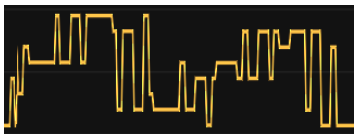
“BUMPER” by Allieway Audio

An Owner’s Guide



Concepts: WTF is a “Bumper”!?

In programming, we always have numbers changing over time. We call the places in memory we store these numbers to *“variables”*. When manipulating these variables, it is common to want to *“increment”* a variable (that is, to increase it by a fixed amount, like 1), or to *“decrement”* it back down by a similar fixed value. Some programming languages have a handy way to do this quickly, known as *“bumping”* - for example, in C++ if I wanted the variable called **“♥”** to go up by 1, I could simply type **“♥++”** and this would cause the variable to *“bump”* up by 1. If **“♥”** was worth 200, after bumping it would be 201. Bump it up again, and it’ll be 202! Back down **(“♥--”)**, and it’s 201 again. Nifty.

This concept of bumping variables up and down gave me an idea - *what if I gave players a bank of several stored voltages, which they could “bump” up or down via triggers and gates?* And what if there were *upper and lower boundaries* which, when reached, would cause the value to *“wrap around”,* starting over from the other boundary? With this, the Bumper concept was born!

At its simplest, *Bumper takes triggers/gates and will create stepped/jumpy voltages which hop up and down by different amounts!* Oftentimes this creates something akin to a *“staircase”, step-sequence, or low-fi sawtooth wave.* Another way to think about Bumper is that at its heart are *4 sample-&-holds* - only, instead of sampling and holding a voltage when there is a trigger, *the voltage stored is shifted up or down by a defined amount,* wrapping around when it exceeds its voltage boundaries! *Triggers are also output when wrapping occurs, making sub-rhythms and grooves!*

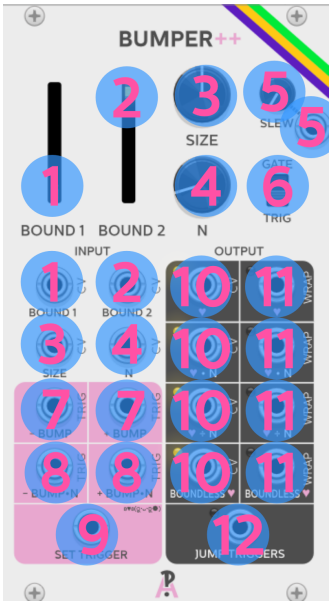
Now, I describe Bumper as a *“Quad Stepped Digital Accumulator”.* It is *“Quad”* because it has *four related but fundamentally independent* stepped digital accumulators running simultaneously, which I call *“bumpers” (lowercase “b” distinguishing them from the module “Bumper”)* for short. The four are friends named: **“♥”** , **“♥ •N”** , **“♥ / N”**, and **“Boundless ♥”.** To simplify things, however, we’ll just focus on just the top unit, named **“♥”.**

“♥”, like all 4 of the bumpers, has two outputs - a **CV output** and a **Trig/Gate output**. The CV output follows the behavior I’ve previously described - when a trig is received at the **“+ Bump”** or **“- Bump”** inputs, *the CV will jump up or down* by the amount you set using the **“Size”** parameter. If the jump would place the resulting voltage *outside of the boundaries* you’ve set using the **“Bound”** parameters, then this CV will *WRAP-AROUND* - if the CV would move over the upper boundary, *it’ll continue it’s travel from the lower one,* and vice-versa.

As mentioned, at the moment this *“wrap-around”* occurs, the **Trig/Gate Output** will fire, outputting a trigger if the module is set to Trig mode, or outputting a high gate until the next bump occurs if set to Gate mode. As a result, the **Trig/Gate Output** will always output a *rhythmic subset of all the gates received at the inputs,* the density of which is dependent on how often we wrap around. *Bumper can act as a clock-divider, or drum sequencer!*

Finally, there is another way we can bump up or down - by using the **“+ Bump•N”** and **“- Bump•N”** trigger inputs. These sound complicated, but all they do is multiply the size of the jump by whatever number you set with the **“N” knob/input.** By default **“N”** is *quantized* to integers 1 through 8 (*round numbers only*), resulting in even clock multiplication/divisions, and more interesting CV patterns when used in conjunction with the other gate inputs. There are a few more little intricacies to this module, but this is basically all that each “bumper” unit is - *stored CVs being bumped up and down by various amounts, and firing out gates whenever they wrap-around!*

Overview: What does each part do?



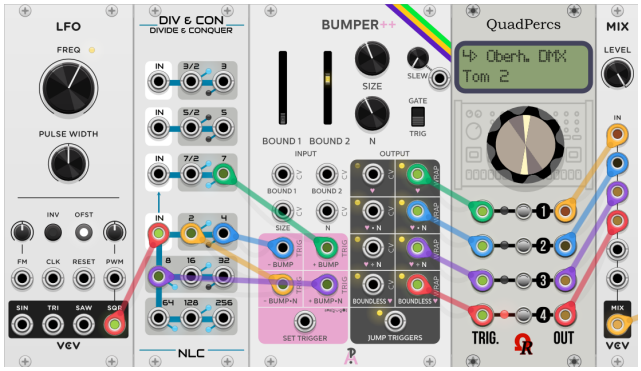
1. **Bound 1 Slider / Input.** Sets the voltage at which the bumpers will *wrap-around* to the other Bound. The slider's light indicates its current value, and the voltage range it spans is determined via the right-click menu (see Appendix). *Only "Boundless ♥" IGNORES these Bounds.*
2. **Bound 2 Slider / Input.** Behaves much the same as Bound 1. Please note that Bound 1 and Bound 2's voltage range spans the same interval *(by default from -5v to +5v)*, so *Bound 1 can serve as either the "high boundary" OR the "low boundary". Whichever* boundary is lower is treated as the "low boundary" - *they can swap these roles on the fly!* As a final quirk, these two sliders also determine the behavior of the *Set Trigger Input (9)* - read on for that...
3. **Bump Size Knob / Input.** Determines the distance traveled with each bump triggered. All the way to the left, there is no movement at all from the bumps. Can be configured via the right-click menu (see Appendix).
4. **N Knob / Input.** N is simply a number which impacts the behavior of a few different parts of Bumper to be discussed further onward - *for now just see it as a number we set.* By default it spans the round numbers from 1-8, though this can also be configured via the right-click menu (see Appendix).
5. **Slew Knob / Input.** Applies a *linear slew* to the four CV outputs (10). All the way to the left, there is *no effect,* but as you increase slew there is a *glide* from value to value, *smoothing out* the stair-steps into *linear ramps*. At the rightmost value, the bumper's CV values are *HELD IN PLACE,* and *will not change* until slew is lowered again, allowing for track-&-hold behavior. *Note that this slew does not impact the four bumper's jump/wrap behaviors in any way, it is simply a slew / filter that is applied to those 4 outputs (10).*
6. **Gate / Trig Switch.** Sets the behavior of all four of the bumper's wrap outputs on the rightmost side of the module (11). When set to *Gate Mode,* if the bumper wraps-around the gate will stay high until the next *non-wrapping* bump occurs, setting it low again. When set to *Trig Mode,* there will simply be a trigger fired *each time* a wrap-around occurs. Both modes have their uses!
7. **+ Bump / - Bump Trigger Inputs.** When a trigger is received at these inputs, it will cause the four bumpers to jump up (+) or down (–) by the amount set via the Size parameter (3), according to their own peculiarities discussed at (10). They are cumulative, so If both these inputs receive a trigger simultaneously there will be no change. The same goes for (8).
8. **+ Bump•N / - Bump•N Trigger Inputs.** When a trigger is received at these inputs, it will cause the four bumpers to jump up (+) or down (–) by *the amount found by multiplying the Size parameter (3) by the N parameter (4).* So, if *Size* was set to "1 volt" and *N* was set to "3", then a trigger here would cause a 3v jump.
9. **Set Trigger Input.** Causes all four bumpers to "reset" - but the specific behavior of this trigger input depends on the position of sliders (1) and (2). *If the left slider is BELOW the right one,* then the four bumpers will be reset to the midpoint between the two boundaries. *If the left slider is ABOVE the right one,* then the four bumpers will be reset to the lower boundary instead. Each behavior has its uses!
10. **Bumper CV Outputs.** Outputs the respective CV of each of the four bumpers, bumping up and down and wrapping when moving past it's bounds. Each of the four bumpers has a unique behavior:
    - ♥ **:** Behaves *exactly as you'd expect,* acting based on whatever Bump Size and Bounds you've set up.
    - ♥ **•N :** Behaves like ♥ , *except that all bumps will go N TIMES as far!* So if N is set to "8" and ♥ is jumping up 2 V, ♥ ***•N*** *will travel 16 volts instead* (still wrapping around the bounds you've set).
    - ♥ **/ N :** Behaves like♥ **•N**, except instead of going N times as far as ♥, every voltage jump is *DIVIDED* by N. So, if N is set to "8" and ♥ is jumping up 2 V, ♥ **/ N** will only travel ¼ a volt!
    - **Boundless ♥ :** Behaves like ♥, except *this bumper doesn't care what Bounds you've set - it'll just use the highest and lowest voltage available* (set via right-click menu, see Appendix).
11. **Bumper Wrap Outputs.** Outputs a gate or trig (6) whenever it's corresponding bumper wraps around.
12. **Jump Triggers Output.** Last but not least, this output simply passes a trigger every time a bump occurs.

Tips, Tricks, and Patch Ideas

▶▶▶ If you're still having a hard time *wrapping* your head around Bumper, *try this out:* Patch a square LFO into one of the bump inputs (7), and then send the ♥ CV output to control an oscillator's pitch. The result should look like a *staircase/sawtooth* on a scope, and *by playing with the Size parameter (3) you can change the size of each stair-step.* By changing the Bound sliders (1 and 2), *you constrain how far the staircase goes before starting over.* Notice that each bumper's wrap output (11) fires whenever a wraparound occurs. Switch the LFO to *the opposite bump input* and you should see that *the staircase switches directions. Explore the other inputs and outputs to see how they behave with this patch.*

▶▶▶ To make the above patch less *"mario-running-bowser's-infinite-staircase",* try putting a *quantizer* before the oscillator's V/oct input - *instant generative melodies!* Send different rhythms to each of the pink inputs, and *you'll get unique shifting melodies from each of the CV outputs (10), and sub-rhythms from the Wrap trigger outputs (11).*

▶▶▶ Also, try using another sequencer to control the Size parameter (3) for some really pretty generative results!



▶▶▶ Bumper can be a really cool generative rhythm generator! To start, patch some different rhythms or clocks to the pink bump inputs, and *patch each of the Wrap trigger outputs (11) to trigger a different drum / voice!* You'll find each bumper will output *different but related rhythms* - Size (3) and N (4) will act as *"density"* controls, and the bounds will impact the *rhythmic relationship* between the first 3 bumpers and **Boundless ♥.** To make the rhythms repeat, patch a divided clock into the Set Trigger input (9), and *this will cause the rhythms to "reset" every time a trigger is received.*

▶▶▶ Bumper can also act as a *unique lo-fi digital oscillator/noise source,* summoning demonic atari ghosts from years gone by. To summon these demons, simply patch an audio-rate oscillator into one of the pink bump inputs, and listen to one of the CV outputs - play around with the various controls - *the Slew (5) works as a unique flavor of low pass filter* - and experiment with *sending different oscillators to the different bump inputs* for more chaotic results. You can also get PWM square oscillations from the Wrap outputs (11) when in gate mode (6). The results of this patch can be heavily aliased; *one can think of the oscillator triggering bumper as the "sample-rate", so higher pitch oscillator inputs will give cleaner results!* Bumper is capable of generating both harmonics and subharmonics ^_^

▶▶▶ The rhythm and trigger sequences generated by Bumper can often meander endlessly, so *if you want your sequences to be periodic (repetitive), make sure to send an occasional reset trigger to the Set Trigger input (9)!*

▶▶▶ Each of the four bumper units have *their own character and uses* - generally **♥ / N** will be *the most subtle/slow* of the bunch, and **♥ •N** will be the most *hyperactive*. **♥** and **Boundless ♥** behave *identically* except for **Boundless ♥** ignoring the Boundaries you've set, *so it will often act as a nice "counterpoint" to* **♥**, *especially in melodies!*

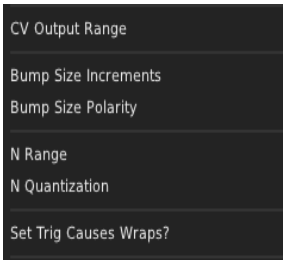
▶▶▶ As mentioned in the Overview section, the slew circuit (5) can act as a track-and-hold for the CV outputs - *experiment with sending a gate signal into the Slew CV input* to cause the CV sequence to freeze at dramatic points!

▶▶▶ Like all of my modules, *Bumper loves to be self-patched!* Any of the bumper outputs *can be fed back to the CV inputs* for all sorts of interesting behaviors - experiment freely, you won't break anything!

▶▶▶ To get even more interesting sequences, try sending the Wrap gate outputs (11) of one Bumper module into another one! *In this way endless Bumper modules can be chained together + fed back for increasingly complex behaviors!*

▶▶▶ Bumper's many outputs can be used for all sorts of things - *try using them to modulate filters, trigger envelopes, accent certain beats, or open and close VCAs!*

Appendix: The Right-Click Menu



Bumper is *highly configurable* - on the physical module, these options might be changed via dip switches, but in VCVrack you can access them by simply right-clicking the module! Remember that, once you have Bumper configured how you like, you can save this configuration as a Preset or Template!

The options are as follows:

- **CV Output Range:** Configures the voltage range of Bumper's CV outputs - this is the range of voltages the Bound controls cover. *By default this is set to Bipolar, from -5v to +5v.*
- **Bump Size Increments:** Allows the bump size parameter to be quantized - for example, if this is set to "8" then the Size will be quantized to ⅛ increments. *Unquantized (w/ Exponential scaling) by default.*
- **Bump Size Polarity:** Configures whether bump size will span both positive and negative sizes (with the center detent being a Size of 0), or just the positive sizes (with 0 at the minimum position). If bump size is negative, all jumps will go backwards (e.g. + bump will make voltages go down!) *Default is positive-only.*
- **N Range:** Controls the range of possible N values, from 1-8 to 1-1024. *Defaults is spanning #s 1-8.*
- **N Quantization:** Chooses whether N values can be decimals, or only round numbers. *Default is quantized.*
- **Set Trig Causes Wraps?:** A weird nuance, but determines whether a trigger signal received at the "Set Trigger" input will cause all 4 bumper's Wrap outputs to fire. *Default is true.*

Appendix: Use Cases

I like to design open-ended modules which can be used for many different kinds of tasks.
Why would you ever choose to use Bumper?
Here is an non exhaustive list of things Bumper can be used for:

- Generative sequencing tool for modulation, rhythms, or melodies (optionally with a quantizer)
- A highly flexible clock-divider / clock-modulator / drum sequencer
- Lo-fi digital noise / ramp / square oscillation generator with built-in filter and PWM
- Flexible Logic module
- Ultra-gnarly digital audio processor (try sending audio signals into the pink bump inputs!)
- Chaos generator (especially with feedback patching and random gate inputs)
- Function generator for opening/closing VCA/LPGs

Appendix: Design Inspirations + Special Thanks

- My major inspirations for this module were found in the field of computer science - many thanks to the ingenious computer scientists who made modern tech a reality. We stand on the shoulders of giants!
- Bastl Instruments Popcorn
- My testers showed me a module called Clep Diaz by Noise Engineering with a more minimal take on a similar concept - check it out! :)
- Meng Qi, a huge inspiration for me - 非常感谢!
- The VCVrack team and community, with a special thanks to Andrew for his dedicated work and passion
- My lovely beta testers and supportive synthfriends (especially @retoid for all his help with graphic design!)
- You for reading this far and for supporting my dream ^^

<3 Allie // alliewayaudio@gmail.com // https://github.com/AlliewayAudio