



## Joranalogue Compare 2 Dual Window Comparator User Guide

[Home](#) » [joranalogue](#) » Joranalogue Compare 2 Dual Window Comparator User Guide 

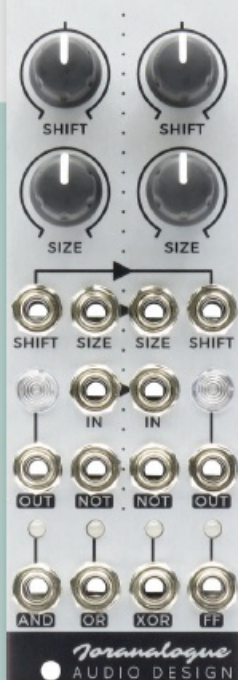
### Contents

- 1 Joranalogue Compare 2 Dual Window Comparator
- 2 INTRODUCTION
- 3 CHAPTER 1: COMPARE 2 BRIEF THEORY
  - 3.1 COMPARE 2 PARAMETERS
  - 3.2 LOGIC OUTPUTS
  - 3.3 BYPASSING THE COMPARATOR CIRCUIT
- 4 CHAPTER 2: COMPARE 2 AUDIO RATE PATCHES
  - 4.1 PWM
  - 4.2 SAW WAVE PHASE ANIMATOR
  - 4.3 PHASE LOCKED LOOP
  - 4.4 BITCRUSHING
  - 4.5 MAKE ANYTHING STEREO (STEREO BIT CRUSHING)
  - 4.6 XOR RINGMODULATION
  - 4.7 WAVE RECTIFICATION
- 5 CHAPTER 3: COMPARE 2 LOGIC AND DC PATCHES
  - 5.1 TRIGGER TO (DELAYED) GATE CONVERTOR
  - 5.2 (RANDOM) EVENT SELECTOR
  - 5.3 DIVIDE BY 2
  - 5.4 INVERT TRIGGERS AND GATES
  - 5.5 VOLTAGE CONTROLLED SWING
  - 5.6 DRUM PATTERNS
  - 5.7 SET/RESET LATCH
  - 5.8 (NOT A) VCA COMPRESSOR
- 6 Documents / Resources
  - 6.1 References
- 7 Related Posts

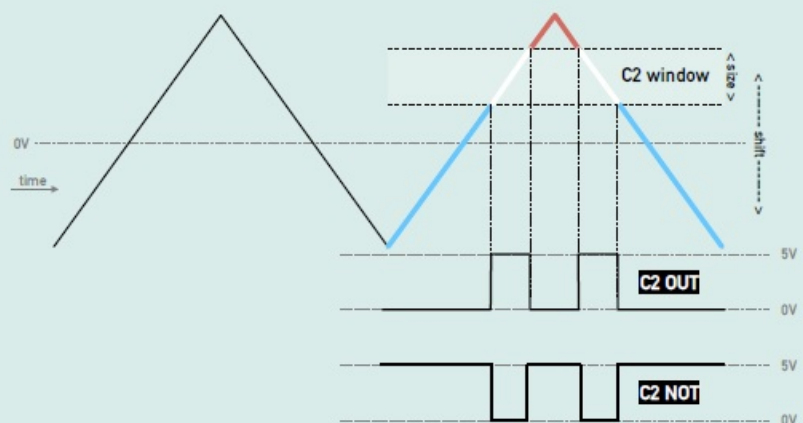
## Joranalogue Compare 2 Dual Window Comparator



COMPARE 2  
DUAL WINDOW COMPARATOR



## C2 PRACTICAL USER GUIDE



## INTRODUCTION

I assume that everyone who ever wondered about how a (window) comparator works at some point came across a diagram quite similar to the one above. While these schematic representations describe what a window comparator does very accurately it's not very intuitive when you start patching things. I've thought about this and I think one of the main reasons COMPARE 2 is so hard to figure out intuitively is because it's not easy for a human brain to convert voltages that are usually represented on a vertical axis to gates that are almost always living on a horizontal axis (time dimension). Having to deal with 2 window comparators in one module makes it even more confusing... but it also means that a lot of interesting patches are possible.

In this practical guide about Joranalogue Audio Design's COMPARE 2 I'll explain the basics of a window comparator. I'll show you a few useful applications, how to apply them in a patch and I'll explain why these patches work in the first place. Instead of going the theoretical route I opted for a very hands-on approach. If you have access to a scope, now would be a good time to use it. If you don't have a scope, you'll be ok as well.

I hope by working your way through this document you'll gain a deeper understanding of what COMPARE 2 and window comparators in general can do for your patches. Have fun!

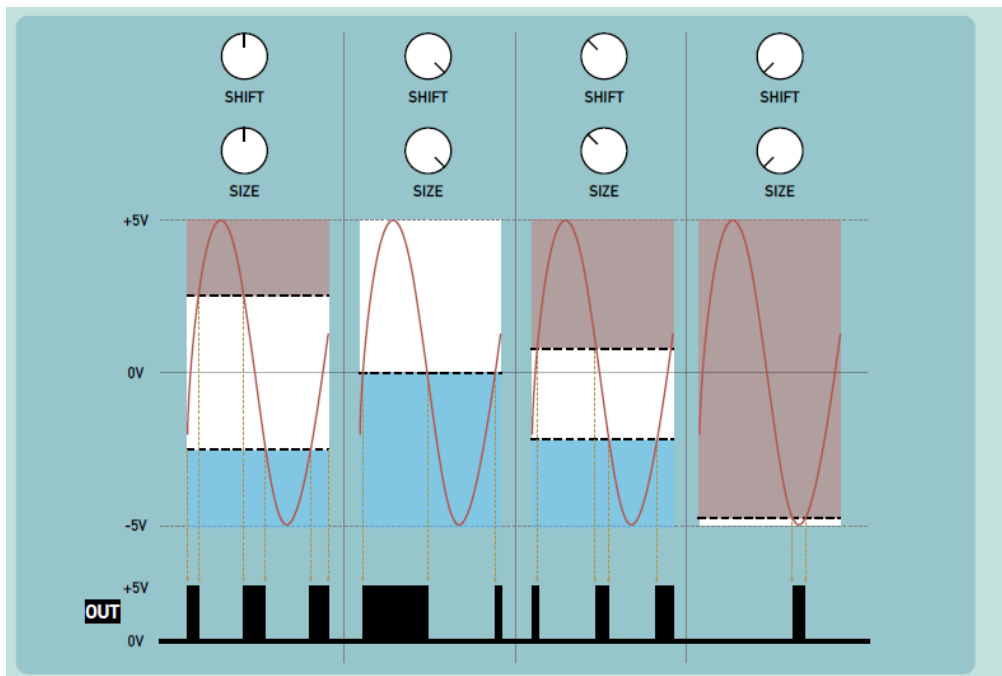
## CHAPTER 1: COMPARE 2 BRIEF THEORY

### COMPARE 2 PARAMETERS

The basics of a window comparator are very well illustrated by the diagram on the previous page. A comparator compares any voltage that you send into it to something else and sends out a signal when it sees that one or the other is higher, lower, equal to, etc. In the case of a window comparator like COMPARE 2 the voltage that you send into it is compared to a range of voltages that fall between two outer edges. This range of voltages is called the window. Whenever the voltage that you send into the comparator is within this range the comparator will generate some kind of signal. COMPARE 2 sends out a 5V gate signal from its OUT jacks when the signal is inside the window. The signal from the NOT outputs is 5V when the voltage you send into COMPARE 2 is not inside the window. COMPARE 2 has two such comparator circuits that can be used separately by patching different signals into its inputs. The input for the second channel is normalised to the first input so you can use both comparator circuits for one signal without having to use a mult .

With the SHIFT knob set to the middle position the window is centered around 0V. Turning it completely to the right adds 5V of offset to the center of the window. Turning it completely to the left shifts the window's center to -5V. The size knob determines the size of the window i.e. the outer edges of this window. When it's turned completely clockwise the size of the window spans 10V. When it's turned counter-clockwise the window is just shy of being completely closed. The LEDs on COMPARE 2 can help understand the inner workings: the LED turns white when the incoming voltage is within the window. It turns blue when it's below the window and red when the voltage is above the voltage set by the window. It turns off when the voltage is inside a window of negative size.

To make sense of the ranges of the size and shift parameters let's imagine a signal with a maximum amplitude of +5V and -5V. The gate outputs (black rectangles) at the bottom are the solution to the simplified question: 'when does the red line cross the white plane?'



## LOGIC OUTPUTS

Because COMPARE 2 has 2 comparator circuits it's possible to perform logic operations with the gate signals of the outputs of the 2 channels. Of course Joran went ahead and included some very useful Boolean (binary) logic functions in COMPARE 2. These logic outputs compare the 2 comparator OUTs in several ways: AND, OR, XOR and FF.

To understand what these Boolean operators do let's take a look at the table below. The 1 and 0 in the first two columns (vertical) represent the states of the gates at the OUT jacks of the 2 window comparators in COMPARE 2 (labeled A and B OUT).

BOOLEAN LOGIC TRUTH TABLE				
A OUT	B OUT	AND	OR	XOR
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

RELATIONSHIP BETWEEN XOR and FF	
XOR	
FF	

The AND output will send out a 5V gate when both the comparator outputs are active. The OR output sends out a gate when either or both are active. The XOR output (exclusive OR) sends out a gate when one or the other comparator outputs are active, but not when they both are. The FF or Flip Flop output switches between sending out a 5V gate and 0V every other XOR gate. The gate stays high until the next time the XOR gate goes high, then it goes low until the XOR gate goes high again, etc. There are other types of Boolean operators like NAND (NOT AND) and XNOR (NOT XOR) that are really interesting if you want to create funky grooves from just a few simple clock sources, but you'll see that the set on COMPARE 2 is a useful bunch. You can create NAND and XNOR signals by running the AND and XOR outputs from COMPARE 2 through a separate NOT logic circuit (SELECT 2

can be used for this as well).

## BYPASSING THE COMPARATOR CIRCUIT

I can imagine a lot of scenarios where you would just want to use the logic section of COMPARE 2 for combining gates or triggers. For standard 5V gates simply connect them to the inputs of COMPARE 2, set the shift knob to the max. position and the size knob to the center position. As long as 0V is not in the window and 5V is this will work. If you're dealing with 10V gates you will need to increase the size parameter.

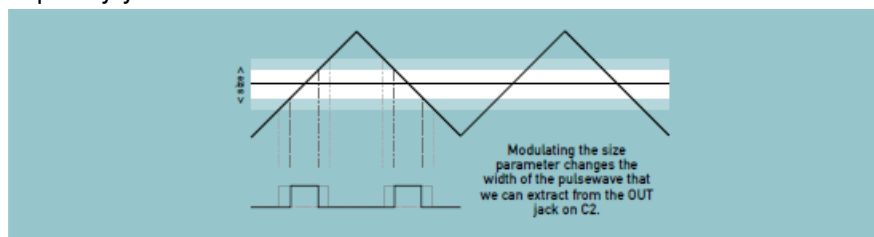
## CHAPTER 2: COMPARE 2 AUDIO RATE PATCHES

### PWM

Patching up a Pulse Width Modulated wave is probably the first thing most people try with COMPARE 2 when it comes to audio rate uses. I think it's the least exotic use of a window comparator and it's also relatively easy to wrap your head around how it works. Let's have a look at a possible patch.



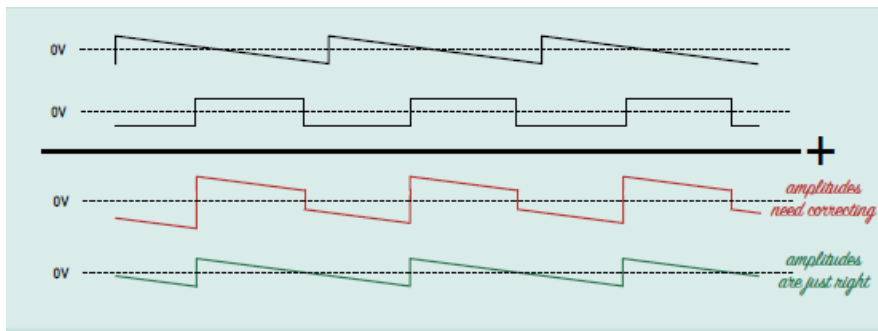
You can use any wave you want to create PWM (everything except a pulse wave) but in this example I'll stick to a triangle wave (those are easiest to draw accurately). For asymmetrical pulses try moving the center of the window around (with the shift parameter). You can use the OUT jack to monitor your PWM wave but you can just as well use the NOT output. Depending on the settings on COMPARE 2 the resulting pulse wave can be shifted one octave up from the frequency you send in.



When modulating the size parameter it's possible to get it to go through zero, i.e. to become negative. Therefore it's perfectly possible to create PWM waves that get so narrow they disappear. If you want to avoid this you should attenuate the level of your modulation before sending it into COMPARE 2. Try patching up a stereo pair for ultra-wide space arps. If you run other sounds through this type of setup the resulting sound might be more accurately described as bitcrush or bit reduction (see 2.4 and 2.5).

### SAW WAVE PHASE ANIMATOR

Every output of COMPARE 2 produces 5V gates which, when they have a period can be described as pulse (or square) waves. Something mathematical happens when we add a saw wave and a pulse wave at just the right amplitudes. Have a look at the following diagram to see what's going on...



You can see that by adding a saw wave to a pulse wave with the same frequency you end up with a phase shifted saw wave. I first discovered this concept in the online manual of the Doepfer A-137-2 and it's stated in that manual that this isn't a true phase shift circuit but merely a circuit based on logic operations (mathematics). Everything really very much depends on the relative levels of the two waves. If you really need the end result to be perfect you should look at the resulting wave on a scope to get the levels just right. It's also possible (but not necessary) to offset the output of COMPARE 2 to make a bipolar pulse wave (like in the diagram above) before mixing it with the saw wave. Modulating the shift or size parameters moves the relative phases around but the range might be limited. Careful tuning of the modulation ranges through an attenuator is necessary. Patch two of these and mix in the original saw to make a saw wave cluster (unison/supersaw). Stereo panning two of these is something you should definitely try. You can use this technique to animate other types of waveforms (without bothering about the attenuation) to achieve really interesting (clusters of) waveshapes.



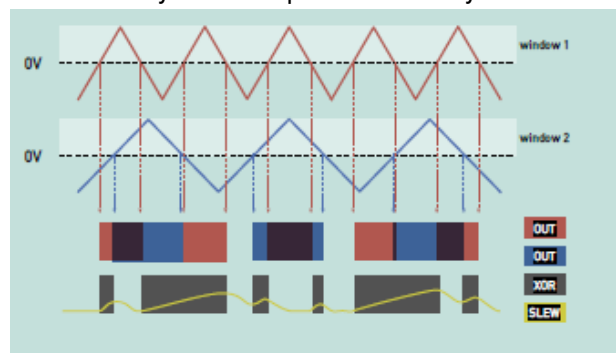
## PHASE LOCKED LOOP

There's a Doepfer module (A-196 PLL) that does phase locked loops exclusively. It's built around a phase comparator that looks at the relative phases of 2 oscillators to get them, as the name of the circuit suggests, locked in phase. Luckily COMPARE 2 can function as a phase comparator as well to make a PLL possible. To get this to work you need to use the XOR output of COMPARE 2 and run this signal through a slew limiter before patching it into the V/oct input of a slave VCO. With some careful nudging something wonderful emerges. I think you'll find that leaving in some inaccuracies is much more rewarding than getting it perfect.



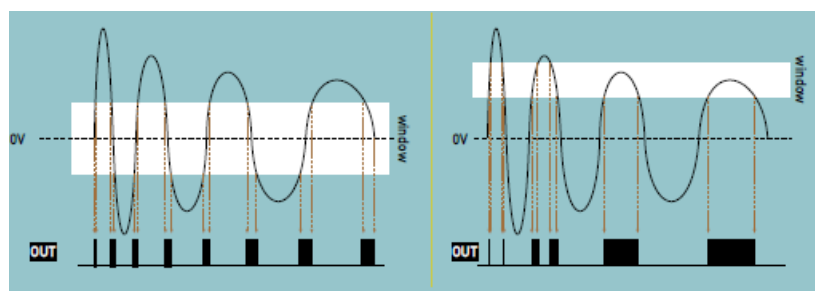


The diagram below is a really (really really) simplified visualisation of what's happening. By setting the two windows to exactly the same values COMPARE 2 acts as a phase comparator. The XOR output sends out gates whenever the phases of the two waves are completely different (in this example it compares whether the phase is positive or negative). By sending the XOR gates through a slew limiter these differences are then used to 'nudge' the frequency of the slave VCO (the black panel Pickle VCO) just enough so that the phases align. Notice that this circuit creates a feedback loop where the phases are constantly adjusted while being analysed. This setup will only work when the frequency of the slave VCO is set to be lower than the frequency of the master VCO. If the VCOs don't lock in immediately try tweaking the frequency of the slave VCO until you hear it lock in. The setting of the slew limiter adjusts how accurate the pitch tracking is. Try small amounts of slew – even a filter at the lowest frequency might work here. You can use any wave output to audition your sound.



## BITCRUSHING

Bitcrushing audio might be the easiest thing to do with COMPARE 2, and while it works at practically every setting you can further shape the tone using the shift and size knobs (or CV) and you can use every output of COMPARE 2 for this effect as they all will produce 1 bit versions of the signal you send into the input. If you connect two sounds to the inputs of COMPARE 2 you can mixcrush them – which is a lot of fun – but let's stick to one signal for now.



In the two examples shown in the diagram above you can see that depending on the setting of the window the output of COMPARE 2 will be different. Looking at these schematic representations it might seem surprising that when you listen to any of the outputs of COMPARE 2 the audio you hear will be very distinguishable, even when it's converted into 1 bit gate information. Almost none of the amplitude information will be retained, but the information about the frequency of the oscillations will still be present (unless you set the window size to maximum spread, because at that point the output of COMPARE 2 will just be a continuous gate).

To get a bipolar (1,5 bit?) signal try this patch where the output of the second channel of COMPARE 2 is inverted before it's mixed in with the output of the first channel.



### MAKE ANYTHING STEREO (STEREO BIT CRUSHING)

So you made a nice drumloop, bass or marimba patch but you want to make it stereo without using any of your dedicated stereo effects. If you don't mind to crush your sound into one bit pieces COMPARE 2 can be your best asset for this. Just connect your source to the first input on COMPARE 2 and set the shift and size knobs of the two window comparators to slightly different settings. Then use the signals from both the OUT jacks (or both the NOT) jacks panned left and right on the mixer. You can use the AND and OR outputs as well, these will produce similar frequencies as the original sound. If you use the XOR or FF outputs you will notice that at some settings the frequency might seem respectively higher or lower. If one of the logic outputs doesn't produce an audio rate signal this probably means that the windows of the two comparators are too similar or there is some kind of overlap that's causing a logic output to produce a static gate. Try changing the shift and size parameters and you should be ok. Don't forget that you can modulate these parameters with LFOs and audio rate signals! This will result in really gnarly fuzzy sounds, I promise.



### XOR RINGMODULATION

While this should technically be in the section about logic patches, I think most people will use ringmod at audio rates so I think it's ok to put it here. XOR ringmod is really similar to regular ringmodulation but it uses square waves and binary logic instead of a four quadrant multiplier. To get the full grasp of what ringmodulation is I urge you to take a look at the website 'synthesizeracademy.com' but in the tables below I'll try to show the resemblance between regular ringmod and XOR ringmod.



multiplication in a four quadrant multiplier			multiplication in an XOR multiplier		
	-	+		0	1
+	-	+	1	1	0
-	+	-	0	0	1

It's ok if none of this means anything but it's nice to know that a lot of vintage synths actually used XOR logic to produce ringmodulation sounds (the Arp Odyssey and the Korg MS20 for example). To achieve this with COMPARE 2 you need to patch 2 sounds into the two signal inputs. The signal coming from the XOR output will be a XOR logic multiplied version of the 2, containing interesting sidebands. Play around with the relative tunings of both sounds, you can even adjust the windows of both comparators (manually or with CV) further increasing the sonic variation (this is similar to doing PWM on an MS20 in ringmod mode – on both VCOs).

In the patch below two sine waves are used, but you can use any wave or sound you want (voices, drums, samples, Daleks, radiobroadcasts, ...). When you find a timbre that you like and you want to use it in a sequence, try sequencing both of the sources with the same pitch CV. Run the output through a filter for a more vintage sound.

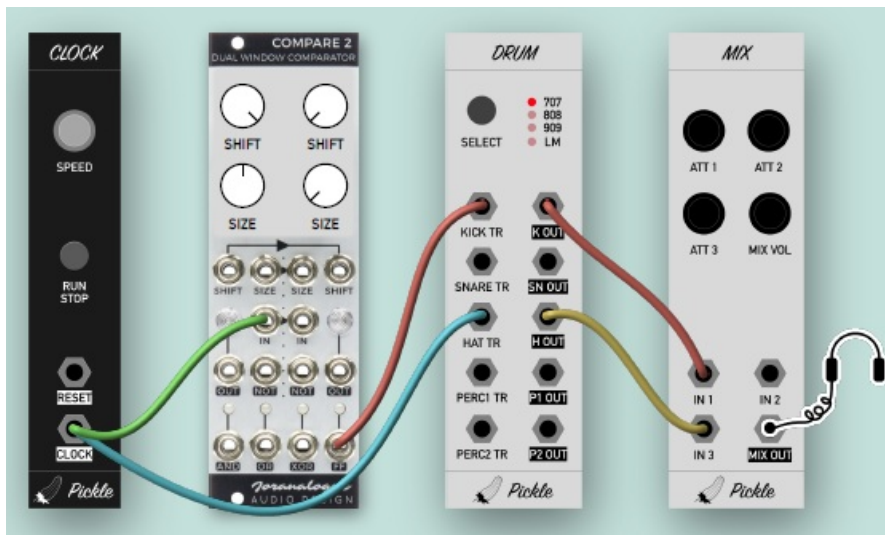
Try listening to the other logic outputs as well, there might be some interesting surprises!



## WAVE RECTIFICATION

Using COMPARE 2 to half-rectify a wave is really straightforward although I have to mention that it can be very tedious to get the knobs set up precisely right (don't rely on this patch if your reputation is at stake).

All you need to pull this off is a VCA or a switch (SELECT 2 for example) that only lets the signal pass through when it's above 0V. COMPARE 2 can analyse the wave to see when that happens. Set it up in such a way that it sends a gate from the OUT jack (opening the VCA) whenever the wave is above zero: the size knob should be left in the center position (5V spread) and the shift knob should point to the 3 o'clock position (+2,5V). This patch can of course be used for low rate signals as well.



You can take this one step further to patch a messy full wave rectifier using an inverter (or polarizing mixer) and a few extra cables. There's easier ways to achieve precise full wave rectification but it's a fun patch to try.

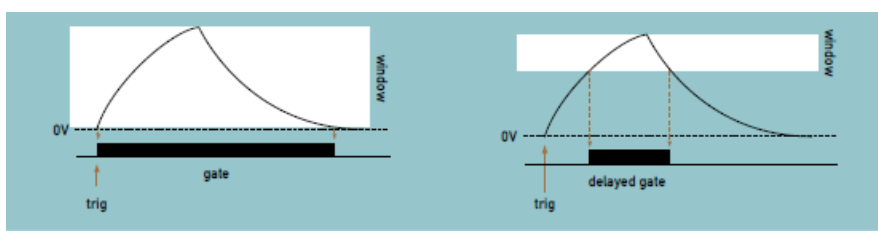


## CHAPTER 3: COMPARE 2 LOGIC AND DC PATCHES

### TRIGGER TO (DELAYED) GATE CONVERTOR

This patch is as straightforward as it gets, and still it might not be the first thing that comes to mind when looking at the skillset of COMPARE 2. There's a lot of uses for trigger to gate convertors and it's not something that's easy to do without a module that has some kind of dedicated end of attack, end of cycle, etc output. While most function generators have this feature if all you have is an envelope generator you probably won't have access to this function. Delayed gates are also really useful in some patches and not easy to achieve without dedicated hardware... Use them to trigger delayed modulation, effects, ...

**Let's have a look at how it works.**



Patching an envelope into the input of COMPARE 2 gives you several options. Depending on how you set the size parameter you should be able to create a gate with a length of your choosing. You could modulate the size parameter to create gates of different lengths (interesting when you patch these gates back into ADSR envelopes,

function generators or slew limiters).

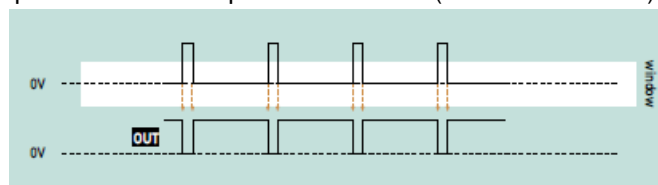
By adjusting the shift parameter just right you can specify the delay of the gate (following the initial trigger for the envelope). Note that the window in the left diagram is slightly above zero to make sure the gate closes when the envelope reaches 0 volts (finetuning is needed). In the following example I'm using a slew limiter as an AR envelope that receives the delayed gate, opening a VCA. Another way of using a chain like this could be to add delayed modulation to a voice. Vibrato would be a classic example of this.



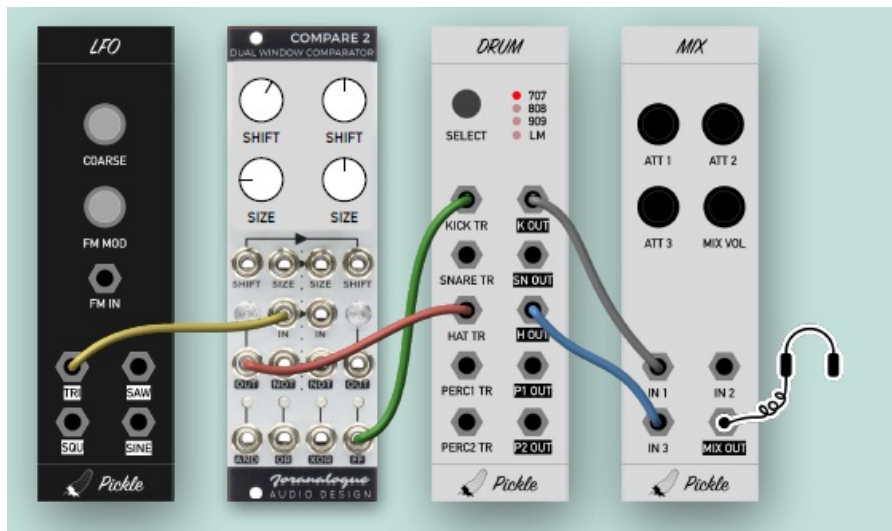
## (RANDOM) EVENT SELECTOR

Using a random voltage is the obligatory modular thing to do, but even this can get a bit repetitive (?) or you might just want a little bit more control over what happens when certain voltages are reached. Using a comparator to trigger or gate events around your patch at set ranges might be an interesting experiment that you can even use in a regular repetitive sequence (for example to trigger a vibrato on the high notes and a filter wobble on the bassy notes, or to change the setting for the decay time for different octave ranges in your sequence, to add noise to your high notes or to gate a sub to accompany the low register). The only limit is your imagination (or the amount of cables you have at your disposal).

Take a look at the following pitch sequence. By carefully adjusting the two windows on COMPARE 2 it's possible to extract separate gates for the low (A) and high (B) parts of the melody. You can adjust the windows to include all the notes, create some overlap or leave some space in between (like shown below).



Use the following patch as a guide to use gates of the according outputs to open a VCA to send modulation to your VCO (for the high notes) and to add a square wave to your bass notes. The settings of the knobs are indicative, adjust these to fit your sequence. Don't forget that the NOT outputs (and logic outputs) can be used alongside the OUT gates to create interesting patches.



## DIVIDE BY 2

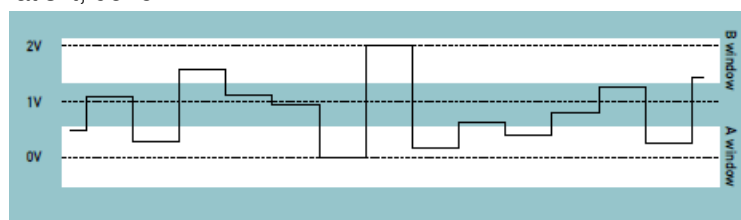
You might have a clock or a pattern that you only want to use every other pulse of. You can use a clockdivider for this but COMPARE 2 can be set up to perform this task as well using the FF or FLIP-FLOP output. By connecting a clock, trigger or pulse (even audio rates are possible) to one of the inputs on COMPARE 2 and making sure that one channel of the comparator reacts to the incoming pulse while the other one is either never active or always, the signal from the FF output will send out an alternating gate signal.

The example shown below is using a steady clock, but don't let this unimaginative patch stop you from trying to make subdivisions to other types of patterns!



## INVERT TRIGGERS AND GATES

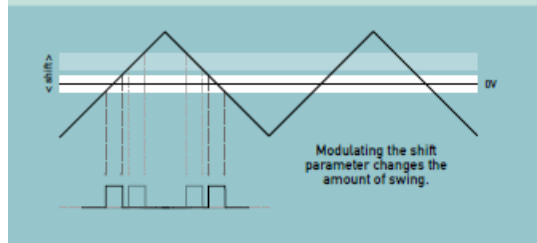
At some point you might need the inverted versions of your gate or trigger patterns. When I first came across this need I instinctively reached for a polarizing mixer but soon realized that, for this to work, an offset needs to be added to the inverted signal. There are some modules that can do this without much patching but using COMPARE 2 for this task is really hassle free. To invert triggers or gates you connect the trigger or gate to a signal input on COMPARE 2, leave the shift and size knob in the center position. At this point any voltage between -2,5V and 2,5V should be inside the window so the OUT jack should generate a 5V gate when no trigger or gate is present. Whenever there's a trigger or gate the voltage rises above the window so the gate from the output of COMPARE 2 goes low. That's it, done.



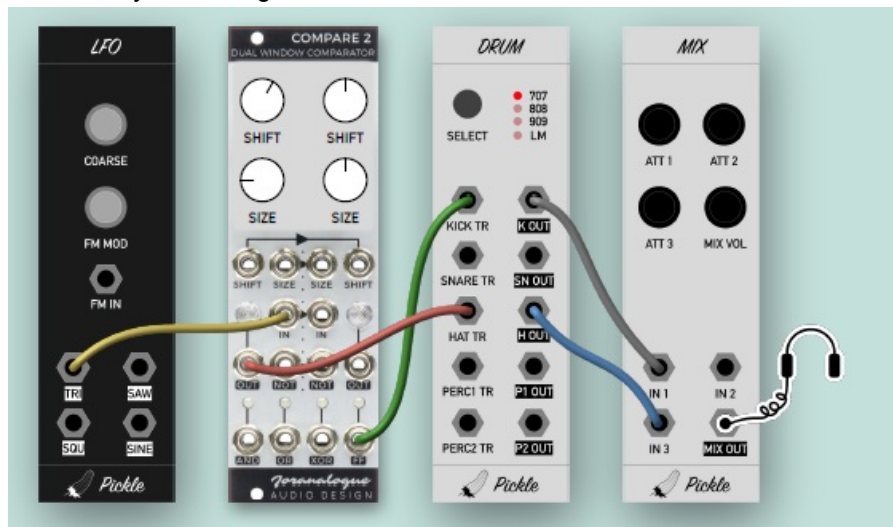
## VOLTAGE CONTROLLED SWING

Using COMPARE 2 to achieve voltage controlled swing is mentioned briefly in the manual. It's essentially the same concept as the previously explained PWM patch. The main difference is that we're extracting pulses at LFO rates instead of audio rates.

Connecting a triangle or sine LFO to the signal input is the starting point. Turn the size knob to the left to make sure that you get short pulses. For a straight clock without swing or shuffle leave the shift knob centered. To introduce swing turn the shift knob either to the left or to the right. You can control the amount of swing externally by modulating the shift parameter with CV.



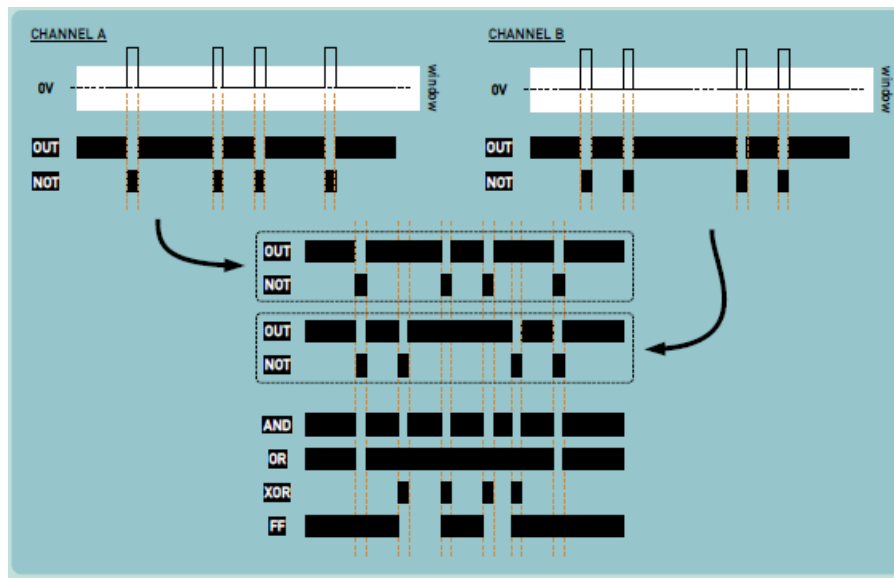
In the patch example below the swinging clock is used to trigger a hi-hat sample, but you can use this shuffling pulse-train (choo-choo) for anything you want. One obvious use for this patch would be as a clock for your sequencer. It's also very rewarding to play around with the settings of the second channel and use the logic outputs to trigger or gate other things in your system. In the example below I used the FF output to trigger a kickdrum. You can achieve tidy shuffled grooves as well as off-kilter beats.



## DRUM PATTERNS

Using Boolean logic is a fast and easy way to create interesting rhythms from basic clocks. Because COMPARE 2 is a dual window comparator it's even possible to create gated patterns from LFOs and other CV sources. Even though there's no dedicated NAND and XNOR outputs (really interesting for drums) these can be made by running the AND and XOR outputs through a separate NOT logic circuit (you can use SELECT 2 for this). Take a look at the schematic example below for an idea of the possibilities...



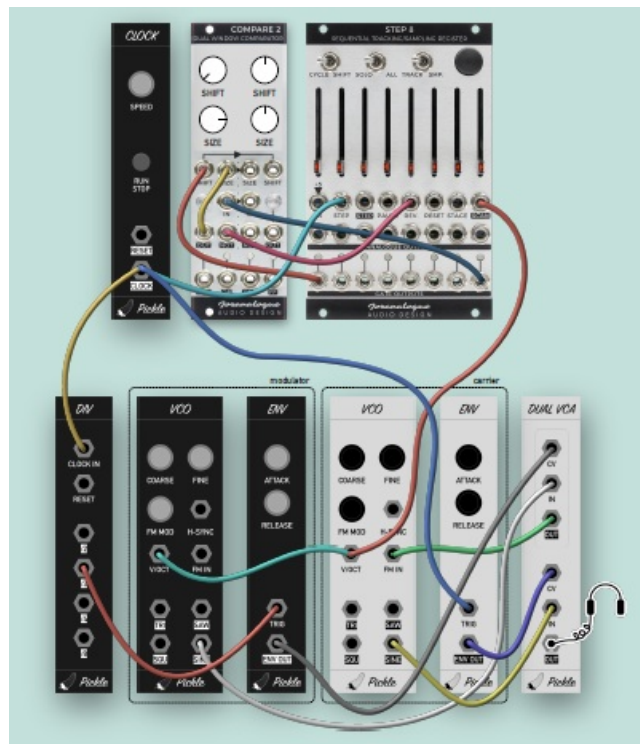


In the patch below we'll just use a clock and a division (/2) to generate interesting patterns to trigger percussion samples. Use a drumsequencer instead of clocks if you want to get really confused.



## SET/RESET LATCH

This is a patch straight from the manual of COMPARE 2. I asked Joran about this patch and he told me that one of his personal uses for it was to create a pendulum sequence with STEP 8 (where the sequencer plays forward until the last step, then reverses and plays backwards until it reaches the first step, then reverses again etc). Below is how to patch it. The bottom row is a dynamic FM voice and really not a part of the set/reset latch routing.



**(NOT A) VCA COMPRESSOR**

Every VCA compressor uses some kind of side-chain circuit with a copy of the audio signal to extract an envelope (not just an envelope follower). This envelope is used to control a VCA that attenuates the audio when it gets too loud. On VCA compressors there's usually separate controls for threshold (the level or volume from where we want to start to compress the audio), the ratio (the relationship between the increase in volume of the original sound vs. the sound after compression) and the attack and release times (how fast the compressor reacts to volumes that exceed the set threshold. Sometimes there's a soft-knee setting (in dB) but let's not get into that.

DISCLAIMER: This is not how the detection circuit on a VCA compressor, or any other type of compressor works. I included this patch because it's interesting to try to get your head around it – and as a starting point for a more elaborate version of this patch. At certain settings the behaviour might even not resemble compression at all. Use this idea at your own discretion.

In this patch COMPARE 2 takes care of the threshold part of the circuit: it detects whether a signal is inside the window. Everything outside the window is above the threshold. If the amplitude of a signal is too loud COMPARE 2 sends out a gate from the NOT output which we slew (creating an attack and decay) and invert (so it becomes a ducking envelope). We will use this signal to compress the multed audio. For this to work we first need to add a positive offset and connect this signal to the CV input of a VCA. Adjusting the frequency of the slew limiter adjust the attack and release times. You could use a function generator instead of a slew limiter to set separate attack and release times. Attenuating the inverted, slewed NOT output adjusts the compression ratio. Some fine tuning will be necessary.




Thanks to all my supporters and enthusiasts. Special thanks to these wonderful people:

César for kickstarting this deep dive into COMPARE 2:


hoofjaw for submitting a few really interesting ideas and patches and for proofreading the guide; Nico for being a genuine person and for making a difference;

Joran for making awesome precision gear and for being so very supportive of my work.

## Documents / Resources

	<a href="#">Joranalogue Compare 2 Dual Window Comparator</a> [pdf] User Guide Compare 2 Dual Window Comparator, Compare 2, Dual Window Comparator, Window Compar ator, Comparator
---	---

## References

-  [The Synthesizer Academy](#)
- [User Manual](#)

[Manuals+](#), [Privacy Policy](#)

This website is an independent publication and is neither affiliated with nor endorsed by any of the trademark owners. The "Bluetooth®" word mark and logos are registered trademarks owned by Bluetooth SIG, Inc. The "Wi-Fi®" word mark and logos are registered trademarks owned by the Wi-Fi Alliance. Any use of these marks on this website does not imply any affiliation with or endorsement.