

Snapper Demo Ensemble

Overall structure

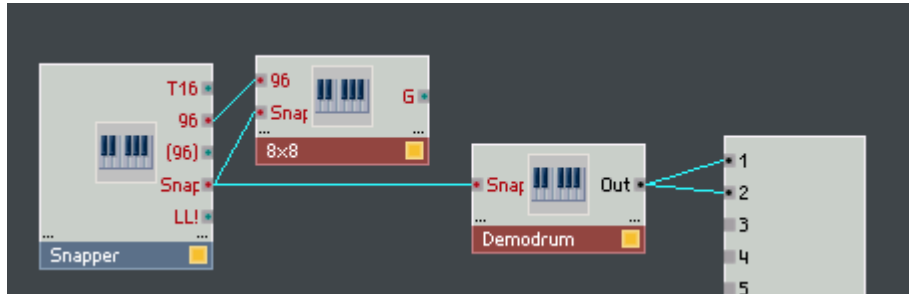


Abbildung 1: Snapper Demo Ensemble

The structure is simple, the snapshot sequencer control the snaps and the timing of the 8x8 sequencer, as well as the snaps of the demodrum instrument. The MIDI In of the demodrum is wired to the midi out of the 8x8 sequencer.

DemoDrum Instrument

The DemoDrum instrument is a simple percussion generator, but I like the sounds, so I'll have a deeper look into it :). It is also interesting to see how the snapshot are recalled by the Snapper.

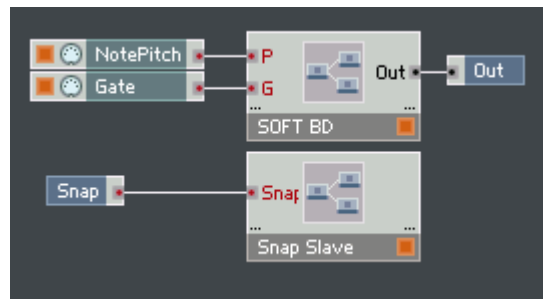


Abbildung 2: DemoDrum instrument

The MIDI In Pitch and Gate are received from the 8x8 sequencer, and sent to the SOFT BD Macro. The audio output of the SOFT BD goes into the main output. The Snap input, which comes from the Snapper, is sent into the Snap Slave macro, which is part of the “Snapper Macros” (it can also be found in Macro -> Building Blocks -> Sequencers). The Snap Slave is simple. An incoming Snap event is sent to the Snp port of the Snapshot module, and in a second time, a snapshot recall is triggered.

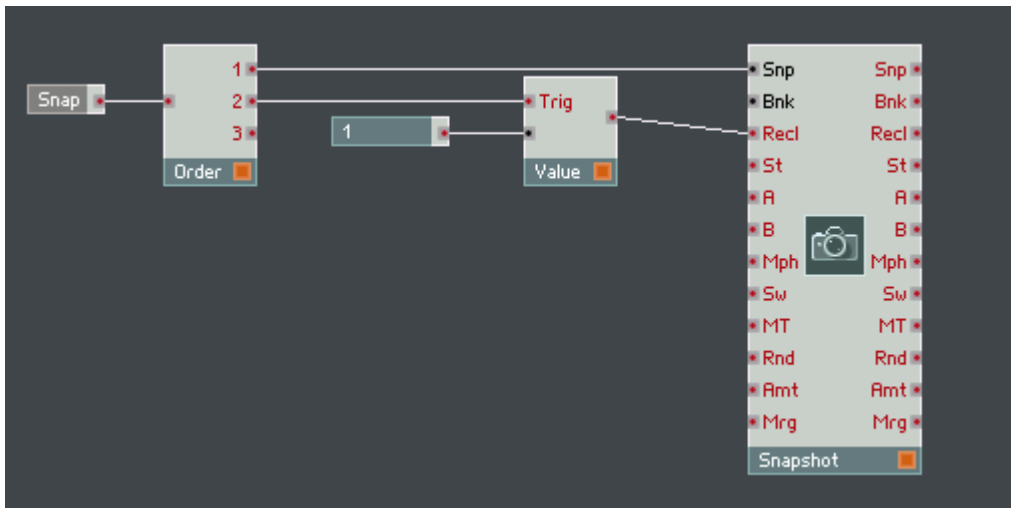


Abbildung 3: Snap Slave Macro

The SOFT BD Macro is pretty simple. It consists of a sound-generating macro “bd”, which is fed parameters given by knobs. The incoming midi Pitch is added to Pitch (the control) – 60. The midi Gate triggers the “bd” Gate.

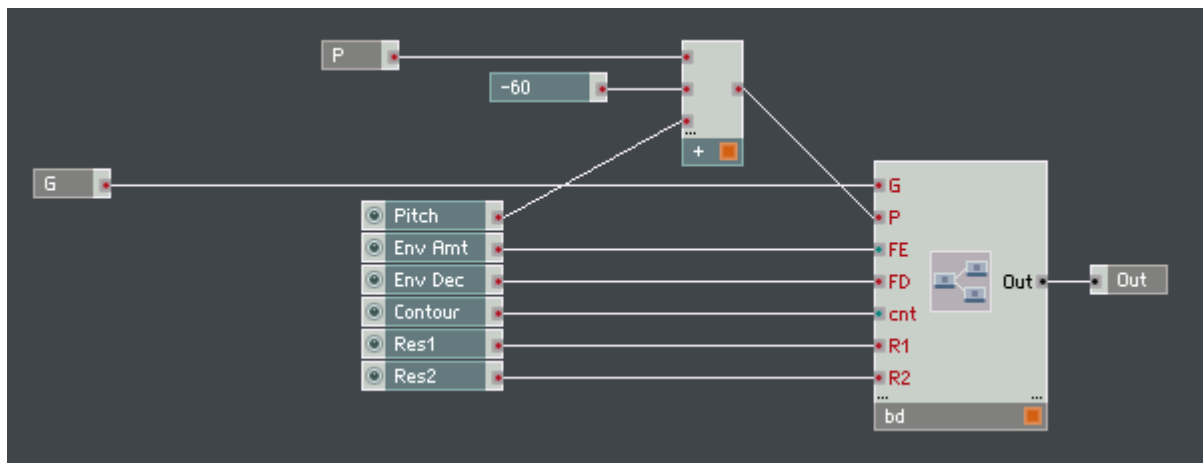


Abbildung 4: SOFT BD Macro

The Gate triggers a decay envelope, going up to the MIDI velocity. This envelope selects between the Pitch and the Pitch + Envelope amount using a linear selector. The output of the selector is converted to a frequency using a “P to F”-Exp Module. The frequency is multiplied by the square of the “Contour” control, and modulates the cutoff frequency of a Pro-52 Filter (which is set to -300). The resonance of the filter is set by the “R1” control. The frequency curve is fed as input into the Pro-52 filter. The output of this filter modulates the cutoff frequency of a second Pro-52 Filter (which is set to -300). The resonance of this second filter is controlled by the “R2” parameter. The input to this filter is the HP filtered envelope triggered by the gate. The cutoff frequency of the HP filter is equal to the incoming pitch. The first high-pass filter lets only the “click” of the envelope through. This click will make the pro-52 filters resonate at their resonating frequency, which will make a nice percussion sound (see the percussion synthesis article in synth secrets in sound and sound, and the percussion synthesis chapter in the nord modular book).

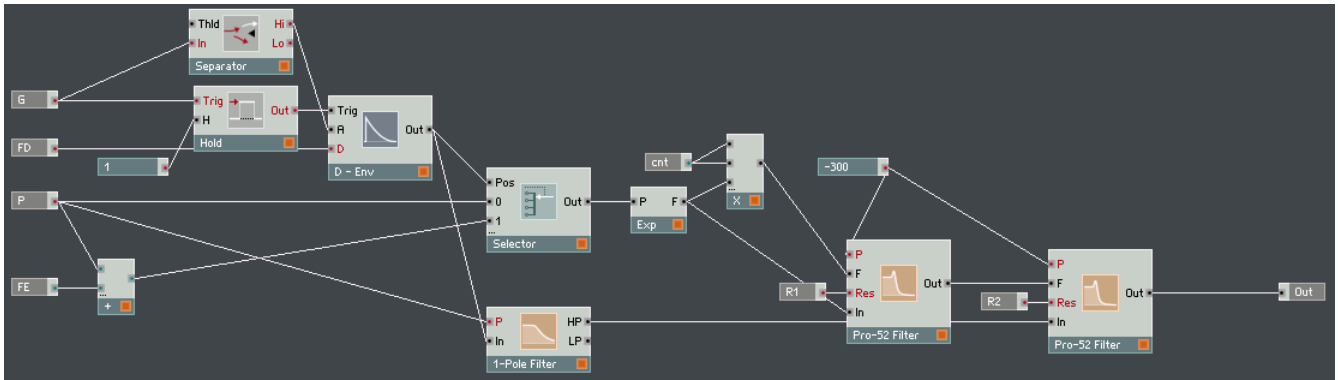


Abbildung 5: bd Macro

8x8 Sequencer

This is not a new addition to Reaktor, but it is very interesting nevertheless. This 8x8 sequencer has been modified to include a Snap Slave macro, and to get its timing from the Snapper instrument. This is why we have an unconnected “Song Position” macro lying around.

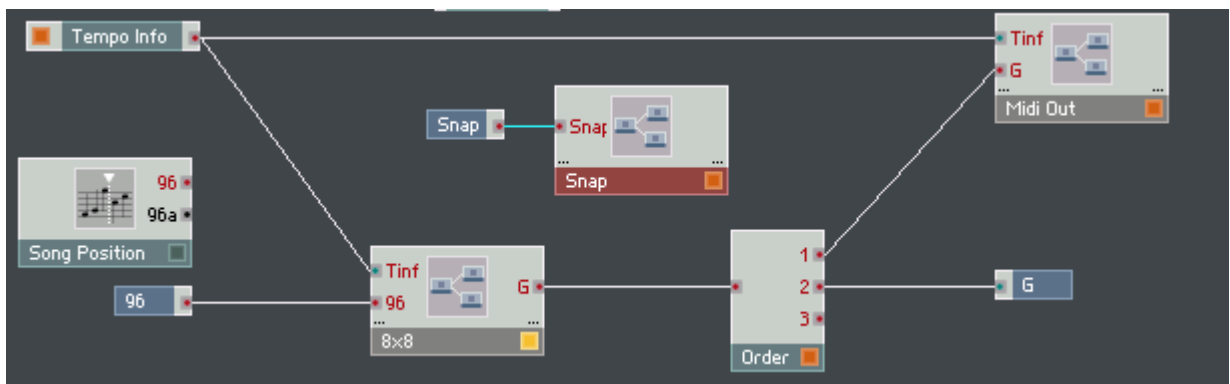


Abbildung 6: 8x8 Instrument

The Snapshots are recalled by the “Snap” macro which is the same as in the demodrum instrument. The tempo information and the current position in 96th are fed into the 8x8 macro, and the gate signal out of the 8x8 macro is fed into both midi out and the G output. In the Midi Out Macro, each G event is held for the duration of a 48th (the third of the duration of a 16th, which is 250 / Tempo Info in ms). The outgoing pitch is controlled by the “Note & sw” Macro.

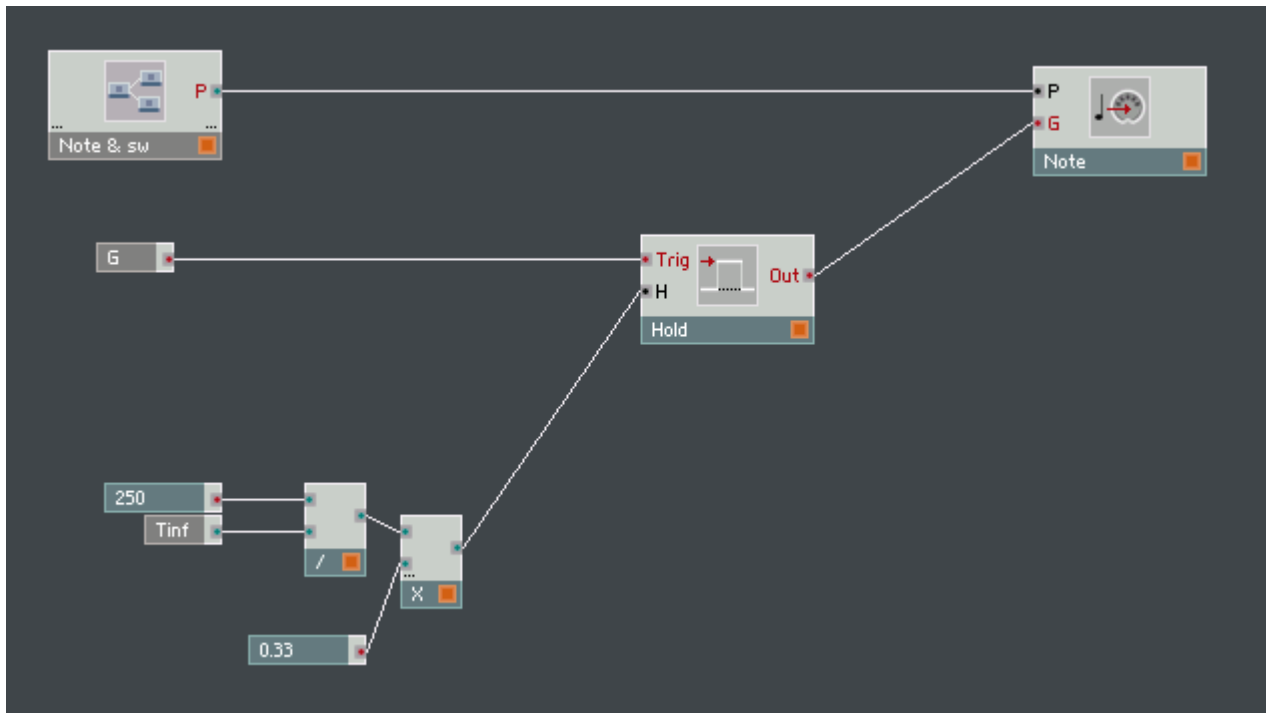


Abbildung 7: MIDI Out Macro

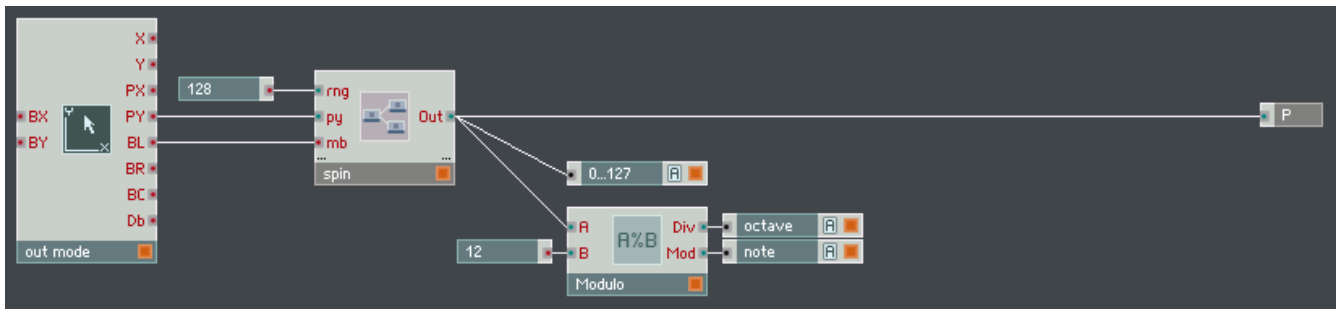


Abbildung 8: Note & sw Macro

The Y coordinate of the mouse area, which is layed out above the pitch information display in the panel, is interpreted by the “spin” macro. The structure of the spin macro is quite complicated because it doesn't take advantage of the incremental mouse setting of the mouse area, which it implements itself. On a mouse click, the Y coordinate is latched, as well as the previous stored pitch value (out of the snap value module). This previous value is added onto all subsequent Y coordinates (this is the incremental behaviour which the mouse area implements by itself). The Y value on mouse click is subtracted from subsequent coordinates, and the movement range is divided by 10 (10 pixels for 1 note), and quantized to integers. After the previous value has been added, the output is taken modulo the range (here 128), and stored in the snap value while the button is pressed down. The value is displayed by a numerical display showing the MIDI pitch, and two multitext displays showing the octave and the note. Using the mouse area incremental mouse option, the Note & sw macro could look like this. The Snap Value this time is set directly after the output of the mouse area, and before the scaling, so that its output can be fed back directly into the mouse area. When a snapshot is recalled, clicking the mouse area will produce no value hiccup because the BY input is set to something different.

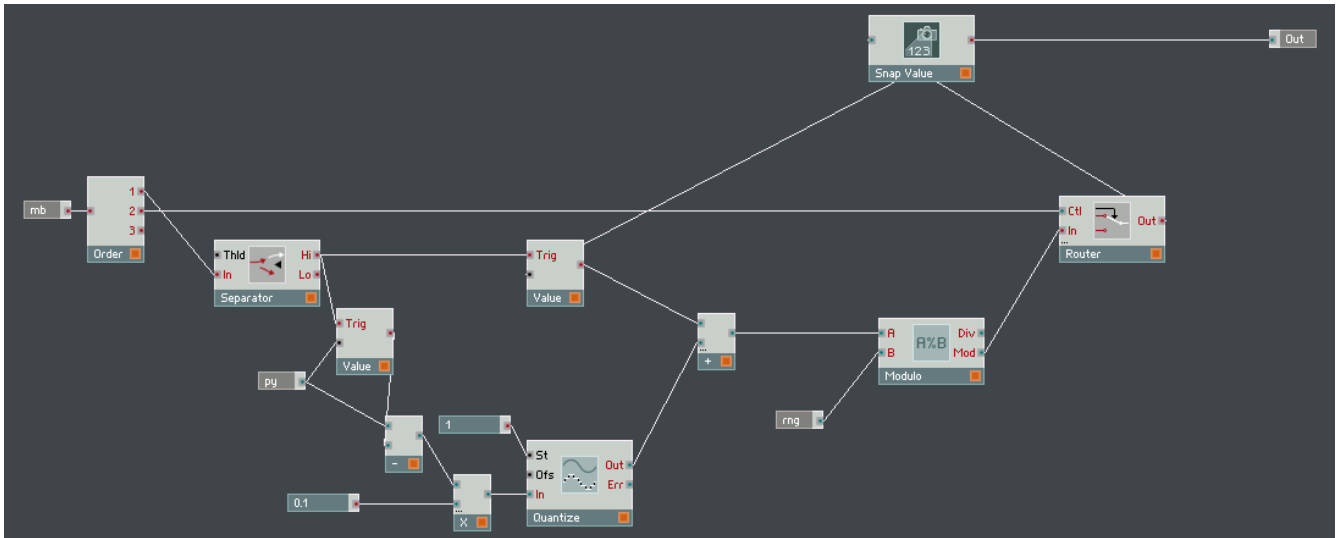


Abbildung 9: spin Macro

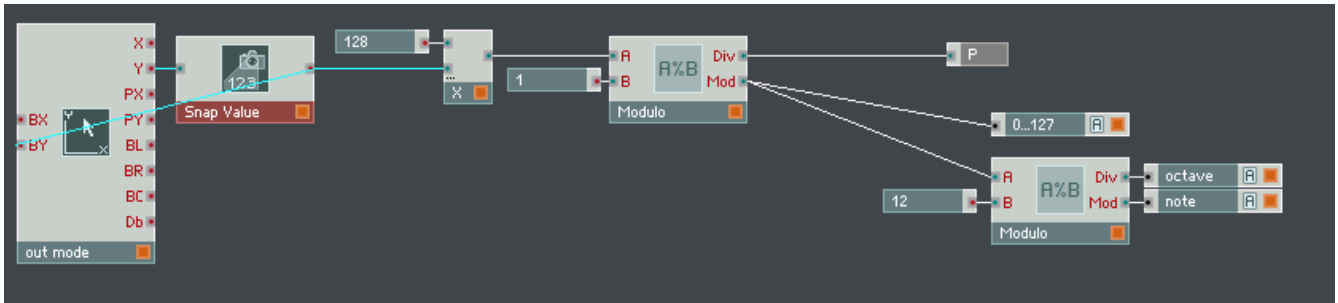


Abbildung 10: Alternative Note & sw Macro

8x8 Macro

The interesting part of the instrument obviously is in the “8x8” Macro, which is quite complicated. The mouse area in the 8x8 Macro covers the drawing area of the sequencer. The output of the mouse area is interpreted by the “draw” macro, as well as by the “zoom” and “rect select” macro, which handle the right mouse button. The timing information is interpreted by the “Song Pos & Shffle” macro at the top left. The actual sequencer data is held in a 64-voice polyphonic Snap Value at the bottom right, and reading the current step is handled by the “play” structure next to it. Displaying the current sequencer structure is handled by three PolyDisplays in the “screen” structure. Let's start with the timing macro.

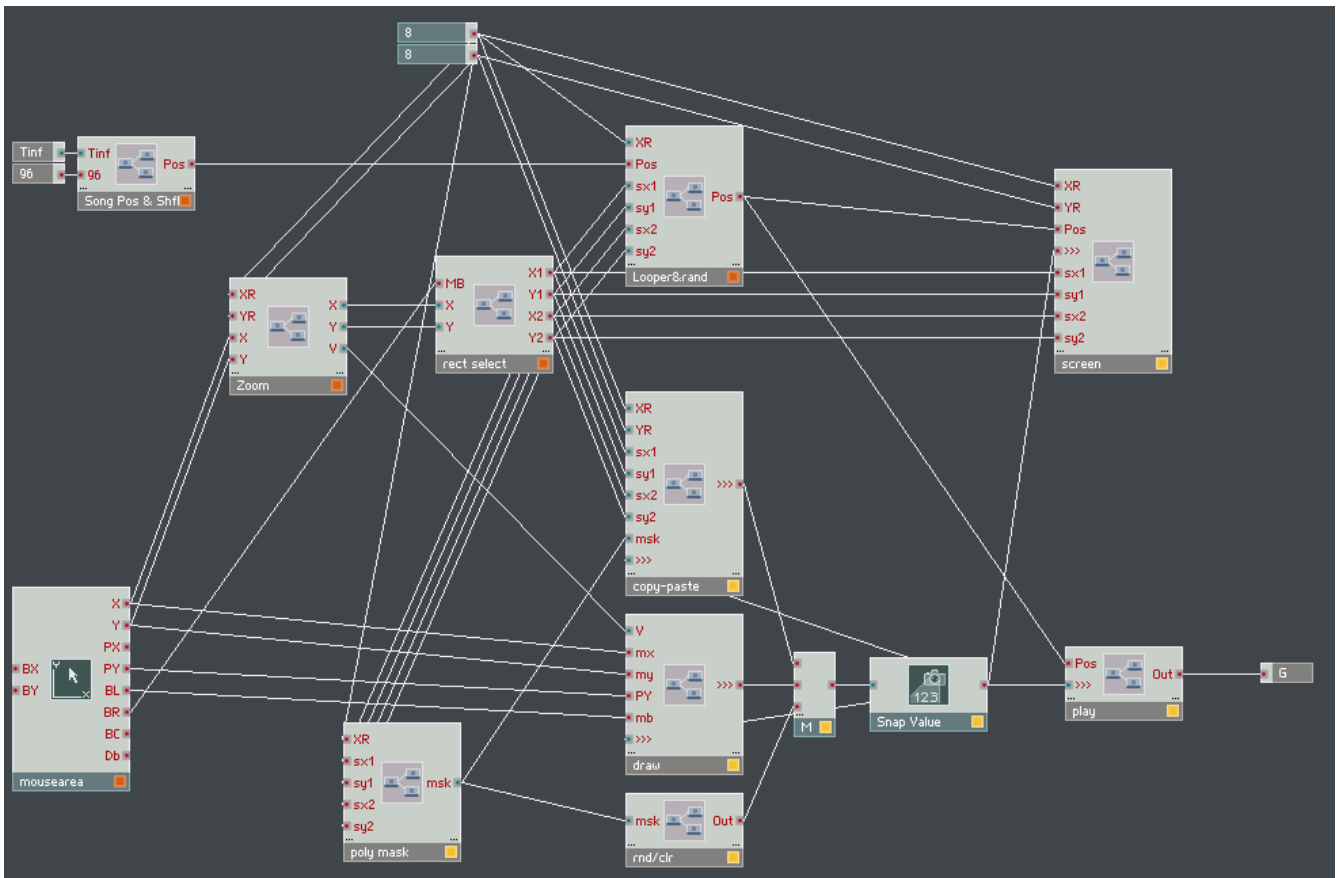


Abbildung 11: 8x8 Macro

Song Pos & Shfl Macro

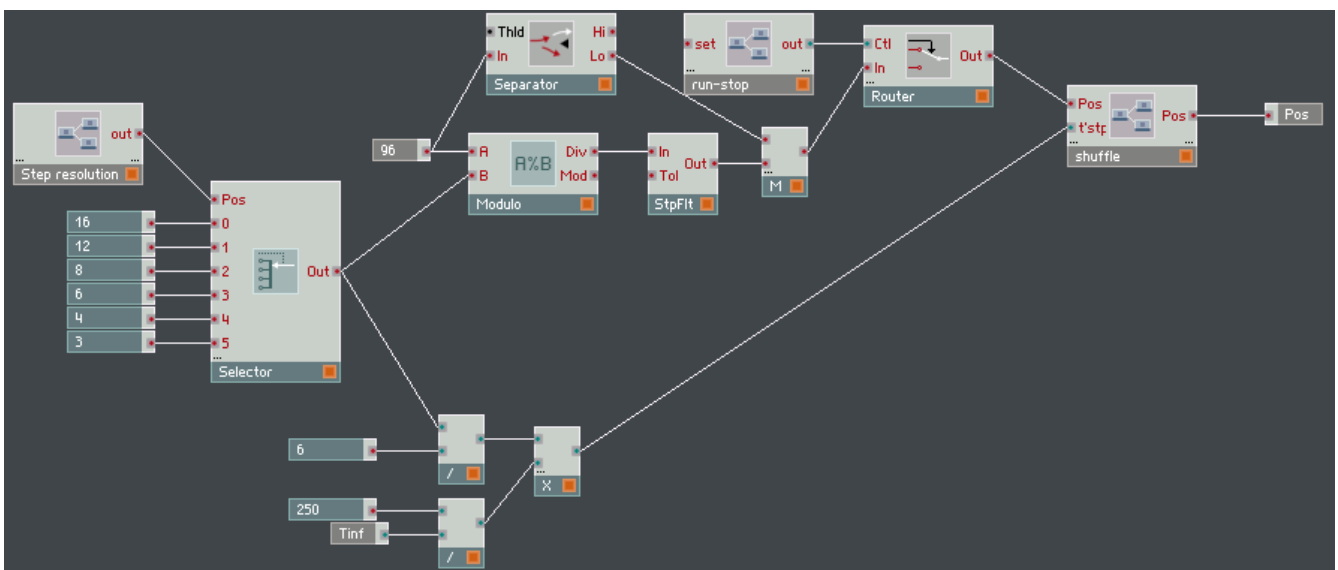


Abbildung 12: Song Pos & Shfl Macro

The Step Resolution is implemented using a mouse area and a spin macro like for the output pitch in “Midi out”. It chooses between a step resolution of 32th up to 6th notes. The output of the selector is the

value by which the 96th position has to be divided. The duration of a step in ms is computed by the math modules at the bottom. 6 is the divider for 16th notes ($96 / 6 = 16$), and 250 / bps is the duration of a 16th in ms, so the output of this is the duration of a step in ms. This is fed into the shuffle macro at the top right. The 96 position is divided by the computed divisor, step-filtered and merged with 0 position events (maybe so that each restart is captured? 0 divided by something will always be 0 so normally a 0 position would go through the bottom step filter). The position events are only let through to the shuffle macro if the run button is toggled. The run button is implemented by the run-stop macro, which implements a “settable” knob using a mouse-area and a display (one could also use a normal knob and an IC send, and link them, for example).

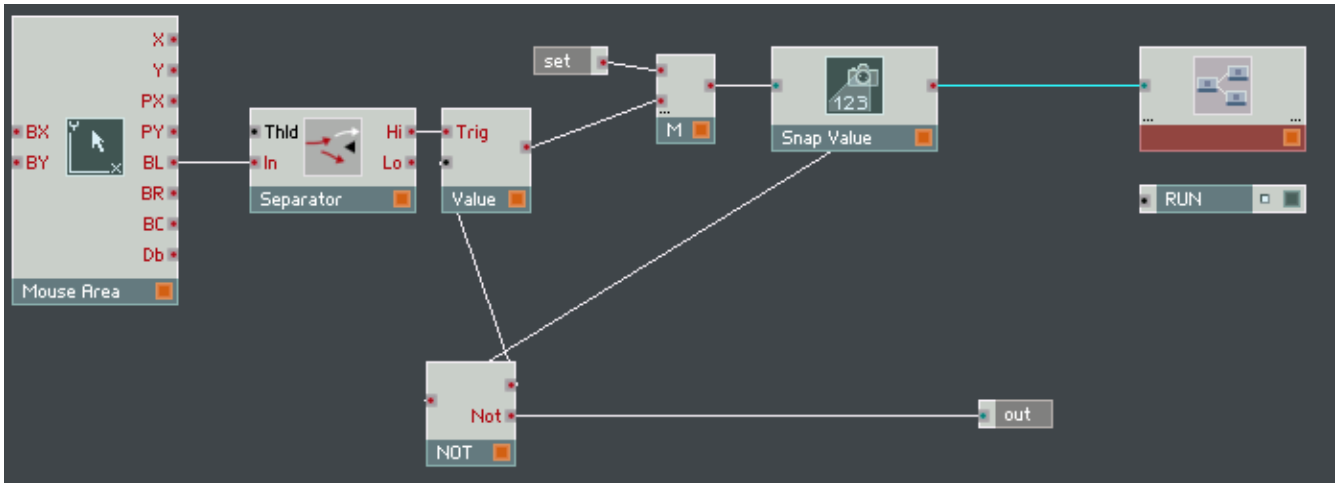


Abbildung 13: run-stop macro

The shuffle macro delays each second step by the step duration multiplied with the shuffle control, which is implemented using another mouse area and display modules. This mouse area uses the incremental function though, and the value is displayed using a PolyDisplay.

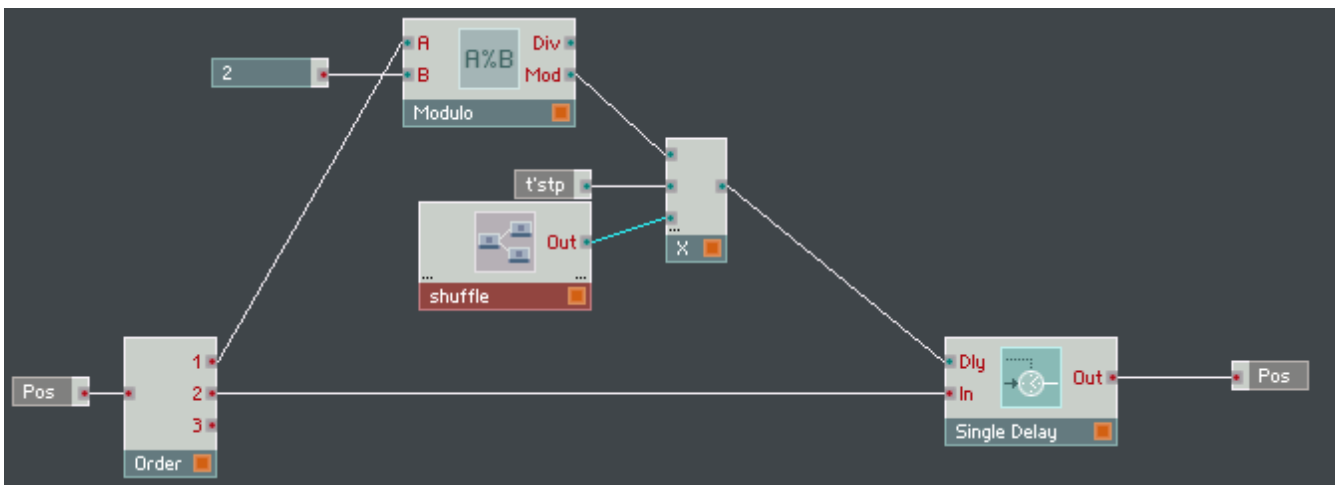


Abbildung 14: shuffle Macro

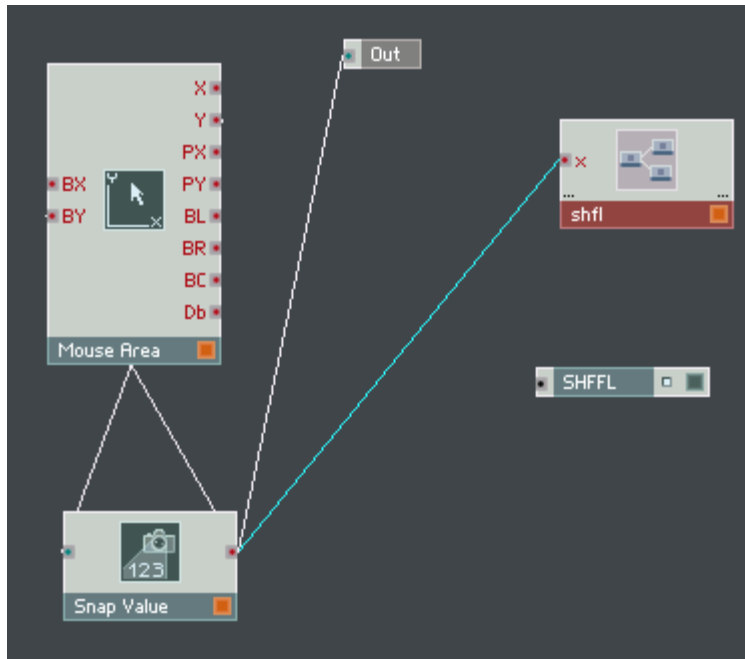


Abbildung 15: shuffle button Macro

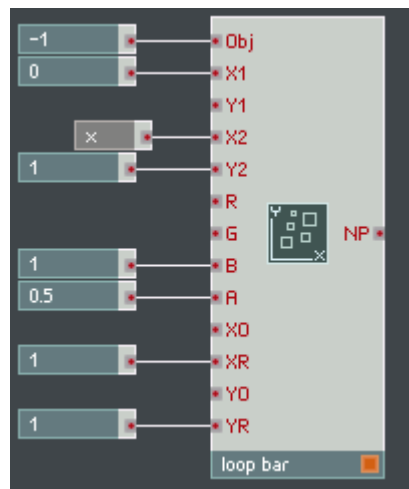


Abbildung 16: shfl display macro

Snapper

The Snapper Instrument has 5 outputs. T16 is the duration of a 16th in ms at the current tempo, 96 is the song position in 96th, (96) is the song position in 96th reset at each snapshot recall, Snap is the snapshot number to be recalled, while LL! outputs an event at each loop cycle begin. The Snapper functionality is implemented in the Snapper Macro.

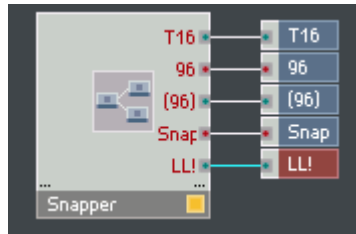


Abbildung 17: Snapper Instrument

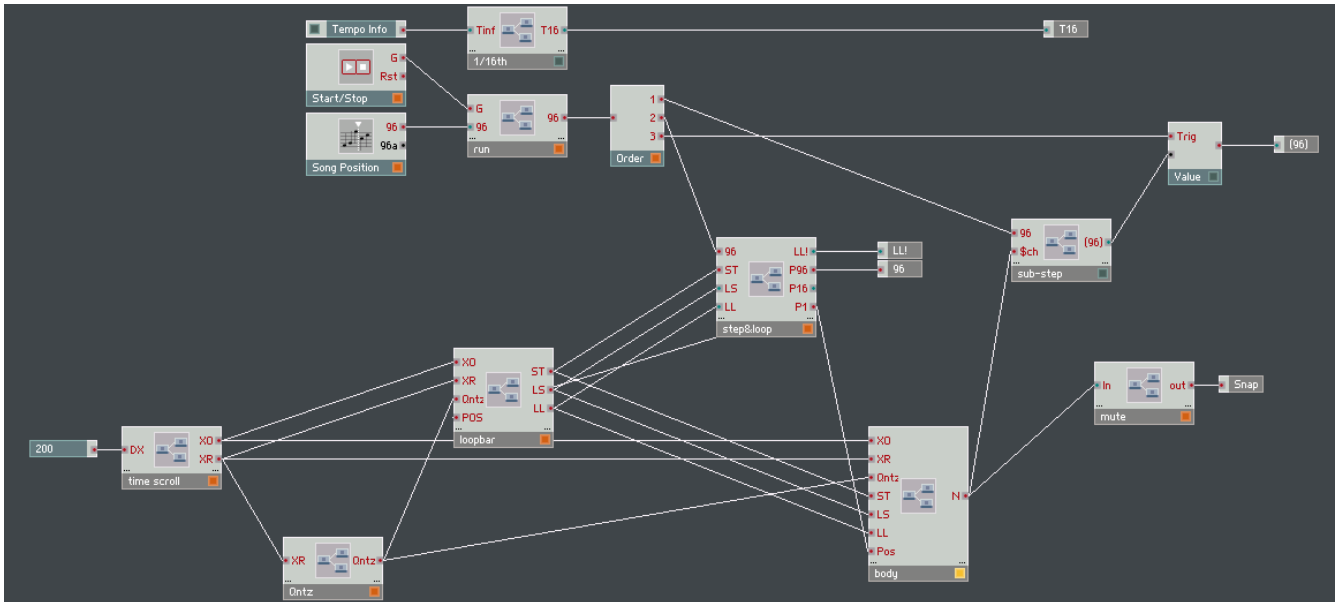


Abbildung 18: Snapper Macro

At the top of the macro, we have the song position, start/stop and tempo info analysis. The macro $1/16^{\text{th}}$ calculates the duration of a 16^{th} in ms according to the current tempo. The macro is a standard construction, Tempo Info outputs the beats per seconds, the inverse of that is the seconds per beat, multiplied by 1000 is the milliseconds per beat, divided by 4 the milliseconds per $1/16^{\text{th}}$, so $250 / \text{Tempo Info}$ is the duration of a 16^{th} in ms. This duration is output to the T16 port.



Abbildung 19: 1/16th macro

The run macro handles the song position and the running gate, as well as the run button at the top of the snapper GUI. The Run macro in "run" is a custom button implementation, using a mouse area, a snap value, a feedback loop with a not to implement a flipflop, and an "imp" macro containing a PolyDisplay to display the run button. I don't really see the advantage over a skinned toggle button, but whatever :)

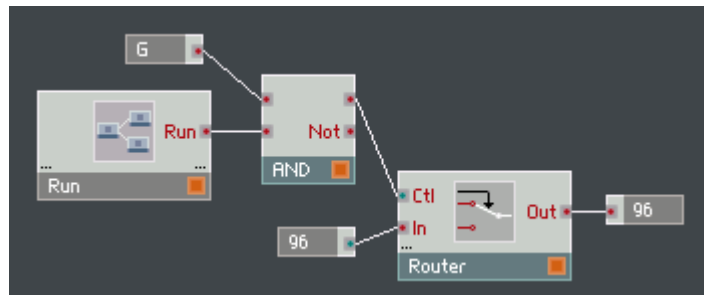


Abbildung 20: run Macro

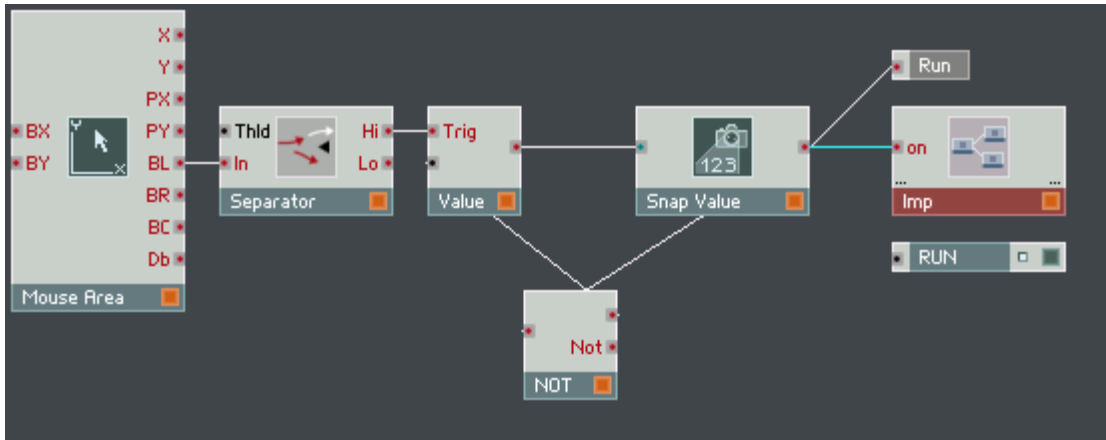


Abbildung 21: Run Button Macro

The current song position is output only if both the ensemble is running and the run button is pressed. This song position is first passed to the actual snapshot sequencer before latching the output of the “sub-step” macro to the (96) port. The sub-step macro resets the song position when a new snapshot is recalled. It receives the “real” 96 position in the 96 port, as well as the \$ch which is triggered when a new snapshot is recalled. When the real song position is equal to 0 is merged with the snapshot-recall events and used to reset an accumulator which is incremented by 1 on each 96th. This is the “snapshot-reset” 96th position, which is sent to the (96) output.

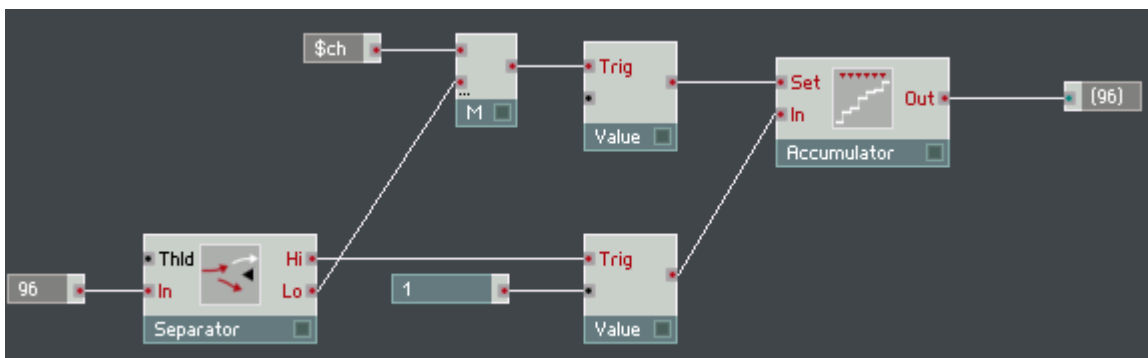


Abbildung 22: sub-step Macro

Most macros are used to control the display of the Snapper sequencer. The “timescroll” Macro implements the time scroll (the timeline at the bottom, with the darkgreen position scroll, as well as the L->Z button at the top which zooms the main display to the length of the looped region. The maximum

length of the timescroll is 200 bars (input DX).

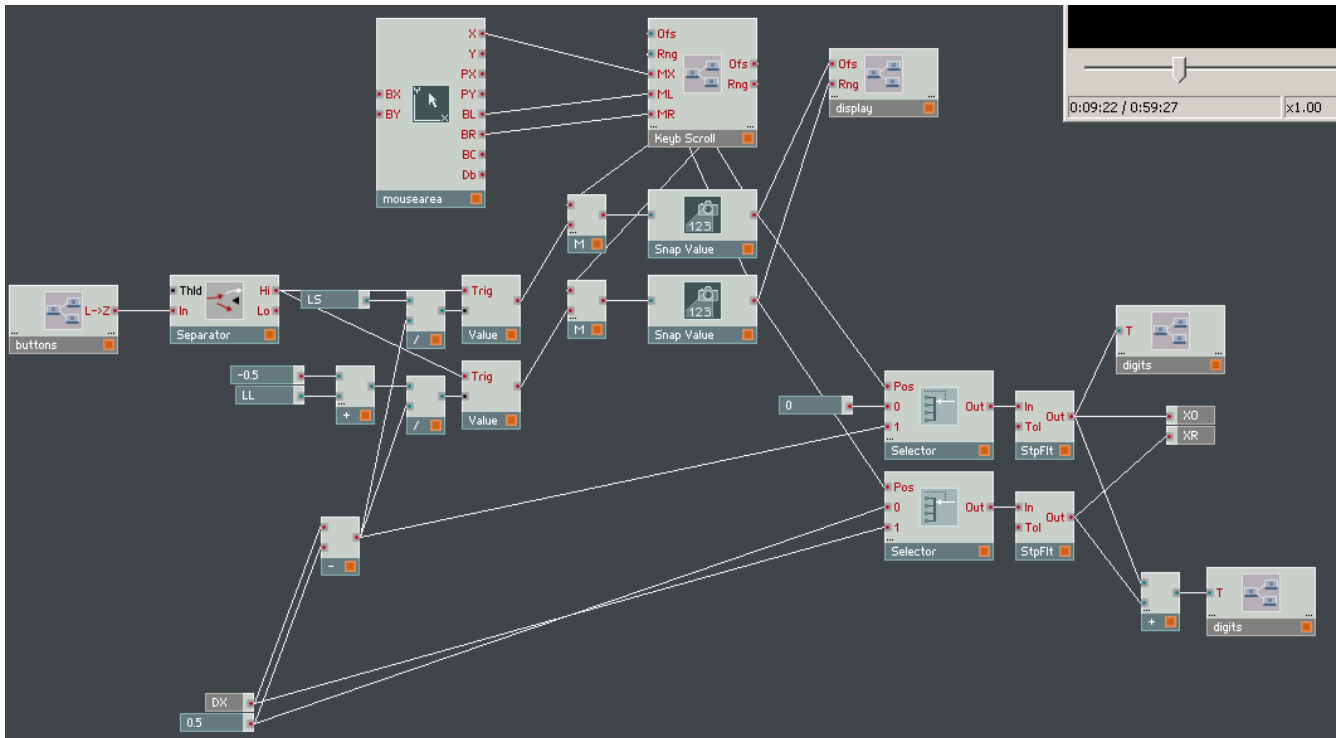


Abbildung 23: timescroll Macro

The macro uses a mouse area on top of the timescroll. The output of the Mouse Area (X coordinate, right click, left click) is analysed by the keyb scroll macro. On the left side, a macro implements the L->Z button. Again, I don't really know why a skinned button wasn't used. Clicking the L->Z button latches the Loop Length and the Loop start into the snap values at the middle, which store the current view settings for the timescroll: the start offset of the time scroll, and the range of the time scroll. The output of the Keyb Scroll macro are merged into these snap values too. The start offset and the end position (which is the start offset + the range) are displayed using the macro "digits", while the current start position and range is also displayed as a greenish rectangle below the digits in the "display" macro.

The keyb scroll macro modifies the current offset and range values according to incoming mouse area events. A left mouse click moves the start position, while a right mouse click modifies the range.

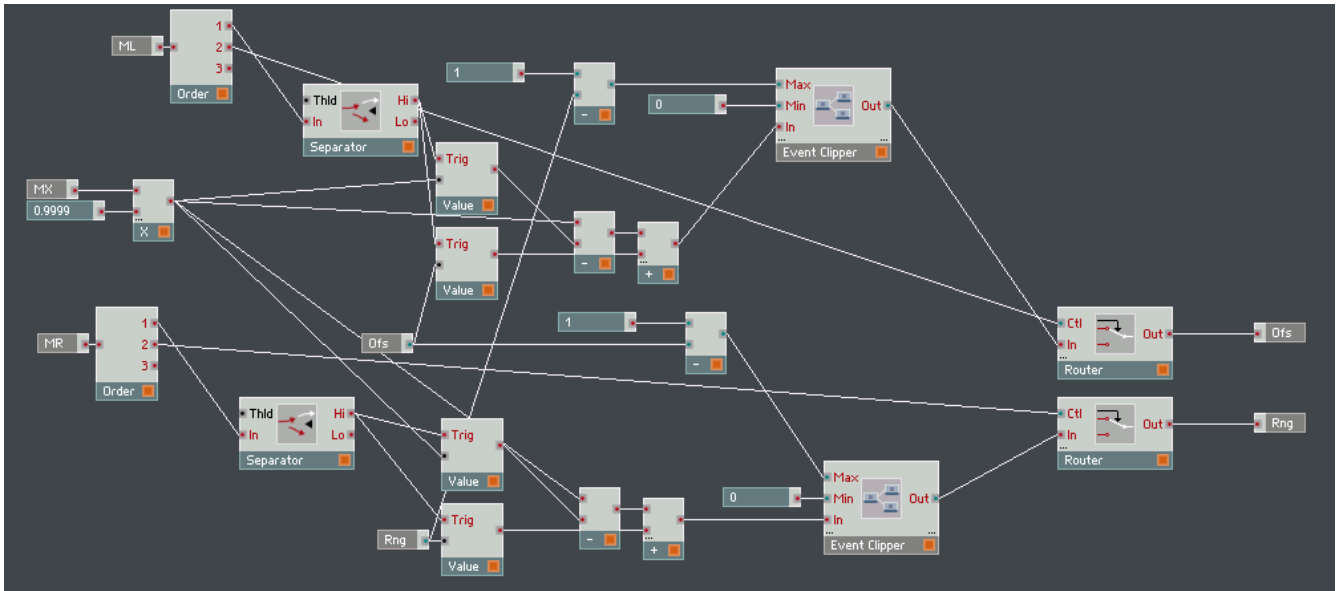


Abbildung 24: keyb scroll Macro

The current display range of the time scroll is used to calculate a quantization parameter in the Qntz macro.

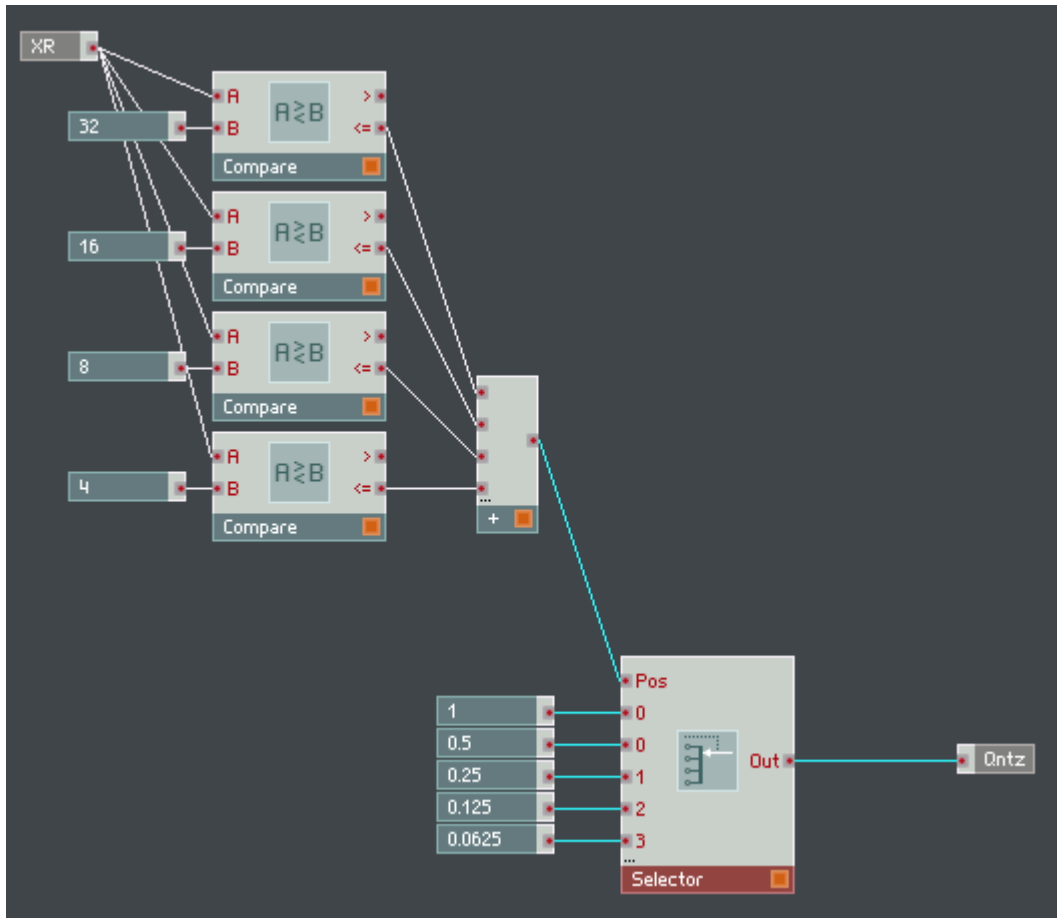


Abbildung 25: Qntz macro

If the range is smaller than 4, the quantization is set to 0.0625, if it is smaller than 8, 0.125, etc... This is

achieved by adding the times XR is smaller than a value. If it is smaller than 4, it is also smaller than 8, etc..., and the selector is controlled by the value 4, which chooses the bottom value. Nice trick.

The loopbar (the green bar on top of the snapshot sequencer) is implemented by the “loopbar” macro, which is given the start offset and the display range, as well as the quantization parameter and the current position.

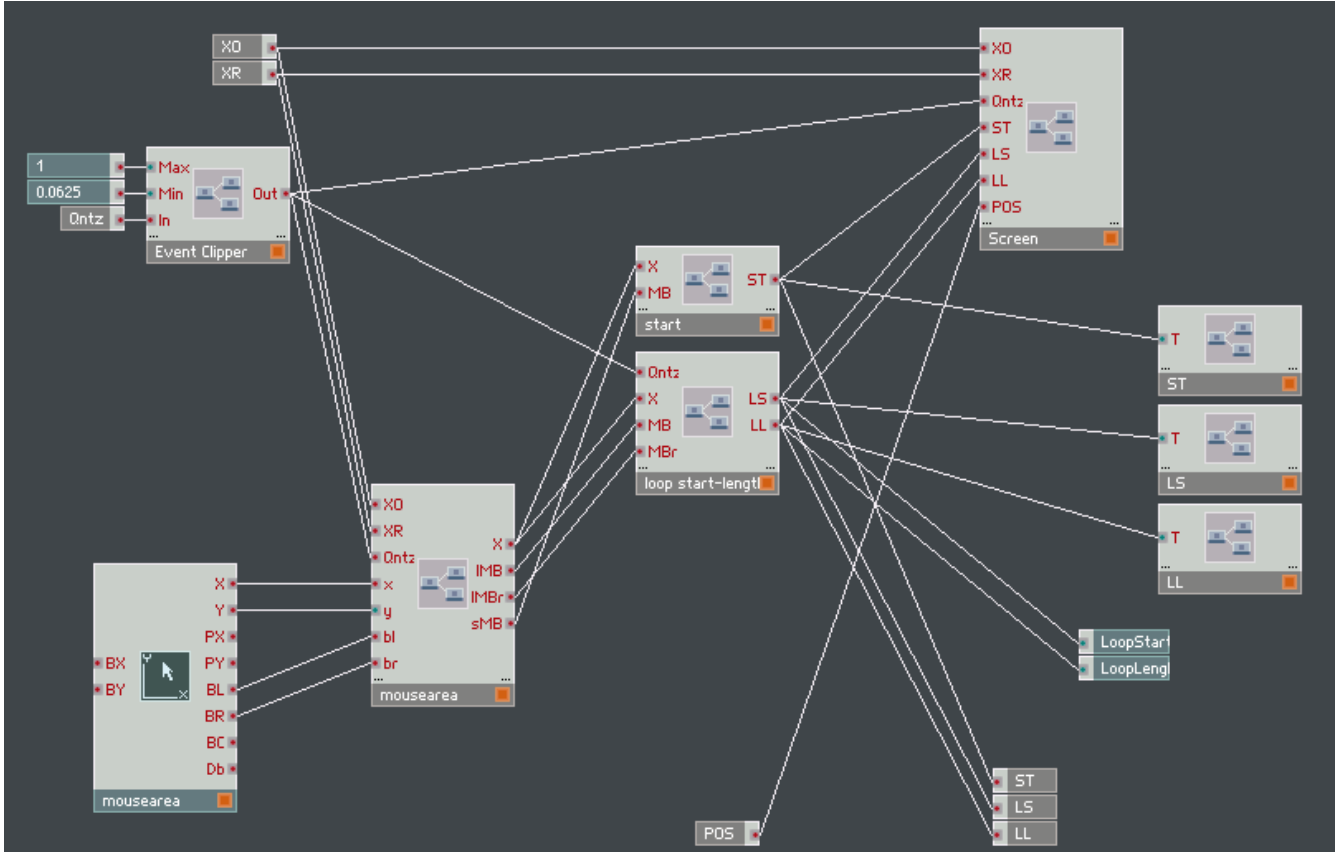


Abbildung 26: loopbar Macro

Quantization is clipped between 1 and 0.0625. The output of the mousearea is interpreted by the “mousearea” macro. The mousearea is divided into two areas, the upper area which is used to set the start marker, and the lower area which is used to set the loop start and loop length and which responds to left and right mouse clicks.

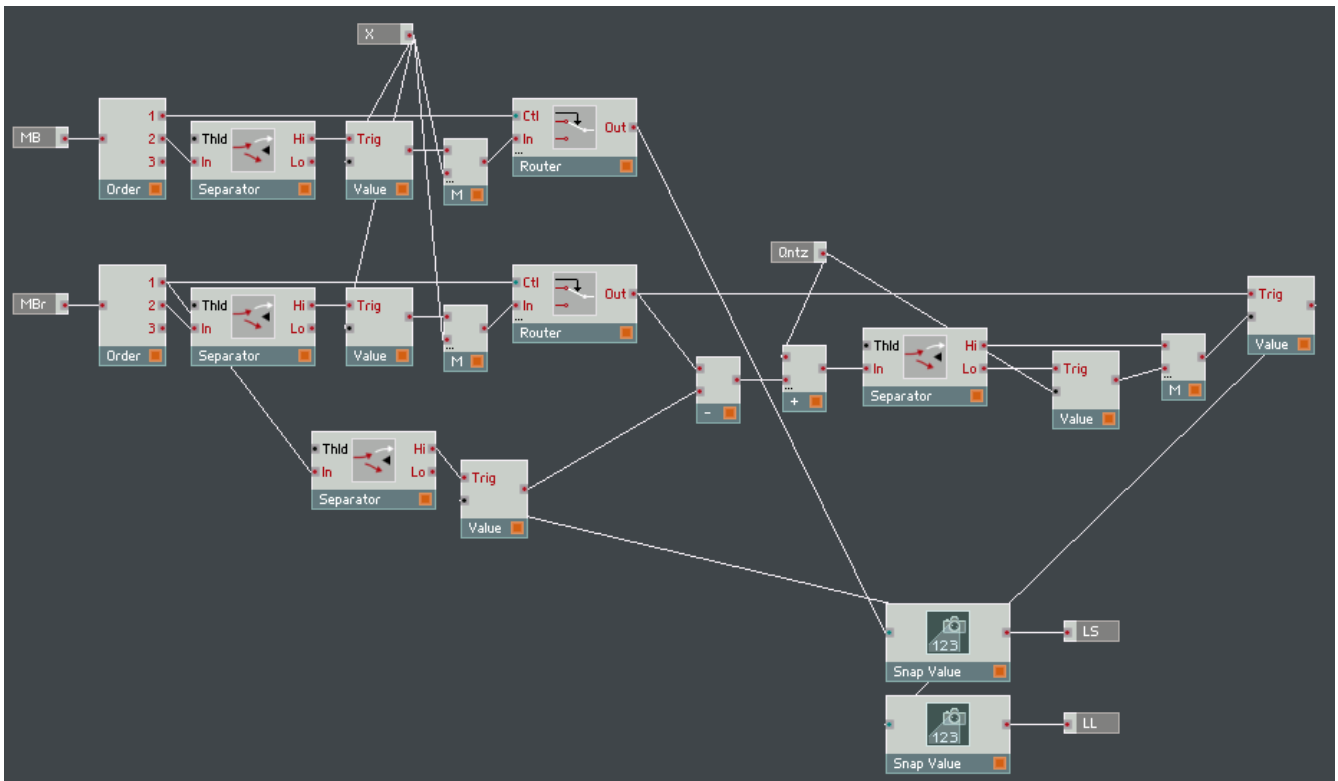


Abbildung 29: loop start-length Macro

The loop start-length Macro works in a similar way, doing further calculation for the loop length by subtracting the loop start from the right-click value (loop-end), and clipping the loop length at the bottom by the Qntz value (an empty loop is not possible).

The loopbar is displayed by the “screen” macro.

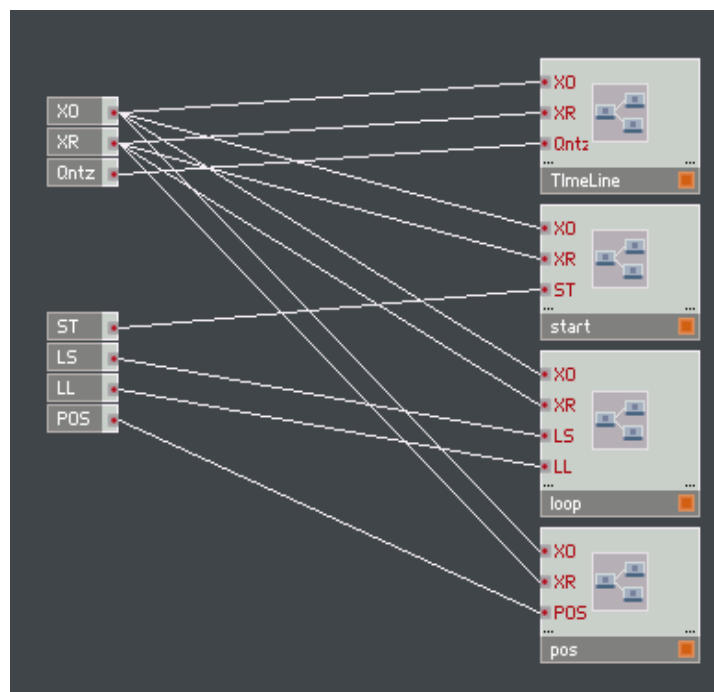


Abbildung 30: screen Macro

The grid is displayed by the TimeLine macro, which only uses the X0 (time display start), XR (time display length) and Qntz values.

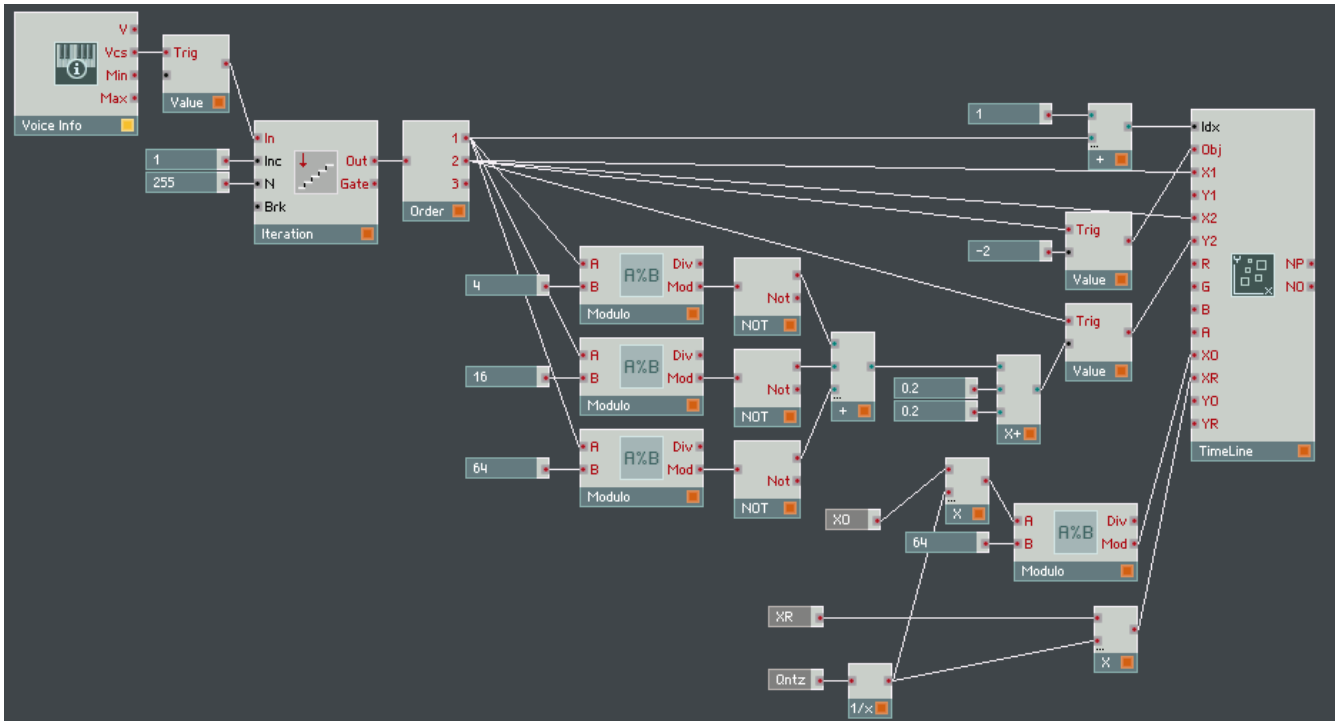


Abbildung 31: TimeLine Macro

TimeLine uses a MultiDisplay to display the ticks. It triggers an iteration at ensemble startup, and draws 256 ticks with different sizes into the MultiDisplay. These ticks span the whole area, and updates to X0, XR and Qntz only change the displayed range. Nice trick.

The start macro displays the start marker. It uses two PolyDisplays, one for the yellow bar, and one for the line at the beginning of the bar.

The loop macro displays the green bar for the loop. The bottom PolyDisplay is hidden and probably was left over from a previous version (showing shows that it has not been moved to the correct place in the interface). It probably was the first try to draw the loop start/end points into the main sequencer (the green lines in the snapshot sequencer). The pos macro is used to draw the red bar to show the current playback position inside the loop. The ST, LS and LL macros are similar to the digits macro in the TimeScroll macro and are used to display the loop parameters at the top of the GUI.

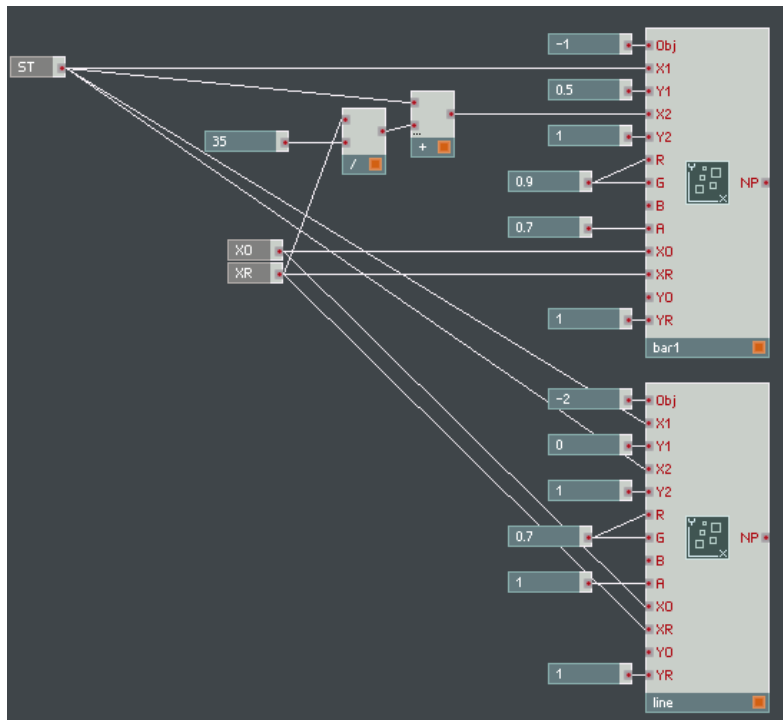


Abbildung 32: start Macro

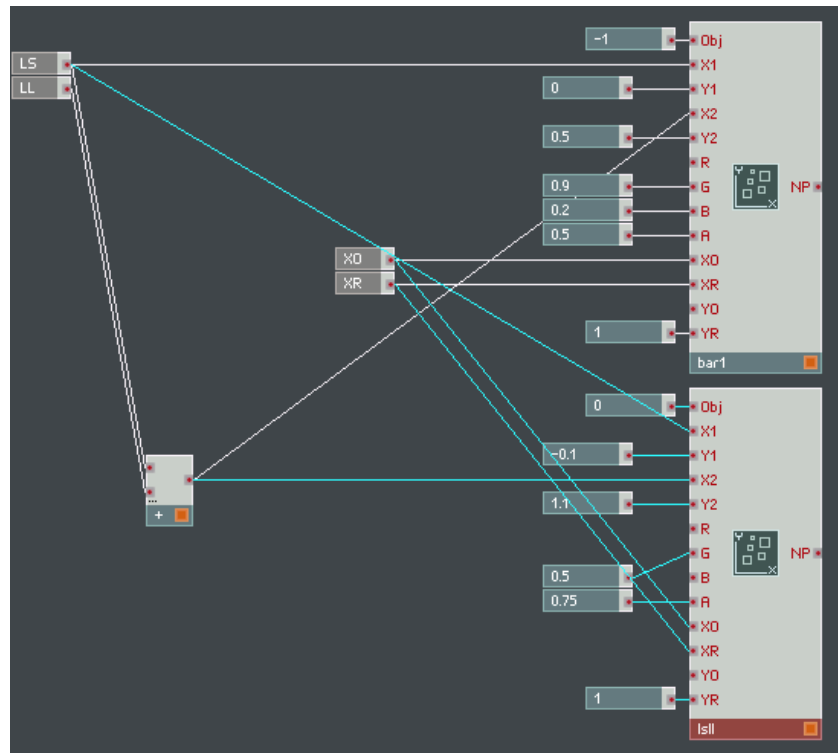


Abbildung 33: loop Macro

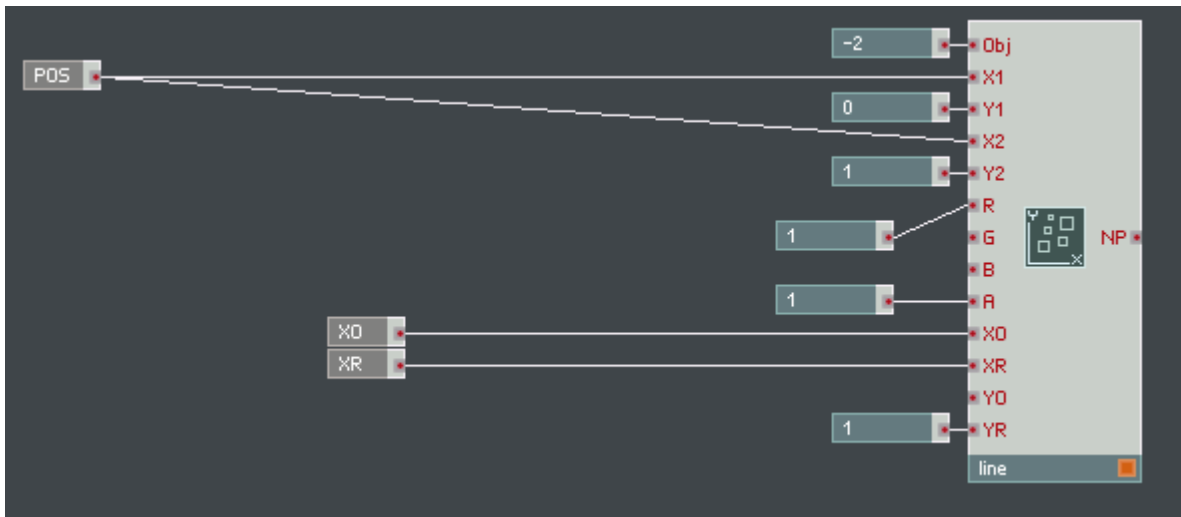


Abbildung 34: pos Macro

The step&loop Macro is used to calculate the current playback position according to the current loop, and to display the position marker in the main sequencer. The structure of the macro is a bit tricky.

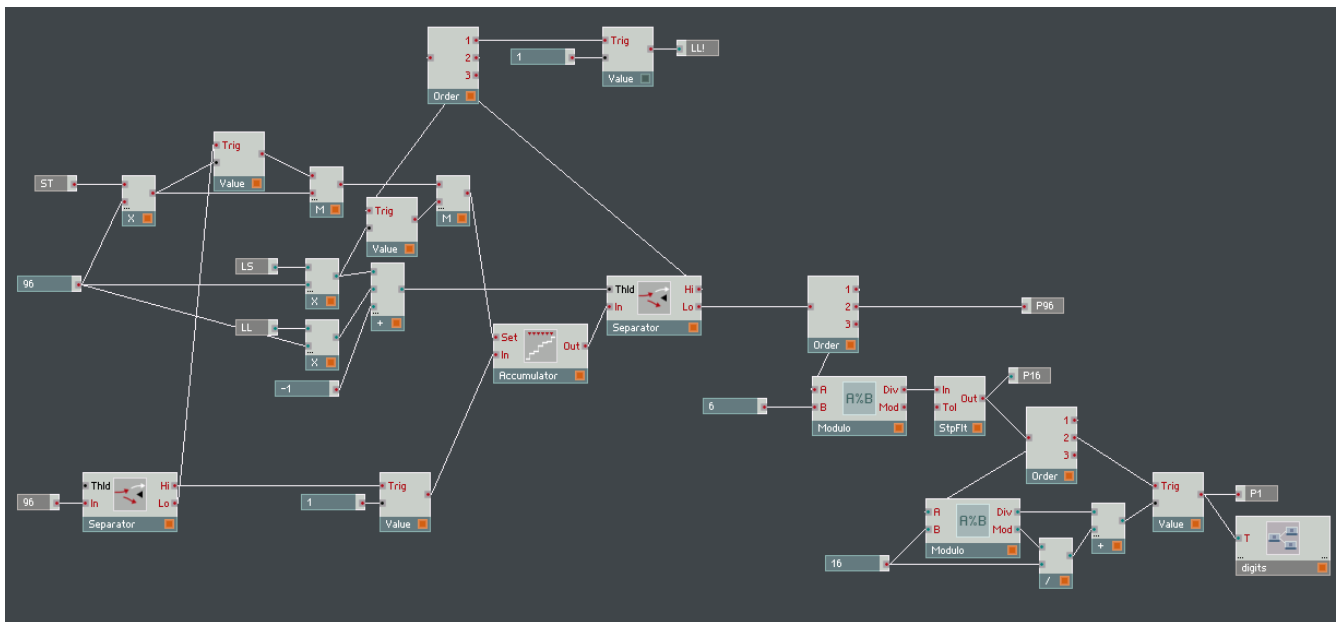


Abbildung 35: step&loop Macro

First, all parameters for the loop are multiplied with 96 to work in 96 steps (loop parameters are set in bars). In the middle of the macro we see an accumulator which is incremented by 1 on each 96th clock step. This accumulator gives the current playback position. It is reset to $ST * 96$ when the “real” clock position is 0 (midi reset), else it is reset to $LS * 96$ when the position is bigger than the loop end (this also triggers a 1 event on the LL! output). This is the separator at the output of the accumulator. The current playback position (output of the accumulator) is sent to P96 (position in 96th). The current position in 96th is also converted to a floating point position in 4th at the lower right end, which is used to display the current position at the top of the snapshot sequencer. The position in 4th is also passed to the P1 output.

Finally, the “body” macro implements the actual snapshot sequencer.

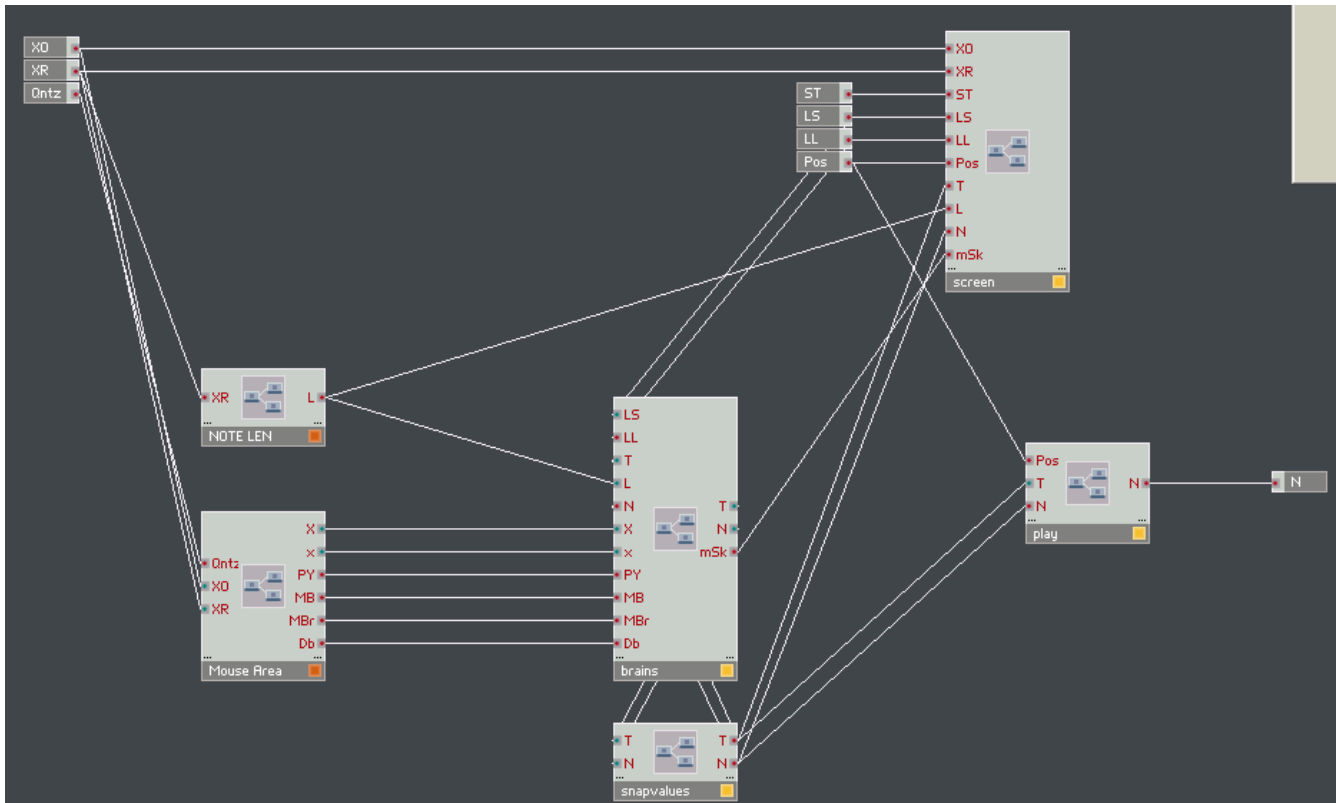


Abbildung 36: body Macro

The mouse area on top of the snapshot sequencer is hidden in the “Mouse Area” macro. This macro outputs the mouse left button, right button and double click, outputs the “real” x coordinate, and also the scaled x parameter that fits the actual time display. These mouse area parameters are interpreted by the “brains” Macro. The outputs of the brains area, two polyphonic values storing the snapshot value and start time of the sequenced snapshot recall, are stored in snapvalues in the bottom “snapvalues” macro.

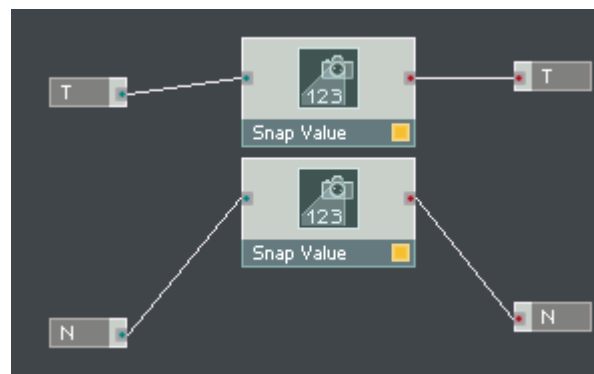


Abbildung 37: snapvalues Macro

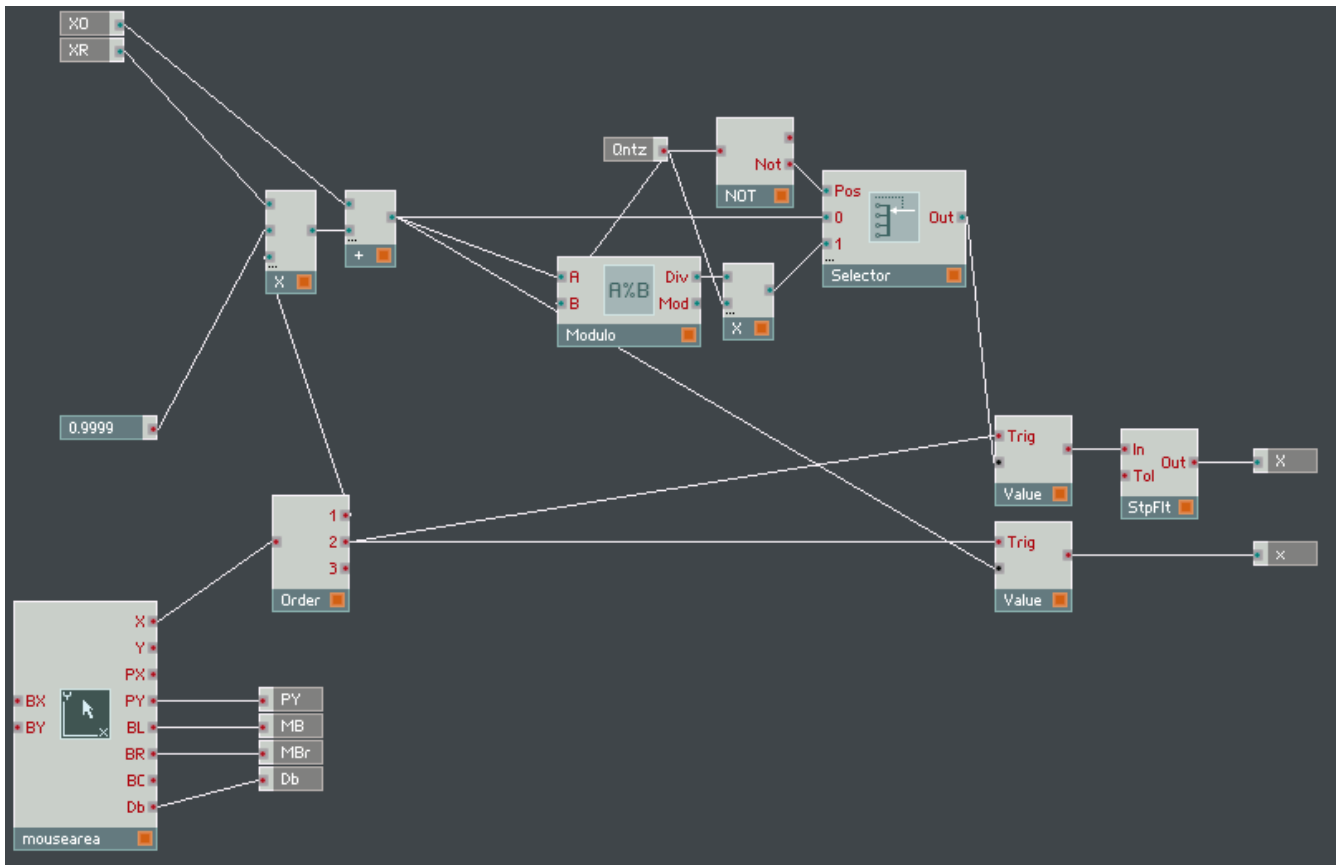


Abbildung 38: MouseArea macro

Brains is the actual snapshot sequencer implementation. It also is quite complex, as it also implements the copy/paste/clear functionality. But nothing really is complex, as we'll see :) Brains get the current sequence data as inputs out the snap values: T (snapshot recall start time) and N (snapshot number). These are polyphonic signals with 100 voices, that means, we can sequence at most 100 snapshot recalls using the sequencer.

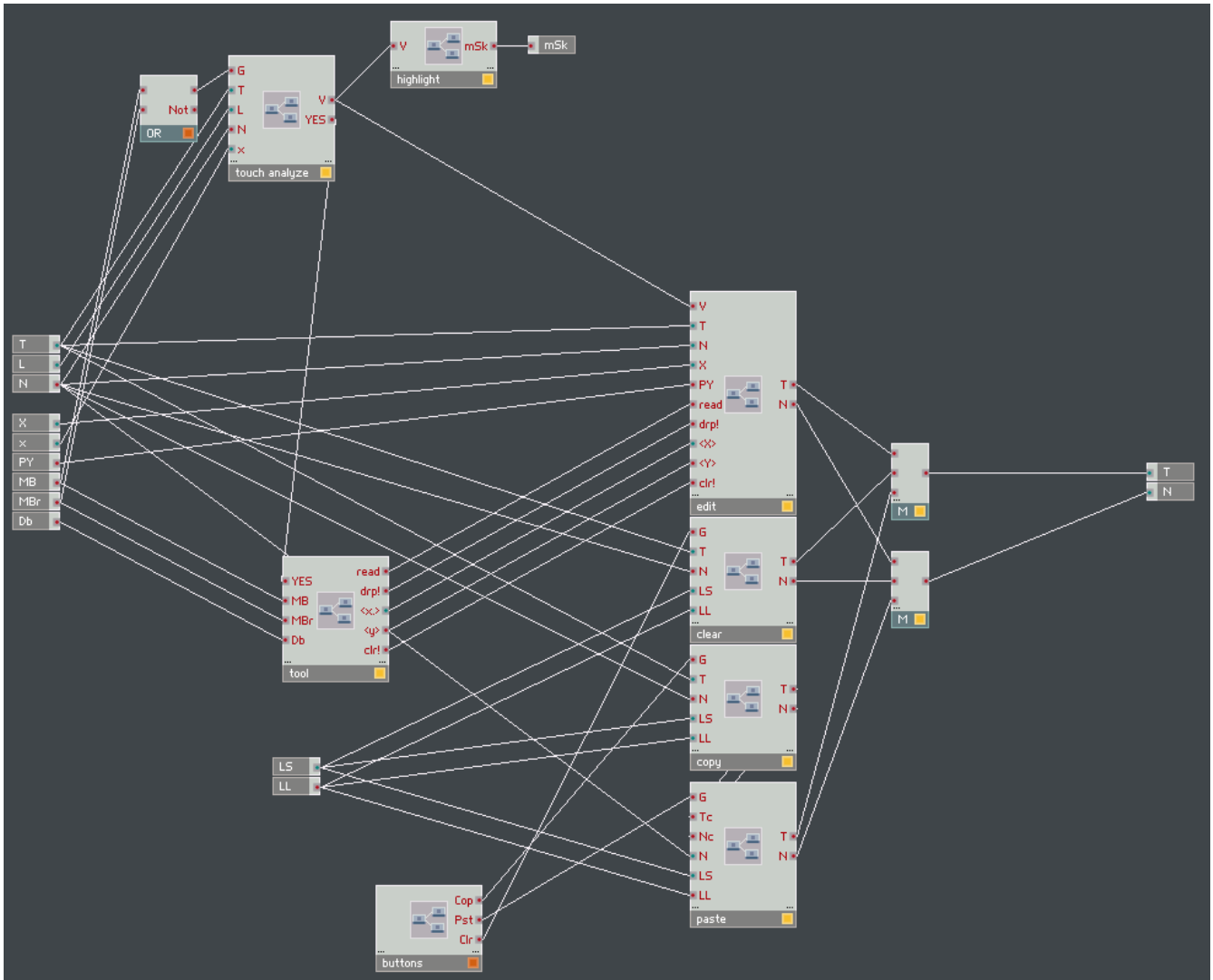


Abbildung 39: brains Macro

T is the input with the current sequencer time data, N is the sequencer data for the snapshot numbers (N is recalled at time T). L is the current note len ($XR / 35$). x is the scaled mouse parameter. The value is analyzed in the “touch analyze” macro. The gate signal (G) of the touch analyze macro is triggered by a button press (either right or left). In “Touch”, the rising gate latches the inputs (X, T, L, N) into the touch? Macro. Touch? Checks if a click is inside an event (between T and T+L). The polyphonic output is then sent to the output. The From Voice macro outputs the number of the voice. This is done by iterating over the voices, breaking the iterator when a true value is found, and outputting the iterator counter value.

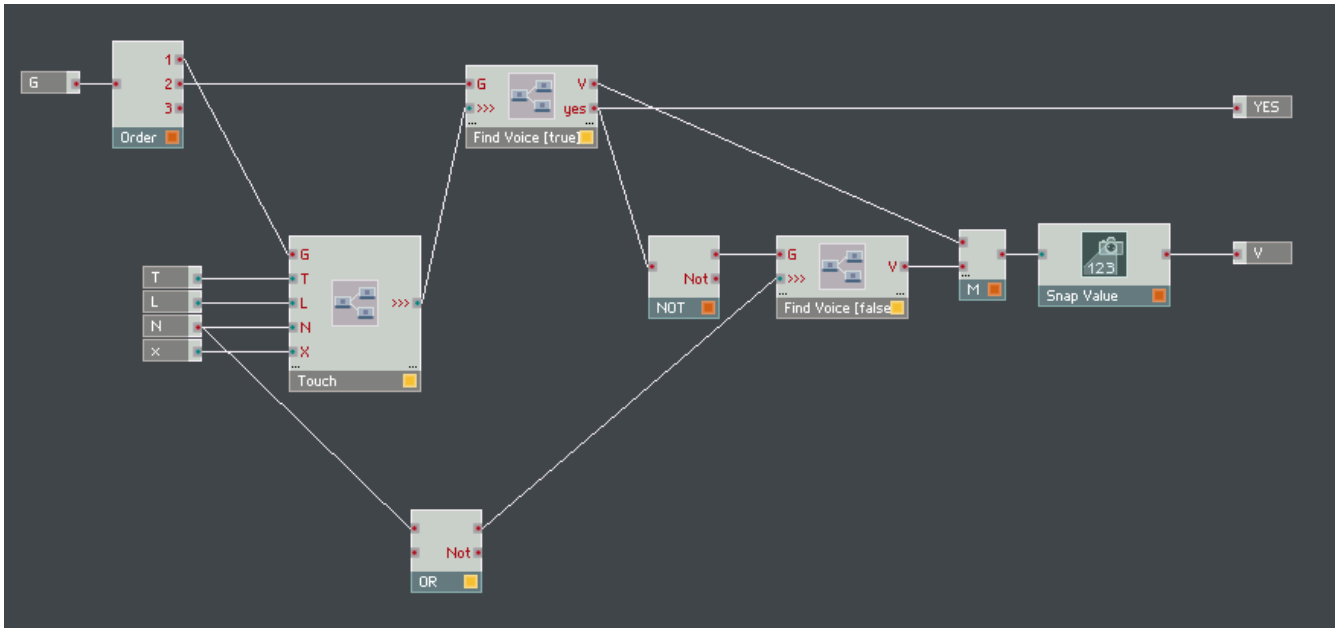


Abbildung 40: touch analyze macro

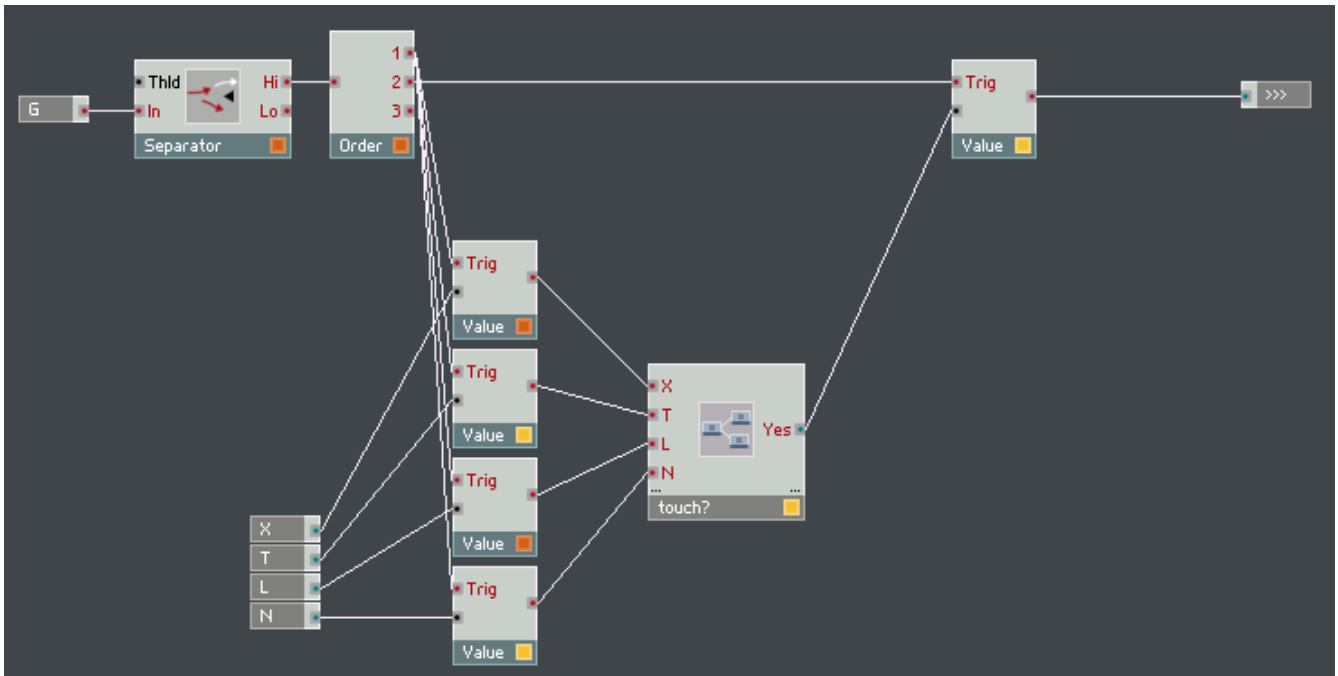


Abbildung 41: Touch Macro

