

ANDB Ensemble

Overall structure

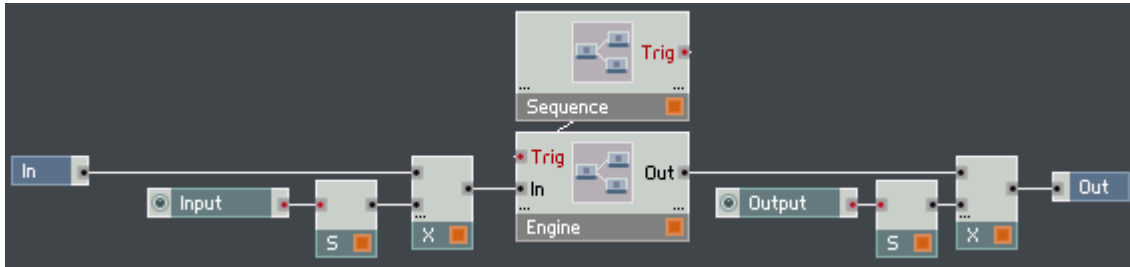


Abbildung 1: ANDB Ensemble

The input is scaled by the “input” parameter, and passed to the recording engine. The playback is triggered by the sequencer, and the output is scaled by the “output” parameter. Nothing fancy here.

Sequencer structure

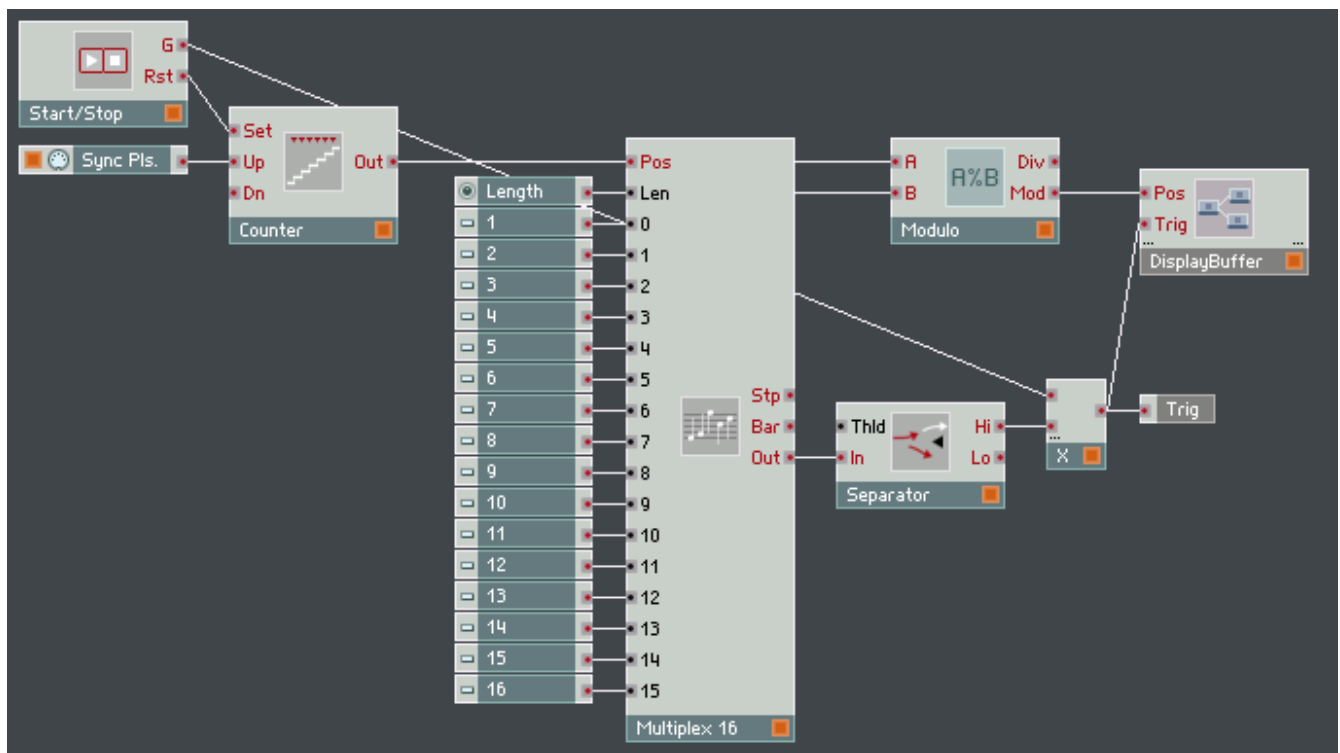


Abbildung 2: Sequence Macro

The sequencer is controlled by the standard MIDI in macro “Start/Stop” and “Sync Pls.”. However, it uses its own position counter, and not the MIDI “96 pos” macro. On start/stop, the counter is reset (the Rst output of the Start/Stop macro is wired to the Set port of the Counter), and on each 16th step the counter is incremented (the Sync Pls. is set to pulse on each 1/16 for the duration of a 1/32). The sequencer consists of 16 buttons polled through a sequencer “Multiplex 16” macro. The output of the

sequencer is filtered to only allow positive events, and multiplied by the “Start/Stop” gate so that it doesn't fire when the ensemble is stopped. The “DisplayBuffer” displays the currently playing buffer.

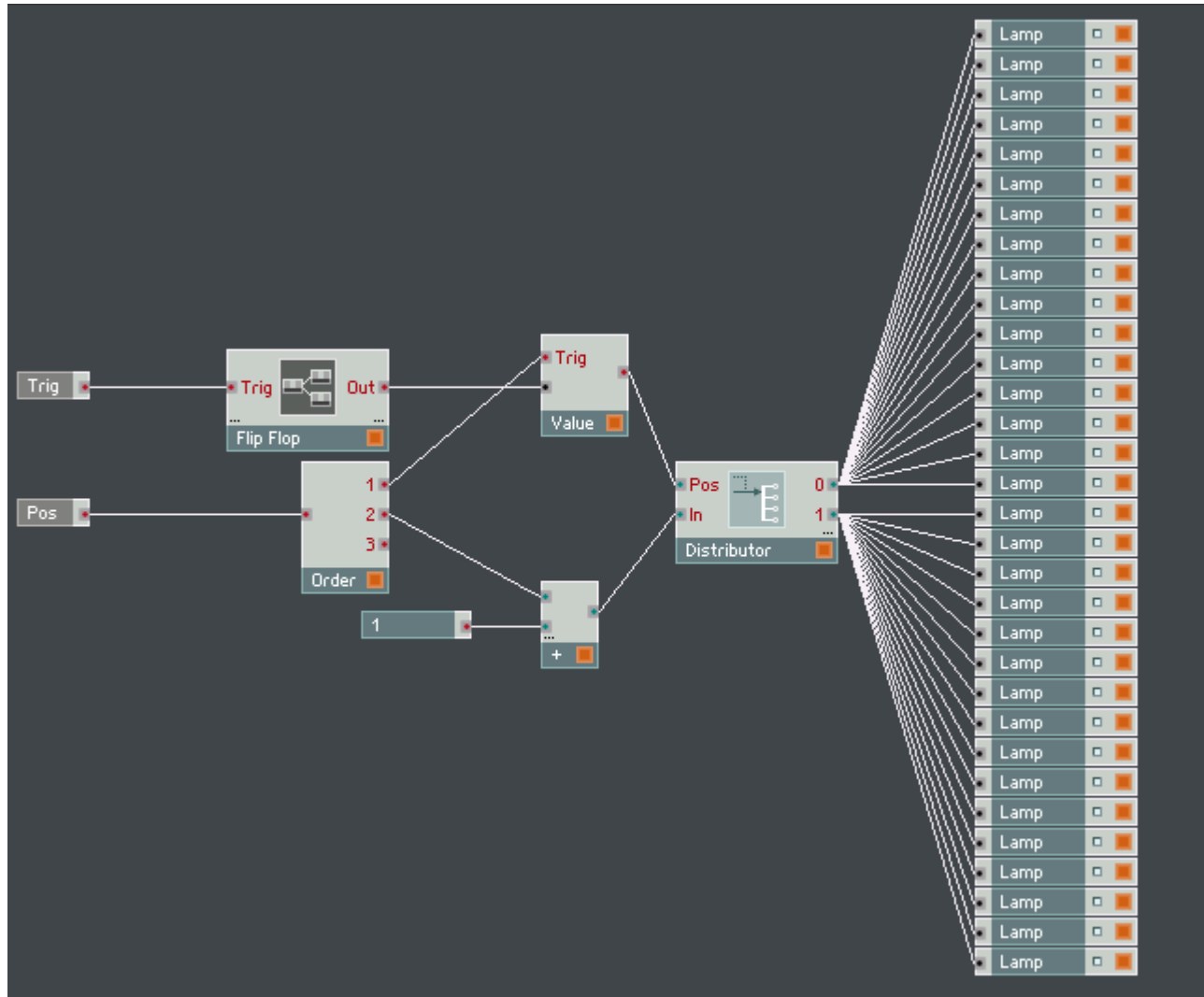


Abbildung 3: DisplayBuffer Macro

On each trigger, a FlipFlop is used to invert a signal (from 0 to 1 to 0 to 1 on each trigger). On each position update, the value of the flipflop is polled and used to distribute the position value to either the top LED row or the bottom LED row.

Engine Structure

The incoming signal is recorded by the ANDB core cell. The output of the ANDB cell is fed back into the input through a Peak EQ, whose pitch is modulated by a DR envelope triggered by the sequencer triggers. The feedback amount is controlled by the “Feedback” control. The sequencer triggers are used to switch the buffers in the ANDB cell. The recorded audio is played back, and the incoming audio is recorded into the previous playback buffer. The crossfade time at switching events is controlled by the “Crossfade” control.

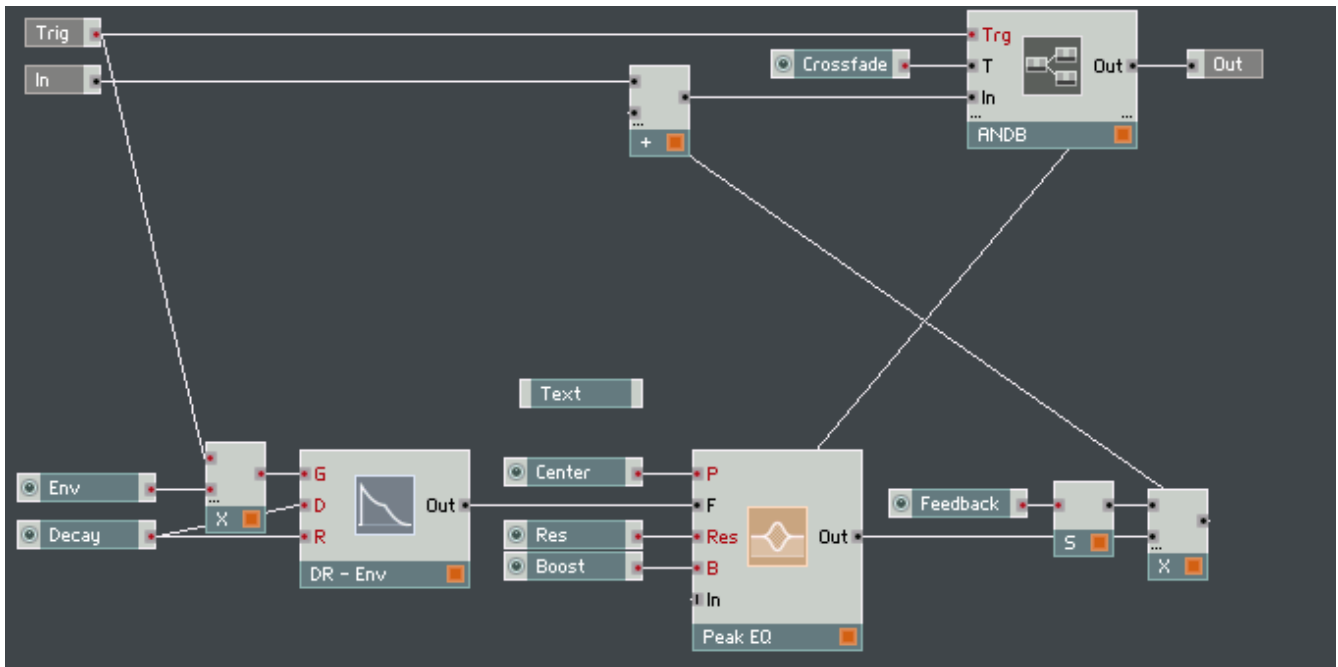


Abbildung 4: Engine Macro

ANDB Core Cell Structure

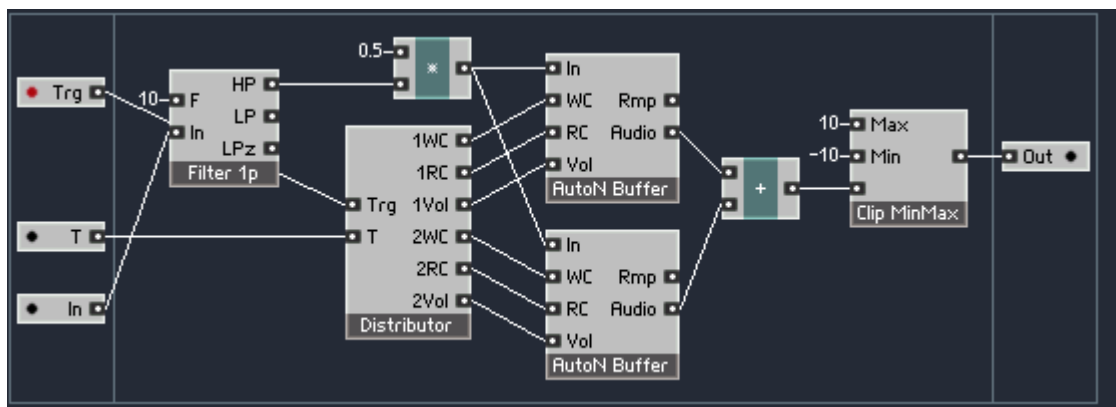


Abbildung 5: ANDB Core Cell

The incoming signal is passed through a highpass filter with a cutoff frequency of 10 Hz, and divided in amplitude (multiplied by 0.5). The core cell features two autonormalized buffers (AutoN Buffer), controlled by the “Distributor” macro. The output of the AutoN Buffers is added and clipped between -10 and 10 before being sent to the output.

Distributor Macro

The distributor consists of a flip-flop that switches on trigger events. The flip-flop distributes the

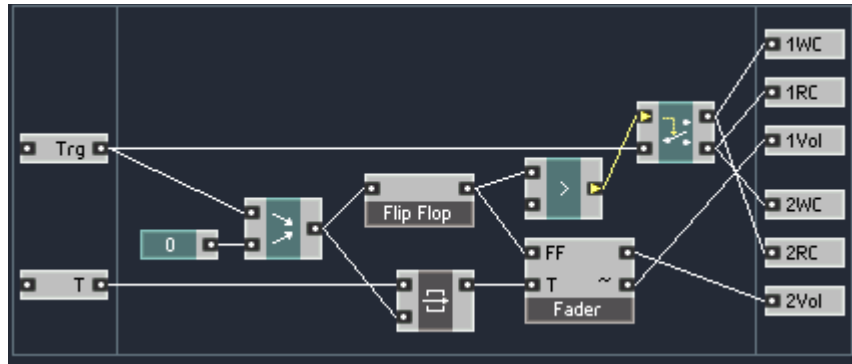


Abbildung 6: Distributor macro

trigger events either to 1WC and 2RC (when the flipflop value is 1), or to 1RC and 2WC (when the flipflop value is 0). The 0 constant fires on core cell initialization and resets the flipflop to 1. The fader time is controlled by the Fader macro, which controls the volume of the first buffer (1Vol) and the second buffer (2Vol).

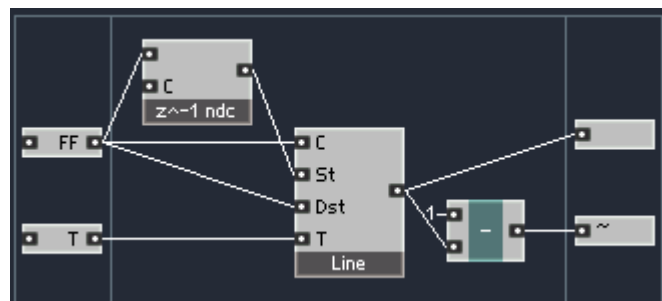


Abbildung 7: Fader macro

The Fader macro fades from the old flipflop value to the new flipflop value in T ms upon a reset of the FF value. The volume and the inverted volume (the 1 – value at the bottom) are used to control the volumes of the playback buffers. The old FF value is obtained using a z^{-1} ndc macro, and fed into the St port of the Line macro. The new value is fed into the Dst port, and a clock pulse is given by wiring FF into the C port. The St and Dst values are latched on C in the Line macro, so the SR.C clocked z^{-1} ndc doesn't matter.

The Line macro is a pretty complicated feedback beast. The Start and Dest values are sampled on the C tick, the T value is converted from ms into samples using the ms-samples macro, and the difference between Start and Dest is divided by the sample count, thus getting the volume increment at each SR.C tick. The Start value is fed into a Merge module which is the central element of the line macro. On init, the Start value gets through. The output of the merge value is “delayed” by a sample using the z^{-1} macros. There are now two similar parts, depending on whether Start is bigger than Dest, or Dest is bigger than Start. The upper part handles the case where $Start < Desc$. It checks whether the current output value is smaller than Dest, and if it is the case, the output is incremented by the increment and sent to the output and back into the feedback loop. The lower part handles the case where $Desc > Start$, and works similarly.

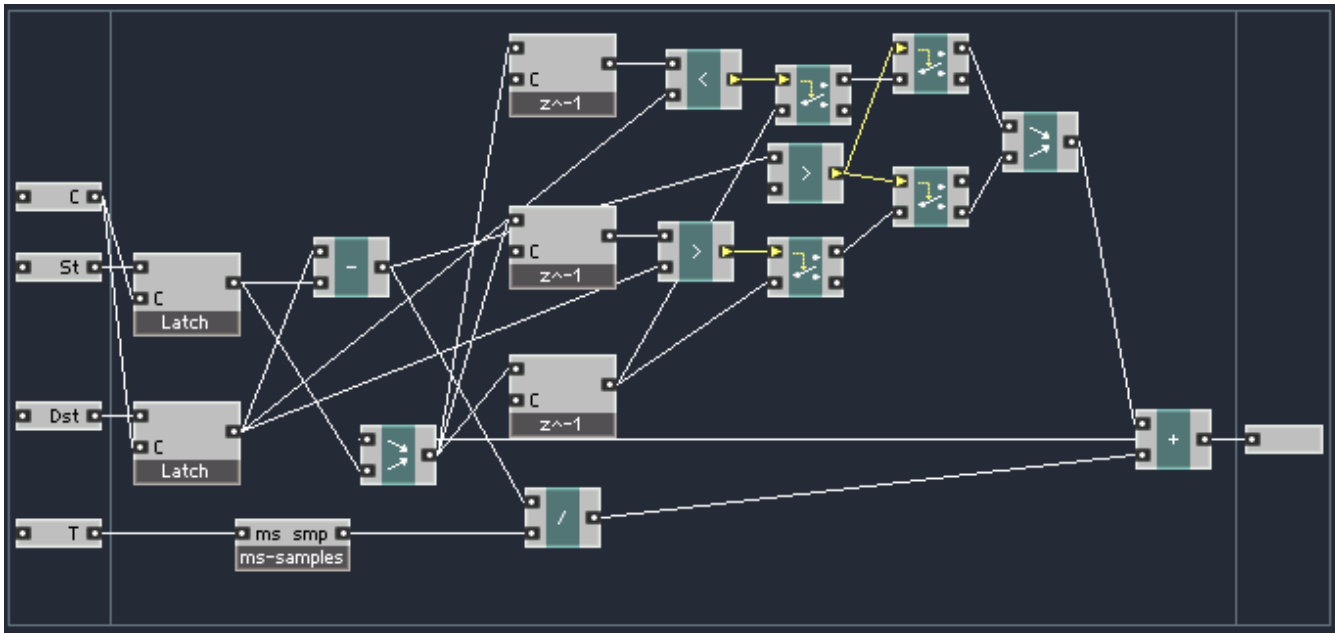


Abbildung 8: Line Macro

AutoN Buffer Macro

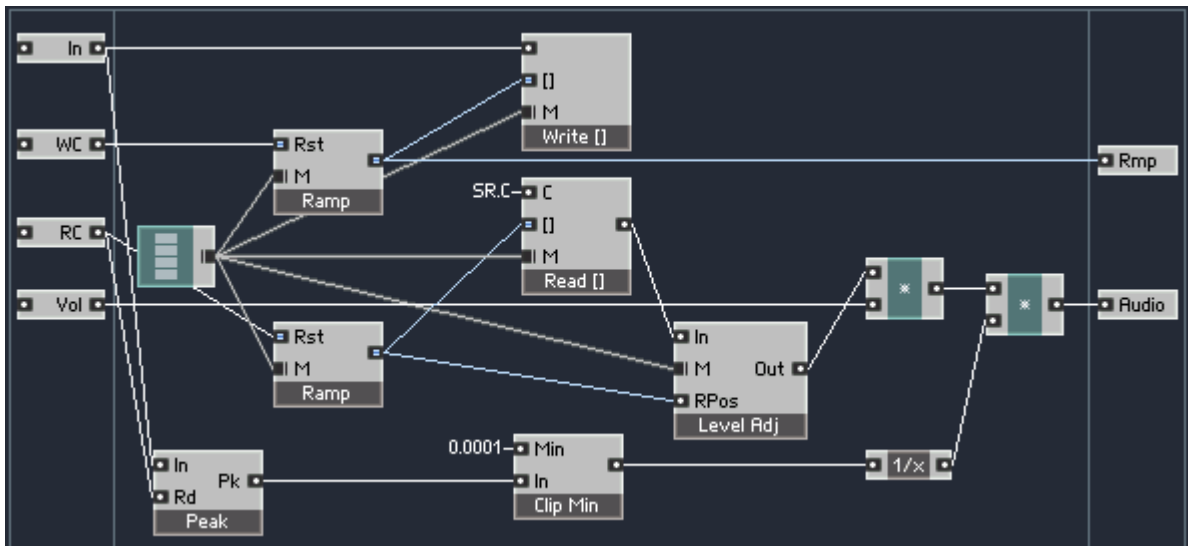


Abbildung 9: AutoN Buffer

The AutoN Buffer can be used to record and playback audio data. The audio data is level adjusted. The In port is the input audio data, the WC input is used to trigger the recording ramp, the RC input is used to trigger the playback ramp, and Vol is used to control the output volume. The “Ramp” macros are a simple upward counter, reset on Rst, and counting up to the size of the latched array. Its structure is pretty similar to the Line macro found in the Fader macro. On reset, a 0 value is latched into a Merge module, and the output is incremented by 1 while it is smaller than the array size.

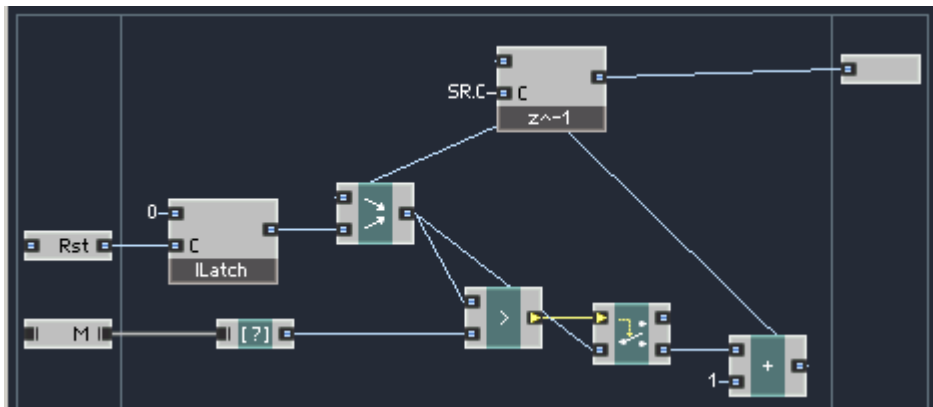


Abbildung 10: Ramp Macro

The incoming audio data is written into the array using the write ramp and the Write [] module. The audio data is played back using the read ramp and the Read [] module.

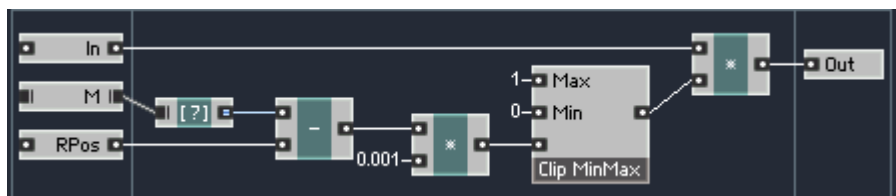


Abbildung 11: LevelAdj Macro

The LevelAdj macro is to implement a fade out on the last 1000 samples of the audio buffer. The length of the array minus the read position are divided by 1000 and clipped between 0 and 1, so most of the time the adjustment is 1, only on the last 1000 samples.

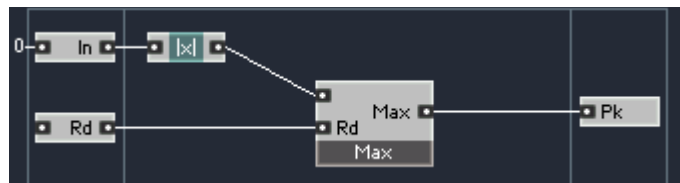


Abbildung 12: Peak Macro

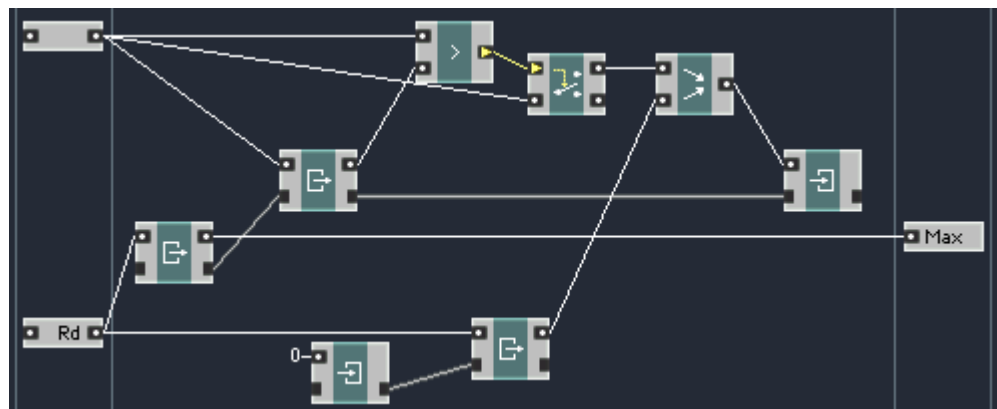


Abbildung 13: Max Macro

The Peak detector works by keeping the maximum absolute audio input value over the whole playback duration. This is achieved by feeding the absolute value of the audio input into a Max cell, which is reset on a Read event, and keeps the maximum value by writing it into a memory cell. If the incoming value is bigger, this value is written to the memory cell. Thus, the output of the cell is the maximum

sample value of the recorded audio (as triggering the recording also sends a 0-event on the RC input). The maximum value is the inverted by a “1/x” module, and multiplied with the read audio signal to normalize the recorded audio.