Contents

RhythML Syntax Specification

RhythML is a plaintext format designed to sequence pitch and CV patterns in a eurorack-style environment. The format is agnostic to what you are using each output for, which allows for specifying both rhythmic triggers and melodic sequences in a compact, easily editable, human readable text format.

Right now, RhythML is only implemented in one place: the 'Spellbook' module for VCV Rack. However, the format is meant to be platform/technology agnostic. In the future, look out for plugins and standalone tools to convert RhythML to and from other common musical notations, or use it in common apps and music writing tools.

Below is a detailed specification of the RhythML syntax as implemented by the `Spellbook` VCV Rack module. Because Spellbook is a VCV Rack module, pitches and percentages are translated into 1v/octave. In other systems, RhythML should be translated into appropriate signals for that system. For example, if there were a RhythML module as a VST plugin for your DAW, it would most likely output everything as MIDI instead of voltages.

Basic Structure

RhythML is structured like a comma separated spreadsheet, or "CSV":

- **Rows**: Each row in the text corresponds to one sequence step.
  - You can have as many or as few rows in a sequence as you like.
  - Sequences are usually played sequentially, for example by stepping to the next line on each clock pulse.
  - Unlike MIDI or sheet music, the steps don't know or tell you how "long" they are; they are stochastic steps in a sequence.
- **Columns**: Individual values within a row are separated by commas `,` to create columns.
  - Each column corresponds to one output signal.
  - You can have as many or as few columns in a sequence as you like.

Why plaintext?

RhythML (short for "Rhythm Markup Language") is a sort of plaintext take on music notation. Very loosely inspired by trackers, Markdown, and experience with other various plaintext formats.

RhythML is NOT an attempt at adapting traditional sheet music into plaintext, nor is it a musical scripting language like ABC Notation or Musical Markup Language. There are no sections or repeats or loops, and there is no concept of "timing" (only "steps"), so unlike ABC or MML, it's more like a modular oriented tablature.

Unlike MIDI or a Tracker, you're not specifiying "note events" with a pitch, duration, etc., but rather you should think in a modular synth way: each cell is parsed and converted into an

output value (such as a voltage), which can be used in your patches however you like. For most use cases you'll need separate columns/voltages for each parameter you want to control: pitch, gate/trigger/envelope, velocity, filter cutoff, pulse width, etc.

Steps don't have an intrinsic "duration"; each step is just the next step in the sequence. With a complex patch you could even do weird things like play sequences backwards, or with a dynamically changing clock speed (you could even control the speed of the clock triggering a RhythML sequence using one of the columns in that very sequence- remember, the secret of modular is that control voltages can do basically *anything*, at many different levels of abstraction in your music!), or other such modular tricks.

The idea is to be able to quickly and easily type out or edit a sequence in a modular synth (especially VCV Rack) context. Anything you could send a voltage to, you can control with a RhythML sequence. Every music notation system has its own strengths, weaknesses, and intentions (sheet music compresses time to make printing and sightreading easier, for example), and this is not meant to replace any of them.

```
0 ? Decimal                                          , T ? Trigger
1.0 ? text after ? is ignored (for comments)!        , X ? Gate with retrigger
-1 ? row 1 comments become output labels             , W ? Full width gate
1 ? (sorry no row 0 / header row... yet!)           , | ? alternate full width gate
                                                     , |
? Empty cells retain the prior/current output...   , ? ...except after gates/trigger symbols, for co
                                                     ,
C4 ? It also parses note names to 1v/oct...          , X
C ? (octave 4 is the default if left out)            , X
m60 ? ...or MIDI note numbers like `m60`...          , X
s7 ? ...or semitones from C4 like `s7`.              , X
10% ? ...or percentages! (useful for velocity)       , X
? Important !!! This is not a Tracker !!!           ,
C4 ? Pitches do NOT automatically create triggers..., ? ...you need a trigger column
                                                  , X ? or triggers from somewhere else
? Or use columns for ANY CV                          , | ? Think modular!
```

The syntax lets you think and write in note names, or voltages, or percentages, or gates and triggers, or whatever other format makes sense to you based on what you're going to use that resulting output voltage for in your patch- which in modular could be *anything*.

Because it's just plain text, you can: - copy and paste it - type it freehand - share or save sequences anywhere you can use text, like chat or email - edit RhythML sequences in any text editor

Because it's simple and human readable it's even easy to adapt RhythML sequences for new contexts or new patches just by changing what you use the outputs for, or how you time the sequence. The goal is to be able to easily write and share useful "text snippets" of musical ideas, patterns, and sequences.

Sequencing using RhythML

Each cell in a grid can contain values in one of these formats. Every cell is parsed independently, so you can freely mix and match formats to express intentions and uses in the music. I usually choose a format for each coulmn, based on whatever that column's intended use in the music is- note names for pitch CVs, percentages of cutoff CVs, semitones for pitch bend CVs, etc. Just by choosing formatting deliberately, a RhythML sequence can convey a lot of intention and meaning even without comments.

1. **Decimal Voltage Levels**:

- Example: `5`, `-3.5`, `0.0`, `12`
- Specifies a voltage directly.
- *Anything that parses to a floating point value is allowed, but note Eurorack/VCV Rack convention is to stay within -10 to +10 volts, with a 10 volt range between min and max value of a given signal, and some modules may behave strangely or not accept voltages outside their expectations.*

2. **Gate and Trigger Commands**:
   - `X` or `R` or `_`: Outputs 0 volts for the first 1ms of the step, then a high signal (10 volts) for the remainder.
     - Guarantees a rising edge, regardless of the prior step, and holds a high signal afterward.
   - `T` or `^`: Outputs 0v for 1ms, then 10v for 1ms, then 0v thereafter.
     - Guarantees a rising edge, regardless of the prior step, and holds a low signal afterward.
   - `W` or `|`: Outputs a full width gate signal (10 volts), equivalent to writing `10` in the cell.
     - Use this to continue a gate across steps.
     - If the output is already high from a prior retrigger or gate, this will NOT create a new rising edge in the signal.
     - *Gates (as with all signals except for Triggers and Retriggers) are 100% width; two consecutive gates will output a continuous signal with no break or edge.*
3. **Scientific Pitch Names**:
   - Format: `<NoteName>[Accidental(s)][Octave]`
   - Example: `C4`, `G#3`, `Db5`
   - Accidentals can be `#` for sharp, `b` for flat, `$` for half-sharp, and `d` for half-flat. Compound accidentals like `C##` (C +2 semitones) or `C#$` (C +1.5 semitones) are allowed, and accidentals can be on any note, so `E#` is the same note as `F` and `Cb` is the same as `B`.
   - *If no octave is included, but it is still valid note name, a default octave of 4 is used. For example "C" will be read as "C4".*
   - *Outputs a voltage corresponding to the specified musical note, in Eurorack 1V/oct standard, where C4 = 0V, C#4 = 1/12V, ..., B4 = 11/12V, C5 = 1V, C3 = -1V, etc.*
4. **MIDI Note Numbers**
   - Format: `m<Number>`
   - Example: `m60`, `m72`
   - MIDI Note 60 = C4 = 0.0v.
   - *Decimals and numbers outside the normal MIDI range of 0 to 127 are allowed, but are usually not going to be respected if you try to send those pitches back into a MIDI environment.*
5. **Semitones**
   - Format: `s<Number>`
   - Example: `s0`, `s7`, `s-12`
   - *Decimals are allowed*
   - *Semitones are numbered relative to C4, which means they are basically the same as MIDI Notes except 0 = C4*
6. **Cents**
   - Format: `<Number>ct`
   - Example: `0ct`, `7ct`, `-12ct`
   - *Decimals are allowed*
   - *Cents relative to C4*
7. **Hertz**
   - Format: `<Number>Hz`
   - Example: `440Hz`, `32Hz`, `1Hz`
   - *Decimals are allowed*
   - *0 or less is undefined, and will output 0 volts*

8. **Percentages**
    - Format: `<Number>%`
    - Example: `50%`, `100%`, `-12.5%`
    - Numbers ending in % are divided by 10, so that 100% becomes 10.0v, -50% becomes -5.0v, etc.
    - *This scaling is specific to eurorack. If you are implementing RhythML in another environment, percentages should be translated according to the standards of that environment. For example, if you were parsing RhythML into MIDI, 0%-100% might become 0-127.*
9. **Empty Cells**:
    - An empty cell or a cell containing only whitespace or comments will leave the output's current voltage as-is, unless that "current voltage" was from a gate, trigger, or retrigger, in which case it reverts to 0.0.

Note that unlike MIDI or a Tracker, a single cell with a pitch in it does NOT automatically generate an associated gate or "note on", it just outputs one voltage for each column, like any typical eurorack CV sequencer. You would need to sequence the rhythm you want using another column, or handle rhythm with another module such as a clock or another sequencer (potentially even another Spellbook module!).

Remember, any column can be for used for anything you need a voltage for: pitches, gates, velocity, CVs, etc. Think modular!

Comments

- Any text following a `?` in a cell is considered a comment (until the next `,`) and is ignored during parsing. Comments cannot contain commas.

Example:

```
5.0 ? This is a comment, 5.5 ? You can put a comment in any cell
```

Consider using comments as labels:

```
E4 ? Pitch, X ? Gate, 10 ? Velocity
C5        , X       , 8
D5        , X       , 6
B4        , X       , 5
```

Or to explain musical intentions:

```
70% ? Melody volume, 50% ? Backing volume, 0% ? Drum chance,  0% ? Reverb, 35% ? Filter, ? NOTES - Soft start - no drums - slight reverb - mild filter
70%           , 50%           , 0%          ,  0%      , 35%      , ? Continue soft intro
80%           , 60%           , 30%         , 50%      , 40%      , ? Increase all volumes - introduce drums sporadically
```

Whitespace

- Whitespace in cell values is ignored.
- Cells are normalized during editing such that each cell in a column has uniform spacing padded with spaces to align columns vertically for readability.
    - Blank lines are NOT ignored, and will become new blank rows, with commas added automatically.

Usage Example

Here's a simple RhythML sequence to get started:

```
E4 ? Pitch, X ? Gate, 10 ? Velocity
C5        , X       , 8
D5        , X       , 6
B4        , X       , 5
```

In this example, each step (row) sets a pitch in column 1, uses column 2 for triggers, and controls a velocity CV in column 3. The sequence is isntructions that play a C major arpeggio, one note per step, and gradually lowering the velocity on each note. The labels, like `? Pitch`, are ignored because of the `?`.

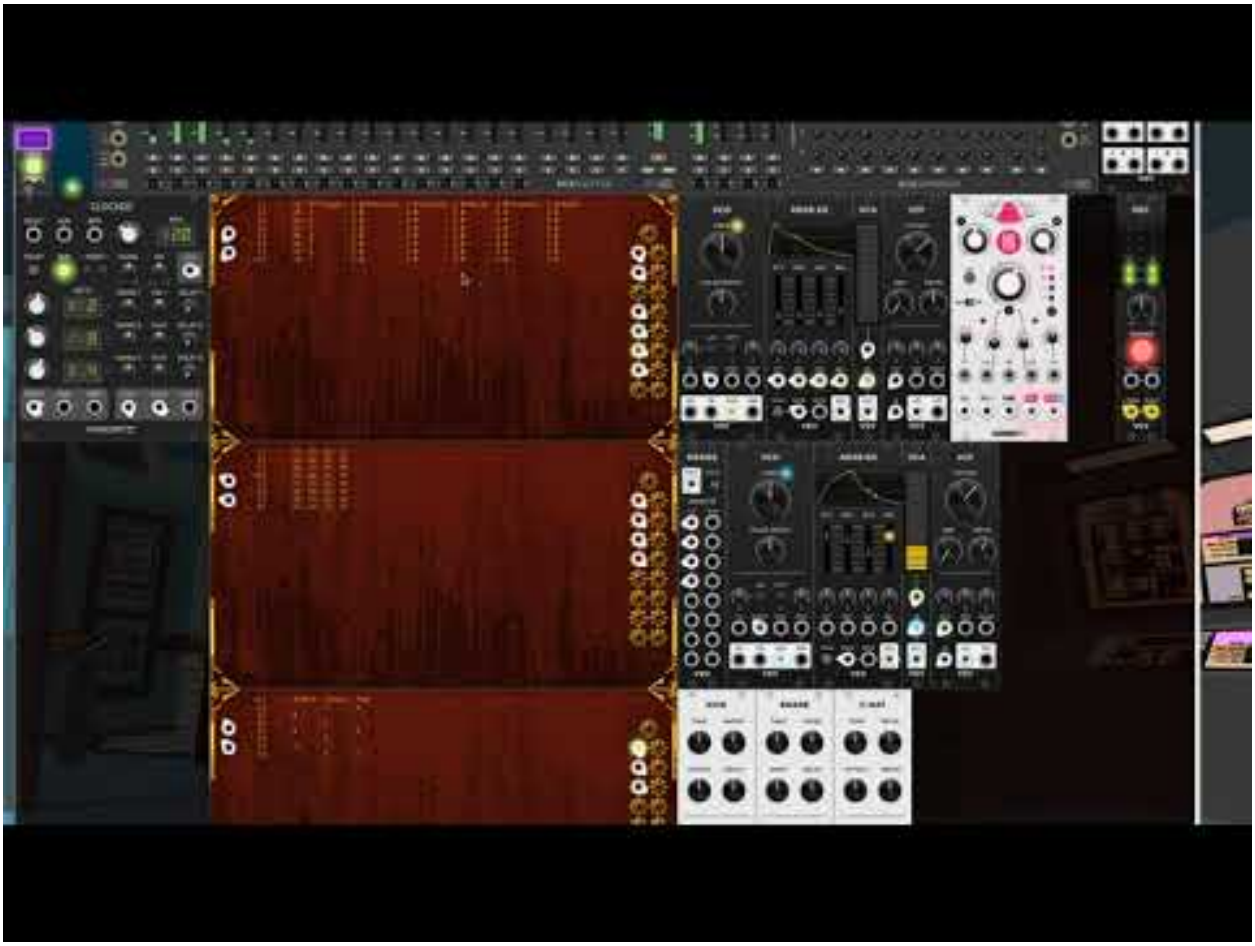
Here's the same arpeggio, but with a little more rhythmic variation, as an 8 step sequence instead of 4:

```
E4 ? Pitch, X ? Gate, 10 ? Velocity
          , |       ,
          , |       ,
C5        , X       , 8
D5        , X       , 6
          ,         ,
B4        , X       , 5
          ,         ,
```

Notice the way this pattern holds the first note for multiple steps by using `|` gates.

In general, don't forget that in modular contexts you need to explicitly and separately output every different voltage/signal you need, like gates, velocities, CVs; everything!

Watch a brief demonstration here:



Other Examples

These are all available as manufacturer presets in Spellbook. Please notice: labels are simply comments in the first row of the sequence, not a separate “header” row.

Generic 8 step, 3 CV, 1 rhythm sequencer:

```
0 ?A, 0 ?B, 0 ?C, X ?T, T ?Start, ?End
0   , 0   , 0   , X   ,         ,
0   , 0   , 0   , X   ,         ,
0   , 0   , 0   , X   ,         ,
0   , 0   , 0   , X   ,         ,
0   , 0   , 0   , X   ,         ,
0   , 0   , 0   , X   ,         ,
0   , 0   , 0   , X   ,         , T
```

This pattern recreates the default state of the core VCV Rack module “Seq3”: three CV channels, a gate channel, and two triggers that fire on the first and last step respectively.

2-5-1 chord progression with timing columns:

```
D3 ? Root, F4 ? Third, A4 ? Fifth, X ? Downbeat, ? Upbeat, T ?ChordChange
         ,           ,           ,             , X       ,
G4       , B5        , D4        , X           ,         , T
```

```
        ,           ,           ,           , X         ,
C4      , E5        , G4        , X         ,           , T
        ,           ,           ,           , X         ,
        ,           ,           , X         ,           ,
        ,           ,           ,           , X         ,
```

This you could clock at the start of each bar or phrase, showing how “steps” don’t have to mean “beats” or “notes”.

Basic Rock Drums:

```
T?Kick, ?Snare, ?Tom1, ?Tom2, ?Rim, ?Clap, T?cHat, ?oHat, ?Crash, ?Ride
      ,       ,      ,      ,     ,      ,       ,      ,       ,
      ,       ,      ,      ,     ,      , T     ,      ,       ,
      ,       ,      ,      ,     ,      ,       ,      ,       ,
      , T     ,      ,      ,     ,      , T     ,      ,       ,
      ,       ,      ,      ,     ,      ,       ,      ,       ,
      ,       ,      ,      ,     ,      , T     ,      ,       ,
      ,       ,      ,      ,     ,      ,       ,      ,       ,
T     ,       ,      ,      ,     ,      , T     ,      ,       ,
      ,       ,      ,      ,     ,      ,       ,      ,       ,
T     ,       ,      ,      ,     ,      , T     ,      ,       ,
      ,       ,      ,      ,     ,      ,       ,      ,       ,
      , T     ,      ,      ,     ,      , T     ,      ,       ,
      ,       ,      ,      ,     ,      ,       ,      ,       ,
      ,       ,      ,      ,     ,      ,       , T    ,       ,
      ,       ,      ,      ,     ,      ,       ,      ,       ,
```

This pattern has one column for each drum on the VCV Rack core module “Drums”, and shows a basic rock beat, which you would likely clock on 16th notes.

32-step single-cycle wave forms:

```
0.000 ?Sine, 0.625 ?Triangle, -5 ?Saw, 5 ?RevSaw, 5 ?Square, 5 ?ShortPulse, 5 ?LongPulse, 0 ?Noise
0.9755    , 1.2500       , -4.6774, 4.6774   , 5.0000   , 5.0000      , 5.0000     , -
5.0000
1.9135    , 1.8750       , -4.3548, 4.3548   , 5.0000   , 5.0000      , 5.0000     , -
1.0000
2.7780    , 2.5000       , -4.0323, 4.0323   , 5.0000   , 5.0000     , 5.0000    , 3.0000
3.5355    , 3.1250       , -3.7097, 3.7097   , 5.0000   , 5.0000      , 5.0000     , -
2.0000
4.1575    , 3.7500       , -3.3871, 3.3871   , 5.0000   , 5.0000     , 5.0000    , 0.0000
4.6195    , 4.3750       , -3.0645, 3.0645   , 5.0000   , 5.0000      , 5.0000     , -
2.0000
4.9040    , 5.0000       , -2.7419, 2.7419   , 5.0000   , 5.0000      , 5.0000     , -
4.0000
5.0000    , 4.3750       , -2.4194, 2.4194   , 5.0000   , -5.0000     , 5.0000     , -
5.0000
4.9040    , 3.7500       , -2.0968, 2.0968   , 5.0000   , -5.0000     , 5.0000     , -
5.0000
4.6195          ,  3.1250                 ,  -1.7742,  1.7742      ,  5.0000      ,  -
5.0000       , 5.0000       , 4.0000
4.1575    , 2.5000       , -1.4516, 1.4516   , 5.0000   , -5.0000     , 5.0000     , -
1.0000
```

```
3.5355         ,  1.8750               ,  -1.1290,  1.1290     ,  5.0000     ,  -
5.0000      , 5.0000      , 1.0000
2.7780         ,  1.2500               ,  -0.8065,  0.8065     ,  5.0000     ,  -
5.0000      , 5.0000      , 5.0000
1.9135         ,  0.6250               ,  -0.4839,  0.4839     ,  5.0000     ,  -
5.0000      , 5.0000      , 1.0000
0.9755    , 0.0000       , -0.1613, 0.1613   , 5.0000   , -5.0000     , 5.0000     , -
1.0000
0.0000    , -0.6250      , 0.1613 , -0.1613  , -5.0000  , -5.0000     , 5.0000     , -
5.0000
-0.9755        ,  -1.2500              ,  0.4839  ,  -0.4839    ,  -5.0000    ,  -
5.0000      , 5.0000      , 5.0000
-1.9135   , -1.8750      , 0.8065 , -0.8065  , -5.0000  , -5.0000     , 5.0000     , -
2.0000
-2.7780        ,  -2.5000              ,  1.1290  ,  -1.1290    ,  -5.0000    ,  -
5.0000      , 5.0000      , 0.0000
-3.5355   , -3.1250      , 1.4516 , -1.4516  , -5.0000  , -5.0000     , 5.0000     , -
1.0000
-4.1575   , -3.7500      , 1.7742 , -1.7742  , -5.0000  , -5.0000     , 5.0000     , -
2.0000
-4.6195   , -4.3750      , 2.0968 , -2.0968  , -5.0000  , -5.0000     , 5.0000     , -
2.0000
-4.9040        ,  -5.0000              ,  2.4194  ,  -2.4194    ,  -5.0000    ,  -
5.0000      , 5.0000      , 4.0000
-5.0000     , -4.3750         , 2.7419 , -2.7419   , -5.0000   , -5.0000       , -
5.0000     , -4.0000
-4.9040     , -3.7500         , 3.0645 , -3.0645   , -5.0000   , -5.0000       , -
5.0000     , 1.0000
-4.6195     , -3.1250         , 3.3871 , -3.3871   , -5.0000   , -5.0000       , -
5.0000     , 1.0000
-4.1575     , -2.5000         , 3.7097 , -3.7097   , -5.0000   , -5.0000       , -
5.0000     , 5.0000
-3.5355     , -1.8750         , 4.0323 , -4.0323   , -5.0000   , -5.0000       , -
5.0000     , -2.0000
-2.7780     , -1.2500         , 4.3548 , -4.3548   , -5.0000   , -5.0000       , -
5.0000     , 3.0000
-1.9135     , -0.6250         , 4.6774 , -4.6774   , -5.0000   , -5.0000       , -
5.0000     , -3.0000
-0.9755     , 0.0000          , 5.0000 , -5.0000   , -5.0000   , -5.0000       , -
5.0000     , 3.0000
```

This pattern could be clocked fast enough that it cycles through the entire sequence at audio frequencies like 440Hz, to get rudimentary audio directly from a plaintext RhythML sequence. Each column is a different wave shape (here normalized to -5/+5, for being treated as voltages in modular). Obviously this isn't as convenient as an actual wavetable synth, but it shows the flexibility of RhythML and the potential for mad science. Are there any CSV datasets you have which might do something interesting if you treat them as RhythML?