne, quelle que soit la valeur apparaissant à l'entrée “Rst”. Nous devons donc envoyer une valeur zéro au module *Write* à chaque fois qu'un évènement arrive à l'entrée “Rst”.

Envoyer un évènement avec une valeur particulière en réponse à un évènement entrant est une tâche assez commune dans Reaktor Core, et nous suggérons d'utiliser à cette fin une macro *Latch*, située dans *Expert Macro > Memory > Latch*:



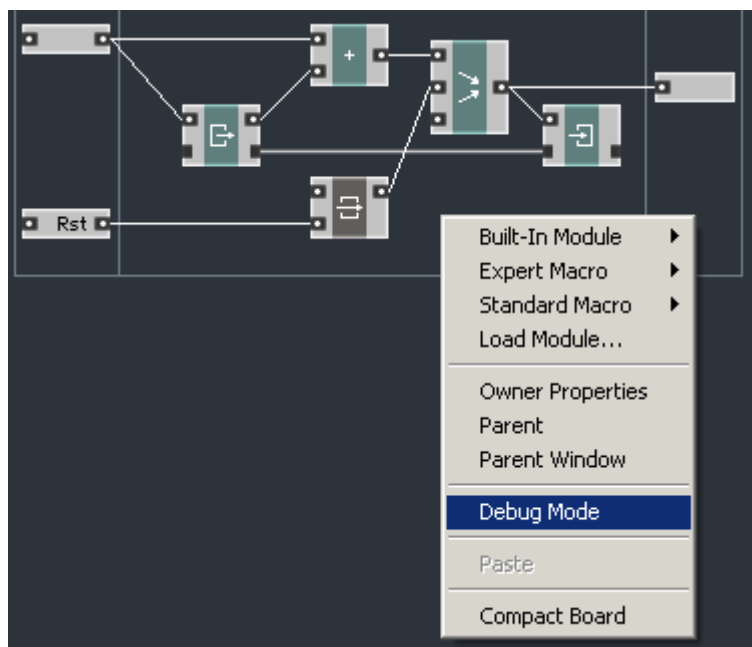
Comme nous l'avons décrit plus haut, le module *Latch* dispose d'une entrée de valeur (en haut) et d'une entrée d'horloge (en bas). Nous souhaitons connecter l'entrée “Rst” à l'entrée d'horloge du *Latch* et une constante nulle à son entrée


Passez en mode *Panel* et observez les valeurs augmentant par pas de 1, toutes les secondes, et redémarrant à chaque pression du bouton.



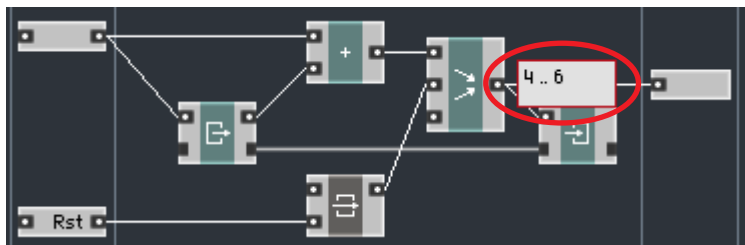
Profitons de cette occasion pour vous présenter le mode de débogage de Reaktor Core. Comme vous l'avez probablement déjà remarqué, contrairement au niveau primaire de Reaktor, les valeurs à la sortie des modules ne s'affichent plus lorsque vous laissez le curseur dessus. C'est un effet secondaire malheureux de l'optimisation interne de Reaktor Core, qui a pour effet que les valeurs à l'intérieur des structures Reaktor Core ne sont pas accessibles depuis l'extérieur.

Comme nous vous entendons déjà gémir, nous avons opté pour un compromis. Vous pouvez désactiver l'optimisation pour une structure core particulière, ce qui vous permet de voir les valeurs de sortie. Essayons avec la structure que nous venons de construire. Faites un clic droit sur le fond et sélectionnez *Debug Mode*:

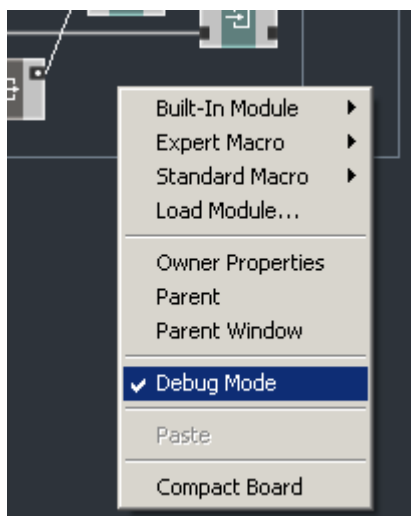


Vous pouvez également activer ce mode via le bouton  dans la barre d'outils.

Maintenant, si vous maintenez votre curseur au-dessus d'une sortie particulière, vous verrez s'afficher la valeur de cette sortie (ou un intervalle de valeurs):

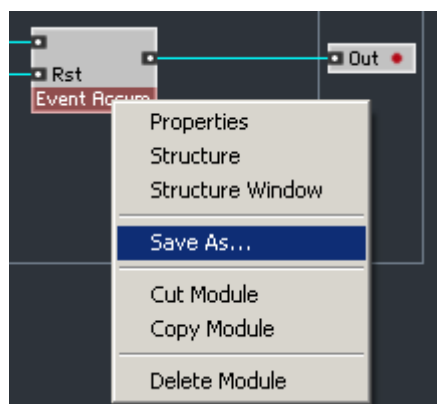


Vous pouvez désactiver le mode de débogage en sélectionnant à nouveau la même commande (ou en appuyant sur le même bouton):

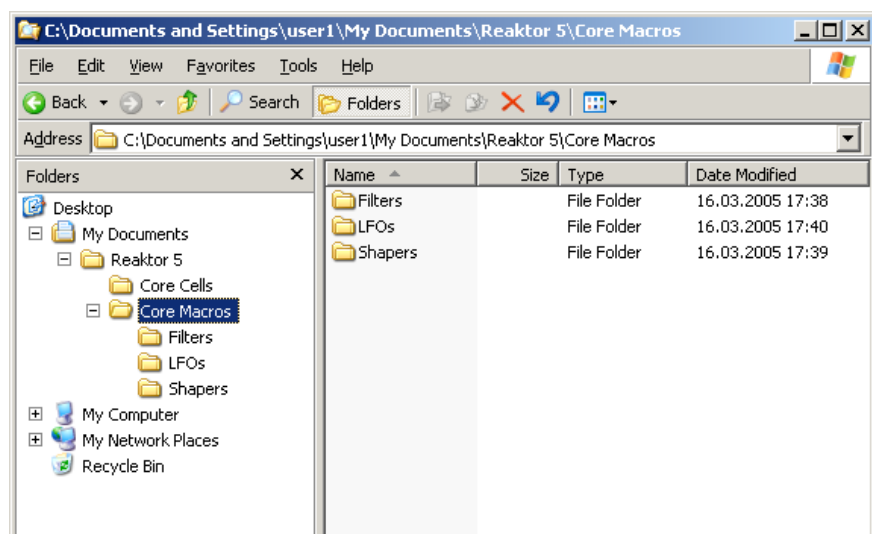


Le mode de débogage est aussi automatiquement désactivé si vous sortez de la structure, vous devrez donc l'activer à nouveau pour toute autre structure.

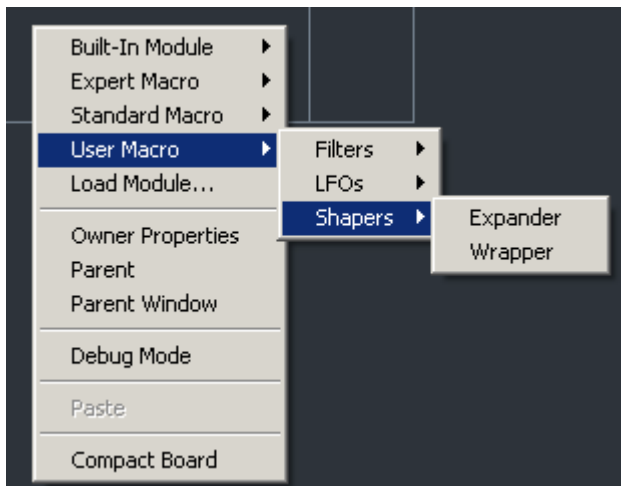
Après avoir “débogué” notre macro, il nous faut penser à l'enregistrer dans un fichier pour un usage ultérieur. Pour ce faire, effectuez un clic droit sur la macro et sélectionnez “Save As...”:



Comme avec les cellules *core*, vous avez la possibilité d'insérer vos propres macros dans le menu. Les macros doivent être placées dans le sous-dossier “*Core Macros*” du dossier utilisateur de Reaktor:



Si des fichiers se trouvent dans ce dossier “*Core Macros*” ou ses sous-dossiers, un nouveau sous-menu apparaît dans le menu contextuel (menu du bouton droit):

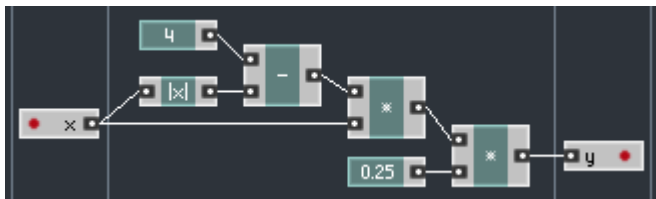


Encore une fois, les mêmes restrictions que pour le dossier “Core Cells” s’appliquent au dossier “Core Macros”:

- les dossiers vides n’apparaissent pas dans le menu,
- ne placez jamais vos propres fichiers dans la librairie système, mais dans la librairie utilisateur.

Réparons le modeleur d’évènements

Nous pouvons maintenant étudier plus en détail ce qui n’allait pas dans le modeleur d’évènements que nous avons construit auparavant:



Le problème vient de l’évènement d’initialisation. Si vous observez comment l’initialisation se propage dans la structure ci-dessus, vous remarquerez les choses suivantes:

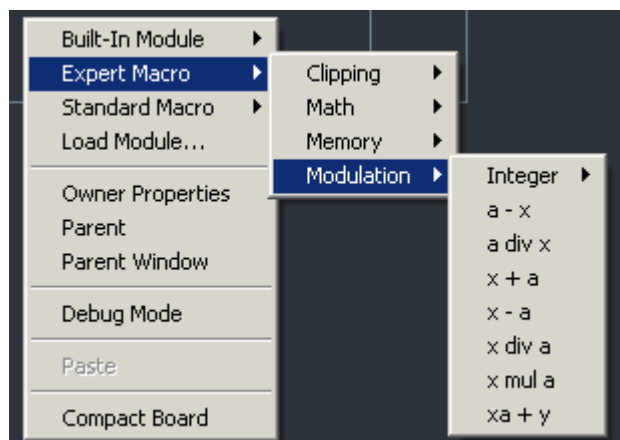
- l’entrée ‘x’ envoie ou n’envoie pas d’évènement d’initialisation, selon qu’elle en reçoit un ou non de la structure primaire à l’extérieur de la structure (c’est la règle générale pour les évènements d’initialisation des entrées évènements des cellules *core*) ;
- les constantes 4 et 0,25 envoient toujours des évènements d’initialisation.

De ce fait, si, pour une raison quelconque, l'évènement d'initialisation ne n'arrive pas à l'entrée du module, sa sortie continuera à recevoir l'évènement issu du dernier multiplicateur et transférera cet évènement à la structure du niveau primaire, à l'extérieur.

Même si ceci n'est pas trop grave pour le traitement du signal (si l'évènement d'initialisation en entrée fait défaut, l'entrée est considérée comme nulle et l'évènement d'initialisation de sortie est tout de même envoyé), ce n'est pas exactement ce que l'on attend intuitivement d'un module de traitement d'évènements. Nous préférons plutôt que le module envoie un évènement en sortie uniquement en réponse à un évènement entrant.

Nous sommes donc confrontés au problème suivant: nos modules de constantes peuvent envoyer des évènements à des moments indus (ie quand il n'y a pas d'évènement en entrée). Comme solution, nous suggérons de remplacer les modules de soustraction et de multiplication, qui ont des constantes en entrée, par leurs homologues à *Modulation*.

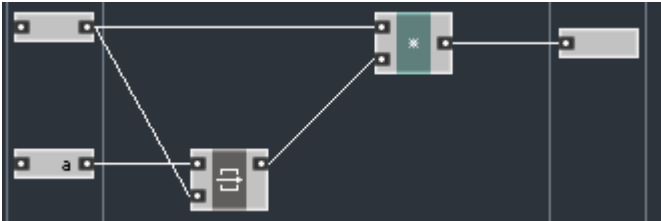
Les macros à "Modulation" sont situées dans la librairie de Reaktor Core dans *Expert Module > Modulation*:



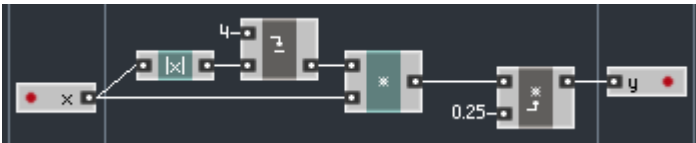
L'appellation "Modulation", même si elle n'est pas correcte à 100 %, reflète pourtant leur fonction, qui est d'utiliser un signal pour en moduler un autre (nous le verrons clairement plus tard, lorsque nous utiliserons des signaux de contrôle pour moduler des signaux audio dans des structures de bas niveau). La plupart de ces macros combinent deux signaux, un "porteur" et un "modulateur" (un peu comme dans la transmission radio). Contrairement aux modules *core* arithmétiques d'usine, les "macros à modulation" génèrent un

événement de sortie uniquement en réponse à un événement à l'entrée du signal "porteur". Les événements à l'entrée "modulateur" ne relancent *pas* le processus de calcul.

L'implémentation interne des macros à modulation est très simple: elles retiennent juste le signal modulateur via un *Latch*, celui-ci étant régulé par le signal porteur. Voici l'exemple de la structure interne du "multiplicateur à modulation" (l'entrée appelée "a" est le modulateur):



Nous remplaçons donc le module de soustraction par la macro à modulation $a - x$ ainsi que le dernier module multiplicateur par la macro à modulation $x \cdot mul\ a$. Voici à quoi ressemble notre structure après ces modifications (nous avons également remplacé les modules *Const* par des *QuickConst*, mais c'est sans importance):



Généralement, dans les macros à modulation, vous pouvez reconnaître l'entrée du modulateur grâce à l'icône de la macro (le petit dessin sur le module): la flèche de l'icône indique la position de l'entrée du modulateur. Dans le cas du module de soustraction, la flèche est en haut, donc l'entrée de modulation aussi. Pour le module de multiplication, c'est l'inverse. Notez également que la sortie de ces modules est située en face de l'entrée du porteur, ce qui est un autre indice. Enfin, vous pouvez placer votre curseur sur les modules et leurs entrées et lire les informations qui s'affichent.

Dans la structure ci-dessus, aucun événement ne sera envoyé, à moins qu'un événement n'arrive à l'entrée de la cellule *core*:

- le module " $|x|$ " est directement commandé par l'évènement d'entrée de la cellule *core* ;
- le module de soustraction consécutif est uniquement commandé par

la sortie du module “|x|”, qui envoie un évènement uniquement en réponse à un évènement d'entrée, la constante *QuickConst* n'ayant pas d'effet de commande ;

- le premier multiplicateur est commandé soit par la sortie du module de soustraction, soit par l'évènement d'entrée de la cellule *core*, mais nous avons déjà vu que les deux ne peuvent survenir que simultanément ;
- le second multiplicateur est commandé uniquement par l'évènement entrant et non par la constante *QuickConst*.

Notre structure se comporte désormais de façon un peu plus intuitive.

Le traitement de l'audio en détail

Les signaux audio

Dans Reaktor Core, les signaux audio ne sont pas d'un type particulier: l'audio est aussi un évènement, qui ne diffère en rien des autres évènements du point de vue de la structure. La seule différence est que, pour les signaux audio, les évènements sont produits “à des instants précis régulièrement espacés” correspondant au taux d'échantillonnage.

Pour produire des évènements régulièrement espacés (ou tout autre suite d'évènements d'ailleurs), nous avons besoin d'une source d'évènements. Comme pour les cellules *core évènements*, les entrées des cellules *core audio* sont aussi des sources d'évènements. Mais nous avons maintenant un type d'entrée supplémentaire:

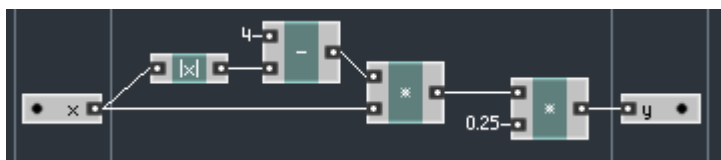
Le **Entrées Audio** envoient régulièrement des évènements *core* à l'intérieur de la structure, à un rythme déterminé par le taux d'échantillonnage utilisé à l'extérieur, dans la structure du niveau primaire. Les évènements sont envoyés simultanément depuis toutes les entrées audio de la cellule *core*.

Les entrées audio envoient également l'évènement d'initialisation dans la structure de la cellule *core*. Cet évènement est envoyé, quoi qu'il arrive dans la structure de niveau primaire. Mais la valeur envoyée par ces entrées pendant l'initialisation dépend, elle, du processus d'initialisation extérieur (dans la structure primaire).

Il existe aussi un nouveau type de sortie, utilisé à la place des sorties évènements:

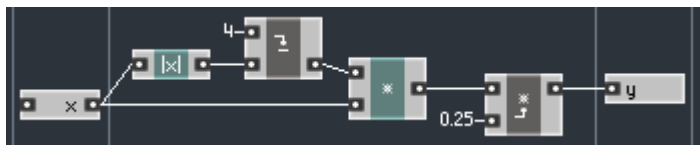
Les **Sorties Audio** envoient dans la structure extérieure de niveau primaire la dernière valeur reçue de la structure *core* intérieure. Comme les sorties audio du niveau primaire n'envoient pas d'évènements, aucun évènement n'est envoyé vers l'extérieur.

Nous allons maintenant essayer de reconstruire le même modeler que précédemment, mais en mode audio au lieu du mode évènement. Globalement, nous pouvons utiliser exactement la même structure, en remplaçant simplement les entrées et sorties évènements par leurs équivalentes audio:

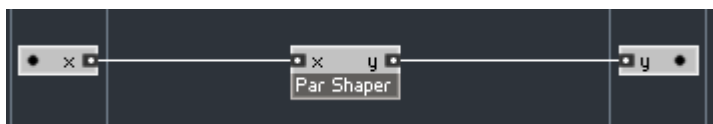


Vous pourriez vous demander pourquoi nous n'avons pas utilisé ici les macros à modulation. La raison en est que nous manipulons maintenant des signaux audio. Ceux-ci envoient toujours un évènement d'initialisation, nous sommes donc sûrs de sa présence aux entrées. Mais vous êtes libre d'utiliser des macros à modulation si vous préférez.

Nous aurions aussi pu essayer d'insérer la structure ci-dessus dans une macro pour pouvoir la réutiliser dans d'autres structures Reaktor Core, pour le traitement des évènements comme pour celui de l'audio. Dans ce cas, il est conseillé d'utiliser ces fameuses macros à modulation, car il n'est plus possible de savoir à l'avance quel type de signal sera traité par le module:

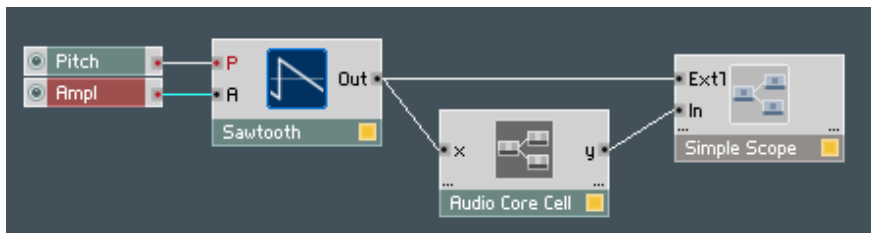


Et voici la structure interne de la cellule *core* audio pour ce cas précis:



Pour la tester, nous allons la connecter à un oscillateur en dents de scie ainsi qu'à un oscilloscope. Vous trouverez un oscilloscope dans *Insert Macro*

> *Classic Modular* > *OO Classic Modular – Display* > *Simple Scope* (dans la structure primaire). N'oubliez pas de vous assurer que le nombre de voix est bien réglé sur 1.



Nous utilisons le contrôle externe de l'oscilloscope pour une meilleure synchronisation aux taux élevés de distorsion (le bouton *Ext* du panneau de l'oscilloscope doit être activé). Modifiez l'intervalle du potentiomètre *Ampl* à environ [0.. 5] pour pouvoir observer le modelage.



Le bus d'horloge du taux d'échantillonnage

Nous devons encore aborder au moins deux points pour pouvoir construire des structures audio. Le premier est la création des cellules core audio sans entrées audio. Bien sûr, nous pouvons les créer, mais comment faire pour obtenir une source d'évènements audio ? Le second point est que de nombreux algorithmes de traitement numérique du signal (DSP) nécessitent la connaissance du taux d'échantillonnage actuel. Nous allons ici traiter de ces deux sujets.

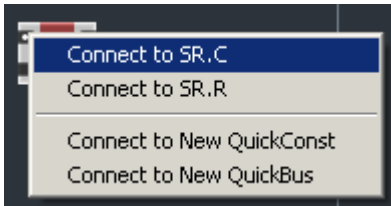
Chaque structure Reaktor Core possède une connexion spéciale, appelée "sampling rate clock bus" ("bus d'horloge du taux d'échantillonnage"). Ce

bus (ou canal de transmission) transporte deux signaux: ceux de l'horloge et du taux d'échantillonnage (en anglais "clock" et "rate").

Clock est une source de signal dans laquelle des évènements sont régulièrement envoyés au taux d'échantillonnage audio. Comme avec tous les signaux audio standard, cette source envoie un évènement d'initialisation. Les évènements du signal d'horloge ont toujours une valeur nulle, mais les structures utilisant ce signal doivent ignorer ses valeurs car elles pourront être amenées à évoluer dans le futur.

Rate est une source de signal dont les valeurs sont toujours égales au taux d'échantillonnage actuel, en Hertz. Les évènements sont envoyés depuis cette source lors de l'initialisation ou lors des modifications du taux d'échantillonnage.

Vous accédez au bus du taux d'échantillonnage via un clic droit sur n'importe quelle entrée de signal et en sélectionnant *Connect to SR.C* pour le signal d'horloge, ou *Connect to SR.R* pour le signal de taux d'échantillonnage.



La connexion s'affiche à côté de l'entrée:



Le bus d'horloge du taux d'échantillonnage ne fonctionne pas à l'intérieur des cellules *core* évènements.

Réinjection d'une connexion

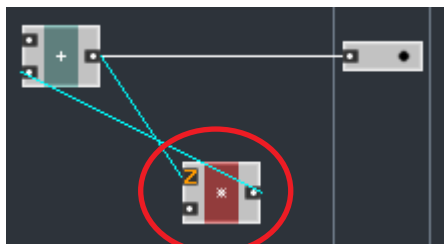
Comme nous l'avons déjà vu, les règles concernant l'ordre de traitement ne peuvent s'appliquer si la structure contient une ou plusieurs réinjections. C'est pourquoi nous avons besoin de règles supplémentaires qui définissent le traitement des réinjections.

La règle principale est: Les structures Reaktor Core ne savent pas manipuler la réinjection.

Non, pas tout à fait comme ça. Vous pouvez effectuer des connexions de réinjection dans Reaktor Core. Mais comme le moteur de Reaktor Core ne sait pas manipuler les structures avec réinjection, il va les résoudre. Résoudre une réinjection signifie que votre structure est modifiée (en interne, vous ne verrez rien à l'écran) de façon à supprimer la réinjection.

Si jamais vous ne le saviez pas, la réinjection sans délai n'est pas pensable dans le monde numérique. Il doit y avoir un délai d'au moins un échantillon dans le chemin de réinjection. Et c'est justement ce que Reaktor Core va faire pendant la résolution de la réinjection: le moteur va introduire un module de délai d'une durée égale à un échantillon (Z^{-1}) dans le chemin de réinjection.

Comme vous le savez déjà, les ports où les délais implicites ont été insérés par Reaktor Core sont indiqués par un grand Z orange:

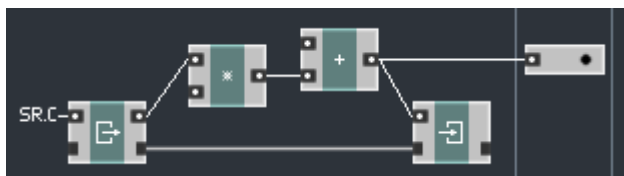


Nous avons déjà vu une structure basée sur un module *Read* et un module *Write* réalisant cette fonctionnalité Z^{-1} . Essayons de placer une telle construction de *Read* et *Write* dans notre structure. Nous allons la placer sur le câble de la réinjection:



D'abord on écrit, puis on lit (notez que le module *Read* est piloté par l'horloge

SR.C pour s'assurer que la lecture est effectuée une fois par cadence audio). Ainsi, la valeur lue a toujours un échantillon de décalage avec la valeur écrite. Et il n'y a pas de réinjection dans la structure. Vous n'en êtes pas sûr(e) ? D'accord, déplaçons un peu les modules (sans toucher aux connexions):



Vous êtes d'accord maintenant ? Évidemment !

En insérant un module Z^{-1} , on supprime formellement la réinjection de la structure, tout en la conservant, logiquement parlant (avec un délai d'un échantillon audio).

En fait, la structure interne d'une macro Z^{-1} est un peu plus compliquée qu'une simple paire de *Read* et *Write*. Nous verrons ceci plus en détail dans la section suivante.

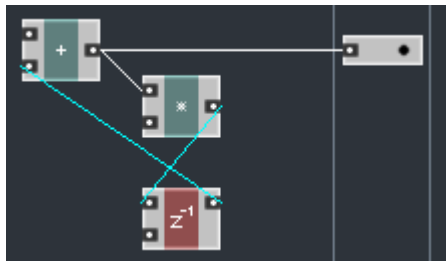
Vous n'avez aucune influence sur l'endroit où la résolution automatique de réinjection va être insérée. Elle est insérée sur n'importe quel câble de la boucle de réinjection. Il n'est même pas garanti que cette résolution soit toujours placée sur le même câble: elle peut changer dans la prochaine version du logiciel, ou en réponse à une autre modification quelque part dans la structure, elle peut même être différente la prochaine fois que vous chargerez cette structure depuis le disque dur.

Cette résolution automatique de la réinjection est conçue pour les structures pour lesquelles l'endroit exact de la résolution n'a pas d'importance. En particulier, ceci est bien pratique pour les utilisateurs pas vraiment calés en traitement numérique de signal et ayant du mal à envisager de telles problématiques. La résolution automatique de réinjection leur permet d'obtenir des résultats satisfaisants.

Si vous avez besoin d'un contrôle précis sur les points de résolution de réinjection, vous pouvez insérer explicitement des modules Z^{-1} dans vos structures. Ces modules élimineront formellement les réinjections et aucune résolution interne ne sera nécessaire.

Voici une version de la structure ci-dessus, avec une macro Z^{-1} insérée (elle

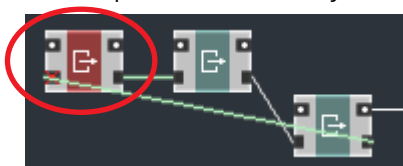
se trouve dans le sous-menu *Expert Macro > Memory*):



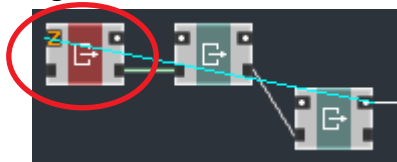
Comme vous pouvez le voir, le gros Z orange a disparu, et vous pouvez également constater que le point de délai d'un échantillon n'est plus au même endroit que celui qui avait été automatiquement inséré: ce dernier était sur le câble allant de la sortie de l'additionneur à l'entrée du multiplicateur, alors qu'il est maintenant sur le câble entre la sortie du multiplicateur et l'entrée de l'additionneur.

La raison d'être de la seconde entrée du module Z^{-1} sera expliquée plus loin. Vous la laisserez généralement déconnectée.

La réinjection sur les connexions OBC et sur tous les autres types de connexions "non-signal" (introduites plus loin) n'a pas de sens, c'est pourquoi elle n'est pas permise. Si une boucle de réinjection sans câble de signal apparaît, une des connexions est indiquée comme invalide et considérée comme absente. La marque "invalide" est symbolisée par une croix rouge sur le port:



En revanche, les boucles de réinjection contenant des types "mixes" de connexions, ie avec des câbles transportant du signal et d'autres types de câbles, sont tout-à-fait acceptées et sont résolues de la manière habituelle, la résolution étant insérée sur l'un des câbles "normaux" (ie transportant du signal):



En fait, cela signifie que les connexions non-signal ne sont pas affectées par la résolution de réinjection, à moins de faire une boucle de réinjection entièrement sans signal, ce qui n'est pas palpitant.

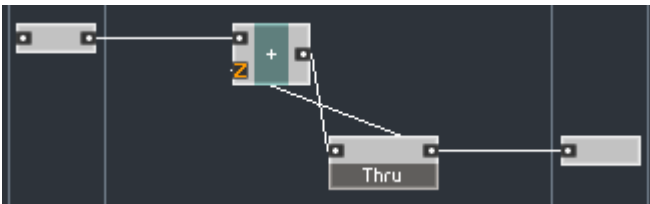
Réinjection et macros

De manière générale, en ce qui concerne la résolution des réinjections, les macros sont traitées comme les modules d'usine.

Considérons une macro qui transfère juste le signal d'entrée en sortie. Voici sa structure interne:



Supposons maintenant que nous construisons une structure avec réinjection en utilisant cette macro:



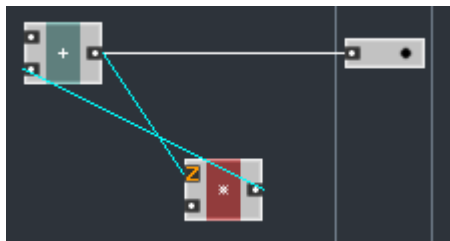
La boucle de réinjection passe par deux câbles dans la structure ci-dessus, ainsi que par un câble à l'intérieur de la macro. Où la résolution va-t-elle se produire ? (d'accord, on peut voir sur la figure ci-dessus qu'elle se produit à l'entrée de l'additionneur *dans ce cas particulier*, mais nous savons qu'elle aurait pu se produire ailleurs).

Imaginez un instant que *Thru* ne soit pas une macro mais un module d'usine. Dans ce cas, il est évident que la résolution de la réinjection ne peut pas apparaître dans ce module, mais uniquement à l'extérieur.

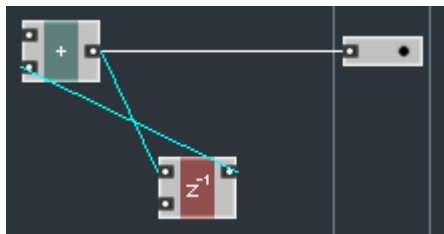
Nous nous efforçons de rendre les macros aussi semblables aux modules d'usine que possible. Pour cette raison, *par défaut*, la résolution d'une telle boucle de réinjection aura lieu à l'extérieur de la macro. L'endroit exact n'est pas spécifié, mais on sait que ce sera à l'extérieur de la macro *Thru*.

La règle générale est la suivante: la résolution de la réinjection apparaît dans le niveau de structure le plus élevé de la boucle de réinjection.

Vous pouvez cependant modifier ce comportement par défaut, c'est-à-dire autoriser la résolution de réinjection à l'intérieur des macros. En fait, vous auriez déjà pu vous demander, si les macros sont vraiment traitées comme les modules d'usine, comment une macro Z^{-1} peut-elle résoudre une réinjection. En effet, considérons la structure suivante:

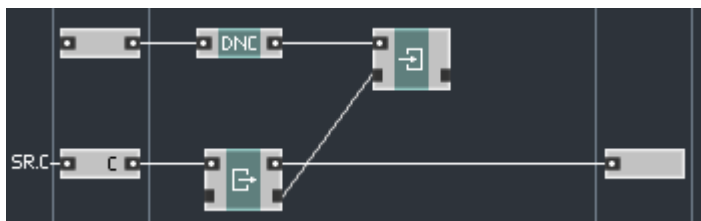


Si les macros et les modules d'usine étaient équivalents, rien ne devrait changer lorsque l'on remplace le multiplicateur par une macro Z^{-1} :



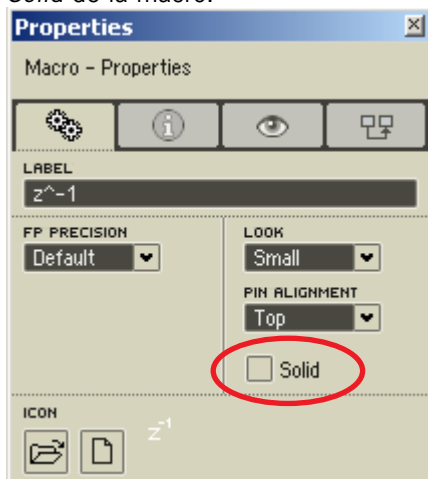
Mais cette structure *est* différente, car la réinjection implicite a disparu. Cette macro Z^{-1} doit bien avoir quelque chose de spécial... et en effet, elle a quelque chose de spécial.

Si nous regardons à l'intérieur de cette macro, nous y voyons presque la même structure que celle de notre propre implémentation de la fonction Z^{-1} (cf. plus haut):



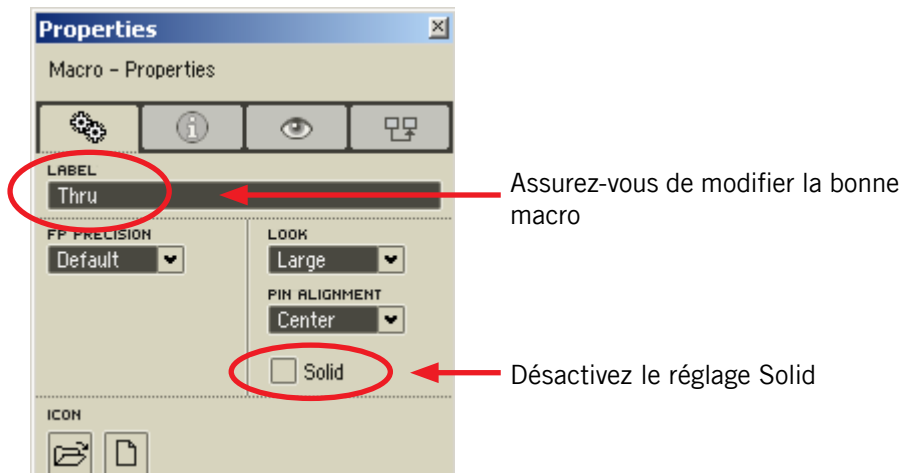
Comme vous pouvez le constater, l'entrée d'horloge de la macro est connectée au module *Read* interne. La connexion par défaut pour cette entrée n'est pas une constante nulle mais *l'horloge audio*, car c'est ce dont vous avez besoin dans la plupart des cas. Le module connecté entre l'entrée supérieure et le module *Write* sera expliqué plus loin. Nous pouvons l'ignorer pour l'instant.

Jusque là, rien de spécial dans cette structure, si ce n'est qu'elle semble réaliser la fonction Z^{-1} dont nous avons déjà parlé. Pourtant, comment le moteur de Reaktor Core sait-il que cette structure sert à résoudre les boucles de réinjection ? Bien sûr, le moteur sait que cette structure *peut* résoudre les boucles de réinjection, mais quant à savoir si elle est *prévue* pour ? Ceci est géré par les propriétés de la macro, et plus particulièrement par le réglage *Solid* de la macro:

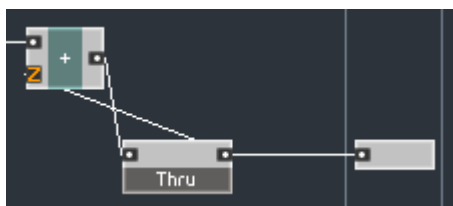


Cette propriété dit au moteur de Reaktor Core si la macro doit être considérée comme un module d'usine "solide" pour la résolution de réinjection, ou si elle doit être considérée comme transparente. **Dans 99 % des cas, vous conserverez cette propriété activée.** En effet, vous ne souhaitez généralement pas que la résolution de réinjection apparaisse à l'intérieur de vos macros.

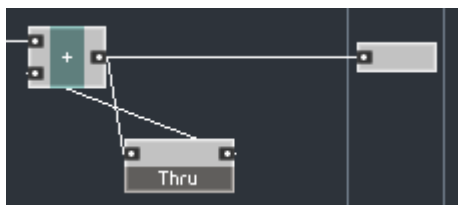
Une bonne raison à cela est que la résolution se produisant à l'intérieur d'une macro passe inaperçue, à moins d'ouvrir la structure interne de cette macro, d'où le risque que certains délais implicites de réinjection échappent à votre attention. Par exemple, si nous prenons notre structure avec la macro Thru, et si nous désactivons le réglage *Solid* (assurez-vous de modifier ce réglage pour la bonne macro, vous pouvez le vérifier via le texte "Thru" apparaissant dans le champ de nom des propriétés):



Votre structure externe est probablement toujours la même (nous disons "probablement", car on ne sait jamais où se trouve la résolution de réinjection):



Mais si vous modifiez votre structure en connectant la sortie à un autre module, elle pourrait bien ressembler à ceci:



Notre délai de résolution de réinjection semble avoir disparu. Dans une structure plus grande et plus complexe, nous pourrions facilement oublier qu'il y a un délai implicite. Où est passé le délai ? Bien sûr, il est maintenant dans la macro, le seul endroit que nous ne pouvons voir de l'extérieur:



Une deuxième bonne raison pour conserver la propriété Solid activée est que, dans certains cas, si vous la désactivez, la fonction de la macro peut être modifiée une fois la macro placée dans la boucle de réinjection. Bref, facilitez-vous la vie et ne désactivez cette propriété que si vous construisez des macros spécifiquement conçues pour résoudre les réinjections... ce qui ne devrait pas arriver tous les quatre matins !

Revenons à notre module Z^{-1} . Comme la propriété Solid est désactivée pour cette macro, les limites de cette macro sont complètement transparentes pour la résolution de la réinjection. Ainsi, la macro Z^{-1} n'est pas vraiment traitée comme un module d'usine et elle peut résoudre la réinjection de la manière décrite plus haut dans ce texte.

Les valeurs dénormales

Dans les structures que nous avons construites jusqu'à présent, à l'intérieur de l'ordinateur, les valeurs des signaux sont représentées par des données binaires d'un type particulier, appelées les *nombres à virgule flottante*, en bref les *flottants*. Ce type de données permet une représentation efficace d'un grand éventail de valeurs.

Le terme *nombres à virgule flottante* n'indique pas exactement comment les nombres sont représentés. Cette appellation décrit simplement l'approche utilisée pour les représenter, laissant une grande marge de manœuvre quant à leur implémentation.

Les processeurs équipant les ordinateurs personnels d'aujourd'hui utilisent la norme standard de virgule flottante de l'IEEE (Institute of Electrical and Electronics Engineers). Ce standard définit exactement comment les nombres à virgule flottante doivent être représentés et ce que doivent être les résultats des opérations entre eux (p.ex. comment manipuler les limitations de précision, etc.). En particulier, ce standard dit que, pour un groupe de valeurs à virgule flottante particulièrement petites qui, pour des raisons de précision limitée, ne peuvent être représentées correctement de la façon "normale", un autre type de représentation doit être utilisée. Cette autre représentation est appelée "dénormale" (traduction directe de l'anglais "denormal", terme utilisé uniquement dans cette acception et ce contexte – NdT).

Pour une virgule flottante en 32-bit, l'intervalle des valeurs "dénormales" est approximativement de 10^{-38} à 10^{-45} et de -10^{-38} à -10^{-45} . Les valeurs plus petites que 10^{-45} en valeur absolue ne peuvent être représentées et sont considérées comme nulles.

Comme ces nombres ont une représentation quelque peu différente de celle des nombres "normaux", certains processeurs ont des difficultés à les manipuler. En particulier, les opérations sur ces nombres peuvent être parfois effectuées beaucoup, beaucoup plus lentement (dix fois, voire pire...) que les opérations sur les nombres "normaux".

Une situation classique dans laquelle les nombres dénormaux apparaissent pour des *durées prolongées* est celle des valeurs exponentiellement décroissantes, comme dans les filtres, certaines enveloppes et autres structures de réinjection. Dans de telles structures, après que le signal d'entrée est revenu à zéro, les signaux décroissent *asymptotiquement* vers zéro. *Asymptotiquement* signifie que ces signaux se rapprochent indéfiniment du zéro, sans jamais l'atteindre. Dans cette situation, les nombres dénormaux peuvent apparaître et rester dans la structure pendant un certain temps, entraînant une augmentation significative de l'utilisation du processeur.

Les nombres dénormaux peuvent également apparaître lorsque vous modifiez la précision d'un flottant d'une valeur élevée (64-bit) à une valeur plus faible (32-bit). En effet, une valeur de 10^{-41} n'est pas dénormale lorsqu'elle a une précision de 64 bits, mais le devient avec 32 bits (le changement de précision des flottants sera discutée plus loin).

Considérons la modélisation d'un filtre passe-bas analogique à 1 pôle avec une fréquence de coupure fixée à 20 Hz. Nos valeurs de signal numérique correspondent aux tensions analogiques en volts. Imaginons que le niveau du signal d'entrée est égal à 1 V (volt) sur une durée assez longue. La tension à la sortie du filtre est aussi égale à 1 V. Maintenant, nous passons brusquement la tension d'entrée à zéro. La tension de sortie va décroître selon la loi:

$$V_{out} = V_0 e^{-2\pi f_c t}$$

où f_c est la fréquence de coupure du filtre (en Hertz), t est le temps (en secondes) et la tension initiale.

La tension de sortie va évoluer comme suit:

après 0,5 sec	$V_{out} \approx 10^{-29}$ volt
après 0,6 sec	$V_{out} \approx 10^{-33}$ volt
après 0,7 sec	$V_{out} \approx 10^{-38}$ volt
après 0,8 sec	$V_{out} \approx 10^{-44}$ volt

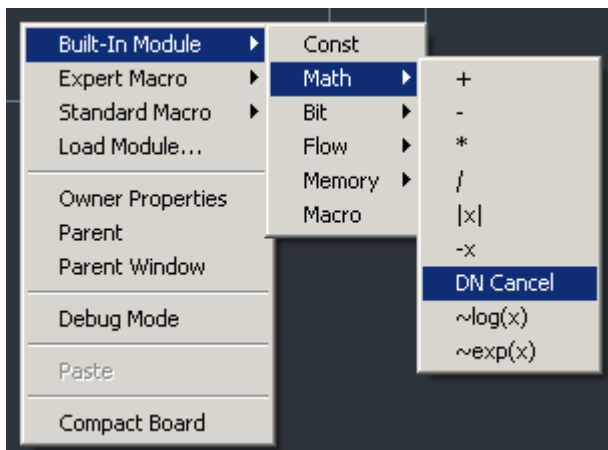
Oups... les nombres entre 10^{-38} et 10^{-45} sont dans l'intervalle des valeurs dénormales. Donc, entre 0,7 et 0,8 secondes à peu près, notre tension est représentée par une valeur dénormale. Et ce pas seulement dans le filtre: la sortie du filtre est probablement traitée plus loin par la structure en aval, ce qui oblige aussi les modules suivants à traiter ces valeurs dénormales.

Au taux d'échantillonnage de 44,1 kHz, l'intervalle de temps de 0,1 seconde correspond à 4410 échantillons. En supposant que la taille de tampon ASIO classique est de quelques centaines d'échantillons, nous devons produire de nombreux tampons réclamant une grosse utilisation du processeur (puisque tous ces tampons sont remplis de valeurs dénormales). Si le traitement de ces tampons nécessite une utilisation processeur de près de 100 % (ou plus), des clics audio apparaissent.

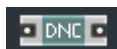
De ces considérations, vous devez tirer l'enseignement suivant: les valeurs dénormales sont très mauvaises pour l'audio en temps réel...

Les modules du niveau primaire de Reaktor sont programmés pour surveiller les valeurs dénormales susceptibles de survenir en leur sein. Pour ce faire, les algorithmes de DSP (traitement numérique du signal) ont été modifiés afin qu'ils produisent le moins de valeurs dénormales possible. Si vous concevez vos propres structures de bas niveau dans Reaktor Core, vous devez aussi

tenir compte de ces valeurs dénormales. Pour vous aider dans cette tâche, nous avons conçu le module *Denormal Cancel*, disponible dans le sous-menu *Built In Module > Math*:



Le module *Denormal Cancel* dispose d'une entrée et d'une sortie. Il essaie de modifier légèrement la valeurs entrante pour éviter qu'une valeur dénormale n'arrive à la sortie:



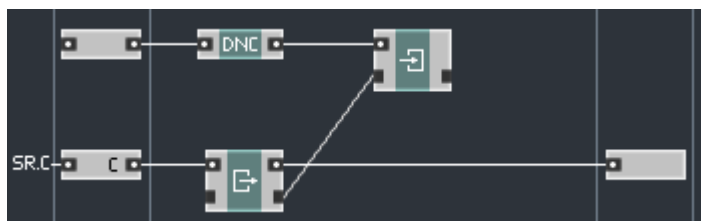
La manière dont ce module modifie le signal n'est pas fixe, elle peut varier d'une version de l'application à l'autre, voire d'un endroit à l'autre dans une structure. Actuellement, cette opération est effectuée en additionnant une toute petite constante à la valeur d'entrée. À cause de la limitation en précision, cette addition ne modifie pas les valeurs suffisamment grandes (les valeurs plus grandes que 10^{-10} ne sont pas modifiées du tout). À cause de cette même limitation en précision, il est aussi très improbable que le résultat de l'addition soit une valeur dénormale (dans la plupart des cas, c'est même impossible).

Si, pour quelque raison que ce soit, le module Denormal Cancel ne fonctionne pas pour votre structure, vous êtes évidemment libre d'utiliser vos propres techniques d'élimination des valeurs dénormales. Mais une technique fonctionnant sur une plateforme peut ne pas fonctionner sur une autre. Notre algorithme DN Cancel s'adapte à chaque plateforme gérée. Il est donc conseillé d'utiliser le module DN Cancel autant que faire se peut. Nous allons même réfléchir à la construction d'algorithmes alternatifs dans ce module – vos contributions sur notre forum sont les bienvenues !

Certains processeurs proposent l'option de contourner le standard IEEE en désactivant la production de nombres dénormaux, forçant le résultat à valoir zéro. Cette option est parfois disponible, parfois non. Comme les structures Reaktor Core sont censées être indépendantes de la plateforme, il est fortement recommandé de vous occuper vous-même de l'élimination des valeurs dénormales dans vos structures, même si votre système particulier s'en occupe déjà.

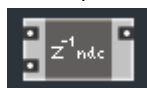
Comme les situations les plus classiques d'apparition de nombres dénormaux sont les boucles de réinjection exponentiellement décroissantes, et comme en traitement du son la plupart des boucles de réinjection décroissent exponentiellement (p.ex. les filtres et les structures à réinjection avec retards), nous avons décidé d'insérer l'élimination des valeurs dénormales dans la macro standard Z^{-1} .

Comme vous vous en souvenez, l'intérieur de cette macro ressemble à ceci:



Vous pouvez maintenant expliquer à quoi sert le module *Denormal Cancel* (*DNC*). Comme vous allez souvent utiliser la macro Z^{-1} dans des structures à réinjection, il y a une forte probabilité que des valeurs dénormales y apparaissent. C'est pourquoi nous avons décidé de mettre ce module *DNC* directement dans la structure de la macro Z^{-1} .

Il existe une autre version de cette macro, nommée $Z^{-1} \text{ ndc}$, qui n'effectue pas cette élimination des valeurs dénormales (*ndc* pour *no denormal cancel*). Vous pouvez l'utiliser dans les structures pour lesquelles vous êtes certain(e) qu'elles ne produisent pas de valeurs dénormales (par exemple les filtres FIR – à Réponse Impulsionnelle Finie):



Les autres mauvais nombres

Les nombres dénormaux ne sont pas les seuls nombres méchants qui peuvent s'infiltrer dans vos structures à état interne, en particulier les boucles de réinjection. Il existe d'autres types de mauvais nombres: les INF, les NaN et les QNaN. Nous n'allons pas détailler chacun d'eux, une somme conséquente d'informations les concernant est déjà disponible, notamment sur Internet. L'important pour nous est d'éviter que ces nombres n'apparaissent dans nos structures.

Généralement, de telles valeurs sont le résultat d'opérations invalides. Une division par zéro est le cas le plus évident. D'autres cas entraînent l'apparition de nombres trop grands pour être représentés par des virgules flottantes (ceux au-dessus de 10^{-38} en valeur absolue), ou allant au-delà de l'intervalle raisonnable pour une opération particulière.

De tels nombres ont tendance à se répandre dans les structures, plus encore que les nombres dénormaux. Par exemple, dès que vous ajoutez une valeur normale à une valeur dénormale, le résultat sera non-dénormal (bref normal), à moins que la deuxième valeur soit presque dénormale. En revanche, si vous ajoutez une valeur normale à INF, le résultat sera toujours INF.

En plus de cette tendance à se répandre pour toujours dans les structures (... jusqu'au prochain reset de la structure), ces nombres ont la fâcheuse habitude d'utiliser beaucoup plus de temps de calcul sur certains processeurs. C'est pourquoi il nous faut tout faire pour tenter de prévenir leur apparition.

"Tout faire" signifie par exemple de vérifier, à chaque fois que vous divisez un nombre par un autre, qu'aucune division par zéro ne puisse survenir. La phase d'initialisation requiert une attention toute particulière. Considérons par exemple l'élément de structure suivant:



Si, pour une raison quelconque, l'évènement d'initialisation n'arrive pas à l'entrée inférieure du module diviseur, il en résulte une division par zéro lors du processus d'initialisation. Dans ce cas, il peut être utile d'utiliser à la place une macro à modulation, ou carrément de trouver une autre solution, selon vos besoins.

Construction d'un filtre passe-bas à 1 pôle

Un filtre passe-bas simple à 1 pôle peut être construit en utilisant l'équation:

$$y = b * x + (1 - b) * y_{-1}$$

où

x est l'échantillon d'entrée,

y est le nouvel échantillon de sortie,

y_{-1} est l'échantillon de sortie précédent et

b est le coefficient définissant la fréquence de coupure du filtre.

La valeur du coefficient b peut être prise égale à la fréquence de coupure cyclique normalisée, qui peut être calculée à partir de la formule:

$$F_c = 2 * \pi * f_c / f_{SR}$$

où

f_c est la fréquence de coupure désirée (en Hertz),

f_{SR} est le taux d'échantillonnage (en Hertz),

π est 3,14159...

F_c est la fréquence de coupure cyclique normalisée (en radians pour les curieux).

En fait, le coefficient b est égal seulement approximativement à la fréquence de coupure normalisée, l'erreur augmentant avec les hautes fréquences de coupure. Mais, pour nos besoins, cela devrait faire l'affaire, en particulier si nous n'avons pas besoin d'un réglage précis de la fréquence de coupure de notre filtre.

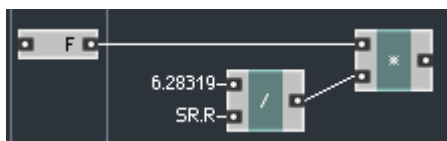
Commençons par créer une cellule core audio avec deux entrées: l'une pour l'entrée audio et l'autre pour la fréquence de coupure. Dans cette version du module, nous allons utiliser une entrée événement pour la fréquence de coupure.



Prenons la bonne habitude de construire nos structures Reaktor Core en tant que macros core, pour pouvoir les réutiliser plus facilement. Nous allons aussi créer notre filtre en tant que macro. Créons donc une macro à l'intérieur de la cellule et créons les mêmes entrées pour la macro:



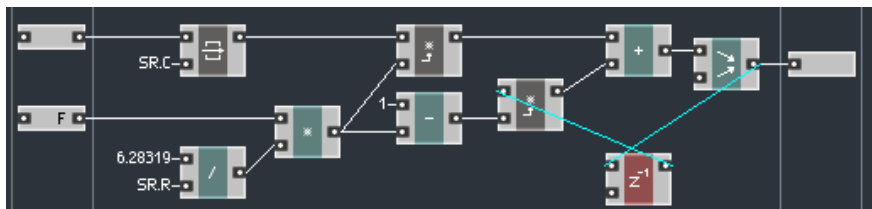
Construisons maintenant la circuiterie pour convertir la fréquence de coupure en coupure cyclique normalisée:



6.28319 est le $2*\pi$, qui est ensuite divisé par le taux d'échantillonnage pour former la valeur à multiplier avec la fréquence de coupure. Nous n'avons pas besoin d'un multiplicateur "à modulation", car "F" est logiquement une entrée de signal de contrôle, nous effectuerons donc la multiplication initiale même s'il n'y a pas d'évènement d'initialisation à l'entrée "F".

Nous effectuons la division avant la multiplication car la division est une opération relativement lourde, et le taux d'échantillonnage ne change pas si souvent que cela. Si seule la fréquence de coupure change, aucun évènement n'est envoyé au module diviseur et ainsi la division ne doit pas être à nouveau effectuée. Ceci est un exemple d'optimisation standard pouvant être réalisée par le concepteur de la structure *core*.

Construisons la circuiterie réalisant l'équation du filtre:



Au cas où des évènements asynchrones arriveraient à l'horloge audio standard, l'entrée audio est contrôlée par un *Latch*. Ceci ne serait pas nécessaire dans une structure de cellule *core* pour laquelle nous savons que l'entrée audio envoie les évènements aux instants corrects. Mais, de façon générale, cette pratique reste toujours une bonne idée.

On utilise deux multiplicateurs à modulation pour empêcher que l'entrée "F" (qui peut, dans l'absolu, changer de valeur à tout instant) ne lance le calcul dans la boucle de réinjection. L'appellation "macros à modulation" devrait maintenant être plus claire: ici, le signal issu de la fréquence de coupure sert à moduler les gains de la boucle de réinjection.

L'utilisation du *Latch* est une technique standard de Reaktor Core pour s'assurer que les événements entrants ne relancent pas le calcul de façon intempestive. Elle est également largement employée par les macros à modulation et dans d'autres situations semblables.

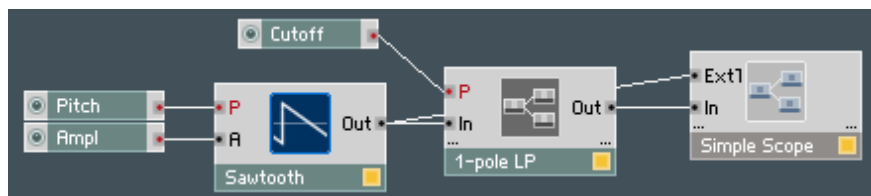
Le module *Z⁻¹* sert à enregistrer la valeur de la sortie précédente et envoie automatiquement un événement avec cette valeur à chaque battement de la cadence audio. Le module prévient aussi l'apparition éventuelle de valeurs dénormales. Les utilisateurs habitués au traitement numérique du signal devraient remarquer que la structure ressemble beaucoup aux diagrammes de filtres DSP standard.

Le module *Merge (Mélangeur)* situé à la sortie de l'additionneur garantit que l'état du filtre est toujours à zéro après l'initialisation, même si le signal d'entrée a une valeur non nulle à cet instant.

N'oubliez pas de placer le convertisseur pitch-vers-fréquence dans la structure de la cellule core, et vous êtes prêt(e) pour le test:



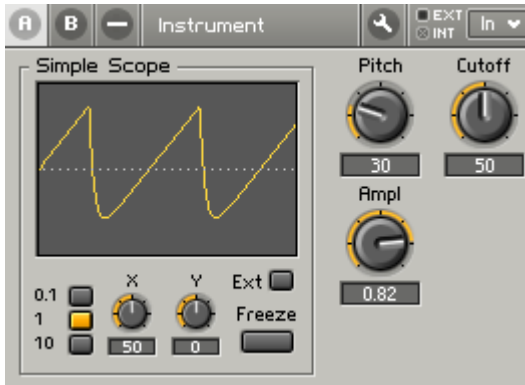
Nous suggérons d'utiliser la structure de test suivante (n'oubliez pas de régler la polyphonie de l'instrument sur 1 voix):



Le potentiomètre *Cutoff (Fréquence de coupure en anglais)* doit être réglé dans l'intervalle [0.. 100]. Faites attention aux fréquences de coupure trop élevées: en raison de l'erreur croissante sur le coefficient du filtre, le filtre devient instable aux hautes fréquences.

Un meilleur design du filtre limiterait au moins les valeurs de la fréquence de coupure à l'intervalle pour lequel le filtre est stable. Dans notre cas, il faudrait limiter le coefficient b à l'intervalle $[0.. 0,99]$ ou quelque chose comme ça. Les techniques de limitation des valeurs seront décrites plus loin dans ce texte.

Voici ce que vous devriez désormais voir en mode *Panel*:



Déplacez le potentiomètre de la fréquence de coupure (*Cutoff*) et observez l'évolution de la forme du signal.

Le traitement conditionnel

Le routage des événements

Dans Reaktor Core, les événements ne doivent pas systématiquement voyager par les mêmes chemins. Il est possible de modifier dynamiquement ces chemins. Pour ce faire, vous pouvez utiliser le module *Router* (*Built In Module > Flow > Router*):



Le module *Router* reçoit des événements à son entrée signal (celle du bas) et les envoie soit vers sa sortie 1 (supérieure), soit vers sa sortie 0 (inférieure). La sortie vers laquelle les événements sont envoyés dépend de l'état courant du routeur, contrôlé par l'entrée *Ctl* (l'entrée du haut).

L'entrée *Ctl* reçoit une connexion d'un nouveau type, compatible ni avec les signaux normaux, ni avec les signaux OBC. Il s'agit d'un signal de type *BoolCtl* ("boolean control", "contrôle binaire" en anglais). Le signal *BoolCtl* ne peut être que dans l'état *true* ou l'état *false* (*vrai* ou *faux*, *on* ou *off*, 1 ou 0...). Si le signal de contrôle est dans l'état *true*, les évènements sont envoyés à la sortie 1. Si le signal de contrôle est dans l'état *false*, les évènements sont envoyés à la sortie 0.

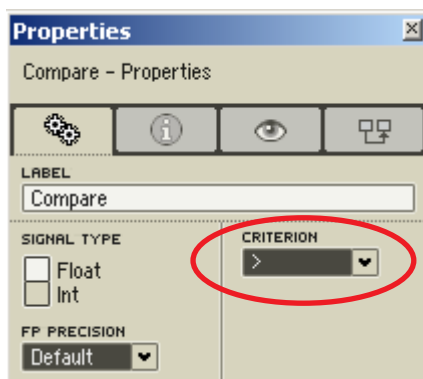
Dans Reaktor Core, les signaux de contrôle ont une différence notable avec les signaux normaux: ils ne transmettent pas d'évènements et de peuvent donc piloter seuls aucun traitement.

Pour contrôler un routeur, vous avez bien sûr besoin d'une source de signal de contrôle. La source la plus commune est sans doute le module *Comparison*, disponible dans *Built In Module > Flow > Compare*:



Ce module compare les deux signaux d'entrée et transmet le résultat sous la forme d'un signal *BoolCtl*. L'entrée supérieure est supposée être à gauche du signe de comparaison et celle du bas à droite du signe. Le module affichant '>' produit donc un signal de contrôle *true* si la valeur à l'entrée supérieure est plus grande que celle à l'entrée inférieure, et un signal *false* dans le cas contraire.

Vous pouvez changer le critère de comparaison dans les propriétés du module:

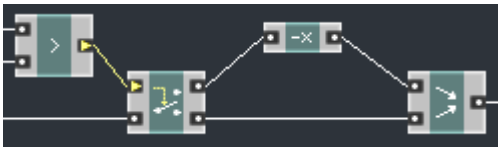


Les critères disponibles sont:

=	égal
!=	différent (\neq)
<=	inférieur ou égal (\leq)
<	strictement inférieur
>=	supérieur ou égal (\geq)
>	strictement supérieur

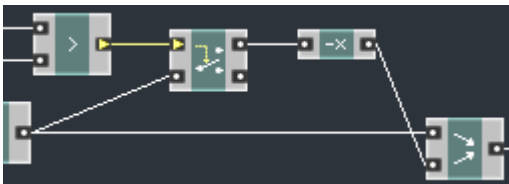
Il est bien sûr possible de connecter plusieurs routeurs au même module de comparaison. Ils changeront d'état simultanément.

Le module *Router* sépare le chemin d'évènements en deux branches. Assez souvent, ces branches se rejoignent en aval:



En fonction du résultat de la comparaison, la structure ci-dessus va soit inverser le signal d'entrée, soit le laisser inchangé.

Une implémentation alternative de cette structure est possible:



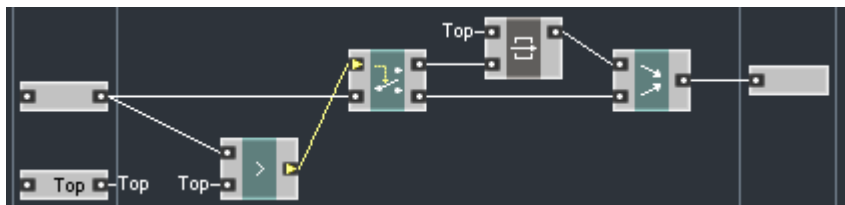
Dans cette version, la sortie 0 du *Router* est déconnectée, le routeur fonctionne donc comme un interrupteur (ou un *gate*), laissant passer les évènements seulement s'il est dans l'état *true*. La valeur inversée arrive alors à la deuxième entrée du module *Merge* (mélangeur), écrasant la valeur non inversée qui arrive toujours à la première entrée. Si le routeur est dans l'état *false*, l'inverseur ne reçoit pas d'évènement et n'envoie rien à la seconde entrée du *Merge*, qui transmet donc le signal non modifié à sa sortie.

Les branches sont le plus souvent mélangées avec un module *Merge*. Mais, théoriquement, vous pourriez utiliser à la place n'importe quel autre module (p.ex. un module arithmétique comme l'additionneur, le multiplicateur, etc.).

Les routeurs traitent l'évènement d'initialisation exactement comme les autres évènements. Ainsi, en utilisant des routeurs, on peut filtrer l'évènement d'initialisation pour l'empêcher d'apparaître dans certaines parties de la structure.

Construction d'un limiteur de signal

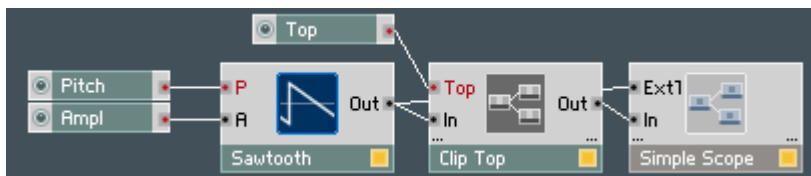
Construisons la structure d'une macro Reaktor Core qui limitant le signal audio à un niveau spécifié):



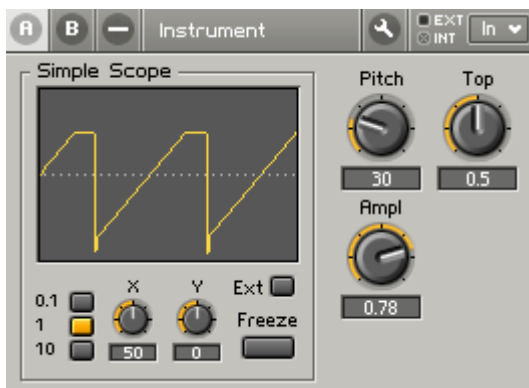
Si le signal d'entrée est inférieur au seuil, il est envoyé à la sortie 0 du routeur puis, via le module *Merge*, à la sortie de la structure. Sinon, le signal est envoyé à la sortie 1 où il contrôle le loquet (*Latch*), envoyant au module *Merge* la valeur seuil à la place du signal. La même chose se produit lors de l'initialisation.

Veuillez remarquer que la sortie de cette structure ne change pas automatiquement quand le seuil change. Le nouveau seuil sera simplement utilisé pour tous les évènements ultérieurs de l'entrée de signal. Ceci est assez similaire au comportement des macros à modulation, pour lesquelles les changements du modulateur ne produisaient pas d'évènements de sortie.

Voici une structure permettant de tester le module limiteur que nous venons de construire (une cellule core audio est utilisée):



Et voilà ce que vous voyez sur le panneau:



En fait, vous trouverez un grand nombre de macros de limiteurs “à modulation” de ce genre dans le menu *Expert Macro > Clipping*.

Construction d’un oscillateur en dents de scie simple

Construisons maintenant un simple oscillateur en dents de scie qui génère une forme d’onde en dents de scie d’amplitude 1 à une fréquence spécifiée. Nous allons utiliser l’algorithme suivant: “Augmente le niveau de sortie à une vitesse constante et, dès qu’il dépasse 1, rabaisse-le de 2”.

On pourrait également proposer de réinitialiser le niveau à -1 au lieu de le rabaisser de 2, mais c’est généralement une mauvaise idée car nous pourrions difficilement maintenir la fréquence d’oscillation demandée.

La vitesse de croissance du signal définit la fréquence de l’oscillateur selon l’équation suivante:

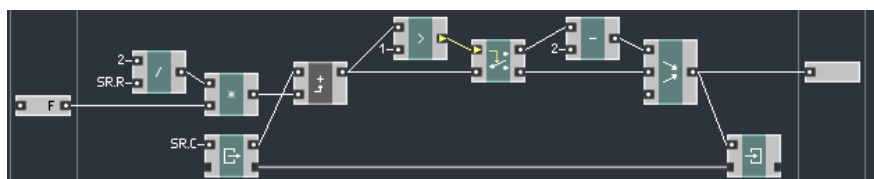
$$d = 2f / f_{SR}$$

où d est l’augmentation du niveau pour un échantillon audio, f la fréquence de l’oscillateur et f_{SR} le taux d’échantillonnage.

Nous allons d’abord construire la circuiterie qui traitera cette vitesse de croissance:



Nous avons maintenant besoin de la boucle d’incrémement. Il est plus que temps d’utiliser les modules *Read* et *Write* exactement comme nous l’avons fait précédemment avec l’accumulateur:



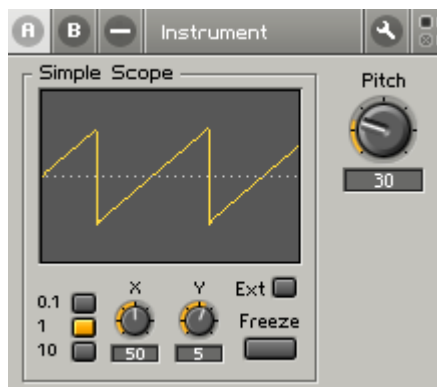
Le module *Read* lance l'incrémentation du niveau à chaque évènement audio. La somme de l'ancien niveau et de l'incrément est ensuite comparée à 1 et envoyée soit directement à l'écriture du résultat, soit à la circuiterie de bouclage.

La troisième entrée du module *Merge* assure l'initialisation correcte de l'oscillateur à zéro. Théoriquement, le module qui soustrait 2 au niveau du signal devrait être une macro à modulation, mais cela nous importe peu ici, puisque le *Merge* écrase de toute façon le résultat de l'initialisation.

Voici la structure de test que nous vous suggérons (n'oubliez pas le convertisseur *P2F* à l'intérieur de la cellule *core*):



Et voilà la vue en mode *Panel*:



Les autres types de signaux

Les signaux flottants

Le type de signal le plus couramment utilisé en DSP (traitement numérique du signal) sur les ordinateurs personnels modernes est la virgule flottante (ou *flottant* pour faire court). Les valeurs flottantes peuvent représenter une large gamme de valeurs, jusqu'à 10^{38} (en mode 32-bit) voire même 10^{308} (en mode 64-bit). Aussi utiles soient-elles, les virgules flottantes ont un inconvénient: elles sont d'une précision limitée. La précision est plus élevée en mode 64-bit, mais elle reste limitée.

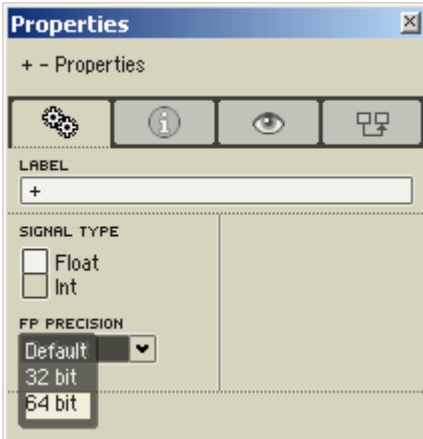
La précision des valeurs en virgules flottantes est limitée pour des raisons techniques: si elles n'étaient pas limitées, ces valeurs nécessiteraient une quantité infinie de mémoire pour les stocker. C'est pour la même raison que vous ne pouvez pas écrire la représentation décimale complète d'un nombre transcendantal (comme π) sur une feuille de papier "finie", aussi grande soit-elle. Même si vous connaissiez tous les chiffres après la virgule (mais comment le pourriez-vous, puisqu'il y en a un nombre infini, mais même si vous le pouviez), essayez juste de les écrire tous: 3, 1, 4, 1, 5, 9, etc. Il arrivera un moment où vous manquerez de papier. Le traitement de tels nombres aurait également demandé un processeur infiniment rapide.

Les signaux et l'espace mémoire que vous avez utilisés jusqu'ici sont représentés par des nombres en virgules flottantes en 32-bit. Reaktor Core vous offre la possibilité d'utiliser des flottants en 64-bit au cas où vous auriez besoin d'une précision supérieure (ou d'un intervalle de valeurs plus grand, bien qu'il soit difficile d'imaginer que l'intervalle allant de 10^{-38} à 10^{38} puisse être insuffisant).

Par défaut, tous les traitements de Reaktor Core sont effectués en flottants de précision 32-bit. Ceci ne signifie pas forcément que les signaux sont traités comme des flottants en 32-bit, mais que le traitement utilisera au *minimum des flottants en 32-bit* (bien que des flottants en 64-bit puissent occasionnellement être utilisés comme résultats intermédiaires).

Vous pouvez modifier la précision des flottants pour les modules individuels ou pour les macros complètes. Pour les modules individuels, vous effectuez

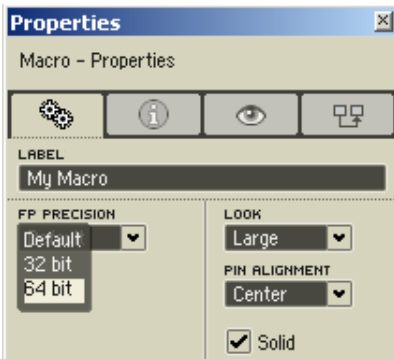
ce réglage via la propriété *FP Precision* (“floating point precision”, “précision de la virgule flottante” en anglais) de ces modules:



- default* utilise la précision par défaut pour la structure en question
- 32 bit* utilise une précision d'au moins 32 bits
- 64 bit* utilise une précision d'au moins 64 bits

Changer la précision pour un module signifie que, dans ce module, le traitement sera effectué en utilisant la précision spécifiée, et que la valeur de sortie sera aussi générée en utilisant cette précision.

Vous pouvez également changer la précision par défaut pour des structures entières si vous faites un clic droit sur le fond et sélectionnez *Owner Properties* pour ouvrir les propriétés du module en question:



La précision par défaut sera effective pour tous les modules à l'intérieur de la structure en question, y compris les macros, tant qu'ils ou elles ne définis-

sent pas à leur tour leur propre précision (ou bien, pour les macros, un autre précision par défaut pour les éléments qu'elles contiennent).

Les signaux en virgule flottante normaux, d'une précision de 32 ou 64 bits, sont parfaitement compatibles entre eux et peuvent être librement interconnectés. Les signaux OBC de précisions différentes ne sont pas compatibles entre eux (car vous ne pouvez pas avoir de stockage qui soit à la fois en 32 et en 64 bits). Ainsi, pour les signaux OBC, les réglages "default", "32 bit" et "64 bit" sont considérés comme étant différents et incompatibles entre eux, car leur précision effective par défaut peut être modifiée en changeant les propriétés de l'une des macros en question.

Les modules d'entrée et de sortie des structures du niveau supérieur des cellules core envoient et reçoivent toujours des flottants de 32 bits, car c'est le type de signal utilisé dans les connexions audio et événements du niveau primaire de Reaktor.

Les signaux entiers

Voici un autre type de données couramment gérées par les processeurs modernes. Ce type de données est en fait plus important encore que les virgules flottantes. Il s'agit des *entiers*. Les nombres entiers sont représentés et traités avec une précision infinie. Bien que la précision des entiers soit *infinie*, l'intervalle des valeurs entières représentables est limité. Pour les entiers représentés sur 32 bits, les valeurs peuvent aller jusqu'à plus de 10^9 (un milliard).

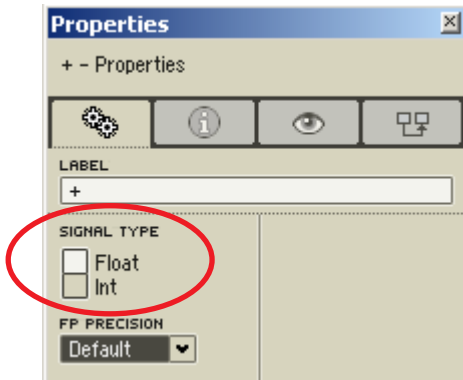
La précision infinie des entiers, tant pour le stockage que pour le traitement, est possible uniquement parce qu'ils n'ont aucun chiffre après la virgule. On peut donc les écrire avec un nombre fini de chiffres. Écrivons le nombre de secondes qu'il y a dans une heure: 3, 6, 0, 0, et c'est tout. C'est aussi simple que ça! Encore une fois, si vous essayez d'écrire la valeur de π , vous n'aurez jamais fini: 3, 1, 4, 1, 5... avec les entiers, point de tout cela.

Alors que la virgule flottante est une représentation naturelle pour les valeurs changeant continûment (comme les signaux audio), les entiers sont une représentation plus appropriée pour les valeurs changeant de manière discrète (p.ex. les compteurs).

De nombreux modules de Reaktor peuvent être commutés en mode entier, dans lequel ils attendent des valeurs entières en entrée, les traitent comme telles (ie avec une précision infinie) et produisent des valeurs entières en sortie. Comme exemples, on peut citer l'additionneur, le multiplicateur ou le soustracteur. Certains modules ne fonctionnent même qu'en mode entier.

Dans Reaktor Core, les valeurs entières ont une longueur minimale de 32 bits garantie.

La commutation entre les types entier et flottant (si le module en est capable) est effectuée via la propriété *Signal Type* du module:



Un module réglé sur le mode entier traite les valeurs d'entrée comme des entiers et produit des valeurs entières en sortie. Vous pouvez déterminer si un module est en mode entier grâce à l'apparence différente de ses entrées et sorties:



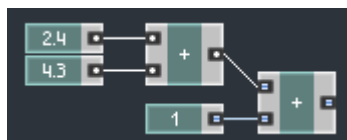
Ce réglage n'existe pas pour les macros, car normalement vous ne construisez pas les structures exactement de la même façon selon que vous leur réservez des entiers ou des flottants (sauf peut-être pour les structures très très simples).

Les signaux entiers peuvent être tout à fait connectés à des flottants, mais les câbles reliant des signaux de types différents effectuent automatiquement la conversion, ce qui peut nécessiter une certaine quantité de ressources processeur. À l'heure où nous écrivons ces lignes, cette quantité est décelable sur les PCs

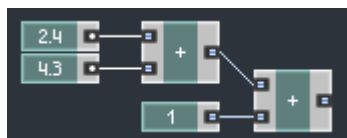
et assez significative sur les Macs. Les connexions OBC de types entier et flottant sont bien sûr incompatibles entre elles.

Une telle conversion peut entraîner une perte d'informations. En particulier, les grands entiers ne peuvent pas être précisément représentés par des flottants, et le contraire est encore plus vrai. Les grands flottants (plus grands que le plus grand entier représentable) ne peuvent pas être représentés par des entiers du tout, auquel cas la conversion est indéfinie. Durant la conversion "flottant vers entier", les valeurs sont arrondies à l'entier approximativement le plus proche. Le terme "approximativement" signifie, par exemple, que la valeur 0,5 sera arrondie soit à 0, soit à 1, même si vous pouvez être à peu près sûr que la valeur 0,49 sera arrondie à 0 et 0,51 à 1.

Il est important de comprendre que la commutation d'un traitement complet en mode entier n'est pas la même chose que la simple conversion de sa sortie en entier. Prenons l'exemple suivant: nous additionnons les deux nombres 2,4 et 4,3 en tant que flottants. Le résultat est 6,7, ce qui, une fois converti en entier, donne 7. La sortie de la structure suivante est donc 8:



Si nous passons maintenant le premier additionneur en mode entier, au lieu d'additionner 2,4 et 4,3, nous allons additionner leurs valeurs arrondies, soit respectivement 2 et 4, ce qui donne 6. Le résultat final sera donc 7:



Les entrées d'horloge ignorent complètement leurs valeurs entrantes, c'est pourquoi elles sont toujours en virgule flottantes. La conversion du type de signal ne sera donc pas effectuée pour les signaux utilisés uniquement comme horloges:

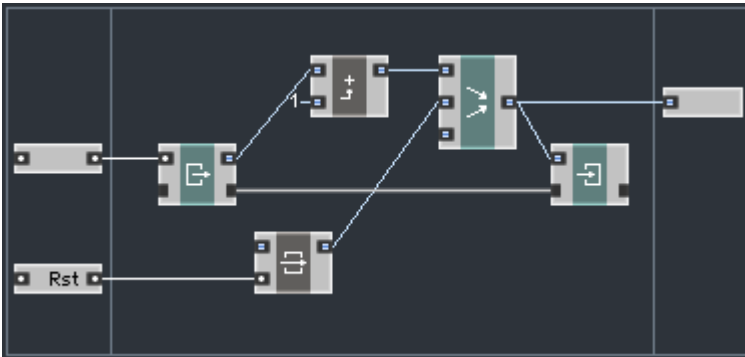


Ici, l'entrée d'horloge du module *Read* reste flottante, bien que le module ait été passé en mode entier (les ports OBC ont la même apparence, qu'ils soient en mode entier ou flottant).

La réinjection entière est résolue de la même manière que la réinjection flottante – en insérant un module Z^{-1} en mode entier (bien sûr, il n'y a plus besoin du système d'élimination des valeurs dénormales).

Construction d'un compteur d'évènements

Construisons un module compteur d'évènements. La fonction de cette macro est similaire à celle de l'accumulateur d'évènements construit plus haut, mais au lieu de sommer les valeurs des évènements, ce module va simplement les compter. Le type entier semble le choix logique pour compter:



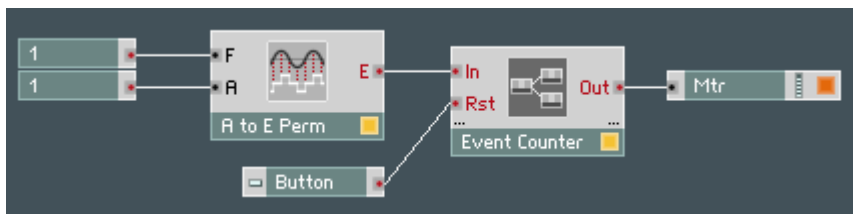
Tous les modules préfabriqués ainsi que la sortie ont été réglés en mode entier. La macro *ILatch* est utilisée ici à la place du *Latch* pour redémarrer le circuit. Les deux macros font exactement la même chose (et se trouvent dans le même menu), mais la première travaille sur les signaux de type entier. Une macro à modulation pour les entiers est aussi utilisée (vous la trouvez dans le menu *Expert Macro > Modulation > Integer*). Les deux entrées n'ont pas besoin d'être réglées en mode entier, car elles fournissent uniquement des signaux d'horloge.

Jetons un œil sur la structure de la cellule core évènement contenant la macro:



Nous voyons que la sortie de ce module n'est pas réglée sur le mode entier (d'ailleurs c'est impossible). Ceci est dû au fait que la cellule *core* doit, en tant que module du niveau primaire de Reaktor, sortir un évènement normal du niveau primaire, qui est forcément en virgule flottante.

Voici une structure de test pour ce compteur:



... et l'allure du panneau correspondant:



Building a rising edge counter macro

Nous allons maintenant discuter de la technique de *comparaison de signe*, qui pourra vous être utile dans la construction de structures Reaktor Core. La “comparaison de signe” s’attache au signe des nombres (“plus” ou “moins”) sans prêter attention à leur valeur. Bien sûr, “plus” est plus grand que “moins”. Ainsi, par exemple:

- 3,1 est de signe plus grand que -1,4
- 2,1 de même signe que 5,0
- 4,5 est de même signe que -2,9

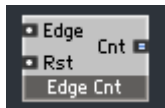
Vous pourriez implémenter la comparaison de signe via plusieurs modules de comparaison et plusieurs routeurs, mais il existe une technique plus efficace. Dans Reaktor Core, la comparaison de signe peut être effectuée via le module *Compare Sign (Built In Module > Flow > Compare Sign)*:

Vous pourriez implémenter la comparaison de signe via plusieurs modules de comparaison et plusieurs routeurs, mais il existe une technique plus efficace. Dans Reaktor Core, la comparaison de signe peut être effectuée via le module *Compare Sign* (*Built In Module > Flow > Compare Sign*):



Le module produit en sortie un signal BootCtl que vous pouvez connecter à un routeur.

L'une des utilisations possibles de ce module est la détection de fronts montants dans le signal entrant. Nous entendons par “front montant” une partie croissante du signal. Nous allons construire ci-dessous un compteur de fronts montants sous forme de macro Reaktor Core:

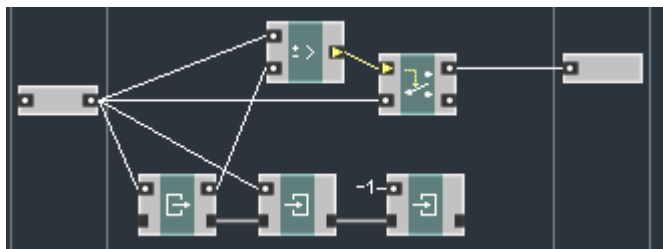


Remarquez que la sortie est en mode entier, puisque le décompte est, à n'en pas douter, une valeur entière.

La première chose dont nous avons besoin est une macro de détection de front qui convertira le front détecté en évènement:



Voici comment la macro de détection peut être implémentée:



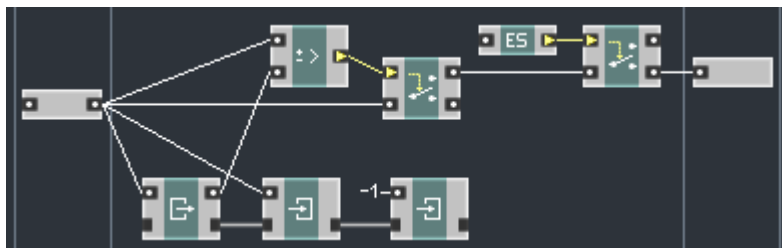
La chaîne OBC en bas de la structure conserve la valeur précédente du signal d'entrée. Comme nous le voyons, la nouvelle valeur est enregistrée après que

l'ancienne a été lue. Le dernier module *Write* de la chaîne s'occupe de la tâche d'initialisation pour l'emplacement mémoire de la valeur précédente. Nous initialisons cet emplacement à la valeur -1 , afin que la première valeur positive soit comptée comme front montant.

Nous avons déjà vu comment utiliser le module *Merge* pour initialiser l'emplacement mémoire. Le module *Write* à la fin de la chaîne OBC est une autre technique d'initialisation de la mémoire interne. Il doit être le dernier module *Write* dans la chaîne, pour pouvoir écraser les résultats déjà enregistrés par les modules *Write* en amont.

Le module *Router*, contrôlé par le module *Sign Comparison*, va sélectionner les événements, ne laissant passer que ceux pour lesquels le signe passe de “négatif” à “positif” (bref les “fronts montants”).

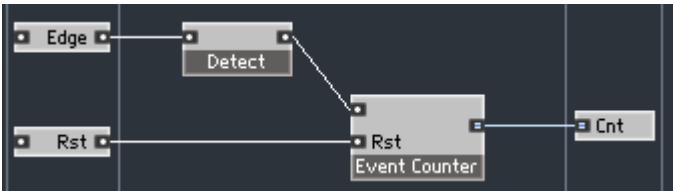
Il n'est pas clair qu'un tel module envoie un événement pendant l'initialisation, en particulier parce que l'emplacement mémoire est encore à zéro lors du traitement des événements d'initialisation, et le signe de zéro est indéfini. Nous pouvons modifier cette structure de manière à éviter qu'un événement ne soit envoyé lors de l'initialisation:



Le module *ES Ctrl* est un “contrôle sensible aux événements”. Le signal de contrôle qu’il produit est dans l’état *true* seulement si un événement arrive à l’entrée du module. Puisque l’entrée est déconnectée dans la structure ci-dessus, autrement dit connectée à une constante nulle, la seule fois où le signal de contrôle est égal à *true* est lors de l’initialisation. Le second routeur va donc bloquer tout événement survenant pendant l’initialisation et laisser passer tous les autres.

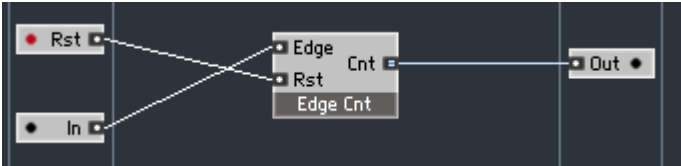
Remarquez que nous avons ici un exemple d’un module qui n’envoie pas d’évènement à sa sortie pendant l’initialisation.

Maintenant que nous avons un module détecteur, nous pouvons le connecter à la circuiterie de comptage que nous avons déjà :

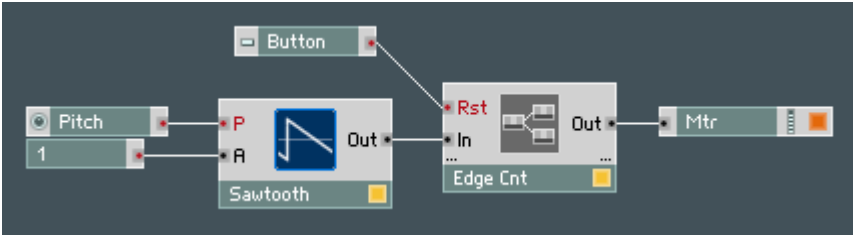


La fonction de la structure ci-dessus doit être claire. Le module *Detect* envoie un évènement à chaque fois qu'il détecte un front montant, et ces évènements sont comptés par le module *Evt Cnt*.

Pour construire un circuit de test, plaçons cette macro dans une cellule *core* audio et comptons les fronts montants d'une forme d'onde en dents de scie. La structure interne de la cellule *core* ressemble à celle-ci :



... et la structure du niveau primaire à ceci :



(n'oubliez pas de régler les propriétés du *Meter* comme indiqué dans les exemples précédents).

Voici ce que vous devriez voir dans le panneau :



La vitesse de changement du nombre doit correspondre à la fréquence de l'oscillateur, définie par le potentiomètre *Pitch*. Avec un *Pitch* valant zéro, la fréquence de l'oscillateur est d'environ 8 Hz, donc le nombre devrait être incrémenté environ 8 fois par seconde.

Les tableaux

Introduction aux tableaux

Imaginez que vous vouliez construire un module sélecteur de signal audio qui, en fonction de la valeur d'un signal de contrôle en entrée, prenne le signal de l'une des quatre entrées audio:



Pour implémenter ce module, une première approche serait d'utiliser des modules *Router*. Mais il y a une autre possibilité. Nous pouvons utiliser une autre fonctionnalité de Reaktor Core: les *tableaux*.

Un *tableau unidimensionnel* est une *série ordonnée* de données *du même type* qui peuvent être référencées par leur rang dans la collection, ie leur *index*. Par exemple, prenons un groupe de 5 nombres en virgule flottante:

5.2 16.1 -24.0 11.9 -0.5

Dans Reaktor Core, les indices des éléments d'un tableau démarrent à zéro: le premier élément a pour index 0. Ici, l'élément d'index 0 est 5,2, celui d'index 1 est 16,1, celui d'index 2 est -24,0 et ainsi de suite.

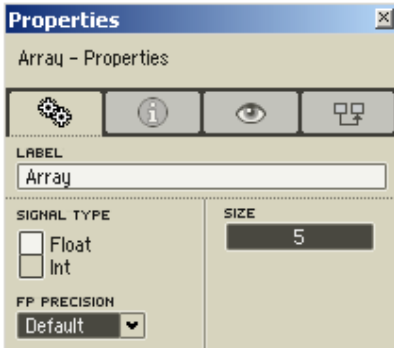
Voici une représentation de ce tableau:

Index	0	1	2	3	4
Valuer	5.2	16.1	-24.0	11.9	0.5

Dans Reaktor Core, les tableaux sont créés grâce au module *Array* (ce qui signifie... *Tableau* en anglais) que l'on trouve dans le sous-menu *Built In Module > Memory > Array*:



Un module *Array* a une seule sortie, de type *Array OBC*. La taille du tableau (le nombre d'éléments) et le type de données contenues sont spécifiés dans les propriétés du tableau:



Par exemple, pour le tableau ci-dessus, nous devons spécifier le type de données *float* (*virgule flottante*) et une taille (*size*) de 5.

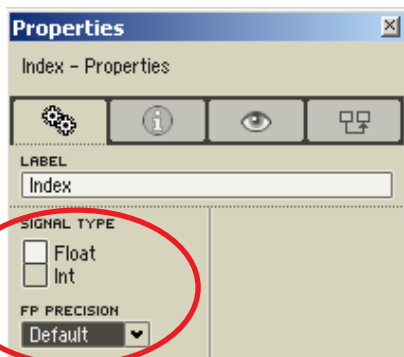
Veuillez noter que, comme les indices des tableaux partent de zéro, l'intervalle d'indices pour un tableau de taille 5 serait de 0 à 4 (vous pouvez aussi le voir dans le tableau ci-dessus).

Les signaux de type *Array OBC* correspondant aux différents types de données ne sont pas compatibles entre eux.

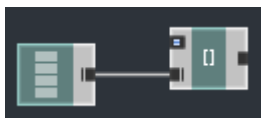
Pour faire référence à un élément d'un tableau, vous devez spécifier son index. Pour ce faire, vous pouvez utiliser le module *Index* situé dans *Built In Module > Memory > Index*:



L'entrée OBC maîtresse (en bas) du module *Index* doit être connectée à la sortie esclave d'un module tableau. Le type de connexion de l'entrée maîtresse doit correspondre au type du tableau, vous devez le spécifier dans les propriétés du module *Index*:



Et maintenant la connexion:

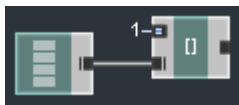


L'entrée supérieure du module *Index* est toujours de type entier et reçoit la valeur d'index. Ici, nous faisons une référence à l'élément d'index 1 dans le tableau:



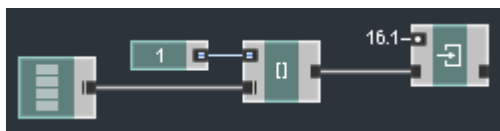
Notez que le module *Const* a aussi été réglé sur le mode entier (indiqué par le symbole de sa sortie). Ceci n'est pas nécessaire, puisqu'une conversion automatique est effectuée sinon, mais c'est plus joli.

Nous aurions aussi pu utiliser une *QuickConst*:



La sortie du module *Index* est de type *Latch OBC*. Ceci signifie que vous pouvez y connecter un module *Read* ou *Write* (ou plusieurs d'entre eux). Bien entendu, vous devez veiller à ce que les modules *Read* et *Write* soient réglés sur le même type de données que les modules *Array* et *Index*.

Ici, l'élément d'index 1 du tableau est initialisé à 16,1:



Si un index hors de l'intervalle est envoyé au module *Index*, le résultat de l'accès au tableau donne une valeur indéfinie. La structure ne plante pas, mais il est impossible de savoir quel élément sera lu, ou même si un élément sera effectivement lu. Ceci signifie que si vous n'êtes pas certain(e) de la taille du tableau, vous devriez la limiter via des modules *Router* ou des macros de la librairie

Construction d'un détecteur de signal audio

Revenons à la construction de notre module sélecteur de signal audio:



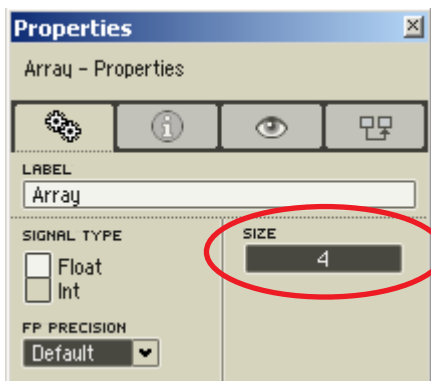
Voici une structure interne vide pour construire ce module:



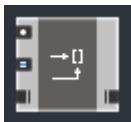
Nous allons utiliser un tableau de 4 éléments en virgule flottante pour stocker nos signaux audio:



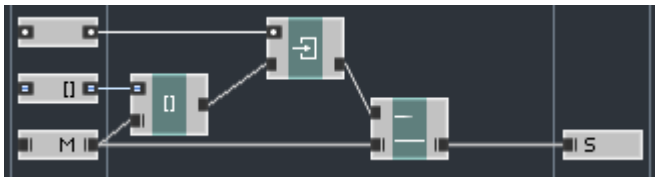
Voici les propriétés du module tableau:



Pour écrire les valeurs d'entrée dans le tableau, nous utilisons une macro standard *Write []* disponible dans *Expert Macro > Memory > Write []*:

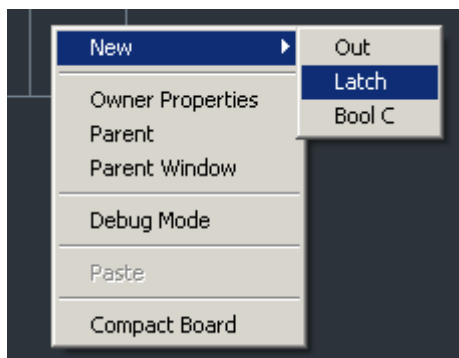


Cette macro dispose en interne d'un module *Index* et d'un module *Write*, effectuant l'écriture dans l'élément du tableau avec l'index spécifié:

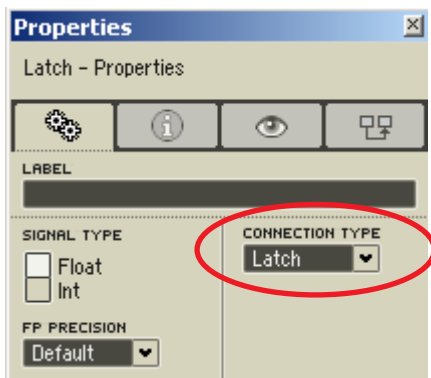


L'entrée supérieure reçoit bien sûr la valeur à écrire. L'entrée "[]" reçoit l'index auquel l'écriture doit avoir lieu. L'entrée "M" reçoit la connexion OBC à un tableau de flottants à la précision par défaut, et la sortie "S" est une connexion "thru" (ie qui prolonge simplement la chaîne OBC), semblable à celle d'autres modules OBC comme *Read* et *Write*.

L'entrée "M" et la sortie "S" sont des ports de macro d'un type différent de celui que nous avons utilisé jusqu'à présent. De tels ports peuvent être insérés en sélectionnant l'entrée "*Latch*" du menu d'insertion de ports (le troisième type de port est le *BoolCtl*):



Les ports de type “*Latch*” peuvent servir aux connexions *OBC Latch* (entre les *Reads* et les *Writes*) ou pour les connexions *OBC Array*. Ces réglages sont à effectuer dans les propriétés du port:



Régler le type de connexion sur “*Latch*” ou “*Array*” définit le type de connexion *OBC*. Pour les ports de la macro *Write []*, ce réglage est visiblement sur “*Array*”.

Le module avec deux lignes horizontales parallèles est le module *R/W Order*, que l'on trouve dans *Built In Module > Memory > R/W Order*:



Il ne fait rien d'autre que transmettre la connexion de son entrée maîtresse (en bas) à sa sortie esclave. L'entrée supérieure n'a absolument aucun effet, si ce n'est celui d'affecter l'ordre de traitement des modules. *Tout ce qui est*

connecté à la sortie “S” de la macro sera traité après le module *Write*, ce qui ne serait pas forcément le cas si ce module *R/W Order* était absent de la structure.

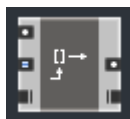
En l’absence du module *R/W Order*, la fonctionnalité de la macro *Write []* n’aurait été ni assurée ni intuitive, car l’utilisateur se serait alors attendu à ce que tout ce qui est connecté en aval de la macro *Write []* soit traité après elle. En général, de tels problèmes surviennent uniquement avec les connexions OBC; vous devez donc veiller à placer des modules *R/W Order* aux endroits nécessaires dans les macros que vous concevez.

Le module *R/W Order*, tout comme les ports OBC, dispose d’une propriété *Connection Type*. Pour ce module, cette propriété ne contrôle que le type des ports “M” et “S”, l’autre entrée étant systématiquement en mode *Latch*. Pour plus de détails, veuillez vous référer à la description du module *R/W Order* dans la section de référence.

Construisons le circuit permettant d’écrire les signaux d’entrée dans le tableau:

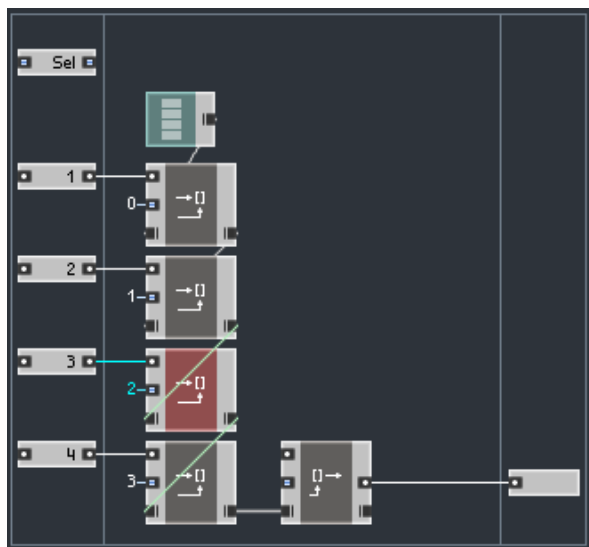


Les quatre modules *Write []* se chargent d'enregistrer les valeurs audio entrantes dans le tableau. Nous avons maintenant besoin de lire l'une des quatre valeurs. Nous suggérons d'utiliser la macro *Read []* (*Expert Macro > Memory > Read []*):

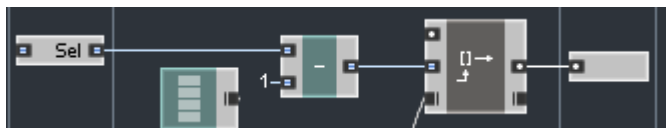


Cette macro effectue la lecture dans un élément d'un tableau, dont l'index est précisé par l'entrée entière au milieu. L'entrée du haut est l'entrée de l'horloge pour les opérations de lecture – la macro envoie la valeur lue à la sortie du haut, en réponse à un évènement entrant à cette entrée du haut. Les ports en bas sont bien sûr les connexions de tableau maître et esclave.

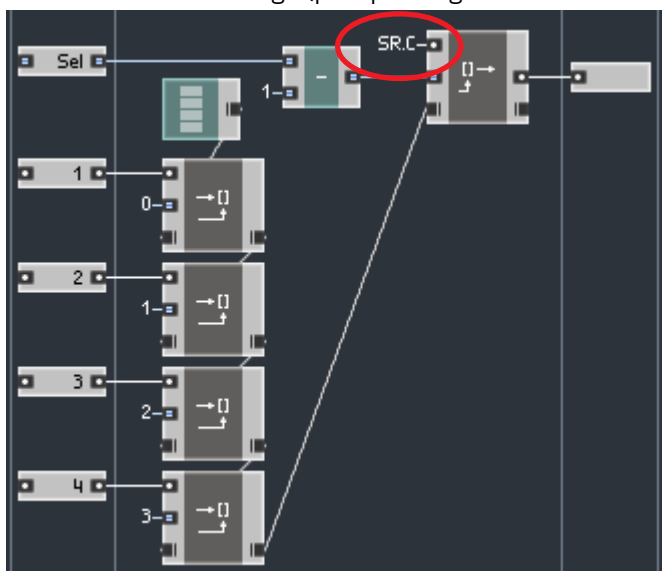
Maintenant, à quoi connectons-nous son entrée maîtresse ? Bien sûr, nous ne pouvons la connecter directement au module tableau, parce que nous voulons que l'opération de lecture soit effectuée après toutes les opérations d'écriture (il pourrait sinon y avoir un délai effectif d'un échantillon, ou pas, bref rien de très fiable). Nous ne pouvons pas non plus la connecter à l'un des modules *Write []*. Cela ne résoud pas notre problème. Nous suggérons plutôt de connecter les modules *Write []* en "ribambelle" derrière le module tableau, autrement dit de les connecter en série, puis de connecter le module *Read []* à la sortie du dernier module *Write []*. Le module *Read []* est alors effectivement le dernier.



Maintenant, que connectons-nous à l'entrée index du module *Read []* ? Si nous voulons que notre valeur de sélection soit entre 1 et 4, nous devons soustraire 1 à la valeur d'entrée "Sel". Notez que nous effectuons une soustraction entière (puisque "Sel" est juste une entrée de contrôle, nous n'avons pas vraiment besoin d'une macro à modulation ici):

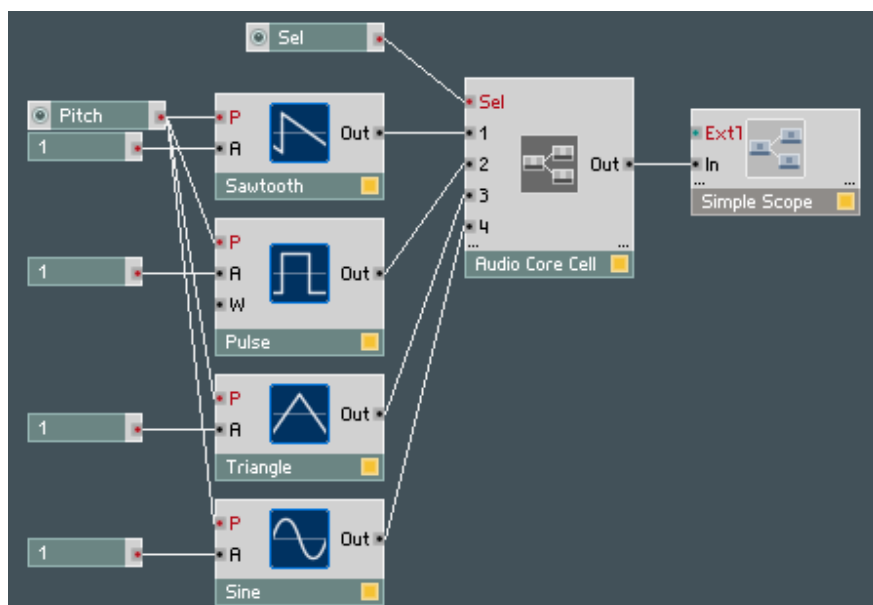


La dernière chose à faire est de cadencer le module de lecture via l'horloge du taux d'échantillonnage (puisque'il s'agit d'un sélecteur audio):



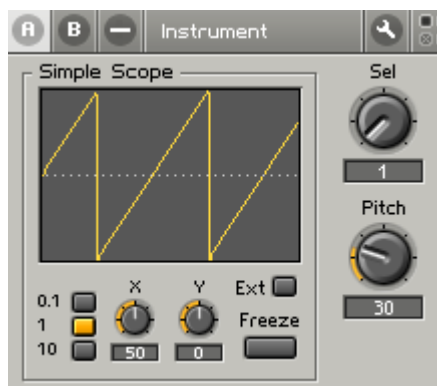
De façon générale, nous aurions dû prendre soin de limiter la valeur d'entrée "Sel" à l'intervalle correct, mais pour nous simplifier la tâche nous n'allons pas le faire maintenant.

Voici la structure de test que nous suggérons (nous avons placé la macro dans une cellule *core* audio):



Le potentiomètre *Sel* est réglé pour commuter entre les quatre valeurs allant de 1 à 4.

Affichez la vue du panneau et observez les différentes formes d'onde en fonction du réglage du potentiomètre:



Construction d'un délai

Maintenant que nous possédons une certaine expérience avec les tableaux, construisons une macro simple de délai audio. Le module doit ressembler à quelque chose comme ça :

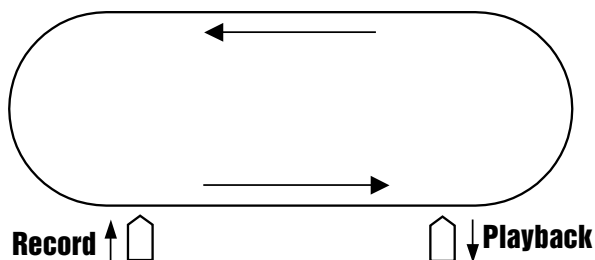


... ou encore mieux, à quelque chose comme ça (pour ce faire nous devons régler la propriété *Port Alignment* de la macro sur “top”) :

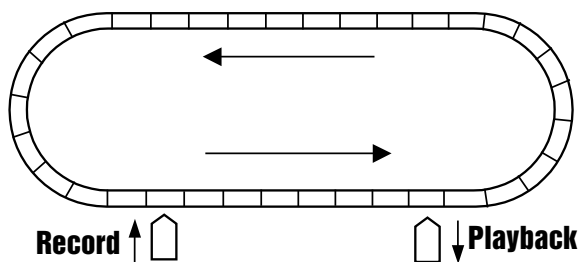


L'entrée “T” reçoit le temps de délai, en secondes.

Si vous jetez un œil sur un appareil de délai analogique à bande, vous verrez une boucle de bande combinée à des têtes de lecture et d'enregistrement. Pour être précis, il y a aussi une tête d'effacement, mais pour faire simple nous pouvons imaginer que la tête d'enregistrement effectue à la fois la tâche d'écriture et celle d'effacement.

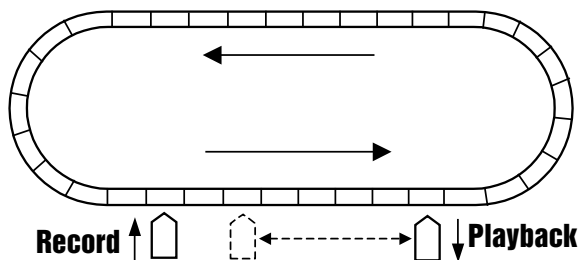


Si nous voulons simuler cet appareil sous forme numérique, il nous faut une sorte de boucle de bande digitale. De par la nature discrète du monde numérique, la boucle de “bande digitale” contiendra un nombre fini d'échantillons audio. Ces échantillons seront enregistrés et lus au taux d'échantillonnage audio :



Un choix naturel pour cette “boucle de bande digitale” est un tableau dont la taille est égale au nombre d’échantillons enregistrés dans la boucle.

Dans un délai analogique à bande, le temps de délai dépend de la distance entre les têtes de lecture et d’enregistrement - qui est fixe - et de la vitesse de défilement de la bande - qui est variable - et ce pour des raisons techniques évidentes: il est bien plus facile de faire varier la vitesse de la bande que l’écartement des têtes... Dans le monde digital, il se trouve que c’est le contraire: faire varier la vitesse de la bande revient à effectuer une conversion du taux d’échantillonnage entre la “bande digitale” et la sortie, alors que faire varier la distance entre les têtes est relativement simple. C’est donc ce que nous allons faire de ce pas:



Il y a une autre différence notable avec le monde analogique. Dans le monde analogique, la bande se déplace. Si nous voulions déplacer notre “bande digitale”, nous serions obligés de copier tous les éléments du tableau dans l’élément voisin *pour chaque battement de l’horloge audio*, ce qui coûterait cher en ressources processeur. À la place, nous allons plutôt déplacer les têtes.

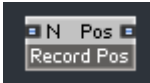
De tout ceci, nous pouvons conclure que nous avons besoin des éléments suivants:

Tableau	– notre simulation de “boucle de bande digitale”
Index d’écriture	– notre tête d’enregistrement
Index de lecture	– notre tête de lecture

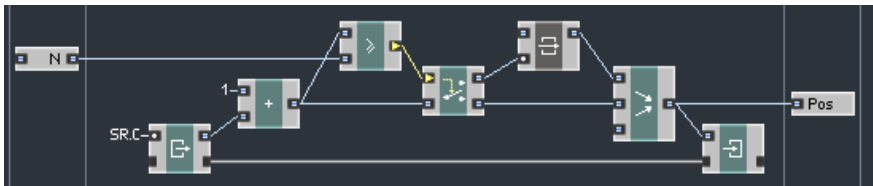
Les index de lecture et d’enregistrement se déplacent donc à travers le tableau, échantillon par échantillon. Au moment où l’un d’eux atteint la fin du tableau, il doit être renvoyé au début du tableau (ce qui correspond à scotcher la fin de la bande avec son début). La différence entre la position de lecture et celle d’écriture correspond au temps de délai mesuré en échantillons.

Cette technique est en fait assez classique en programmation, on l’appelle la technique du “tampon circulaire” (“circular buffer”), ou “tampon en anneau” (“ring buffer”).

Commençons par programmer la “tête d’enregistrement”. Cette fonction est très similaire à l’oscillateur en dents de scie que nous avons déjà programmé, si ce n’est que les calculs sont effectués en mode *entier*. Le pas d’incréméntation vaut 1 (pour chaque tic-tac audio), l’intervalle des valeurs de sortie va de 0 à $N-1$, où N est la taille du tableau. Plaçons cette circuiterie dans une macro “RecordPos”:

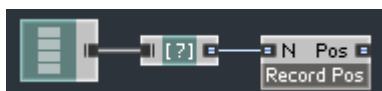


L’entrée “N” doit recevoir le nombre d’éléments du tableau, et la sortie “Pos” doit générer la position courante d’enregistrement (l’index). Voici comment implémenter cette macro (comparez avec l’implémentation de l’oscillateur en dents de scie vu plus haut):

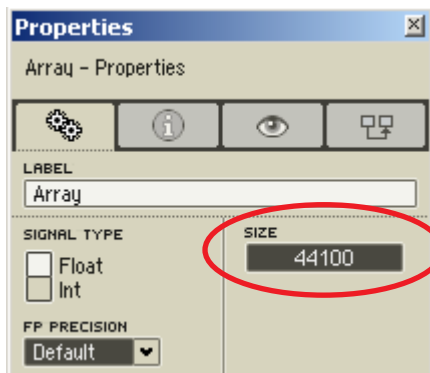


Notez que le module de comparaison est réglé sur “>=”. Le signe “=” était sans importance pour l’oscillateur en dents de scie, mais pour les calculs sur les entiers, la différence se révèle généralement décisive. La condition “>=” garantit que la tête d’enregistrement n’atteindra jamais la valeur N.

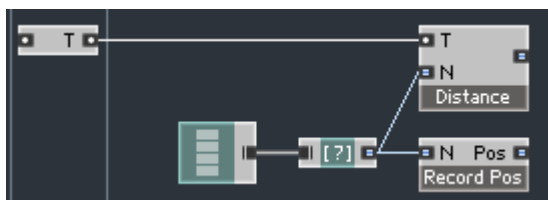
Au niveau supérieur, nous créons un module tableau et le connectons à ce RecordPos via le module `Size []` (disponible dans *Built In Module > Memory > Size []*), qui renvoie la taille du tableau:



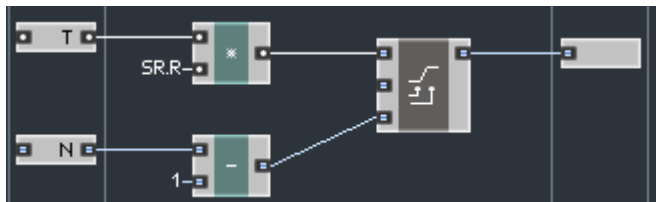
La taille du tableau peut aller jusqu'à 44100 emplacements. Ceci nous autorise jusqu'à 1 seconde de délai (une seconde moins un échantillon, pour être précis) au taux d'échantillonnage de 44,1 kHz:



Il nous faut maintenant calculer les véritables index. Pour ce faire, nous allons construire deux macros. La première macro va convertir le temps de délai demandé en distance, exprimée en échantillons:



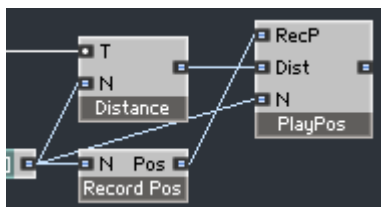
Ceci peut être effectué en multipliant le temps (en secondes) par le taux d'échantillonnage (en Hertz). Nous devons aussi penser à limiter le résultat, la macro *Expert Macro > Clipping > IClipMinMax* devrait faire l'affaire:



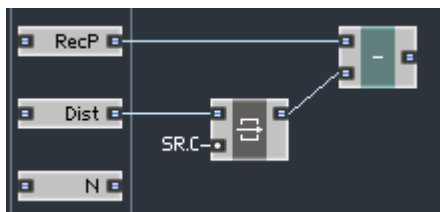
Nous nous limitons à $N-1$ car c'est effectivement la plus grande distance possible entre deux éléments du tableau. Notez que la conversion en entiers est effectuée après la multiplication.

Nous aurions aussi pu limiter directement la valeur d'entrée (à une valeur différente, bien sûr), ce qui est généralement plus approprié, puisque les valeurs flottantes au-delà des limites de représentation des entiers peuvent produire des valeurs entières arbitraires, rendant alors la limitation sans objet.

Nous utilisons ensuite une autre macro pour calculer l'index de lecture à partir de *RecordPos* et *Distance*:



La position de lecture doit être située à *Distance* échantillons derrière la position d'enregistrement, c'est pourquoi nous effectuons leur soustraction:



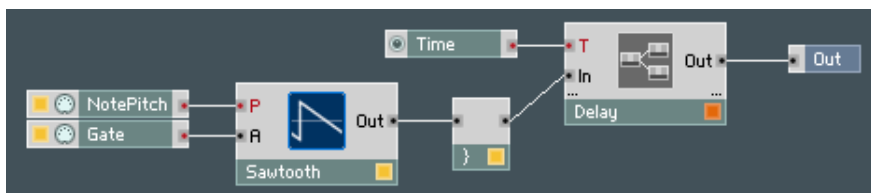
La valeur de la distance passe par un loquet car elle est produite par une entrée de contrôle qui peut potentiellement recevoir des événements à n'importe quel moment, et nous ne voulons pas que la soustraction survienne à d'autres moments que les événements de l'horloge audio.

Si nous nous contentons de soustraire, la différence risque d'être souvent négative. Ceci est dû au fait que notre tableau n'est pas une boucle, ses extrémités ne sont pas connectées entre elles. Nous devons donc *cycliser* le résultat:

- 1 doit devenir $N-1$,
- 2 doit devenir $N-2$,
- 3 doit devenir $N-3$,
- etc.

The diagram illustrates a complex data flow graph with several interconnected blocks and signals. On the left, there are two input ports. The top one connects to a 'Buf' block. Below it, a 'T' signal is connected to a 'Dist' block. A 'Record Pos' block receives input from the 'Buf' block and outputs to the 'Dist' block. The 'Dist' block has two outputs: one labeled 'N' which feeds into a 'PlayPos' block, and another labeled 'N' which feeds into a 'Buf' block. The 'PlayPos' block has an output labeled 'SR.C' which connects to a 'Dist' block. This 'Dist' block has an output labeled 'N' which feeds into a 'Buf' block. The 'Buf' block has an output labeled 'N' which feeds into a 'Record Pos' block. There are also control signals like 'T' and 'N' that influence the flow between blocks.

Voici la structure de test que nous proposons. N'oubliez pas de placer un convertisseur *ms2sec* dans la cellule *core Delay*, ni de régler celle-ci en mode monophonique:



En fait, il est conseillé de passer le délai en mode monophonique dès que possible, car pour chaque voix il consomme environ 200 Ko de mémoire. En effet, 44100 échantillons, utilisant 4 octets (32 bits) chacun, font en tout:

$44100 \times 4 = 176400$ octets = un peu plus de 172 Ko
(un kilooctet contient 1024 octet)

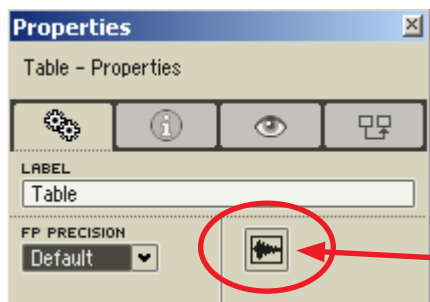
Pour tester la structure ci-dessus, jouez des notes sur votre clavier MIDI et écoutez-les retardées de la durée spécifiée par le potentiomètre *Time*.

Les tables

Il existe un autre module similaire à *Array*. Il s'appelle *Table* et se trouve dans le sous-menu *Built In Module > Memory > Table*:

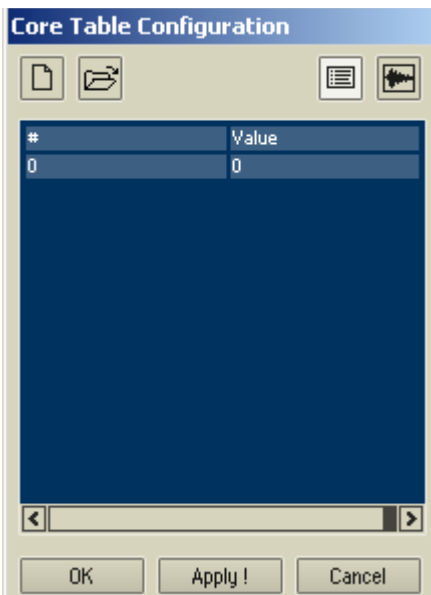


Les tables sont en fait des tableaux un peu particulier, en ceci que vous ne pouvez que lire dans les tables, l'écriture y étant interdite. Les valeurs d'une table sont préinitialisées via les propriétés du module. Pour accéder à la liste des valeurs, appuyez sur le bouton dans la fenêtre de propriétés:

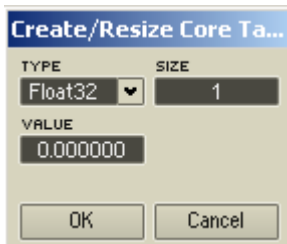


Cliquez ici pour modifier les valeurs

Une nouvelle boîte de dialogue doit apparaître:



Vous voyez une table vide. Elle contient un seul élément, de valeur zéro. Vous pouvez maintenant commencer à saisir les nouvelles valeurs manuellement ou les importer depuis un fichier. Si vous choisissez l'option manuelle, cliquez sur le bouton . La boîte de dialogue suivante apparaît:



Vous devez sélectionner le type de valeurs stockées dans la table, sa taille (le nombre d'éléments qu'elle contient) et une valeur pour initialiser tous les éléments de la table.

Vous pouvez également importer la table depuis un fichier. Le fichier peut être aux formats WAV/AIFF, texte (TXT/ASC) ou Native Table (NTF). Pour importer depuis un fichier, appuyez sur le bouton . Une boîte de dialogue apparaît et vous permet de sélectionner un fichier. Puis une autre boîte de dialogue vous demande de sélectionner le type de données pour les valeurs de la table.

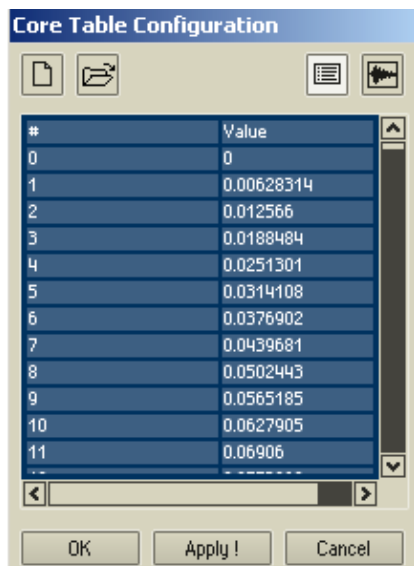
Essayons d'utiliser une table. Nous allons construire une macro d'oscillateur sinusoïdal via l'approche "lecture de table":





Dans cette macro, commençons par créer un module table:



Nous l'initialisons via le fichier *sinetable.txt* que nous vous avons préparé dans le dossier "Core Tutorial Examples", dans le dossier d'installation de Reaktor. Il s'agit d'un fichier texte contenant les valeurs d'une fonction sinus, sur une période entière plus un sample. Importez-les en tant que valeurs flottantes *Float32*:

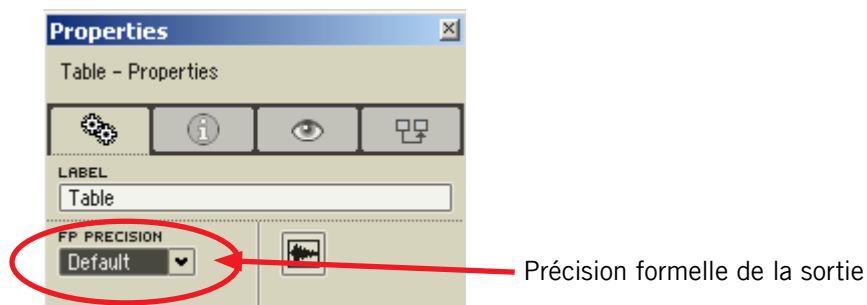


Vous pouvez aussi visualiser les valeurs chargées sous l'aspect d'une forme d'onde. Les boutons  et  commutent entre les deux affichages "liste" et "forme d'onde".

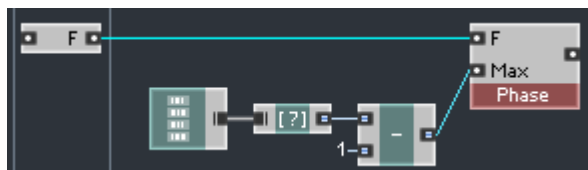
Appuyez sur OK pour fermer la boîte de dialogue et transférer les valeurs à la table.

Il existe également une propriété *FP Precision*, que vous trouverez dans la fenêtre de propriétés de la table. Elle ne contrôle pas vraiment la précision

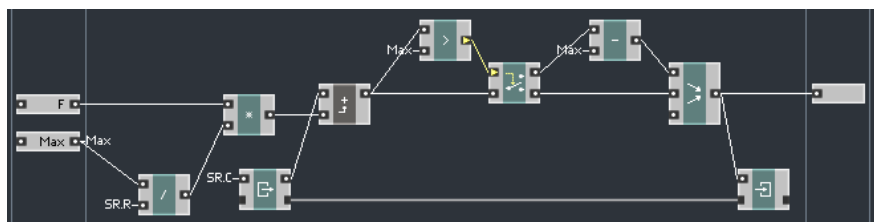
des valeurs dans la table (celle-ci aurait dû être sélectionnée lors de l'importation des valeurs depuis un fichier ou lors de leur saisie manuelle). Elle définit plutôt la précision "formelle" de la sortie du module *Table*. De façon générale, laissez cette propriété sur "Default":



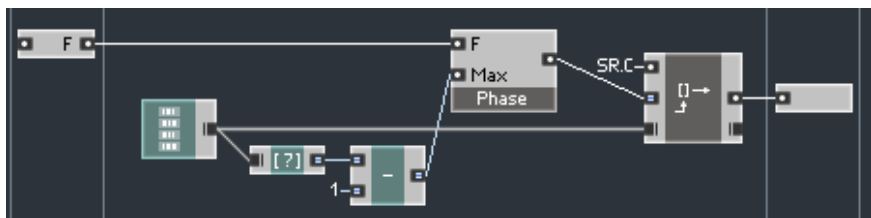
Maintenant que nous avons la table, continuons à construire l'oscillateur. Il sera basé sur un "oscillateur de phase" qui générera un signal en dents de scie montantes, de 0 jusqu'à la taille de la table moins un:



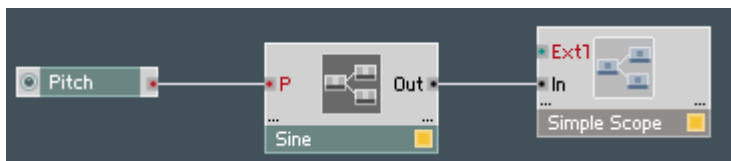
L'implémentation de l'oscillateur de phase est assez semblable à celle d'un oscillateur en dents de scie ou à celle de la position d'enregistrement dans le délai précédent:



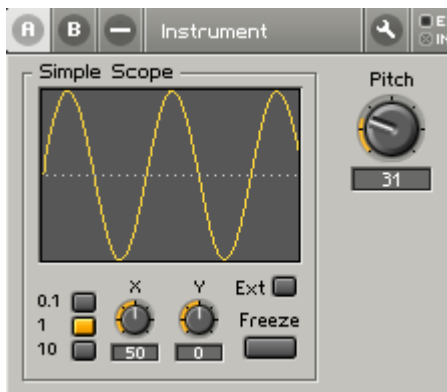
Un module *Read []*, connecté à l'oscillateur de phase et cadencé par l'horloge du taux d'échantillonnage, accède à l'élément correspondant dans la table et extrait sa valeur.



La structure de test que nous suggérons est la suivante (n'oubliez pas le convertisseur *P2F* dans la cellule *core*):



... dont voici le panneau:



Bien entendu, le sinus produit ne sonne pas très bien car nous n'avons pas effectué d'interpolation. Nous vous laissons le plaisir de construire une version avec interpolation si vous le souhaitez.

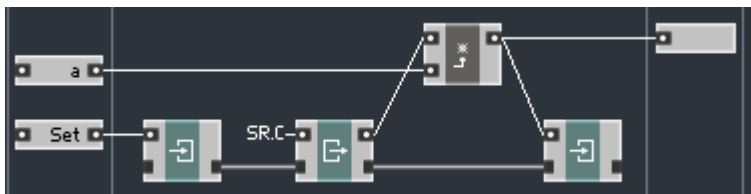
Construction de structures optimisées

La règle dit: aucun outil n'est parfait. La technologie Reaktor Core ne fait pas exception. Cela ne signifie pas qu'elle soit mauvaise, bien au contraire, nous pensons que cette technologie est tout simplement géniale 😊. *Pas parfait* signifie *réel*. Cela implique que vous deviez avoir quelques connaissances sur comment tirer le meilleur parti de cette technologie. Appelez cela "Trucs et astuces" ou quoi que ce soit d'autre. Nous allons ici discuter de ces questions d'optimisation.

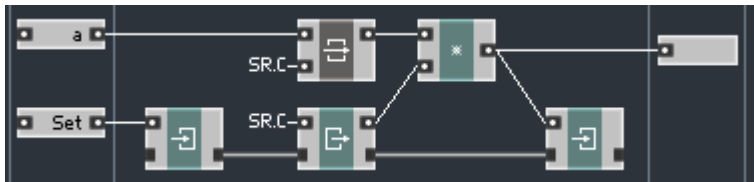
Les Latches et les macros à modulation

Utilisez les *Latches* (loquets) et/ou les macros à modulation à tous les endroits nécessaires, pour être sûr(e) de retarder les évènements jusqu'à ce qu'ils doivent effectivement être traités.

Voici une structure qui utilise une macro à modulation pour la multiplication dans la boucle d'itération audio. L'utilisation de la macro à modulation évite que le traitement soit lancé par les évènements de l'entrée "a":



Nous pourrions tout aussi bien utiliser un loquet explicite dans la structure:



Nous avons déjà mentionné de nombreux exemples de cette technique plus haut dans ce manuel.

L'utilisation des loquets intéresse à la fois l'optimisation des performances et *l'exactitude* de vos structures. Certaines erreurs typiques de la programmation de structures sont liées à l'envoi d'évènements à certains modules aux mauvais moments.

Ne craignez pas que les loquets diminuent les performances de vos structures. Les loquets ne consomment pas beaucoup, voire *pas du tout* de ressources processeur.

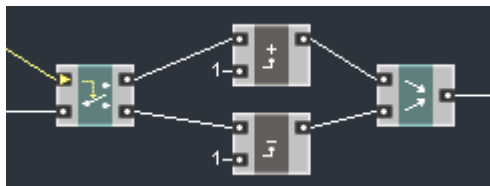
Les loquets sont généralement préférés aux routeurs pour le filtrage des évènements, car ils consomment moins de puissance processeur. Autant que faire se peut, utilisez les routeurs uniquement lorsque la logique du traitement impose leur emploi.

Routage et mélange

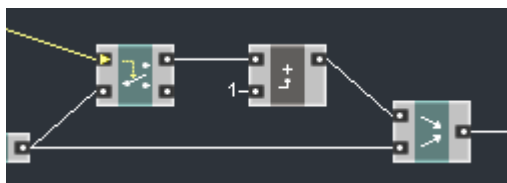
Le routage peut être plus ou moins coûteux en termes de puissance de calcul, selon la situation et votre plateforme. Si vous pouvez éviter le routage sans ajouter d'opérations coûteuses à votre structure, faites-le !

Le routage *ES Ctl* peut être parfois remplacé par des *Latches*. Si possible, ayez recours à ce remplacement.

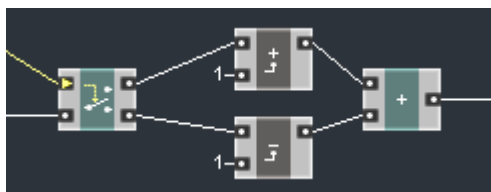
Si vous séparez le chemin d'évènements en deux branches via un routeur, il est souhaitable de mélanger ces branches après leur traitement différentiel:



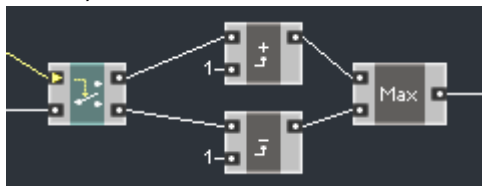
Il est aussi conseillé de mélanger au routeur les évènements entrants (non séparés):



Le mélange ne doit pas forcément être effectué via un module *Merge*. N'importe quel module arithmétique ou similaire fait l'affaire:



Le mélange peut aussi se faire dans une macro (en fonction de sa structure interne):



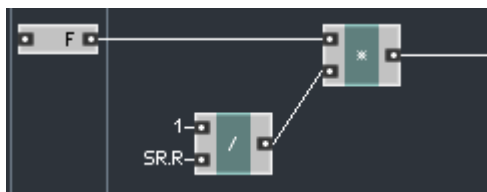
Il peut être nécessaire ou raisonnable de mélanger les branches générées par différents routeurs, mais veillez alors à la consommation de puissance processeur.

Les opérations numériques

Pour les nombres en virgule flottante, l'addition, la soustraction, la valeur absolue et l'opposé sont généralement les opérations les plus économiques. Pour les entiers, ce sont l'addition, la soustraction et l'opposé. La valeur absolue d'un entier est aussi relativement économique. L'élimination des valeurs dénormales du module *DN Cancel* est, comme vous vous en rappelez, une simple addition.

En revanche, la division de flottants, la multiplication et la division d'entiers consomment en moyenne plus de puissance.

Il est conseillé de regrouper vos opérations d'une manière telle que les plus coûteuses soient effectuées le plus rarement possible. Par exemple, si vous devez calculer une fréquence normalisée en divisant la fréquence en Hertz par le taux d'échantillonnage, il est judicieux de calculer d'abord l'inverse du taux d'échantillonnage puis de multiplier le résultat par la fréquence:



Dans la structure ci-dessus, la division est effectuée seulement lors des changements de taux d'échantillonnage, qui devraient être plutôt rares. Les changements de la fréquence ne relancent que la multiplication.

Comparez avec l'implémentation directe de la même formule:

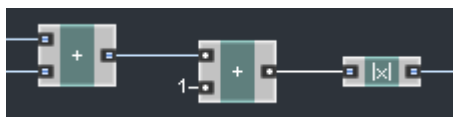


dans laquelle la division est effectuée en réponse à chaque changement de la fréquence.

Conversions entre flottants et entiers

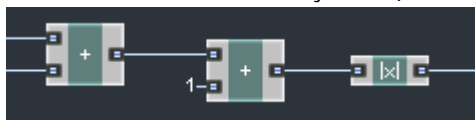
De manière générale, évitez toute conversion inutile entre flottants et entiers. Selon la plateforme, ces conversions peuvent utiliser une part non négligeable de la puissance processeur. Les conversions nécessaires doivent être maintenues, bien sûr...

Bien que la structure suivante fonctionne comme prévu, elle comprend en fait deux conversions inutiles entre entiers et flottants:



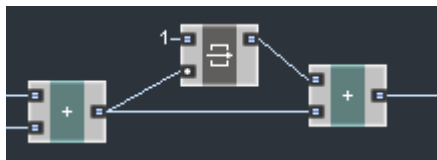
La première conversion a lieu à l'entrée du module additionneur, au milieu. Ce module est réglé pour travailler avec des flottants, mais il reçoit un signal entier en entrée. Il convertit donc les entiers en flottants. La seconde conversion a lieu à l'entrée du module de valeur absolue, qui est réglé en mode *entier* alors qu'il reçoit les flottants issus de la sortie de l'additionneur. Il convertit donc les flottants en entiers.

Voici une bien meilleure façon de procéder:



Tous les modules sont réglés en mode *entier*, il n'y donc aucune conversion à faire.

Les signaux d'horloge doivent généralement être de type flottant, mais si un signal entier est utilisé, ce n'est pas un problème:



Même si l'entrée d'horloge du *Latch* est flottante, elle est cadencée par un signal entier. Comme la valeur de l'horloge est sans importance, aucune conversion n'est effectuée.

Annexe A. Interface utilisateur de Reaktor Core

A.1. Les cellules core

Une cellule core peut être créée dans la structure du niveau primaire de Reaktor (sauf dans la structure de l'ensemble) via un clic droit sur le fond et en sélectionnant *Core Cell > New Audio* ou *Core Cell > New Event*.

Les cellules core des bibliothèques (utilisateur comme système) se trouvent dans le même menu "*Core Cell*". Vous pouvez également charger les cellules core via la commande *Core Cell > Load...*

Pour supprimer une cellule core, sélectionnez-la et appuyez sur la touche *Suppr*, ou bien faites un clic droit sur la cellule core et sélectionnez la commande *Delete*. Il est aussi possible de supprimer une sélection multiple.

Pour enregistrer une cellule core dans un fichier, faites un clic droit sur la cellule core et sélectionnez la commande *Save Core Cell As...*

Pour modifier la structure interne de la cellule core, double-cliquez dessus. Pour ressortir de sa structure, double-cliquez sur le fond.

Pour modifier les propriétés externes de la cellule core, faites un clic droit sur la cellule et sélectionnez *Properties*. Si la fenêtre de propriétés est déjà ouverte, il vous suffit de cliquer sur la cellule.

Pour modifier ses propriétés internes, vous devez vous rendre dans sa structure interne, faire un clic droit sur le fond de l'écran et sélectionner *Owner Properties*. Si la fenêtre de propriétés est déjà ouverte, il vous suffit de cliquer sur le fond.

A.2. Les macros et modules core

Pour créer un module ou une macro core normal(e), faites un clic droit dans la partie centrale (la plus grande) de la structure core et sélectionnez l'un(e) des modules/macros dans les menus *Built In Module*, *Expert Macro*, *Standard Macro*, ou *User Macro*. Vous pouvez aussi charger un(e) module/macro par un clic droit sur le fond et en sélectionnant la commande *Load Module...*

Une macro vide peut être créée depuis le menu *Built In Module*.

Pour enregistrer un(e) module/macro *core* dans un fichier, faites un clic droit dessus et sélectionnez la commande *Save As...*

Pour supprimer un(e) module/macro *core*, sélectionnez-le(la) et appuyez sur la touche *Suppr*, ou bien faites un clic droit dessus et sélectionnez la commande *Delete*. Il est aussi possible de supprimer une sélection multiple.

Pour modifier la structure interne du module / de la macro *core*, double-cliquez dessus. Pour ressortir de sa structure, double-cliquez sur le fond.

Pour modifier les propriétés du module / de la macro *core*, vous devez vous rendre dans sa structure interne, faire un clic droit sur le fond de l'écran et sélectionner *Owner Properties*. Si la fenêtre de propriétés est déjà ouverte, il vous suffit de cliquer sur le fond.

Vous pouvez également accéder aux propriétés du module / de la macro *core* depuis l'extérieur, via un clic droit dessus et en sélectionnant *Properties*. Si la fenêtre de propriétés est déjà ouverte, il vous suffit de cliquer sur le module / la macro.

A.3. Les ports core

Pour créer un port *core*, effectuez un clic droit dans la zone de gauche (entrées) ou de droite (sorties) de la structure *core* et sélectionnez l'un des types de ports disponibles dans le sous-menu *New*.

Pour supprimer un port *core*, sélectionnez-le et appuyez sur la touche *Suppr*, ou bien faites un clic droit dessus et sélectionnez la commande *Delete*. Il est aussi possible de supprimer une sélection multiple (y compris une sélection mixte de modules/ports).

A.4. Modification de la structure core

Pour déplacer un module *core*, cliquez dessus et déplacez-le vers l'endroit souhaité. Les ports peuvent être déplacés uniquement verticalement, leur ordre vertical définissant leur ordre d'apparition à l'extérieur.

Pour créer une connexion entre une entrée d'un module et une sortie d'un autre, cliquez sur l'un des ports et déplacez le curseur sur l'autre port.

Pour supprimer une connexion, cliquez sur le câble de connexion pour le sélectionner, puis appuyez sur la touche *Suppr*. Vous pouvez aussi cliquer sur

l'entrée et déplacer le curseur n'importe où sur le fond de l'écran. Pour créer une *QuickConst*, effectuez un clic droit sur l'entrée d'un module et sélectionnez *Connect to New QuickConst*. Pour accéder aux propriétés de la *QuickConst*, cliquez simplement sur celle-ci.

Pour créer un *QuickBus*, effectuez un clic droit sur une entrée ou une sortie d'un module et sélectionnez *Connect to New QuickBus*. Pour connecter une entrée ou une sortie de module à un *QuickBus* existant, cliquez sur cette entrée ou cette sortie, et sélectionnez l'un des bus disponibles dans le menu *Connect to QuickBus*.

Annexe B. Le concept de Reaktor Core

B.1. Les signaux et les évènements

Les signaux peuvent être de type flottant ou entier. Les ports flottants ressemblent à ceci: , et les ports entiers à cela: .

Les signaux se propagent sous la forme d'évènements à travers les connexions depuis les sorties jusqu'aux entrées connectées. Un évènement est une action basique qui se produit à une sortie particulière, changeant la valeur de cette sortie (dans certains cas, la valeur est changée en elle-même).

Tous les évènements provenant de la même source d'évènements sont considérés comme "simultanés". Ceci signifie que si deux évènements arrivent à différentes entrées d'un même module, ils arrivent "simultanément".

"La même source" signifie la même sortie: dans certaines circonstances, plusieurs sorties peuvent être considérées comme "une même source d'évènements". Par exemple, toutes les entrées audio et connexions d'horloge du taux d'échantillonnage standard sont considérées comme une même source d'évènements. Dans ce contexte, "la même source" ne signifie pas forcément "la même valeur" mais plutôt leur "simultanéité".

À moins qu'un module donné soit une source d'évènements, la seule chose pouvant commander le traitement de ses valeurs entrantes est un ou plusieurs évènements arrivant à ses entrées. S'il s'agit d'évènements multiples, un seul évènement est produit en sortie, puisque les évènements en entrée arrivent simultanément.

B.2. L'initialisation

L'initialisation des structures s'effectue de la façon suivante. D'abord, toutes les valeurs sont remises à zéro. Puis l'évènement d'initialisation est envoyé simultanément depuis toutes les sources d'initialisation. Généralement, il s'agit des constantes, des entrées des cellules core (mais pas toujours) et des horloges. C'est tout.

B.3. Les connexions OBC

Les OBC (Object Bus Connections) sont des connexions entre modules qui n'envoient aucun signal mais déclarent juste que les modules partagent un même état commun (mémoire). La connexion de ce type la plus courante est celle entre des modules *Read* et *Write* accédant à une même valeur enregistrée.

B.4. Le routage

Vous pouvez utiliser les modules *Router* pour diriger le flux d'évènements entre deux chemins possibles. Si le *Router* choisit l'un des chemins de sortie pour les évènements entrants, l'autre sortie ne reçoit pas d'évènement (cela signifie en particulier que la valeur de cette autre sortie ne peut pas être modifiée).

Le Router est contrôlé par une connexion d'entrée de type *BoolCtl*. À l'autre extrémité de cette connexion, vous avez généralement un module *Compare* (parfois, d'autres modules intermédiaires comme des ports de macro *BoolCtl* peuvent se trouver entre le *Compare* et le *Router*).

En utilisant les *Routers*, vous pouvez dynamiquement activer ou désactiver les calculs dans certaines parties de vos structures.

En général, après avoir séparé le chemin du signal en deux via un *Router*, vous voudrez remélanger les deux branches via un module *Merge* ou un autre module. Souvent, vous voudrez mélanger une des branches au signal original non séparé. De manière générale, vous êtes libre de faire ici ce qu'il vous plaira (mais faites tout de même attention à la consommation de ressources processeur).

B.5. Le “latching”

Le “latching” (utilisation des *Latches*, ou loquets) est probablement la technique la plus courante dans Reaktor Core. Elle consiste à utiliser des modules *Latch* pour éviter que des évènements ne soient envoyés aux mauvais instants. Par exemple, vous ne voudriez pas qu'un évènement de signal de contrôle ne

lance un calcul dans une boucle audio.

Vous pouvez aussi utiliser les macros du groupe *Expert Macros > Modulation*, qui est un ensemble des combinaisons de *Latches* les plus courantes, avec quelques modules de traitement arithmétique.

B.6. Les horloges

Les horloges sont des sources d'évènements. D'habitude, les évènements d'horloge arrivent à des intervalles réguliers qui correspondent à la fréquence de l'horloge. Vous avez besoin d'horloges pour piloter différents modules comme les oscillateurs, les filtres, etc. La plupart des implémentations de ces modules n'ont pas besoin d'une connexion d'horloge explicite venant de l'extérieur: ils utilisent implicitement une source d'horloge standard disponible dans les structures *core*. Cette source est l'horloge du taux d'échantillonnage, qui tourne à la fréquence audio par défaut.

Veuillez noter que, dans les cellules *core* évènements, bien que la connexion à l'horloge du taux d'échantillonnage soit disponible, le signal d'horloge lui-même n'est pas disponible. C'est pourquoi la plupart des oscillateurs, filtres et autres modules similaires ne fonctionnent pas dans les cellules *core* évènements.

Annexe C. Les ports des macros core

C.1. In



Reçoit un évènement entrant depuis l'extérieur et le transfère inchangé à sa propre sortie interne.

La connexion d'entrée interne peut être utilisée pour forcer la signification par défaut (déconnectée) de ce port.

C.2. Out



Reçoit un évènement arrivant à l'entrée interne et le transfère inchangé à l'extérieur

C.3. Latch (entrée)



Transfère une connexion OBC de l'extérieur de la macro à l'intérieur de la macro.

La connexion d'entrée interne peut être utilisée pour forcer la signification par défaut (déconnectée) de ce port.

C.4. Latch (sortie)



Transfère une connexion OBC de l'intérieur de la macro à l'extérieur de la macro

C.5. Bool C (entrée)



Transfère une connexion BoolCtl de l'extérieur de la macro à l'intérieur de la macro.

La connexion d'entrée interne peut être utilisée pour forcer la signification par défaut (déconnectée) de ce port.

C.6. Bool C (sortie)



Transfère une connexion BoolCtl de l'intérieur de la macro à l'extérieur de la macro.

Annexe D. Les ports des cellules core

D.1. In (mode audio)



Donne accès au signal audio venant de l'extérieur du module. Envoie des événements réguliers à la cadence du taux d'échantillonnage (synchronisés à SR.C, l'horloge globale du taux d'échantillonnage).

ÉVÈNEMENT D'INITIALISATION: envoie un événement d'initialisation. Sa valeur est définie par l'initialisation extérieure.

D.2. Out (mode audio)



Donne accès depuis l'extérieur du module aux valeurs reçues à l'intérieur. À tout instant, la dernière valeur reçue est transférée à l'extérieur.

D.3. In (mode événement)



Convertit les événements du niveau primaire de Reaktor venant de l'extérieur en événements Reaktor Core et les transfère à l'intérieur.

ÉVÈNEMENT D'INITIALISATION: envoie un événement d'initialisation si un événement d'initialisation est reçu de l'extérieur.

D.4. Out (mode événement)



Convertit les événements Reaktor Core venant de l'intérieur en événements du niveau primaire de Reaktor et les transfère à l'extérieur. Si plusieurs sorties en mode événements reçoivent simultanément des événements Reaktor Core, les événements correspondant du niveau primaire sont envoyés dans l'ordre des sorties supérieures vers les sorties inférieures.

Annexe E. Les bus intégrés

E.1. SR.C

Envoie des évènements d'horloge réguliers à la cadence du taux d'échantillonnage.

ÉVÈNEMENT D'INITIALISATION: envoie toujours un évènement d'initialisation.

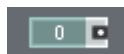
E.2. SR.R

Fournit le taux d'échantillonnage courant, en Hertz. Envoie des évènements avec les nouvelles valeurs en réponse aux changements du taux d'échantillonnage.

ÉVÈNEMENT D'INITIALISATION: envoie toujours un évènement d'initialisation avec le taux d'échantillonnage initial.

Annexe F. Les modules d'usine

F.1. Const



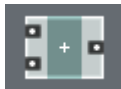
Génère un signal de valeur constante. Cette valeur est affichée dans le module.

ÉVÈNEMENT D'INITIALISATION: durant l'initialisation, envoie l'évènement avec la valeur spécifiée. C'est le seul instant où ce module envoie un évènement.

PROPRIÉTÉS:

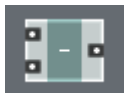
Value la valeur à envoyer en sortie

F.2. Math > +



Génère en sortie la somme des signaux en entrée. L'évènement de sortie est envoyé à chaque fois qu'un évènement arrive à l'une des entrées (ou aux deux simultanément).

F.3. Math > -



Génère en sortie la différence des signaux en entrée (le signal à l'entrée inférieure est soustrait au signal à l'entrée supérieure). L'évènement de sortie est envoyé à chaque fois qu'un évènement arrive à l'une des entrées (ou aux deux simultanément).

F.4. Math > *



Génère en sortie le produit des signaux en entrée. L'évènement de sortie est envoyé à chaque fois qu'un évènement arrive à l'une des entrées (ou aux deux simultanément).

F.5. Math > /



Génère le quotient des signaux en entrée (le signal à l'entrée supérieure est divisé par le signal à l'entrée inférieure). En mode entier, produit une division avec reste, le reste étant abandonné. L'évènement de sortie est envoyé à chaque fois qu'un évènement arrive à l'une des entrées (ou aux deux simultanément).

F.6. Math > |x|



Génère en sortie la valeur absolue du signal entrant. L'évènement de sortie est envoyé à chaque fois qu'un évènement arrive en entrée.

F.7. Math > -x



Génère en sortie la valeur opposée (inversion du signe) du signal entrant. L'évènement de sortie est envoyé à chaque fois qu'un évènement arrive en entrée.

F.8. Math > DN Cancel



Modifie le signal entrant de manière à supprimer les nombres dénormaux. L'implémentation actuelle ajoute une toute petite constante. Fonctionne uniquement avec les nombres en virgule flottante. L'évènement de sortie est envoyé à chaque fois qu'un évènement arrive en entrée.

F.9. Math > ~log

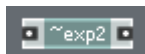


Calcule une approximation du logarithme de la valeur entrante. L'évènement de sortie est envoyé à chaque fois qu'un évènement arrive en entrée.

PROPRIÉTÉS:

Base	base du logarithme
Precision	précision de l'approximation (une meilleure précision demande plus de puissance processeur)

F.10. Math > ~exp



Calcule une approximation de l'exponentielle de la valeur entrante. L'évènement de sortie est envoyé à chaque fois qu'un évènement arrive en entrée.

PROPRIÉTÉS:

Base	base de l'exponentielle
Precision	précision de l'approximation (une meilleure précision demande plus de puissance processeur)

F.11. Bit > Bit AND



Effectue la fonction binaire ET sur les signaux entrants. Fonctionne uniquement sur les entiers. L'évènement de sortie est envoyé à chaque fois qu'un évènement arrive à l'une des entrées (ou aux deux simultanément).

F.12. Bit > Bit OR



Effectue la fonction binaire OU sur les signaux entrants. Fonctionne uniquement sur les entiers. L'évènement de sortie est envoyé à chaque fois qu'un évènement arrive à l'une des entrées (ou aux deux simultanément).

F.13. Bit > Bit XOR



Effectue la fonction binaire OU EXCLUSIF sur les signaux entrants. Fonctionne uniquement sur les entiers. L'évènement de sortie est envoyé à chaque fois qu'un évènement arrive à l'une des entrées (ou aux deux simultanément).

F.14. Bit > Bit NOT



Effectue l'inversion binaire du signal entrant. Fonctionne uniquement sur les entiers. L'évènement de sortie est envoyé à chaque fois qu'un évènement arrive en entrée.

F.15. Bit > Bit <<



Décalle vers la gauche les bits de la valeur à l'entrée supérieure (vers les bits plus significatifs). La quantité du décalage est spécifiée par l'entrée inférieure. Le résultat pour $N < 0$ et $N > 31$ est indéfini (donc utilisez cette fonction seulement pour $0 \leq N \leq 31$). Fonctionne uniquement sur les entiers. L'évènement de sortie est envoyé à chaque fois qu'un évènement arrive à l'une des entrées (ou aux deux simultanément).

F.16. Bit > Bit >>



Décalle vers la droite les bits de la valeur à l'entrée supérieure (vers les bits moins significatifs). Aucune extension de signe n'est effectuée. La quantité du décalage est spécifiée par l'entrée inférieure. Le résultat pour $N < 0$ et $N > 31$ est indéfini (donc utilisez cette fonction seulement pour $0 \leq N \leq 31$). Fonctionne uniquement avec les entiers. L'évènement de sortie est envoyé à chaque fois qu'un évènement arrive à l'une des entrées (ou aux deux simultanément).

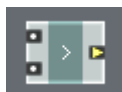
F.17. Flow > Router



Assigne le signal de l'entrée signal (entrée inférieure) à l'une des deux sorties en fonction de l'état du signal de contrôle (entrée supérieure). Si le signal de contrôle est dans l'état true, le module assigne le signal à la sortie 1 (supérieure), et à la sortie 0 (inférieure) si le signal de contrôle est dans l'état

false. L'évènement de sortie est envoyé à l'une des sorties à chaque fois qu'un évènement arrive à l'entrée signal.

F.18. Flow > Compare

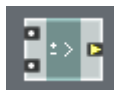


Produit un signal BoolCtl en sortie, indiquant le résultat de la comparaison des valeurs d'entrée. La valeur à l'entrée supérieure est placée à gauche du signe de comparaison et la valeur à l'entrée inférieure à droite du signe (p.ex. le module montré ci-dessus vérifie si la valeur du haut est plus grande que celle du bas).

PROPRIÉTÉS:

Criterion l'opération de comparaison à effectuer.

F.19. Flow > Compare Sign



Produit un signal BoolCtl en sortie, indiquant le résultat de la comparaison du signe des valeurs d'entrée. La valeur à l'entrée supérieure est placée à gauche du signe de comparaison et la valeur à l'entrée inférieure à droite du signe (p.ex. le module montré ci-dessus vérifie si le signe de la valeur du haut est plus grand que le signe de la valeur du bas).

La comparaison de signe est définie comme suit:

- + est égal à +
- est égal à –
- + est plus grand que –

Le signe d'une valeur nulle est indéfini, donc une valeur arbitraire peut être générée si l'une des valeurs comparées est nulle.

PROPRIÉTÉS:

Criterion l'opération de comparaison à effectuer.

F.20. Flow > ES Ctl



Produit en sortie un signal BoolCtl indiquant la présence momentanée d'un évènement en entrée (le signal vaut *true* s'il y a un évènement à l'entrée de ce module à cet instant précis).

F.21. Flow > ~BoolCtl



Produit en sortie un signal BoolCtl qui est l'inversion du signal BoolCtl d'entrée (*true* se change en *false* et réciproquement).

F.22. Flow > Merge

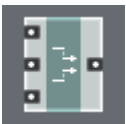


Envoie un évènement en sortie à chaque fois qu'il y a un évènement à l'une des entrées, ou à plusieurs simultanément. Si une seule entrée reçoit un évènement à un instant donné, la valeur de l'évènement en sortie est égale à la valeur de cet évènement en entrée. Si plusieurs entrées reçoivent des évènements simultanément, la valeur de l'entrée la plus basse (parmi celles recevant un évènement à cet instant) est transmise. Par exemple, si la deuxième et la troisième entrée (en partant du haut) reçoivent un évènement, la valeur de l'évènement à la troisième entrée sera transmise.

PROPRIÉTÉS:

Input Count nombre d'entrées du module

F.23. Flow > EvtMerge



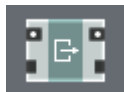
La fonction de ce module est similaire à celle du module *Merge*, si ce n'est qu'ici les valeurs des entrées sont ignorées. La valeur de la sortie est indéfinie. Ce module est conçu pour générer des signaux d'horloge. Il fonctionne

uniquement en mode virgule flottante, puisque de toute façon la valeur n'est pas censée servir à quoi que ce soit.

PROPRIÉTÉS:

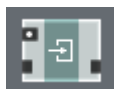
Input Count nombre d'entrées du module

F.24. Memory > Read



Lit la valeur stockée dans la mémoire de la chaîne OBC à laquelle ce module appartient. La lecture se produit en réponse à un événement à l'entrée supérieure (horloge) et la valeur lue est envoyée à la sortie supérieure. Les ports du bas sont les connexions OBC maîtresse (à gauche) et esclave (à droite).

F.25. Memory > Write



Écrit la valeur arrivant à l'entrée supérieure dans la mémoire de la chaîne OBC à laquelle ce module appartient. L'écriture se produit en réponse à un événement à l'entrée supérieure. Les ports du bas sont les connexions OBC maîtresse (à gauche) et esclave (à droite).

F.26. Memory > R/W Order



Ce module n'effectue aucune action. Il peut être inséré dans une structure pour contrôler l'ordre de traitement des modules connectés par OBC. Les ports OBC du bas sont les connexions OBC maîtresse et esclave, connectées en interne en "thru" (transmission simple). L'entrée OBC supérieure est la connexion "secondaire", qui permet de placer ce module logiquement après le module connecté à cette entrée.

L'entrée secondaire ne peut être connectée qu'à des signaux de type OBC Latch normaux. Les ports maître et esclave peuvent quant à eux être connectés à des signaux de type OBC Latch ou Array (loquet ou tableau), selon les réglages des propriétés du module *R/W Order*. En tout cas, le type de signal et la

précision doivent être les mêmes pour toutes les connexions de ce module (p.ex. vous ne pouvez pas connecter en même temps l'entrée secondaire à un *Read* en mode entier et les connexions maîtresse et esclave à des modules en mode flottants).

PROPRIÉTÉS:

Connection Type type de connexion du port "thru" (loquet ou tableau).

F.27. Memory > Array



Définit un objet mémoire de type tableau. Le module lui-même n'effectue aucune action. Toutes les opérations sur le tableau doivent être effectuées par des modules connectés à la sortie du tableau, qui est une connexion OBC esclave de type tableau.

PROPRIÉTÉS:

Size nombre d'éléments dans le tableau.

F.28. Memory > Size []



Renvoie la taille de l'objet tableau connecté à l'entrée. La taille est une valeur entière constante.

ÉVÈNEMENT D'INITIALISATION:

pendant l'initialisation, envoie en sortie un évènement ayant pour valeur la taille du tableau. C'est le seul moment où ce module envoie un évènement.

F.29. Memory > Index



Donne accès à un élément du tableau. L'accès est donné sous la forme d'une connexion OBC Latch associée à l'élément du tableau. L'association est établie et/ou modifiée en envoyant un évènement à l'entrée supérieure (index)

du module *Index*, qui *démarre* à zéro et est toujours en mode entier. L'entrée inférieure est la connexion OBC maîtresse vers le tableau. La sortie fournit la connexion OBC Latch à l'élément tableau, sélectionné par l'entrée d'index. Le type de valeur et la précision sont bien sûr les mêmes pour les connexions OBC d'entrée et de sortie, ils sont contrôlés par les propriétés du module.

F.30. Memory > Table



Définit un tableau préinitialisé en lecture seule. Le module lui-même n'effectue aucune action. Toutes les opérations sur la table doivent être effectuées par des modules connectés à la sortie de la table, qui est une connexion OBC esclave de type tableau.

PROPRIÉTÉS:



FP Precision

modifie les valeurs dans la table.

contrôle la précision formelle de la connexion de sortie.

F.31. Macro



Fournit un conteneur pour une structure interne. Le nombre d'entrées et de sorties n'est pas fixé, il est défini par la structure interne.

PROPRIÉTÉS:

FP Precision

contrôle la précision formelle de la connexion de sortie.

Look

commute entre les apparences Large (label et noms des ports visibles) et Small (label et noms des ports invisibles).

Pin Alignment

contrôle l'alignement des ports dans la vue externe de la macro

Solid

contrôle le traitement de la macro par le moteur de Reaktor Core. Si cette option est désactivée, les limites de la macro sont transparentes, notamment pour la résolution des réinjections. Laissez cette option ACTIVÉE, à moins que vous ne sachiez ex-ac-te-ment ce que vous faites !

Icon

le bouton



charge une nouvelle icône pour la macro, et

le bouton



efface l'icône (aucune icône assignée).

Annexe G. Les macros expertes

G.1. Clipping > Clip Max / IClip Max



Le signal à l'entrée supérieure est borné supérieurement par la valeur de seuil à l'entrée du bas. Les changements de seuil ne génèrent pas d'évènement.

G.2. Clipping > Clip Min / IClip Min



Le signal à l'entrée supérieure est borné inférieurement par la valeur de seuil à l'entrée du bas. Les changements de seuil ne génèrent pas d'évènement.

G.3. Clipping > Clip MinMax / IClipMinMax



Le signal à l'entrée supérieure est borné inférieurement par la valeur de seuil à l'entrée du milieu, et supérieurement par la valeur de seuil à l'entrée du bas. Les changements de seuil ne génèrent pas d'évènement.

G.4. Math > 1 div x



Calcule l'inverse de la valeur d'entrée.

G.5. Math > 1 wrap



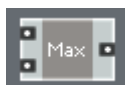
Boucle les valeurs entrantes dans l'intervalle [-0,5.. 0,5] (la période de bouclage vaut 1).

G.6. Math > Imod



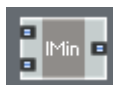
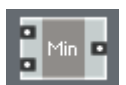
Calcule le reste de la division de la valeur supérieure par la valeur inférieure. L'évènement de sortie est envoyé à chaque fois qu'il y a un évènement à l'une des deux entrées (ou aux deux simultanément).

G.7. Math > Max / IMax



Calcule le maximum des valeurs d'entrée. L'évènement de sortie est envoyé à chaque fois qu'il y a un évènement à l'une des deux entrées (ou aux deux simultanément).

G.8. Math > Min / IMin



Calcule le minimum des valeurs d'entrée. L'évènement de sortie est envoyé à chaque fois qu'il y a un évènement à l'une des deux entrées (ou aux deux simultanément).

G.9. Math > round



Arrondit la valeur entrante à l'entier le plus proche. L'arrondi des valeurs exactement au milieu entre deux entiers n'est pas défini. Par exemple, 1,5 sera arrondi à 1 ou à 2.

G.10. Math > sign +/-



Génère en sortie soit 1, soit -1, en fonction du signe de l'entrée (les nombres positifs génèrent 1, les négatifs -1, et zéro ne génère pas de sortie).

G.11. Math > sqrt (>0)



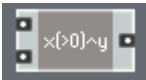
Approximation de la racine carrée. Fonctionne uniquement pour les entrées positives.

G.12. Math > sqrt



Approximation de la racine carrée.

G.13. Math > x(>0)^y



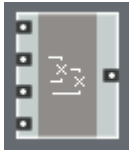
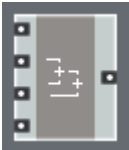
Approximation de x^y (“x puissance y”). x doit être positif. L’évènement de sortie est envoyé à chaque fois qu’il y a un évènement à l’une des deux entrées (ou aux deux simultanément).

G.14. Math > x^2 / x^3 / x^4



Calcule x à la puissance 2/3/4.

G.15. Math > Chain Add / Chain Mult



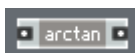
Additionne/multiplie les signaux entre eux, du bas vers le haut. L’évènement de sortie est généré s’il y a un ou plusieurs évènements à une ou plusieurs entrées.

G.16. Math > Trig-Hyp > 2 pi wrap



Boucle les valeurs entrantes sur l’intervalle $[-\pi..\pi]$ (la période de bouclage est 2π).

G.17. Math > Trig-Hyp > arcsin / arccos / arctan



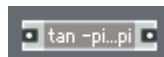
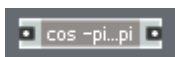
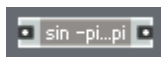
Approximation de l'arcsinus/arccosinus/arctangente.

G.18. Math > Trig-Hyp > sin / cos / tan



Approximation du sinus/du cosinus/de la tangente.

G.19. Math > Trig-Hyp > sin $-\pi..pi$ / cos $-\pi..pi$ / tan $-\pi..pi$



Approximation du sinus/du cosinus/de la tangente (fonctionne seulement sur l'intervalle $[-\pi..pi]$).

G.20. Math > Trig-Hyp > tan $-\pi/4..pi/4$



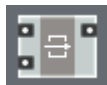
Approximation de la tangente (fonctionne seulement sur l'intervalle $[-\pi/4..pi/4]$).

G.21. Math > Trig-Hyp > sinh / cosh / tanh



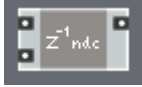
Approximation du sinus/du cosinus/de la tangente hyperboliques.

G.22. Memory > Latch / ILatch



“Bloque” (retarde) le signal à l'entrée supérieure jusqu'à ce qu'un évènement d'horloge arrive à l'entrée inférieure. Si deux évènements arrivent simultanément aux deux entrées, le signal entrant passe directement en sortie.

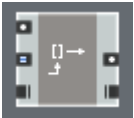
G.23. Memory > z^{-1} / z^{-1} ndc



Envoie en sortie la dernière valeur reçue à l'entrée supérieure, en réponse à l'évènement d'horloge de l'entrée inférieure. Si l'entrée d'horloge est déconnectée, le module utilise l'horloge audio standard (SR.C) à la place, fonctionnant alors de fait comme un délai d'un échantillon.

Les deux modules peuvent automatiquement résoudre les boucles de réinjection, la version z^{-1} assurant de plus l'élimination des valeurs dénormales. La version z^{-1} ndc ne doit être utilisée que lorsque les valeurs dénormales ne sont pas attendues en entrée.

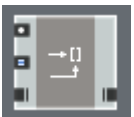
G.24. Memory > Read []



Lit une valeur dans un tableau à un index donné (spécifié par l'entrée du milieu), en réponse à un évènement d'horloge entrant (entrée supérieure). L'entrée inférieure (OBC) est la connexion au tableau.

Utilisez la sortie OBC pour créer des chaînes OBC et "sérialiser" les opérations d'accès au tableau !

G.25. Memory > Write []



Écrit une valeur (reçue à l'entrée supérieure) dans un tableau, à un index donné (spécifié par l'entrée du milieu). L'opération d'écriture est commandée par toute valeur entrante. L'entrée inférieure (OBC) est la connexion au tableau.

Utilisez la sortie OBC pour créer des chaînes OBC et "sérialiser" les opérations d'accès au tableau !

G.26. Modulation > $x + a$ / Integer > $lx + a$



Ajoute un paramètre (entrée inférieure) au signal (entrée supérieure) en réponse à un signal entrant. Les changements du paramètre ne génèrent pas d'évènements.

G.27. Modulation > $x * a$ / Integer > $lx * a$



Multiplie le signal (entrée supérieure) par un paramètre (entrée inférieure) en réponse à un signal entrant. Les changements du paramètre ne génèrent pas d'évènements.

G.28. Modulation > $x - a$ / Integer > $lx - a$



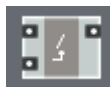
Soustrait un paramètre (entrée inférieure) au signal (entrée supérieure) en réponse à un signal entrant. Les changements du paramètre ne génèrent pas d'évènements.

G.29. Modulation > $a - x$ / Integer > $la - x$



Soustrait le signal (entrée supérieure) à un paramètre (entrée inférieure) en réponse à un signal entrant. Les changements du paramètre ne génèrent pas d'évènements.

G.30. Modulation > x / a



Divise le signal (entrée supérieure) par un paramètre (entrée inférieure) en réponse à un signal entrant. Les changements du paramètre ne génèrent pas d'évènements.

G.31. Modulation > a / x



Divise un paramètre (entrée inférieure) par le signal (entrée supérieure) en réponse à un signal entrant. Les changements du paramètre ne génèrent pas d'évènements.

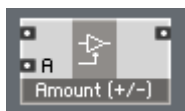
G.32. Modulation > xa + y



Multiplie le signal à l'entrée supérieure par un paramètre de gain (entrée du milieu) et ajoute le résultat au signal de l'entrée inférieure. Les évènements à l'une des deux entrées signal (ou aux deux) génèrent une nouvelle valeur, contrairement aux évènements arrivant à l'entrée paramètre.

Annexe H. Les macros standard

H.1. Audio Mix-Amp > Amount



Fournit un contrôle linéaire et inversible sur l'amplitude d'un signal audio.

A = 0	coupe le signal
A = 1	laisse le signal intact
A = -1	inverse le signal

Usage typique: contrôler la quantité de réinjection audio.

H.2. Audio Mix-Amp > Amp Mod



Module l'amplitude du signal audio par une certaine quantité (AM) sur une échelle linéaire

AM = 1	double l'amplitude
AM = 0	pas de changement
AM = -1	coupe le signal

Usage typique: trémolo, modulation d'amplitude.

H.3. Audio Mix-Amp > Audio Mix



Mixe deux signaux ensemble.

H.4. Audio Mix-Amp > Audio Relay



Commute entre les deux signaux audio d'entrée. Si 'x' est plus grand que 0, le module prend le signal 1, sinon il prend le signal 0.

H.5. Audio Mix-Amp > Chain (amount)

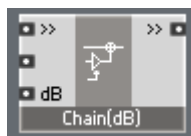


Pondère l'amplitude du signal audio par une certaine quantité (A) et mélange le résultat avec le signal audio enchaîné (>>).

A = 0	le signal est coupé (mis en sourdine)
A = 1	le signal reste inchangé
A = -1	le signal est inversé

Usage typique: chaînes de mixage audio, contrôle de la quantité de réinjection audio.

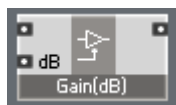
H.6. Audio Mix-Amp > Chain (dB)



Pondère l'amplitude du signal audio par une certaine quantité de décibels (dB) et mélange le résultat avec le signal audio enchaîné (>>).

Usage typique: chaînes de mixage audio.

H.7. Audio Mix-Amp > Gain (dB)



Pondère l'amplitude du signal audio par une certaine quantité de décibels (dB).

+6 dB	double l'amplitude
0 dB	pas de changement
-6 dB	diminue l'amplitude de moitié

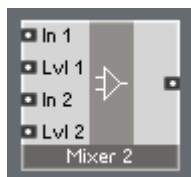
Usage typique: contrôle du volume du signal, en échelle de dB.

H.8. Audio Mix-Amp > Invert



Inverse la polarité du signal audio.

H.9. Audio Mix-Amp > Mixer 2 ... 4



Mixe les signaux audio entrants (In1, In2, ...) en atténuant leurs niveaux d'une certaine quantité de dB (Lvl1, Lvl2, ...).

H.10. Audio Mix-Amp > Pan



Règle la balance du signal audio en utilisant une courbe parabolique.

Pos = -1 tout à gauche

Pos = 0 centre

Pos = 1 tout à droite

H.11. Audio Mix-Amp > Ring-Amp Mod



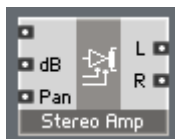
Le signal audio porteur (entrée du haut) est modulé par le signal audio de l'entrée "Mod". Le type de modulation est contrôlé par l'entrée "R/A", qui effectue un fondu entre la modulation en anneau et la modulation d'amplitude.

R/A = 0 modulation en anneau

R/A = 1 modulation d'amplitude

(pour une vraie modulation d'amplitude, l'amplitude du modulateur doit pas dépasser 1)

H.12. Audio Mix-Amp > Stereo Amp



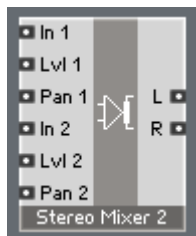
Amplifie un signal audio monophonique d'une certaine quantité de décibels et le place dans le champ panoramique à la position spécifiée. La position panoramique est définie comme suit:

-1 tout à gauche

0 centre

1 tout à droite

H.13. Audio Mix-Amp > Stereo Mixer 2 ... 4



Mixe les signaux audio entrants (In1, In2, ...) en atténuant leurs niveaux d'une certaine quantité de dB (Lvl1, Lvl2, ...) et en les plaçant dans le champ stéréo aux positions spécifiées (Pan1, Pan2, ...). Les positions panoramiques sont définies comme suit:

- 1 tout à gauche
- 0 centre
- 1 tout à droite

H.14. Audio Mix-Amp > VCA



Amplificateur audio avec contrôle linéaire de l'amplitude.

- A = 0 coupe le signal (sourdisse)
- A = 1 laisse le signal inchangé

Usage typique: connecter une enveloppe d'amplitude à l'entrée A.

Remarque: pour une amplification négative, utilisez le module *Audio Amount*.

H.15. Audio Mix-Amp > XFade (lin)

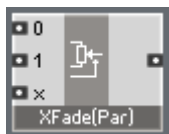


Crossfader (mélangeur) audio avec courbe linéaire.

- x = 0 seul le signal 0 est entendu
- x = 0,5 mélange égal des deux signaux
- x = 1 seul le signal 1 est entendu

Remarque: le crossfader parabolique donne généralement de meilleurs résultats sonores.

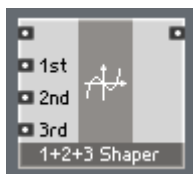
H.16. Audio Mix-Amp > XFade (par)



Crossfader (mélangeur) audio avec courbe parabolique. Donne généralement de meilleurs résultats sonores que le crossfader linéaire.

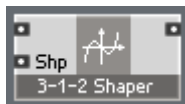
$x = 0$	seul le signal 0 est entendu
$x = 0,5$	mélange égal des deux signaux
$x = 1$	seul le signal 1 est entendu

H.17. Audio Shaper > 1+2+3 Shaper



Fournit un modelage contrôlable du signal audio du 2^{ème} et du 3^{ème} ordre. L'entrée "1st" spécifie la quantité de signal original (1=inchangé, 0=absent). Les entrées "2nd" et "3rd" correspondent respectivement aux quantités de distorsion du deuxième et du troisième ordre.

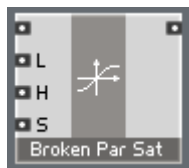
H.18. Audio Shaper > 3-1-2 Shaper



Modeleur du signal audio avec quantité variable de distorsion du 2^{ème} ou du 3^{ème} ordre. La quantité de distorsion et son type sont contrôlés par l'entrée "Shp":

Shp = 0	pas de modelage
Shp > 0	modelage du 3 ^{ème} ordre
Shp < 0	modelage du 2 ^{ème} ordre

H.19. Audio Shaper > Broken Par Sat



Saturation parabolique brisée. Segment linéaire autour du niveau zéro.

L'entrée "L" règle le niveau de sortie de la "saturation totale" (1 par défaut).

L'entrée "H" règle la dureté (entre 0 et 1). Les valeurs plus élevées correspondent à un segment linéaire plus important au milieu.

L'entrée "S" contrôle la symétrie de la courbe de saturation (de -1 à 1). À 0, la courbe est symétrique.

H.20. Audio Shaper > Hyperbol Sat



Saturation hyperbolique simple. L'entrée "L" règle le niveau de sortie de la "saturation totale" (1 par défaut). Cependant, la "saturation totale" n'est jamais atteinte avec ce type de saturation.

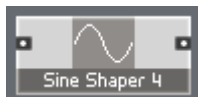
H.21. Audio Shaper > Parabol Sat



Saturation parabolique simple. L'entrée "L" règle le niveau de sortie de la "saturation totale" (1 par défaut).

Remarque: la saturation totale est atteinte avec un niveau d'entrée égal à 2L.

H.22. Audio Shaper > Sine Shaper 4 / 8



Modeleur en sinus du 4^{ème} / 8^{ème} ordre. Le modeleur du 8^{ème} ordre effectue une meilleure approximation du sinus, mais il consomme plus de puissance processeur.

H.23. Control > Ctl Amount

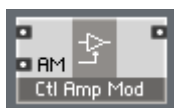


Fournit un contrôle linéaire (éventuellement négatif) sur l'amplitude d'un signal de contrôle.

- A = 0 coupe le signal
- A = 1 laisse le signal intact
- A = -1 inverse le signal

Usage typique: contrôler la quantité de modulation.

H.24. Control > Ctl Amp Mod



Module l'amplitude du signal de contrôle par une certaine quantité (AM) sur une échelle linéaire.

- AM = 1 double l'amplitude
- AM = 0 pas de changement
- AM = -1 coupe le signal

H.25. Control > Ctl Bi2Uni



Convertit un signal bipolaire de -1 à 1 en signal unipolaire. L'entrée "a" contrôle la quantité de conversion: à 0 il n'y a aucun changement, à 1 (valeur

par défaut) la conversion est totale.

Usage typique: connecter immédiatement après un LFO pour ajuster la polarité de la modulation.

H.26. Control > Ctl Chain



Pondère l'amplitude du signal audio par une certaine quantité (A) et mélange le résultat avec le signal audio enchaîné (>>).

- | | |
|--------|---------------------------------------|
| A = 0 | le signal est coupé (mis en sourdine) |
| A = 1 | le signal reste inchangé |
| A = -1 | le signal est inversé |

Usage typique: chaînes de mixage audio.

H.27. Control > Ctl Invert



Inverse la polarité du signal de contrôle.

H.28. Control > Ctl Mix



Mixe deux signaux de contrôle.

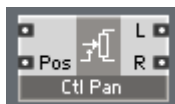
H.29. Control > Ctl Mixer 2



Mixe deux signaux de contrôle (In1 et In2) avec les facteurs de gain spécifiés (A1 et A2).

- | | |
|--------|-----------------|
| A = 0 | pas de signal |
| A = 1 | signal inchangé |
| A = -1 | signal inversé |

H.30. Control > Ctl Pan



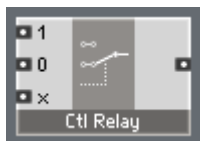
Règle la balance du signal de contrôle en utilisant une courbe parabolique

Pos = -1 tout à gauche

Pos = 0 centre

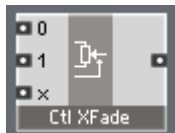
Pos = 1 tout à droite

H.31. Control > Ctl Relay



Commute entre deux signaux de contrôle. Si 'x' est plus grand que 0, le module transmet le signal 1, sinon il transmet le signal 0.

H.32. Control > Ctl XFade



Crossfader (mélangeur) pour signaux de contrôle avec courbe linéaire.

x = 0 seul le signal 0 est transmis

x = 0,5 mélange égal des deux signaux

x = 1 seul le signal 1 est transmis

H.33. Control > Par Ctl Shaper



Applique une double courbe parabolique au signal de contrôle. Le signal d'entrée doit être dans l'intervalle [-1..1]. Le signal de sortie est aussi dans [-1..1]. La quantité de torsion est contrôlée par l'entrée "b" (dont l'intervalle est encore une fois [-1..1]).

b = 0 pas de torsion (courbe linéaire)

b = -1 torsion maximale "vers" l'axe des X

b = 1 torsion maximale "vers" l'axe des Y

Vous pouvez aussi utiliser ce modeleur pour les signaux dans l'intervalle [0..1], auquel cas seule une moitié de la courbe est utilisée.
Usage typique: vitesse et autres modélisations de contrôleurs.

H.34. Convert > dB2AF



Convertit un signal de contrôle de l'échelle de décibels à l'échelle linéaire de gain d'amplitude.

0 dB	→	1.0
-6 dB	→	0.5
etc.		

H.35. Convert > dP2FF



Convertit un signal de contrôle de l'échelle de pitch (hauteur tonale, en demi-tons) à l'échelle des ratios de fréquences.

12 demi-tons	→	2
-12 demi-tons	→	-2
etc.		

H.36. Convert > logT2sec



Convertit le temps logarithmique du niveau primaire de Reaktor (utilisé pour les enveloppes) en secondes.

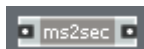
0	→	0.001 sec
60	→	1 sec
etc.		

H.37. Convert > ms2Hz



Convertit une période temporelle en millisecondes dans la fréquence correspondante en Hertz. P.ex. 100 ms → 10 Hz.

H.38. Convert > ms2sec



Convertit le temps en millisecondes en un temps en secondes. P.ex. 500ms → 0.5 sec.

H.39. Convert > P2F



Convertit un signal de contrôle de l'échelle de pitch (hauteur tonale) à l'échelle de fréquences. P.ex. pitch 69 → 440 Hz.

H.40. Convert > sec2Hz



Convertit une période temporelle en secondes dans la fréquence correspondante en Hertz. P.ex. 0.1sec → 10 Hz.

H.41. Delay > 2 / 4 Tap Delay 4p



Délai à 2/4 battements avec 4 points d'interpolation. Les entrées T1 à T4 spécifient la durée de délai (en secondes) pour chacun des battements. La durée maximale de délai est de 44100 échantillons par défaut, ce qui fait 1 seconde à 44,1 kHz. Pour ajuster cette durée, modifiez la taille du tableau dans la macro.

H.42. Delay > Delay 1p / 2p / 4p



Délai 1-point (non interpolé) / 2-points / 4-points. L'entrée T spécifie la durée du délai en secondes. La durée maximale de délai est de 44100 échantillons par défaut, ce qui fait 1 seconde à 44,1 kHz. Pour ajuster cette durée, modifiez la taille du tableau dans la macro.

Utilisez les versions interpolées du délai pour réaliser des délais modulés. Pour les délais non modulés (durée fixe de délai), la version non interpolée donne habituellement de meilleurs résultats.

H.43. Delay > Diff Delay 1p / 2p / 4p



Délai à diffusion 1-point (non interpolé) / 2-points / 4-points. L'entrée T spécifie la durée du délai en secondes. Les entrées Dffs règlent les facteurs de diffusion.

La durée maximale de délai est de 44100 échantillons par défaut, ce qui fait 1 seconde à 44,1 kHz. Pour ajuster cette durée, modifiez la taille du tableau dans la macro.

H.44. Envelope > ADSR



Génère une enveloppe ADSR.

- A, D, R stemps d'attaque, de décroissance (ou chute) et de relâchement (ou extinction) en secondes.
- S niveau de maintien (intervalle entre 0 et 1, à 1 le niveau de maintien est égal au niveau maximal).
- G entrée gate. Les évènements positifs entrants (re)démarrent l'enveloppe. Les évènements nuls ou négatifs arrêtent l'enveloppe.
- GS sensibilité du gate. Pour une sensibilité nulle, le maximum de l'enveloppe a toujours une amplitude de 1. Pour une sensibilité égale à 1, le maximum de l'enveloppe est égal au niveau du gate positif.

RM mode de redéclenchement. Commute entre les modes analogique et digital d'une part, et entre les modes redémarrage et legato d'autre part. En mode "digital", l'enveloppe redémarre toujours de zéro alors qu'en mode "analogique" elle redémarre de son niveau de sortie actuel. En mode "redémarrage", les événements gate positifs redémarrent l'enveloppe, alors qu'en mode legato celle-ci redémarre seulement quand les gates passent d'une valeur négative/nulle à une valeur positive. Voici les valeurs de RM autorisées:

RM = 0 redémarrage analogique (par défaut)

RM = 1 legato analogique

RM = 2 redémarrage digital

RM = 3 legato digital

H.45. Envelope > Env Follower



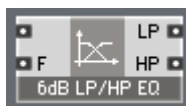
Produit en sortie un signal de contrôle qui "suit" l'enveloppe du signal audio entrant. Les entrées A et D spécifient les temps d'attaque et de décroissance du suivi, en secondes.

H.46. Envelope > Peak Detector



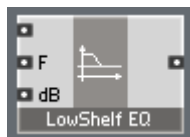
Produit en sortie le "dernier" pic du signal audio entrant, sous forme de signal de contrôle. L'entrée D spécifie le temps de décroissance du niveau de sortie, en secondes.

H.47. EQ > 6dB LP/HP EQ



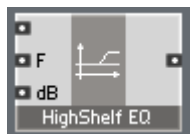
Égaliseur passe-bas (LP) / passe-haut (HP) à 1 pôle (6 dB/octave). L'entrée F spécifie la fréquence de coupure (en Hz) pour les deux sorties LP et HP.

H.48. EQ > 6dB LowShelf EQ



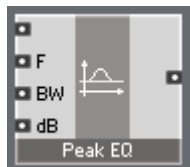
Égaliseur à étage bas à 1 pôle. L'entrée dB spécifie l'accentuation des basses fréquences, en dB (les valeurs négatives atténuent ces fréquences). L'entrée F spécifie la fréquence de transition, en Hertz.

H.49. EQ > 6dB HighShelf EQ



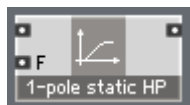
Égaliseur à étage haut à 1 pôle. L'entrée dB spécifie l'accentuation des hautes fréquences, en dB (les valeurs négatives atténuent ces fréquences). L'entrée F spécifie la fréquence de transition, en Hertz.

H.50. EQ > Peak EQ



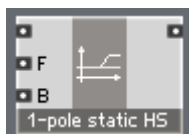
Égaliseur à pic/creux à 2 pôles. L'entrée F spécifie la fréquence centrale en Hz, l'entrée BW spécifie la largeur de bande (en anglais bandwidth) en octaves, et l'entrée dB spécifie la hauteur du pic (les valeurs négatives produisant un creux).

H.51. EQ > Static Filter > 1-pole static HP



Filtre passe-haut statique à 1 pôle. L'entrée F spécifie la fréquence de coupure en Hz.

H.52. EQ > Static Filter > 1-pole static HS



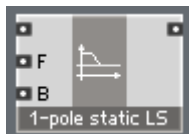
Filtre à étage haut statique à 1 pôle. L'entrée F spécifie la fréquence de coupure en Hz, et l'entrée B l'accentuation des hautes fréquences en dB.

H.53. EQ > Static Filter > 1-pole static LP



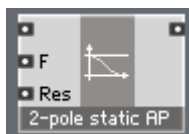
Filtre passe-bas statique à 1 pôle. L'entrée F spécifie la fréquence de coupure en Hz.

H.54. EQ > Static Filter > 1-pole static LS



Filtre à étage bas statique à 1 pôle. L'entrée F spécifie la fréquence de coupure en Hz, et l'entrée B l'accentuation des basses fréquences en dB.

H.55. EQ > Static Filter > 2-pole static AP



Filtre passe-tout statique à 2 pôles. L'entrée F spécifie la fréquence de coupure en Hz, et l'entrée Res la résonance (entre 0 et 1).

H.56. EQ > Static Filter > 2-pole static BP



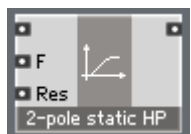
Filtre passe-bande statique à 2 pôles. L'entrée F spécifie la fréquence centrale en Hz, et l'entrée Res la résonance (entre 0 et 1).

H.57. EQ > Static Filter > 2-pole static BP1



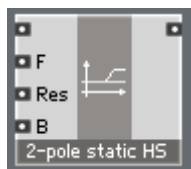
Filtre passe-bande statique à 2 pôles. L'entrée F spécifie la fréquence centrale en Hz, et l'entrée Res la résonance (entre 0 et 1). L'amplification à la fréquence centrale est toujours égale à 1, quelle que soit la résonance.

H.58. EQ > Static Filter > 2-pole static HP



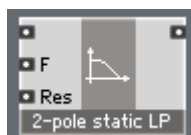
Filtre passe-haut statique à 2 pôles. L'entrée F spécifie la fréquence de coupure en Hz, et l'entrée Res la résonance (entre 0 et 1).

H.59. EQ > Static Filter > 2-pole static HS



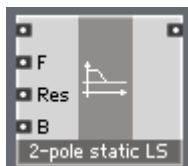
Filtre à étage haut statique à 2 pôles. L'entrée F spécifie la fréquence de coupure en Hz, l'entrée Res spécifie la résonance (entre 0 et 1) et l'entrée B l'accentuation des hautes fréquences en dB.

H.60. EQ > Static Filter > 2-pole static LP



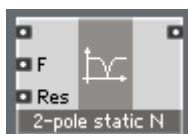
Filtre passe-bas statique à 2 pôles. L'entrée F spécifie la fréquence de coupure en Hz, et l'entrée Res la résonance (entre 0 et 1).

H.61. EQ > Static Filter > 2-pole static LS



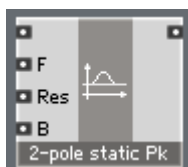
Filtre à étage bas statique à 2 pôles. L'entrée F spécifie la fréquence de coupure en Hz, l'entrée Res spécifie la résonance (entre 0 et 1) et l'entrée B l'accentuation des hautes fréquences en dB.

H.62. EQ > Static Filter > 2-pole static N



Filtre en creux statique à 2 pôles. L'entrée F spécifie la fréquence de coupure en Hz, et l'entrée Res la résonance (entre 0 et 1).

H.63. EQ > Static Filter > 2-pole static Pk



Filtre en pic statique à 2 pôles. L'entrée F spécifie la fréquence centrale en Hz, l'entrée Res la résonance (entre 0 et 1) et l'entrée B l'accentuation à la fréquence centrale en dB.

H.64. EQ > Static Filter > Integrator



Intègre le signal audio entrant (calcule sa primitive) via la méthode des sommes rectangulaires. Un évènement à l'entrée Rst réinitialise la sortie de l'intégrateur à la valeur de cet évènement.

H.65. Event Processing > Accumulator



Calcule la somme des valeurs arrivant à l'entrée supérieure. Un évènement à l'entrée Set réinitialise la sortie à la valeur de cet évènement. La valeur de la sortie inférieure est la somme de tous les évènements précédents, la valeur de la sortie supérieure est la somme de tous les évènements précédents sauf le dernier.

H.66. Event Processing > Clk Div



Diviseur de fréquence d'horloge. Les évènements d'horloge arrivant à l'entrée supérieure sont filtrés, le module ne laissant passer que le 1er, le $N+1^{\text{ème}}$, le $2N+1^{\text{ème}}$, etc. (N est la valeur à l'entrée inférieure, elle spécifie le ratio de la division).

H.67. Event Processing > Clk Gen



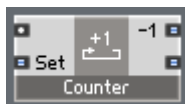
Génère des évènements d'horloge à la fréquence spécifiée par l'entrée (en Hz). Ce module fonctionne uniquement à l'intérieur des cellules core audio.

H.68. Event Processing > Clk Rate



Estime la fréquence et la période des évènements du signal d'horloge entrant. La sortie F est la fréquence en Hz, et la sortie T la période en secondes. Ce module fonctionne uniquement à l'intérieur des cellules core audio. La valeur initiale de la période est zéro, et la fréquence est une valeur très grande. Vous obtenez une estimation raisonnable en sortie seulement après le deuxième évènement d'horloge en entrée.

H.69. Event Processing > Counter



Compte le nombre d'évènements arrivant à l'entrée supérieure. Un évènement à l'entrée Set réinitialise la sortie à la valeur de cet évènement. La valeur de la sortie inférieure est le décompte de tous les évènements précédents, et la valeur de la sortie supérieure est le décompte de tous les évènements précédents sauf le dernier.

H.70. Event Processing > Ctl2Gate



Convertit le signal de contrôle (ou audio) de l'entrée supérieure en signal gate (signal en porte) avec pour amplitude la valeur à l'entrée inférieure A. Les passages du signal à zéro dans le sens croissant ouvrent la porte, les passages à zéro dans le sens décroissant ferment la porte.

H.71. Event Processing > Dup Flt / IDup Flt



Filtre les évènements en double (seuls les évènements avec une valeur différente de l'évènement précédent sont transmis).

H.72. Event Processing > Impulse



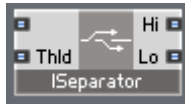
Génère une impulsion d'un échantillon, d'amplitude 1, en réponse à un évènement entrant. Ce module fonctionne uniquement dans les cellules core audio.

H.73. Event Processing > Random



Génère des nombres aléatoires en réponse aux horloges entrantes. L'intervalle de sortie est [-1..1]. Un évènement à l'entrée Seed ("Graine" en anglais) relance le générateur avec la valeur de cet évènement.

H.74. Event Processing > Separator / ISeparator



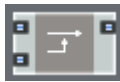
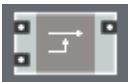
Les évènements à l'entrée supérieure ayant une valeur supérieure à la valeur de l'entrée Thld (pour "Threshold", "Seuil" en anglais) sont envoyés à la sortie Hi. Les autres sont envoyés à la sortie Lo.

H.75. Event Processing > Thld Crossing



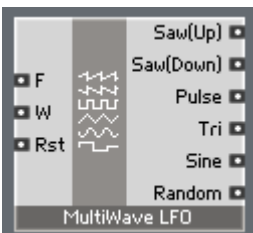
Dès qu'un signal croissant, à l'entrée supérieure, dépasse le seuil fixé par l'entrée inférieure, un évènement est envoyé à la sortie supérieure.

H.76. Event Processing > Value / IValue



Change la valeur de l'évènement arrivant à l'entrée supérieure et lui attribue la valeur de l'évènement disponible à cet instant à l'entrée inférieure.

H.77. LFO > MultiWave LFO



Génère simultanément en sortie plusieurs oscillateurs basse fréquence (LFOs) calés en phase, avec différentes formes d'onde. L'entrée F spécifie la fréquence en Hz, l'entrée W la largeur du train d'impulsions (dans l'intervalle de -1 à 1, valable uniquement pour la sortie "train d'impulsions"), les événements à l'entrée Rst redémarrent le LFO avec la phase spécifiée par la valeur de l'évènement (dans l'intervalle de 0 à 1).

H.78. LFO > Par LFO



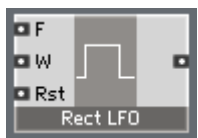
Génère un signal de contrôle basse fréquence parabolique. L'entrée F spécifie la fréquence en Hz et les événements à l'entrée Rst redémarrent le LFO avec la phase spécifiée par la valeur de l'évènement (dans l'intervalle de 0 à 1).

H.79. LFO > Random LFO



Génère un signal de contrôle basse fréquence aléatoire (méthode "random sample-and-hold", littéralement "échantillon-et-maintien aléatoire"). L'entrée F spécifie la fréquence en Hz et les événements à l'entrée Rst redémarrent le LFO avec la phase spécifiée par la valeur de l'évènement (dans l'intervalle de 0 à 1).

H.80. LFO > Rect LFO



Génère un signal de contrôle basse fréquence rectangulaire (ou train d'impulsions). L'entrée F spécifie la fréquence en Hz, l'entrée W contrôle la largeur des impulsions (dans l'intervalle de -1 à 1), et les événements à l'entrée Rst redémarrent le LFO avec la phase spécifiée par la valeur de l'évènement (dans l'intervalle de 0 à 1).

H.81. LFO > Saw(down) LFO



Génère un signal de contrôle basse fréquence en dents de scie descendantes. L'entrée F spécifie la fréquence en Hz et les événements à l'entrée Rst redémarrent le LFO avec la phase spécifiée par la valeur de l'évènement (dans l'intervalle de 0 à 1).

H.82. LFO > Saw(up) LFO



Génère un signal de contrôle basse fréquence en dents de scie montantes. L'entrée F spécifie la fréquence en Hz et les événements à l'entrée Rst redémarrent le LFO avec la phase spécifiée par la valeur de l'évènement (dans l'intervalle de 0 à 1).

H.83. LFO > Sine LFO



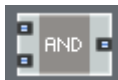
Génère un signal de contrôle basse fréquence sinusoïdal. L'entrée F spécifie la fréquence en Hz et les événements à l'entrée Rst redémarrent le LFO avec la phase spécifiée par la valeur de l'évènement (dans l'intervalle de 0 à 1).

H.84. LFO > Tri LFO



Génère un signal de contrôle basse fréquence triangulaire. L'entrée F spécifie la fréquence en Hz et les événements à l'entrée Rst redémarrent le LFO avec la phase spécifiée par la valeur de l'évènement (dans l'intervalle de 0 à 1).

H.85. Logic > AND



Effectue la conjonction (opération logique ET) de deux signaux: la sortie vaut 1 si et seulement si les deux entrées valent 1. Pour les valeurs d'entrées autres que 0 et 1, le résultat est indéfini.

H.86. Logic > Flip Flop



La sortie est commutée entre 0 et 1 à chaque fois que l'entrée d'horloge reçoit un évènement.

H.87. Logic > Gate2L



Convertit un signal de gate en signal logique. La "porte ouverte" produit la valeur 1 en sortie, la "porte fermée" produit la valeur 0.

H.88. Logic > GT / IGT



Compare les deux valeurs flottantes/entières entrantes puis génère 1 si la valeur du haut est plus grande que celle du bas, et 0 dans le cas contraire.

H.89. Logic > EQ



Compare les deux valeurs entières entrantes et génère 1 si elles sont égales, et 0 dans le cas contraire.

H.90. Logic > GE



Compare les deux valeurs entières entrantes et génère 1 si la valeur du haut est plus grande que (ou égale à) celle du bas, et 0 dans le cas contraire.

H.91. Logic > L2Clock



Convertit un signal logique en signal d'horloge. Le passage du signal d'entrée de 0 à 1 envoie l'évènement d'horloge. Pour les valeurs d'entrées autres que 0 et 1, le résultat est indéfini..

H.92. Logic > L2Gate



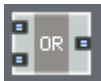
Convertit un signal logique en signal gate. Le passage du signal d'entrée de 0 à 1 ouvre la porte, le passage de l'entrée de 1 à 0 la ferme. Le niveau du signal gate "porte ouverte" est défini par la valeur à l'entrée inférieure (par défaut = 1). Pour les valeurs d'entrées autres que 0 et 1, le résultat est indéfini.

H.93. Logic > NOT



Convertit les 0 en 1 et vice versa. Pour les valeurs d'entrées autres que 0 et 1, le résultat est indéfini.

H.94. Logic > OR



Effectue la disjonction (opération logique OU) de deux signaux logiques: la sortie vaut 1 si au moins une des deux entrées vaut 1. Pour les valeurs d'entrées autres que 0 et 1, le résultat est indéfini.

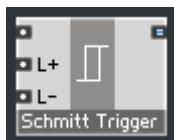
H.95. Logic > XOR



Effectue une disjonction exclusive (opération logique OU EXCLUSIF) de deux

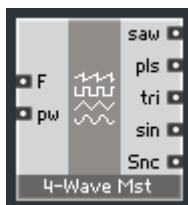
signaux logiques: la sortie vaut 1 si une seule des entrées vaut 1 (et l'autre 0). Pour les valeurs d'entrées autres que 0 et 1, le résultat est indéfini.

H.96. Logic > Schmitt Trigger



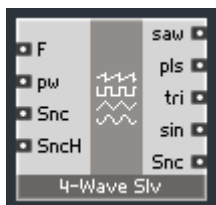
Passe la sortie à 1 si l'entrée devient supérieure à L+ (par défaut 0,67), et à 0 si l'entrée devient inférieure à L- (par défaut 0,33).

H.97. Oscillators > 4-Wave Mst



Génère quatre formes d'onde audio calées en phase. La fréquence est spécifiée par l'entrée F (en Hz). La largeur de l'impulsion (pour le train d'impulsions) est spécifiée par l'entrée 'pw', dans l'intervalle de -1 à 1. Cet oscillateur peut fonctionner avec des fréquences négatives et dispose de plus d'une sortie de synchronisation pour l'oscillateur esclave équivalent, *4-Wave Slv*.

H.98. Oscillators > 4-Wave Slv



Génère quatre formes d'onde audio calées en phase. La fréquence est spécifiée par l'entrée F (en Hz). La largeur de l'impulsion (pour le train d'impulsions) est spécifiée par l'entrée 'pw', dans l'intervalle de -1 à 1.

Cet oscillateur peut fonctionner avec des fréquences négatives. Il peut aussi être synchronisé à un autre oscillateur *4-Wave Mst/Slv* via l'entrée Snc. L'entrée SncH contrôle la dureté de la synchronisation (0 = pas de synchro, de 0 à 1).

= degré croissant de synchronisation souple). Une sortie de synchronisation pour un autre oscillateur *4-Wave S/v* est également disponible.

H.99. Oscillators > Binary Noise



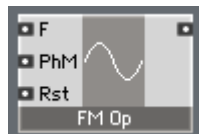
Générateur de bruit blanc binaire. Produit aléatoirement les valeurs -1 ou 1 . Un évènement entrant à l'entrée Seed (ré)initialise le générateur aléatoire interne avec une valeur "graine" donnée (la valeur de l'évènement).

H.100. Oscillators > Digital Noise



Générateur de bruit blanc numérique. Produit des valeurs aléatoires dans l'intervalle $[-1..1]$. Un évènement entrant à l'entrée Seed (ré)initialise le générateur aléatoire interne avec une valeur "graine" donnée (la valeur de l'évènement

H.101. Oscillators > FM Op



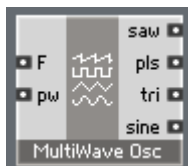
Opérateur classique de FM (Modulation de Fréquence). Produit une onde sinusoïdale dont la fréquence est définie par l'entrée F (en Hz). Le sinus peut être modulé en phase par l'entrée PhM (en radians). Un évènement arrivant à l'entrée Rst redémarre l'oscillateur à la phase spécifiée par la valeur de cet évènement (intervalle de 0 à 1).

H.102. Oscillators > Formant Osc



Génère une forme d'onde avec une fréquence fondamentale spécifiée par l'entrée F (en Hz) et une fréquence de formant spécifiée par l'entrée Fmt (en Hz aussi).

H.103. Oscillators > MultiWave Osc



Génère quatre formes d'onde audio calées en phase. La fréquence est spécifiée par l'entrée F (en Hz). La largeur de l'impulsion (pour le train d'impulsions) est spécifiée par l'entrée 'pw', dans l'intervalle de -1 à 1 . Cet oscillateur ne peut pas fonctionner avec des fréquences négatives.

H.104. Oscillators > Par Osc



Génère une forme d'onde audio parabolique. L'entrée F spécifie la fréquence en Hz.

H.105. Oscillators > Quad Osc



Génère une paire de formes d'onde sinusoïdales calées en phase, avec un décalage de phase de 90° . L'entrée F spécifie la fréquence en Hertz.

H.106. Oscillators > Sin Osc



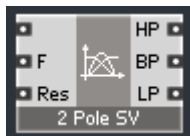
Génère une onde sinusoïdale. L'entrée F spécifie la fréquence en Hz.

H.107. Oscillators > Sub Osc 4



Génère quatre sous-harmoniques calées en phase. La fréquence “fondamentale” est spécifiée par l’entrée F (en Hz). Les nombres sous-harmoniques sont spécifiés par les entrées S1 à S4 (intervalle 1..120). L’entrée Tbr contrôle le contenu harmonique de la forme d’onde de sortie (intervalle 0..1).

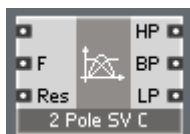
H.108. VCF > 2 Pole SV



Filtre 2-pôles à variable d’état. L’entrée F spécifie la fréquence de coupure en Hz et l’entrée Res la résonance (intervalle 0..0,98).

Les sorties HP/BP/LP produisent respectivement les signaux passe-haut, passe-bande et passe-bas.

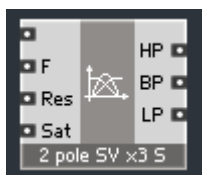
H.109. VCF > 2 Pole SV C



Filtre 2-pôles à variable d’état (version compensée). Offre un meilleur comportement aux fréquences de coupures élevées. L’entrée F spécifie la fréquence de coupure en Hz et l’entrée Res la résonance (intervalle 0..0,98). Vous pouvez également utiliser des valeurs négatives de résonance, elles augmentent encore plus la pente.

Les sorties HP/BP/LP produisent respectivement les signaux passe-haut, passe-bande et passe-bas.

H.110. VCF > 2 Pole SV (x3) S

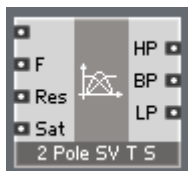


Filtre 2-pôles à variable d’état avec un suréchantillonnage optionnel (version x3) et une saturation. L’entrée F spécifie la fréquence de coupure en Hz, l’entrée Res la résonance (intervalle 0..1) et l’entrée Sat le niveau de saturation

(intervalle typique: de 8 à 32).

Les sorties HP/BP/LP produisent respectivement les signaux passe-haut, passe-bande et passe-bas.

H.111. VCF > 2 Pole SV T (S)



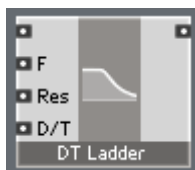
Filtre 2-pôles à variable d'état avec compensation et saturation optionnelle (version S). Offre un meilleur comportement aux fréquences de coupures élevées, quoique légèrement différent de celui de la version 2 Pole SV C. L'entrée F spécifie la fréquence de coupure en Hz, l'entrée Res la résonance (intervalle 0..1) et l'entrée Sat le niveau de saturation (intervalle typique: de 8 à 32). Les sorties HP/BP/LP produisent respectivement les signaux passe-haut, passe-bande et passe-bas.

H.112. VCF > Diode Ladder



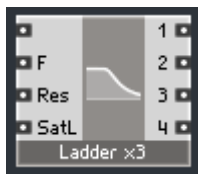
Émulation linéaire d'un filtre à échelle de diodes. L'entrée F spécifie la fréquence de coupure (en Hz) et Res spécifie la résonance (intervalle de 0 à 0,98).

H.113. VCF > D/T Ladder



Émulation linéaire de filtre à échelle, fondu possible entre les comportements "diode" et "transistor". L'entrée F spécifie la fréquence de coupure (en Hz), l'entrée Res spécifie la résonance (intervalle de 0 à 0,98) et l'entrée D/T contrôle la balance entre les diodes et les transistors (0=diode, 1=transistor).

H.114. VCF > Ladder x3



Une émulation d'un filtre à échelle de transistor saturant (suréchantillonné trois fois). L'entrée F spécifie la fréquence de coupure (en Hz), l'entrée Res spécifie la résonance (intervalle de 0 à 1) et l'entrée SatL spécifie le niveau de saturation (intervalle typique entre 1 et 32).

Les sorties 1 à 4 sont prises aux étages correspondants de l'échelle émulée. Prenez l'étage 4 pour le son de filtre à échelle "classique".

Annexe I. Core cell library

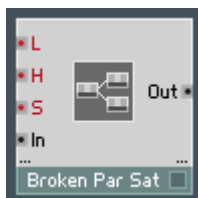
I.1. Audio Shaper > 3-1-2 Shaper



Modeleur du signal audio avec une quantité variable de distorsion du 2^{ème} ou du 3^{ème} ordre. La quantité de distorsion et son type sont contrôlés par l'entrée "Shp":

- Shp = 0 pas de modelage
- Shp > 0 modelage du 3^{ème} ordre
- Shp < 0 modelage du 2^{ème} ordre

I.2. Audio Shaper > Broken Par Sat



Saturation parabolique brisée. Segment linéaire autour du niveau zéro. L'entrée "L" règle le niveau de sortie de la "saturation totale" (1 par défaut). L'entrée "H" règle la dureté (entre 0 et 1). Les valeurs plus grandes correspondent à un segment linéaire plus grand au milieu. L'entrée "S" contrôle la symétrie de la courbe de saturation (de -1 à 1). À 0, la courbe est symétrique.

I.3. Audio Shaper > Hyperbol Sat



Saturation hyperbolique simple. L'entrée "L" règle le niveau de sortie de la "saturation totale" (1 par défaut). Cependant, la "saturation totale" n'est jamais atteinte avec ce type de saturation.

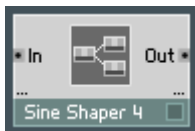
I.4. Audio Shaper > Parabol Sat



Saturation parabolique simple. L'entrée "L" règle le niveau de sortie de la "saturation totale" (1 par défaut).

Remarque: la saturation totale est atteinte avec un niveau d'entrée égal à 2L.

I.5. Audio Shaper > Sine Shaper 4/8



Modeleur en sinus du 4^{ème} / 8^{ème} ordre. Le modeleur du 8^{ème} ordre effectue une meilleure approximation du sinus, mais il consomme plus de puissance processeur.

I.6. Control > ADSR



Génère une enveloppe ADSR.

- A, D, R temps d'attaque, de décroissance (ou chute) et de relâchement (ou extinction) en secondes.
- S niveau de maintien (intervalle entre 0 et 1, à 1 le niveau de maintien est égal au niveau maximal).
- G entrée gate. Les événements positifs entrants (re)démarrent l'enveloppe. Les événements nuls ou négatifs arrêtent l'enveloppe.
- GS sensibilité du gate. Pour une sensibilité nulle, le maximum de l'enveloppe a toujours une amplitude de 1. Pour une sensibilité égale à 1, le maximum de l'enveloppe est égal au niveau du *gate* positif.

RM mode de redéclenchement. Commute entre les modes analogique et digital d'une part, et entre les modes redémarrage et legato d'autre part. En mode "digital", l'enveloppe redémarre toujours de zéro alors qu'en mode "analogique" elle redémarre de son niveau de sortie actuel. En mode "redémarrage", les événements gate positifs redémarrent l'enveloppe, alors qu'en mode legato celle-ci redémarre seulement quand les gates passent d'une valeur négative/nulle à une valeur positive. Voici les valeurs de RM autorisées:

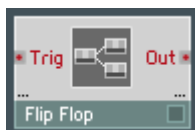
RM = 0	redémarrage analogique (par défaut)
RM = 1	legato analogique
RM = 2	redémarrage digital
RM = 3	legato digital

I.7. Control > Env Follower



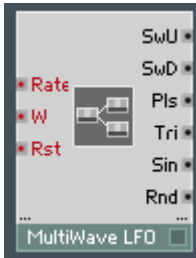
Génère en sortie un signal de contrôle qui "suit" l'enveloppe du signal audio entrant. Les entrées A et D spécifient les temps d'attaque et de décroissance du suivi, en secondes.

I.8. Control > Flip Flop



La sortie est basculée entre 0 et 1 à chaque fois que l'entrée de commande reçoit un événement.

I.9. Control > MultiWave LFO



Génère simultanément en sortie plusieurs oscillateurs basse fréquence (LFOs) calés en phase, avec différentes formes d'onde. L'entrée Rate spécifie la fréquence en Hz, l'entrée W la largeur du train d'impulsions (dans l'intervalle de -1 à 1, valable uniquement pour la sortie "train d'impulsions"), les événements à l'entrée Rst redémarrent les LFOs avec la phase spécifiée par la valeur de l'évènement (dans l'intervalle de 0 à 1).

I.10. Control > Par Ctl Shaper



Applique une double courbe parabolique au signal de contrôle. Le signal d'entrée doit être dans l'intervalle [-1..1]. Le signal de sortie est aussi dans [-1..1]. La quantité de torsion est contrôlée par l'entrée "b" (dont l'intervalle est encore une fois [-1..1]).

b = 0 pas de torsion (courbe linéaire)
b = -1 torsion maximale "vers" l'axe des X
b = 1 torsion maximale "vers" l'axe des Y

Vous pouvez aussi utiliser ce modelleur pour les signaux dans l'intervalle [0..1], auquel cas seule une moitié de la courbe est utilisée.

Usage typique: vitesse et autres modelages de contrôleurs.

I.11. Control > Schmitt Trigger



Passe la sortie à 1 si l'entrée devient supérieure à L+ (par défaut 0,67), et à 0 si l'entrée devient inférieure à L- (par défaut 0,33)..

I.12. Control > Sine LFO



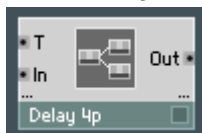
Génère un signal de contrôle basse fréquence sinusoïdal. L'entrée Rate spécifie la fréquence en Hz et les événements à l'entrée Rst redémarrent le LFO avec la phase spécifiée par la valeur de l'évènement (dans l'intervalle de 0 à 1).

I.13. Delay > 2/4 Tap Delay 4p



Délai à 2/4 battements avec 4 points d'interpolation. Les entrées T1 à T4 spécifient la durée du délai (en secondes) pour chacun des battements. La durée maximale du délai est de 44100 échantillons par défaut, ce qui fait 1 seconde à 44,1 kHz. Pour ajuster cette durée, modifiez la taille du tableau dans la macro.

I.14. Delay > Delay 4p



Délai à interpolation 4-points. L'entrée T spécifie la durée du délai en secondes. La durée maximale de délai est de 44100 échantillons par défaut, ce qui fait 1 seconde à 44,1 kHz. Pour ajuster cette durée, modifiez la taille du tableau dans la macro.

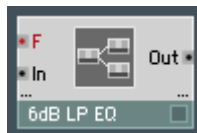
I.15. Delay > Diff Delay 4p



Délai à diffusion à interpolation 4-points. L'entrée T spécifie la durée du délai

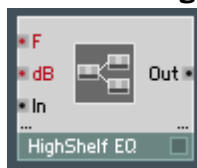
en secondes. L'entrée Dffs règle le facteur de diffusion. La durée maximale de délai est de 44100 échantillons par défaut, ce qui fait 1 seconde à 44,1 kHz. Pour ajuster cette durée, modifiez la taille du tableau dans la macro.

I.16. EQ > 6dB LP/HP EQ



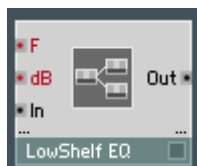
Égaliseur passe-bas (LP) / passe-haut (HP) à 1 pôle (6 dB/octave). L'entrée F spécifie la fréquence de coupure (en Hz).

I.17. EQ > HighShelf EQ



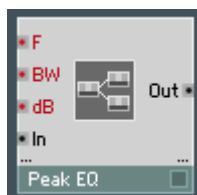
Égaliseur à étage haut à 1 pôle. L'entrée dB spécifie l'accentuation des hautes fréquences, en dB (les valeurs négatives atténuent ces fréquences). L'entrée F spécifie la fréquence de transition, en Hertz.

I.18. EQ > LowShelf EQ



Égaliseur à étage bas à 1 pôle. L'entrée dB spécifie l'accentuation des basses fréquences, en dB (les valeurs négatives atténuent ces fréquences). L'entrée F spécifie la fréquence de transition, en Hertz.

I.19. EQ > Peak EQ



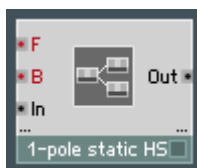
Égaliseur à pic/creux à 2 pôles. L'entrée F spécifie la fréquence centrale en Hz, l'entrée BW spécifie la largeur de bande (en anglais *bandwidth*) en octaves, et l'entrée dB spécifie la hauteur du pic (les valeurs négatives produisant un creux).

I.20. EQ > Static Filter > 1-pole static HP



Filtre passe-haut statique à 1 pôle. L'entrée F spécifie la fréquence de coupure en Hz.

I.21. EQ > Static Filter > 1-pole static HS



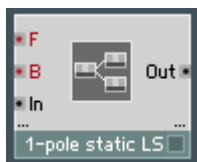
Filtre à étage haut statique à 1 pôle. L'entrée F spécifie la fréquence de coupure en Hz, et l'entrée B l'accentuation des hautes fréquences, en dB.

I.22. EQ > Static Filter > 1-pole static LP



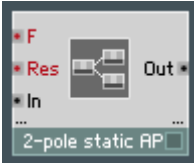
Filtre passe-bas statique à 1 pôle. L'entrée F spécifie la fréquence de coupure en Hz.

I.23. EQ > Static Filter > 1-pole static LS



Filtre à étage bas statique à 1 pôle. L'entrée F spécifie la fréquence de coupure en Hz, et l'entrée B l'accentuation des basses fréquences en dB.

I.24. EQ > Static Filter > 2-pole static AP



Filtre passe-tout statique à 2 pôles. L'entrée F spécifie la fréquence de coupure en Hz, et l'entrée Res la résonance (entre 0 et 1).

I.25. EQ > Static Filter > 2-pole static BP



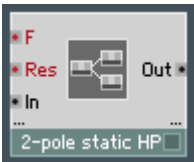
Filtre passe-bande statique à 2 pôles. L'entrée F spécifie la fréquence centrale en Hz, et l'entrée Res la résonance (entre 0 et 1).

I.26. EQ > Static Filter > 2-pole static BP1



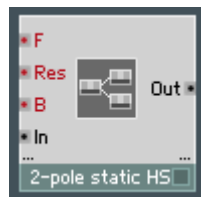
Filtre passe-bande statique à 2 pôles. L'entrée F spécifie la fréquence centrale en Hz, et l'entrée Res la résonance (entre 0 et 1). L'amplification à la fréquence centrale est toujours égale à 1, quelle que soit la résonance.

I.27. EQ > Static Filter > 2-pole static HP



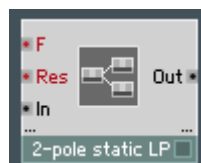
Filtre passe-haut statique à 2 pôles. L'entrée F spécifie la fréquence de coupure en Hz, et l'entrée Res la résonance (entre 0 et 1).

I.28. EQ > Static Filter > 2-pole static HS



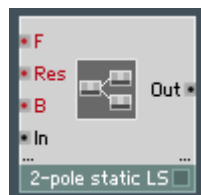
Filtre à étage haut statique à 2 pôles. L'entrée F spécifie la fréquence de coupure en Hz, l'entrée Res spécifie la résonance (entre 0 et 1) et l'entrée B l'accentuation des hautes fréquences en dB.

I.29. EQ > Static Filter > 2-pole static LP



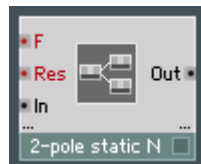
Filtre passe-bas statique à 2 pôles. L'entrée F spécifie la fréquence de coupure en Hz, et l'entrée Res la résonance (entre 0 et 1).

I.30. EQ > Static Filter > 2-pole static LS



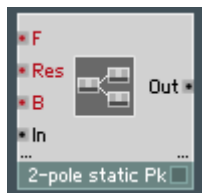
Filtre à étage bas statique à 2 pôles. L'entrée F spécifie la fréquence de coupure en Hz, l'entrée Res spécifie la résonance (entre 0 et 1) et l'entrée B l'accentuation des hautes fréquences en dB.

I.31. EQ > Static Filter > 2-pole static N



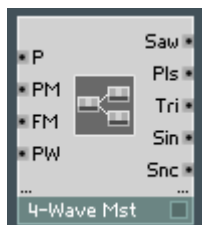
Filtre en creux statique à 2 pôles. L'entrée F spécifie la fréquence de coupure en Hz, et l'entrée Res la résonance (entre 0 et 1).

I.32. EQ > Static Filter > 2-pole static Pk



Filtre en pic statique à 2 pôles. L'entrée F spécifie la fréquence centrale en Hz, l'entrée Res la résonance (entre 0 et 1) et l'entrée B l'accentuation à la fréquence centrale en dB.

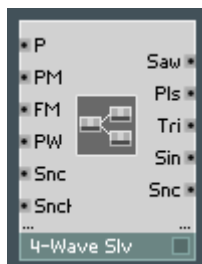
I.33. Oscillator > 4-Wave Mst



Génère quatre formes d'onde audio calées en phase. Le pitch (hauteur tonale) de l'oscillateur est spécifié par l'entrée P (sous forme de numéro de note MIDI). Il peut être modulé par l'entrée PM (en demi-tons, donc exponentiellement) et par l'entrée FM (en Hertz, donc linéairement). La largeur de l'impulsion (pour le train d'impulsions) est spécifiée par l'entrée PW, dans l'intervalle de -1 à 1.

Cet oscillateur peut fonctionner avec des fréquences négatives et dispose de plus d'une sortie de synchronisation pour l'oscillateur esclave équivalent, *4-Wave Slv*.

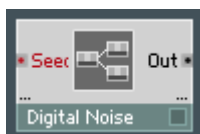
I.34. Oscillator > 4-Wave Slv



Génère quatre formes d'onde audio calées en phase. Le pitch (hauteur tonale) de l'oscillateur est spécifié par l'entrée P (sous forme de numéro de note MIDI).

Il peut être modulé par l'entrée PM (en demi-tons, donc exponentiellement) et par l'entrée FM (en Hertz, donc linéairement). La largeur de l'impulsion (pour le train d'impulsions) est spécifiée par l'entrée PW, dans l'intervalle de -1 à 1. Cet oscillateur peut fonctionner avec des fréquences négatives. Il peut aussi être synchronisé à un autre oscillateur 4-Wave Mst/Slv via l'entrée Snc. L'entrée SncH contrôle la dureté de la synchronisation (0 = pas de synchro, de 0 à 1 = degré croissant de synchronisation souple). Une sortie de synchronisation pour un autre oscillateur 4-Wave S/v est également disponible.

I.35. Oscillator > Digital Noise



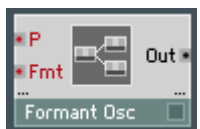
Générateur de bruit blanc numérique. Produit des valeurs aléatoires dans l'intervalle [-1..1]. Un évènement entrant à l'entrée Seed (ré)initialise le générateur aléatoire interne avec une valeur "graine" donnée (la valeur de l'évènement).

I.36. Oscillator > FM Op



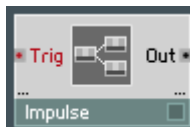
Opérateur classique de FM (Modulation de Fréquence). Produit une onde sinusoïdale dont le pitch est défini par l'entrée P (en numéro de note MIDI). Le sinus peut être modulé en phase par l'entrée PhM (en radians). Un évènement arrivant à l'entrée Rst redémarre l'oscillateur à la phase spécifiée par la valeur de cet évènement (intervalle de 0 à 1).

I.37. Oscillator > Formant Osc



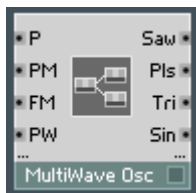
Génère une forme d'onde avec une fréquence fondamentale spécifiée par l'entrée P (en numéro de note MIDI) et une fréquence de formant spécifiée par l'entrée Fmt (en Hz aussi).

I.38. Oscillator > Impulse



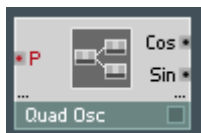
Génère une impulsion de largeur un échantillon, d'amplitude 1, en réponse à un évènement entrant.

I.39. Oscillator > MultiWave Osc



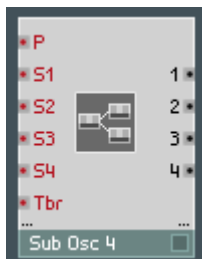
Génère quatre formes d'onde audio calées en phase. Le pitch (hauteur tonale) de l'oscillateur est spécifié par l'entrée P (sous forme de numéro de note MIDI). Il peut être modulé par l'entrée PM (en demi-tons, donc exponentiellement) et par l'entrée FM (en Hertz, donc linéairement). La largeur de l'impulsion (pour le train d'impulsions) est spécifiée par l'entrée PW, dans l'intervalle de -1 à 1. Cet oscillateur ne peut pas fonctionner avec des fréquences négatives.

I.40. Oscillator > Quad Osc



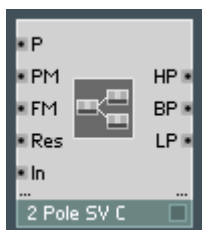
Génère une paire de formes d'onde sinusoïdales calées en phase, avec un décalage de phase de 90°. L'entrée P spécifie le pitch, en numéro de note MIDI.

I.41. Oscillator > Sub Osc



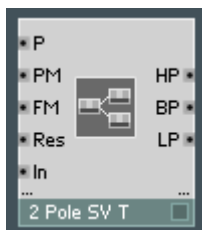
Génère quatre sous-harmoniques calées en phase. La fréquence “fondamentale” est spécifiée par l’entrée P (en numéro de note MIDI). Les nombres sous-harmoniques sont spécifiés par les entrées S1 à S4 (intervalle 1..120). L’entrée Tbr contrôle le contenu harmonique de la forme d’onde de sortie (intervalle 0..1).

I.42. VCF > 2 Pole SV C



Filtre 2-pôles à variable d’état (version compensée). Offre un meilleur comportement aux fréquences de coupures élevées. L’entrée P spécifie la fréquence de coupure (en numéro de note MIDI). Elle peut être modulée par l’entrée PM (en demi-tons, donc exponentiellement) et par l’entrée FM (en Hertz, donc linéairement). L’entrée Res spécifie la résonance (intervalle 0..0,98). Vous pouvez également utiliser des valeurs négatives de résonance, elles augmentent encore plus la pente. Les sorties HP/BP/LP produisent respectivement les signaux passe-haut, passe-bande et passe-bas.

I.43. VCF > 2 Pole SV T



Filtre 2-pôles à variable d'état avec compensation. Offre un meilleur comportement aux fréquences de coupures élevées, quoique légèrement différent de celui de la version 2 Pole SV C. L'entrée P spécifie la fréquence de coupure en numéro de note MIDI. Elle peut être modulée par l'entrée PM (en demi-tons, donc exponentiellement) et par l'entrée FM (en Hertz, donc linéairement). L'entrée Res spécifie la résonance (intervalle 0..1).

Les sorties HP/BP/LP produisent respectivement les signaux passe-haut, passe-bande et passe-bas.

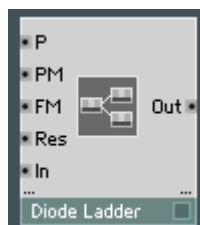
I.44. VCF > 2 Pole SV x3 S



Filtre 2-pôles à variable d'état avec un suréchantillonnage optionnel (version x3) et une saturation. L'entrée P spécifie la fréquence de coupure (en numéro de note MIDI). Elle peut être modulée par l'entrée PM (en demi-tons, donc exponentiellement) et par l'entrée FM (en Hertz, donc linéairement). L'entrée Res spécifie la résonance (intervalle 0..1) et l'entrée Sat le niveau de saturation (intervalle typique: de 8 à 32).

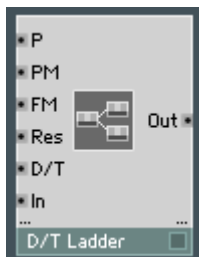
Les sorties HP/BP/LP produisent respectivement les signaux passe-haut, passe-bande et passe-bas.

I.45. VCF > Diode Ladder



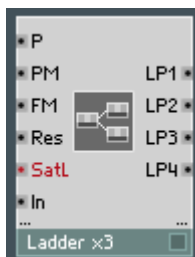
Émulation linéaire d'un filtre à échelle de diodes. L'entrée P spécifie la fréquence de coupure du filtre (en numéro de note MIDI). Elle peut être modulée par l'entrée PM (en demi-tons, donc exponentiellement) et par l'entrée FM (en Hertz, donc linéairement). L'entrée Res spécifie la résonance (intervalle de 0 à 0,98).

I.46. VCF > D/T Ladder



Émulation linéaire d'un filtre à échelle, fondu possible entre les comportements "diode" et "transistor". L'entrée P spécifie la fréquence de coupure (en numéro de note MIDI). Elle peut être modulée par l'entrée PM (en demi-tons, donc exponentiellement) et par l'entrée FM (en Hertz, donc linéairement). L'entrée Res spécifie la résonance (intervalle de 0 à 0,98) et l'entrée D/T contrôle la balance entre les diodes et les transistors (0=diode, 1=transistor).

I.47. VCF > Ladder x3



Une émulation d'un filtre à échelle de transistor saturant (suréchantillonné trois fois). L'entrée P spécifie la fréquence de coupure (en numéro de note MIDI). Elle peut être modulée par l'entrée PM (en demi-tons, donc exponentiellement) et par l'entrée FM (en Hertz, donc linéairement). L'entrée Res spécifie la résonance (intervalle de 0 à 1) et l'entrée SatL spécifie le niveau de saturation (intervalle typique entre 1 et 32). Les sorties 1 à 4 sont prises aux étages correspondants de l'échelle émulée. Prenez l'étage 4 pour le son de filtre à échelle "classique".

Index

Symbols

1/96 Clock	47, 53
1 / Square Root	62
12-Step	113
16-Step	113
4-Ramp	87, 130
4-Step	86
5-Ramp	88, 131
5-Step	87
6-Ramp	88, 133
6-Step	87, 112
8-Ramp	88
8-Step	87, 113

A

a * b+ c	57
Accumulator	169
AD-Env	123
ADBDR-Env	126
ADBDSR-Env	127
Add	55
ADR-Env	124
ADSR-Env	125
AHDBDR - Env	129
AHDSR - Env	128
Allpass 1-Pole	136
Amp/Mixer	68
Appendix	202
AR-Env	123
Arccos	64
Arcsin	63
Arctan	64
A to E	186
A to E (Perm)	186
A to E (Trig)	186

A to Gate	187
Audio Smoother	189
Audio Table	167
Audio Voice Combiner	185
Auxiliary	181

B

Beat Loop	108
Bi-Pulse	84
Bi-Saw	72
Boolean control (BoolCtl)	305
Button	24

C

Cellules.	<i>Voir</i> Cellules Core
Cellules Core	203
(Re-)nommer	225
Audio	284
Dossier utilisateur des	205
Édition basique des	209
Évènement et Audio	215
Évènements	258
Générer	217
Ports de	221
Utilisation des	204
Ch. Aftertouch	45, 51
Chopper	162
Clipper	160
Clock Oscillator	89
Comparaison de signe	316
Compare	59
Compare/Equal	59
Constant	55

Controller.....	45, 51
Counter.....	169
Crossfade.....	66
Ctrl. Shaper 1 BP.....	171
Ctrl. Shaper 2 BP.....	171
Ctrl. Shaper 3 BP.....	172

D

D-Env	119
DBDR-Env.....	121
DBDSR-Env.....	122
Debug Mode.....	278
Differentiator	150
Diffuser Delay.....	154
Distributor.....	67
Divide x/y.....	57
Diviseur de fréquence.....	166
DR-Env	119
DSR-Env	120

E

Entrées.....	<i>Voir Ports</i>
Signaux par	
default de l'entrée	238
Évènements.....	252
Le routage des.....	304
simultanés.....	255
Évènements Core. .	<i>Voir Évènements</i>
Event Smoother	189
Event Table	179
Event V.C. All.....	185
Event V.C. Max	185
Event V.C. Min	185
Expon. (A).....	60
Expon. (F).....	60

F

Fader.....	21
FP Precision.....	237, 311, 338
Frequency Divider	166, 170
From Voice	188

G

Gate	43
Geiger.....	90
Gestion dynamique des ports	20
Grain Cloud	106
Grain Cloud Delay	156
Grain Delay	155
Grain Pitch Former.....	102
Grain Resynth.....	98

H

H-Env	117
High Shelf EQ.....	148
High Shelf EQ FM	149
Hold	178
Horloge du taux	
d'échantillonnage	265, 286,
.....	288, 293, 302, 309, 328
HP/LP 1-Pole.....	135
HP/LP 1-Pole FM	136
HR-Env	118

I

IC Receive.....	200
IC Send	200
Impulse	84
Impulse FM	85
Impulse Sync.....	85
INF	300
Initialisation	269, 318
Initialisation événements.....	269,
.....	281, 285, 300, 318

In Port.....	198	Paramètre Solid	237, 293
Inputs.....	<i>See</i> Ports	Ports de	236
Integrator	151	Macro Write []	324
Invert, -X.....	56	Master Tune/Level	190
Iteration	175	Mauvaises valeurs	295, 300
K		Merge	176
Knob	24	Meter	30
L		MIDI Channel Info.....	192
Ladder Filter.....	145	Mirror 1 Level.....	161
Ladder Filter FM	146	Mirror 2 Levels	161
Lamp.....	28	Mod. Clipper	160
Latch module	264, 266,	Modulateur Chopper	162
.....	267, 275, 307, 315, 341	Module Denormal Cancel (DNC)	299
Level Lamp	29	Module ES Ctl	318
Level Meter	30	Module Merge	273, 275, 306, 342
LFO.....	116	Module R/W Order	325
List	25	Module Read	265, 272, 322
Log (A)	61	Initialisation de	269
Log (F)	61	Module Router	304, 342
Logic AND	173	Modules. <i>Voir</i> Modules Core	
Logic EXOR.....	173	Modules Core	
Logic NOT.....	174	Insérer	218
Logic OR.....	173	L'ordre de traitement.....	257
Low Shelf EQ.....	149	mode entier	313, 315
Low Shelf EQ FM	150	Modules hybrides.....	19
M		Module Write ...	265, 272, 318, 322
Macro Read []	327	Initialisation de	269
Macros.	<i>Voir</i> Macros Core	Module Z1	267, 288,
Macros à modulation	282,	289, 303, 315
.....	285, 315, 341	Modulo	58
Macros Core	236	Mouse Area.....	39
(Re-)nommer	237	Multi-Ramp	87
Création de.....	236	Multi-Sine	79
Dossier Utilisateur	280	Multi-Tap Delay.....	153
		Multi/HP 4-Pole.....	143
		Multi/HP 4-Pole FM	144
		Multi/LP 4-Pole	141
		Multi/LP 4-Pole FM.....	142
		Multi/Notch 2-Pole.....	139
		Multi/Notch 2-Pole FM	140

Multi 2-Pole	137	Pitchbend	42, 50
Multi 2-Pole FM.....	138	Poly Aftertouch.....	46, 51
Multi Display	36	Poly Display.....	36
Multi Picture	31	Ports	
Multiplex16	114	Audio.....	284
Multiply	56	Audio/Évènement	211
Multi Text.....	33	Évènements.....	259
N		Générer.....	221
NaN	300	L'ordre d'émission.....	259
Noise.....	90	L'ordre relatif	210
Note Pitch.....	42	Power x y	61
Note Pitch/Gate	50	Précision de la virgule	
Note Range Info.....	192	flottante.....	<i>Voir FP Precision</i>
O		Pro-52 Filter	145
Object Bus		Program Change	46, 52
Connections (OBC) ..	265, 312, 321	Pulse.....	80
Off Velocity	44	Pulse 1-ramp.....	82
On Velocity.....	44	Pulse 2-ramp.....	83
Order.....	174	Pulse FM	81
Oscillateur dents de scie/pics	72	Pulse Sync	81
OSC Receive	201	Q	
OSC Send	201	QNaN.....	300
Out Port.....	198	Quantize	60
P		QuickBus	227
Panner.....	67	QuickConst	239, 261
Parabol	75	QWERTY.....	202
Par FM	75	R	
Par PWM	77	Ramp	88
Par Sync	76	Random	90
Peak Detector.....	166	Randomizer	170
Peak EQ.....	147	Receive.....	198
Peak EQ FM.....	147	Reciprocal 1/x	57
Picture.....	31	Rect./Sign	58
		Rectifier.....	58
		Réinjection	287
		Indication de	234, 288

Réinjection et macros	291
Résolution de.....	289, 315
Relay 1,2	66
RGB Lamp	29
Router 1,2	177
Router 1-M.....	178
Router M-1	177

S

Sample & Hold	166
Sample Lookup.....	110
Sampler	94
Sampler FM	94
Sampler Loop	96
Saturator.....	159
Saturator 2.....	159
Saw FM	70
Saw Pulse	72
Saw Sync	71
Sawtooth	70
Scanner	65
Scope.....	34
Sel. Note Gate	44
Sel. Poly AT.....	46, 52
Selector	65
Send	198
Separator	175
Set Random	195
Shaper 1 BP	162
Shaper 2 BP	163
Shaper 3 BP	163
Shaper Cubic.....	165
Shaper Parabolic.....	164
Signaux	
Audio.....	229
contrôle	229
entiers	312, 344
événements	245
flottants	310, 344

Horologe. . Voir Signaux d'horloge	
logiques	249
Signaux d'horloge.....	264
Signaux par default	
de l'entrée.....	Voir Entrées
Sine	62, 77
Sine/Cosine	63
Sine FM	78
Sine Sync.....	78
Single Delay	152
Single Trig. Gate	43
Slew Limiter	165
Slow Random	117
Snapshot	193
Snap Value.....	195
Song Pos	48, 53
Sorties.....	Voir Ports
Sorties Audio.....	285
Square Root	62
Stacked Macro.....	41
Start/Stop	47, 52
Step Filter	176
Stereo Amp	68
Stereo Pan	67
Structure Core	210
Subtract.....	56
Switch	27
Sync Clock	48
System Info.....	191
SŽquenceur.....	112

T

Tableaux	320
Tables.....	336
Tapedeck 1-Ch	181
Tapedeck 2-Ch	184
Tempo Info.....	190
Text.....	32
Timer	178

To Voice.....	187
Transposing.....	202
Tri/Par Symm	74
Triangle.....	73
Tri FM	73
Tri Sync	74
Tuning Info.....	191

U

Unison Spread.....	195
Unit Delay.....	158

V

Valeurs dénormales	295
Value.....	176
Voice Info.....	190
Voice Shift	188