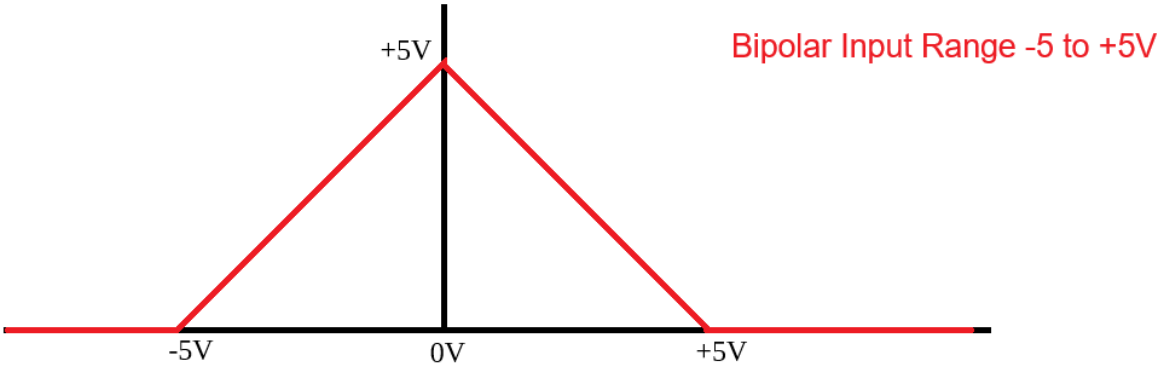
Pyramid Peak from Dome Music Technologies

Introduction

Pyramid Peak is a voltage processor which produces a **unipolar** triangular-peak **output**. It rises from 0V to 5V over the first half of its input range, then falls from 5V back down to 0V over the second half of its input range.

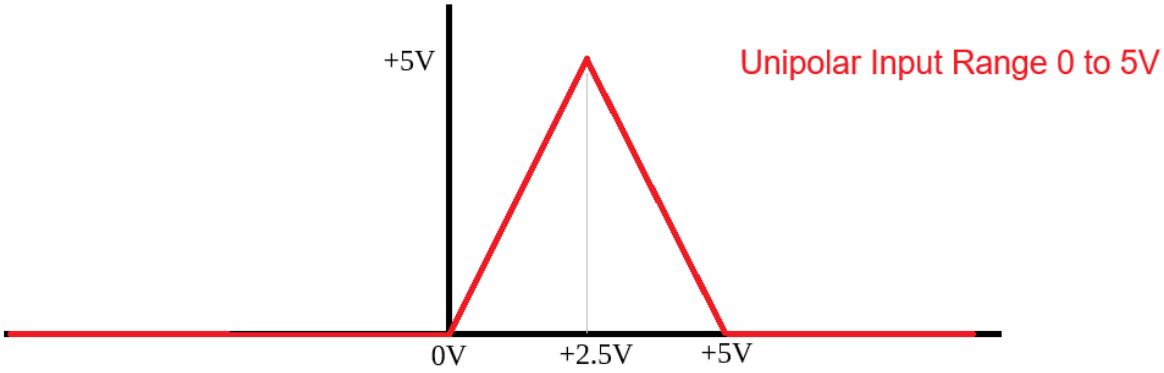
There are two switch-selectable **input ranges**: **Bipolar** (switch to the left position) and **Unipolar** (switch to the right position)

In the Bipolar position, the transfer function looks like this (x axis is input voltage, red trace indicates output voltage on the y axis):



If you use this with the pitch-bend wheel, it will output 0V at the extremes of up and down shift, and 5V in the central position.

In the Unipolar position, the transfer function looks like this (x axis is input voltage, red trace indicates output voltage on the y axis):



If you use this with the modulation wheel, it will output 0V at the lower and upper positions, and 5V in the mid-way position.

If you set the input range to Unipolar, then feed it with a bipolar signal at half its amplitude (+/- 2.5V), you get a half-wave rectifier response:

Although the module was designed primarily to process control voltages, it can also process audio-rate signals. One application of this is to use the Unipolar input range to create a rudimentary wave-folder. This allows you to experiment with simple West-Coast additive synthesis ideas. Each stage only provides one ‘reflection’ of the waveform. However, multiple stages can be cascaded to provide multiple reflections, leading to more rich and complex harmonic content.

The Genesis of Pyramid Peak

I was emailed by long-time Voltage Modular (and Dome Music Technologies) user Jean d'Oran. He wanted to create the two transfer functions of Pyramid Peak using existing VM modules.

The first solution used the "Formula" module:

Unfortunately, "Formula" is quite CPU intensive in operation, so it wasn't practical to have multiple instances in one patch.

The next solution was to use a combination of ACE Comparator and ACE Constants & Multipliers to implement the same formulas.

Although this is a more CPU-friendly solution, it does take up an awful lot of panel space. On top of that, you would have to do a lot of convoluted patching to get your final result.

So, after a bit of discussion with Jean, I decided to implement the functionality in a single, easy-to-use 5HP module, and make it available for anyone to use.