# Biset

## REGEX

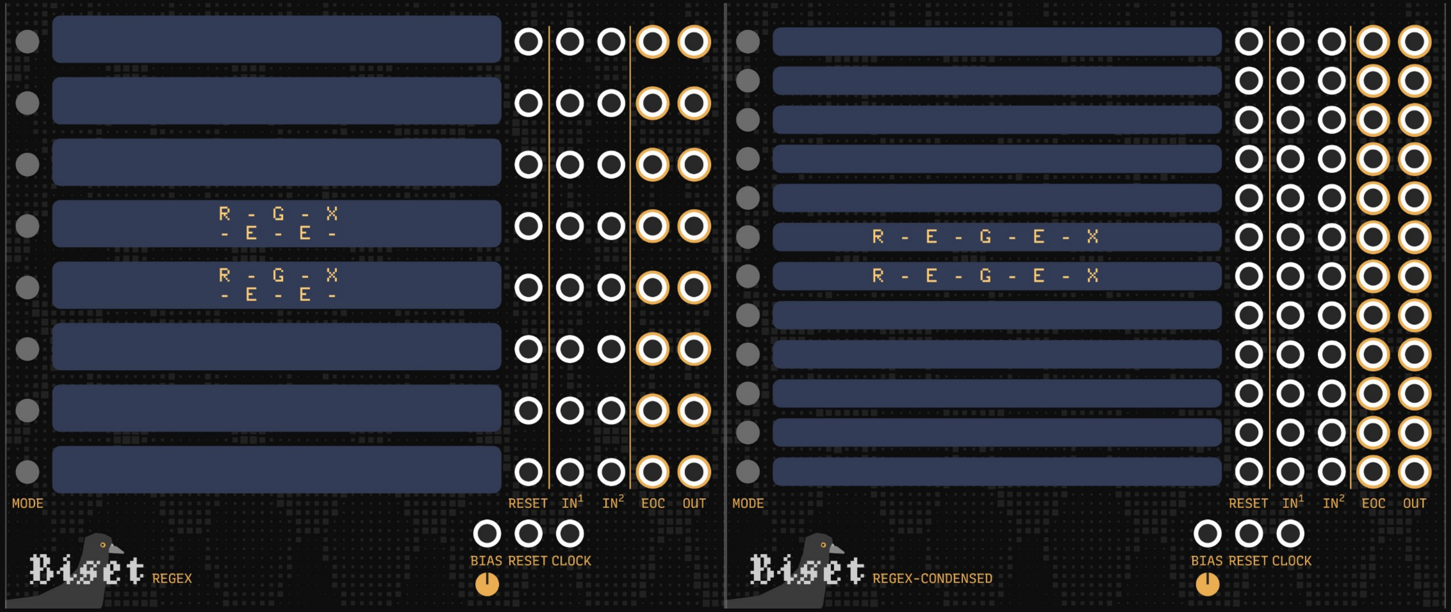
VCV Rack manual

# Biset REGEX

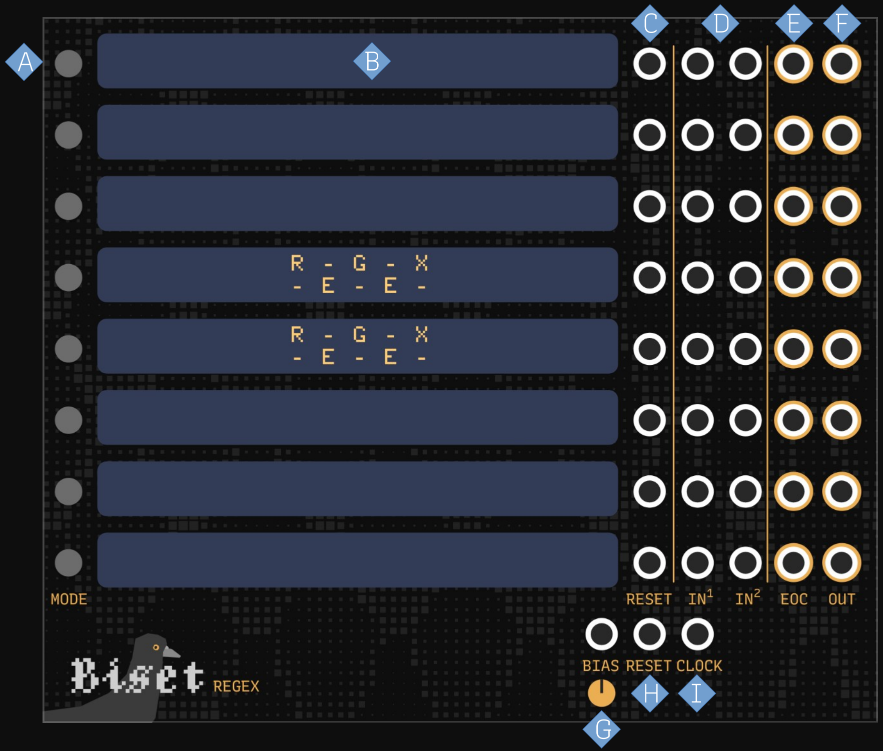
**Regex** is a **text based pattern sequencer**. It can generate both **clock** (rythm) and **pitch** (or modulation) sequences.

It has been thought for **performance** as a **live coding device**. You can easily build **sequences**, **rythm** and/or **pitch/cv** and **connect** them.

**Regex-Condensed** is simply the condensed version of **Regex** with less readability but **more lines** and **connection options**.



# Biset REGEX



| | |
|---|---|
| **A** | Mode switch |
| **B** | Expression editor |
| **C** | Individual reset input |
| **D** | Individual inputs |
| **E** | Individual EOC output |
| **F** | Individual output |
| **G** | Bias knob and input |
| **H** | Global reset input |
| **I** | Global clock input |

**Regex** is made of 8 or 12 lines. A line is made of an **expression editor** where all the code is written, a **switch** allowing you to change the **expression type** (clock or pitch), **inputs** and **outputs**.
**Inputs** and **outputs** behavior depends on the expression type.

## Biset REGEX / clock mode

The **clock** mode generates rythm. It uses the **master clock input** or the individual **input 1** as **main clock** and can use the expression **Input 2** as **additional clock** for more complex rythms.

The sequence values acts as **clock dividers** on the input clock. The **reset** input **resets** and **restarts** the flow of the expression. Everytime the expression **loops**, the **EOC** outputs a trigger.

The **main output** outputs triggers according to the expression.

Here are a few clock valid expressions

```
>1,2                  ••-|
<1,2                  •-•|
^1,2                  ••-•-•|
>1,2,3                ••-•--|
>1,2,3%8              ••-•--|••|
>1,2,3%16             ••-•--|••-•--|•••-•|
>(1,2,((3,4))%1)%16   ••-•--|••-•---|••-|
```

Biset REGEX / pitch mode

The **pitch mode** generates **pitch** on **each trigger** received via the **master clock input** or the individual **input 1**.

The expression **input 2** allows you to **combine pitch sequences**. It acts as an **offset** to the output pitch. The **reset** input **resets** and **restarts** the flow of the expression. Everytime the expression **loops**, the **EOC** outputs a trigger.

The **main output** outputs pitch (cv) according to the expression.

Here are a few pitch valid expressions

| | |
|---|---|
| **>c,b,d,e,f,g,a,b** | *Outputs **major scale*** |
| **>(c,c,d#,?(f,g)%1)** | *Outputs **c,c,d#,f** or **c,c,d#,g*** |
| **?(0,0,0,12,12,24)** | *Outputs **c4** or **c5** or **c6**, useful to offset another expression by an octave* |
| ***c,d#,>((f,g,g#))%1*** | *Outputs **c,d#,f** then **c,d#,g** then **c,d#,g#*** |

Biset REGEX / expression anatomy

An expression is a **string** defining a **pattern** generating **clock** (rythm) or **pitch** (or CV).

It is based on **sequences** that are made of :

**Type** *The sequence type (foreward, backward, pingpong, etc.), optional (foreward by default)*
**Values** *A series of values (number, pitch, another sequence) that can be enclosed in brackets for clarity*
**Modulator** *Defining the sequence behavior through time, optional*

To run an expression, you should press **Enter** while focusing the corresponding **display**. You can also press **Ctrl + Enter** to run all expressions. If an expression is already running, it will waits for its end to update to the new expression. An expression can be stopped by pressing **Escape**. You can use **Ctrl + Arrow** to jump between expressions.

Biset REGEX / expression anatomy / sequence type

By default, a **sequence** is set to the **foreward** type.

Here are the available sequence types

|  |  |
|---|---|
| **>** | *Foreward, read the sequence foreward* |
| **<** | *Backward, read the sequence backward* |
| **^** | *Ping-pong, read the sequence foreward and then backward* |
| **@** | *Shuffle, shuffle the sequence before reading it* |
| **?** | *Random, pick a random item of the sequence* |
| **!** | *X-Random, pick a random item of the sequence avoiding the last picked element* |
| **$** | *Walk, random walk in the sequence* |

## Biset REGEX / expression anatomy / sequence values

Expressions **values** can be **numbers** or **pitch** (with optionnal octave).
You can also use other sequences as values, allowing you to build complex patterns.

In **pitch mode**, values follow the **v/oct** rule, even for numerical values.
**0** returns **0 volt** while **12** returns **1 volt**.
**c4** returns **0 volt** while **c3** returns **-1 volt**.

In **clock mode**, pitch notation are accepted (though they do not really make sense).
Values less than or equal to 0 will be considered as a rest (wait for 1 triggers in input without outputting any trigger).

Here are a few valid values

| | |
|---|---|
| **C** | c4 |
| **c** | c4 |
| **c#** | c4 sharp |
| **cb** | c4 flat |
| **c5** | c5 |
| **3** | b4 or +3 |
| **16** | e5 or +16 |
| **-5** | g3 or -5 |

# Biset REGEX / expression anatomy / sequence modulator

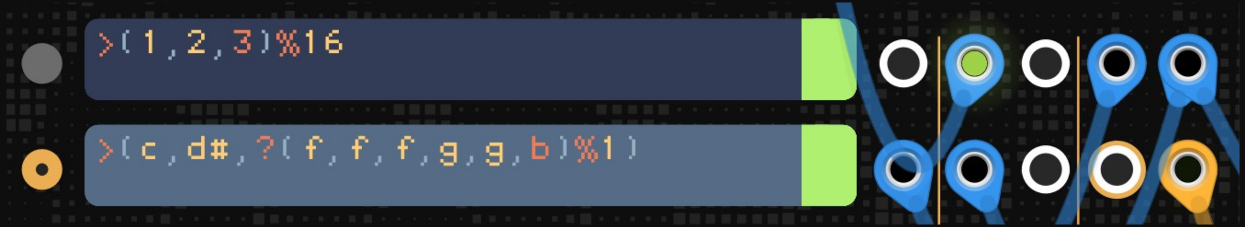
**Modulator** define the sequence **behavior through time.**
When no modulator is used, the sequence is read once (expressions are always looped).

- **xN** - Read the sequence **N times**
- **%N**
  - **Clock mode** - Loops the sequence until **N clock triggers** have been reached on the input, allowing you to easily build "regular" rythms.
  - **Pitch mode** - Loops the sequence until **N values** have been output.
- ***N**
  - **Clock mode** - Advance (and potentially loops) the sequence for **N steps.**
  - **Pitch mode** - Acts like **%N** modulator.

**Inline multiplication** - The **xN** modulator can also be used with single values to multiply them in a sequence (ex: **?(0x3,1x5)** will be translated **?(0,0,0,1,1,1,1,1)**).
Single values cannot be multipled more than 64 times

A **bias** knob and input are available on the bottom of the module (G). **Bias** is a variable that impacts **random sequences** (?, **!** and **$**). Moving bias knob **left** or **right** makes these sequences tend to choose random values on the corresponding **directions**. This can be used to add spice or, at the oposite, regularity, to your sequences by connecting it to an **LFO**. As an example, I like to have a random pitch sequence with notes sorted in pitch. By moving the bias knob on the left, the sequence will tend to play lower notes while moving it the right will tend to play higher notes.

Biset REGEX / examples



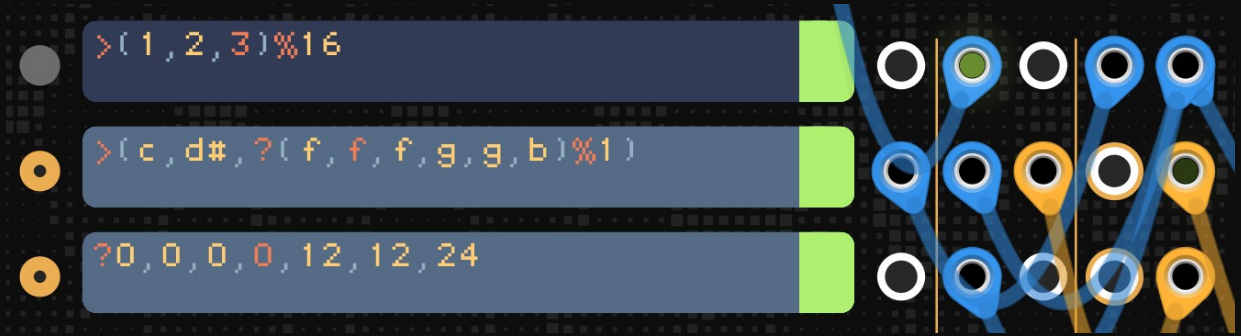
```
>(1,2,3)%16
>(c,d#,?(f,f,f,g,g,b)%1)
```

Here, the **1st** expression is a **clock expression** while the **2nd** is a **pitch sequence.** A clock generator is connected to the **individual input 1** of the clock expression to make it run. Its output is connected to the **input 1** of the pitch expression, thus, connecting them. Its **EOC output** is also connected to the **reset input** of the pitch expression, thus, making them reset at the same time and **loop together.**

Clock cables are blue
Pitch cables are yellow
Modulation cables are green

Biset REGEX / examples

## Bias

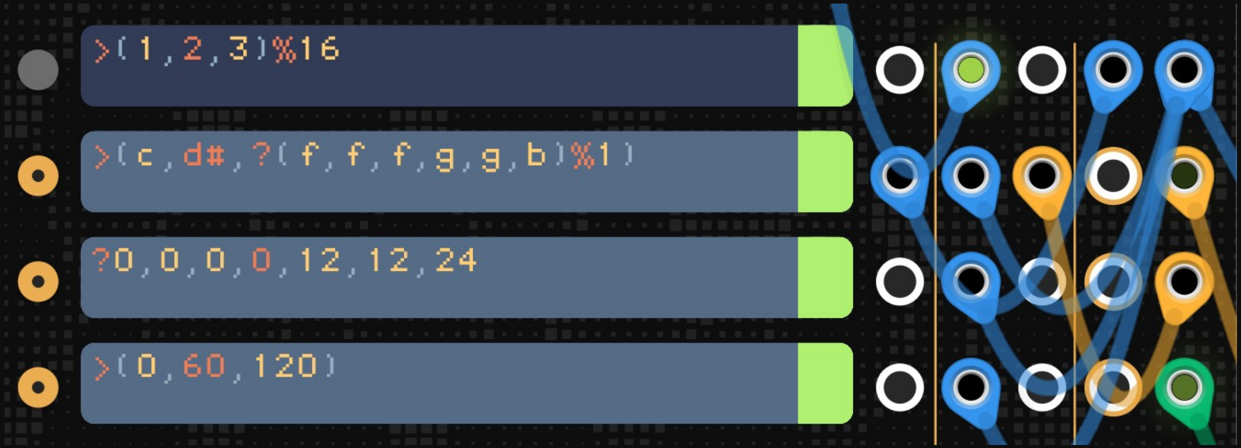


```
>(1,2,3)%16
>(c,d#,?(f,f,f,g,g,b)%1)
?0,0,0,0,12,12,24
```

Here, the first two expressions are the same. We introduce a 2nd pitch expression to make the sequence more interesting. This expression is also **driven** by the clock expression but its **output** is connected to the 1st pitch expression **input 2**, thus, **offseting its pitch value**. As this expression **outputs random octaves** (+0 semitone, +12 semitones or +24 semitones), it will offset the 1st pitch expression to build a pitch sequence that varies more.

# Biset REGEX / examples

## Bias



```
>(1,2,3)%16
>(c,d#,?(f,f,f,g,g,b)%1)
?0,0,0,0,12,12,24
>(0,60,120)
```

This time, we introduce a 3rd "pitch" expression also **driven** by the clock expression. But, as we can see, this new "pitch" sequence outputs a forward sequence of really high values. It is here used as a modulation sequence to add spice to the melody. As said above, pitch is in **Volt Per Octave**, meaning that a value of **12** will result in a CV of **1 volt** and while value of **120** a CV of **10 volt**, this can then really easily be used to produce any modulation sequences !

Biset REGEX / tricks

**Nice arpegios** - It’s important to notice that a sequence restarts only when it reached its own end and not when its parent its end. You can play with that trick to create nice pitch sequences.

Ex: **>(c,c#,>(>(e,f,g,g#))%1)** - Outputs c,c#,e, | c,c#,f, | c,c#g, | c,c#,g#

Here, the **>(>(e,f,g,g#))%1** sequence is a **foreward sequence of length 1** containg an other **foreward sequence of length 4.** If the **%1 modulator** will stop the parent sequence every 1 item pulled, the children sequence will only be paused but not reset and will continue, step by step, every time the whole sequence loops !

- - -

**Nice rhythm** - You can use the ***N** modulator to create changing rhythms.

Ex: **>(1,2,?(3,(1x3))*1)** - Outputs ••-•-- or ••-•••
Ex: **>(1,2,?(3,2)*1)%8** - Outputs ••-•--|•• or ••-•-|••- (even funnier with **%16**)

- - -

**Drums !** - You can use **inline multiplication** to create nice drum patterns with a shuffle sequence.

Ex: **@(0x3,1x5)x8** - Outputs a random drum sequence made of hits and rests that will loop 8 times before reseting and changing to another random drum sequence !