

Contents

RhythML Syntax Specification	1
Basic Structure	1
Why plaintext?	2
Sequencing using RhythML	2
Comments	4
Whitespace	4
Timing	5
Usage Example	5
Other Examples	6
Appendix	8
Acknowledgements	8
Other Notations	9
Traditional Music Notation	9
ABC Notation	9
MML (Music Macro Language)	9
MIDI (Musical Instrument Digital Interface)	9
Tracker Software	10
Glossary of Terms	10

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, we would love to create or see others create 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, mostly as implemented by the Spellbook VCV Rack module. Within the context of Spellbook as a VCV Rack module, pitches and percentages are translated into 1v/octave. In other contexts, RhythML would 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, with output ranges of 0-127 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 specify how “long” they are; they are just 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 specifying “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.

Conceptually, it’s really just a grid sequencer, but with text input instead of a series of knobs or buttons for each step. This means you can have sequences of any length, and any number of channels. Just use commas for more columns, and more lines for more steps.

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.

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 column, 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:

- W or |: Outputs a full width gate signal (10 volts). This is identical to writing 0 or 100% in the cell.
 - Use this to hold a gate open for the entire step.
 - 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 Retrigger) are 100% width; two consecutive gates will output a continuous signal with no break or edge.*
- T or ^: Trigger pulse. 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.
- X or R or _: This is a retrigger, and is often a good default, with caveats. This outputs 0 volts for the first 1ms of the step, then a high signal (10 volts) for the remainder.
 - This guarantees a rising edge, regardless of the prior step, then holds a high 10v gate/signal afterward.
- Gate vs. Trigger vs. Retrigger: Triggers leave the signal low for the duration of the step, so when you're mixing multiple sources, triggers "stay out of each others way" better, while if a gate is already high, more triggers won't have an effect. Sometimes you want one or the other, or retriggers give you a happy medium.

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, so you only have to enter values on steps where a change is desired
- The exception to this is rows following trigger/gate commands: if the previous value was a gate, an empty cell reverts to 0v/0%, so you don't need to manually "close" every gate.

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 in the first row as labels (every cell can have its own comment):

E4 ? Pitch, X ? Gate, 100% ? Velocity

C5 , X , 80%

D5 , X , 60%

B4 , X , 50%

Or to explain musical intentions:

70% ? Melody, 50% ? Backing, 0% ? Drum, ? NOTES

70% , 50% , 0% , ? Continue soft intro

80% , 60% , 30% , ? Increase all volumes

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.

Timing

Sequences are typically played step by step (e.g. using an clock or other trigger source), but you might move or access steps in the sequence in complex modular ways as well.

Importantly, RhythML itself has no concept of “time” or “duration”, only “steps in a sequence”. It’s up to you to decide how to clock or otherwise move through a sequence to actually “play” it, and what each “step” means in the context of the music- is it going to be clocked on 8th notes? Whole notes? Bars? None of the above because you’re doing some modular mad science?

It’s often useful to have one sequence for each musical “layer of abstraction”, so they can be sequenced and timed independantly based on the overall need of the music or the patch, and kept in sync using traditional timing methods such as a master clock and clock dividers. RhythML makes working with multiple clock speeds easy because RhythML sequences can be any arbitrary length; just add or remove rows.

You might step one sequence once per bar, for the chord progression, then you might clock a drum loop in another sequence on 16th notes.

Or you could compose an entire performance in one long sequence, hundreds or thousands of steps long; taking advantage of the ability to copy & paste to build complex meta-patterns.

Gates, Triggers, Retrigger, and Clocks Triggers, gates, retrigger, and clocks are all types of control signals used in modular synthesis, often overlapping in functionality but with distinct characteristics.

- A trigger is a short pulse, typically 1ms long, used to initiate events like starting an envelope or advancing a sequencer.
- A gate is a sustained on/off signal, often used to control the duration of a note or effect.
- A retrigger combines aspects of both: it starts with a brief low pulse (like a trigger) before going high (like a gate), ensuring a new event is initiated even if the signal was already high.
- Clocks are a series of regular trigger or gate signals used to synchronize timing across a system.

While distinct, these signals often serve similar purposes: a clock could be used as a trigger source, a gate could be used where a trigger is expected (many modules accept either), and creative patching often blurs the lines between these signal types in practice.

Usage Example

Here’s a simple RhythML sequence to get started:

```
E4 ? Pitch, X ? Gate, 100% ? Velocity
C5      , X      , 80%
D5      , X      , 60%
B4      , X      , 50%
```

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 instructions 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
      , | ? hold      ,
      , |              ,
C5      , X ? retrigger, 8
D5      , X              , 6
      , ? rest          ,
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, 0 ?SNES Noise
0.98      , 1.25      , -4.68      , -5.00
1.91      , 1.88      , -4.35      , -1.00
2.78      , 2.50      , -4.03      , 3.00
3.54      , 3.13      , -3.71      , -2.00
4.16      , 3.75      , -3.39      , 0.00
4.62      , 4.38      , -3.06      , -2.00
4.90      , 5.00      , -2.74      , -4.00
5.00      , 4.38      , -2.42      , -5.00

```

4.90	, 3.75	, -2.10	, -5.00
4.62	, 3.13	, -1.77	, 4.00
4.16	, 2.50	, -1.45	, -1.00
3.54	, 1.88	, -1.13	, 1.00
2.78	, 1.25	, -0.81	, 5.00
1.91	, 0.63	, -0.48	, 1.00
0.98	, 0.00	, -0.16	, -1.00
0.00	, -0.63	, 0.16	, -5.00
-0.98	, -1.25	, 0.48	, 5.00
-1.91	, -1.88	, 0.81	, -2.00
-2.78	, -2.50	, 1.13	, 0.00
-3.54	, -3.13	, 1.45	, -1.00
-4.16	, -3.75	, 1.77	, -2.00
-4.62	, -4.38	, 2.10	, -2.00
-4.90	, -5.00	, 2.42	, 4.00
-5.00	, -4.38	, 2.74	, -4.00
-4.90	, -3.75	, 3.06	, 1.00
-4.62	, -3.13	, 3.39	, 1.00
-4.16	, -2.50	, 3.71	, 5.00
-3.54	, -1.88	, 4.03	, -2.00
-2.78	, -1.25	, 4.35	, 3.00
-1.91	, -0.63	, 4.68	, -3.00
-0.98	, 0.00	, 5.00	, 3.00

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 above is a different wave shapes (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? I've tried using seismic data, for example. Lots of interesting lists of numbers out there! It's also quite fun to generate sequences programmatically- little Python scripts, Excel formulas, etc.

Appendix

Acknowledgements

RhythML and the Spellbook module were significantly inspired by various sources in the world of music technology and notation.

A particular source of inspiration was Tantacrul's YouTube video "Notation Must Die: The Battle For How We Read Music" (<https://www.youtube.com/watch?v=Eq3bUFgEcb4>), which critically examines traditional music notation and proposes innovative alternatives. This video's exploration of the limitations (both real and imagined) of many different notation systems, and its call for more intuitive, flexible systems strongly influenced the development of RhythML as a modern, modular-synth-oriented approach to musical sequencing and notation. Importantly, it recognizes that there is never going to be a universal standard for music, because music is used in and important to so many radically different contexts, people, and cultures. Flexibility, interoperability and interchangeability is the name of the game here, not centralization or standardization.

The design of RhythML also draws inspiration from tracker software, aiming to capture the directness and efficiency of tracker interfaces while adapting to the unique needs of modular synthesis environments like VCV Rack.

Other Notations

Excellent idea. I'll expand on the introduction and add comparisons to RhythML for each system. This will help readers understand RhythML's unique position and design choices.

To understand the context and motivation behind RhythML, it's helpful to consider some other systems for representing musical information. While each of these systems has its strengths, RhythML was developed to address specific needs in the modular synthesis environment, drawing inspiration from various sources while focusing on flexibility and intuitive use in a voltage-controlled context. Here's how RhythML compares to some established systems:

Traditional Music Notation

The standard system of writing music, using a five-line staff with symbols representing pitch and duration. While comprehensive and widely used, it can be complex to learn and doesn't always intuitively represent electronic or non-Western music concepts. Traditional notation primarily serves as a guide for human performers rather than as direct machine input.

Comparison: Unlike traditional notation, RhythML is designed for direct machine input and is more flexible for representing non-standard tunings and electronic music concepts. However, RhythML lacks the visual intuitiveness of traditional notation for classically trained musicians. The biggest difference is the same problem piano rolls have: empty space or slow sequences would take a lot more visual space (or pages), while traditional notation compresses time for sight-reading.

ABC Notation

A text-based music notation system designed for folk and traditional music. It uses the letters A through G for note names, with additional symbols for rhythm, accidentals, and other musical elements, including common ideas like repeated or looping sections. ABC Notation is compact and readable, making it popular for sharing tunes online and in folk music communities.

Comparison: RhythML shares ABC Notation's text-based approach, making it easy to share and edit. However, RhythML is more focused on modular synthesis concepts and allows for direct voltage specification, which ABC Notation doesn't support.

MML (Music Macro Language)

A music programming language originally developed for early computer sound chips. It uses a series of letters and numbers to represent notes, durations, and other musical parameters, and includes programmatic concepts like conditions, branches, and variables. MML is still used in some chiptune and game music contexts, offering a text-based way to create melodies and harmonies.

Comparison: RhythML and MML both offer text-based music programming. However, RhythML is more tailored to modular synthesis, offering voltage-specific controls and a wider range of input formats, while MML is more focused on chip-based sound generation.

MIDI (Musical Instrument Digital Interface)

A technical standard that defines a protocol, digital interface, and connectors for communication between electronic musical instruments, computers, and other audio devices. MIDI doesn't transmit audio, but rather event messages about notation, pitch, velocity, control

signals, clock signals, and other types of musical information. It's widely used in music production for connecting hardware and software instruments.

Comparison: While MIDI is excellent for real-time performance and recording, RhythML is designed for precise sequencing and voltage control in a modular environment. RhythML could represent MIDI-like note or CC data but also allows direct voltage specification, making it more versatile for modular synthesis applications. Because MIDI has been around so long, people have used it to control non-musical devices too, meaning you could use MIDI as the bridge between RhythML and all sorts of devices, controls, or applications.

Tracker Software

A type of music sequencer software that displays note and instrument data in a grid format. Trackers originated in the 1980s and became popular in early computer music composition, especially in the demoscene. They typically represent music as numeric and letter codes in a vertical, top-to-bottom format, with each column representing a separate instrument or sound channel.

Comparison: RhythML draws significant inspiration from trackers, adopting their grid-based, top-to-bottom approach to sequencing. However, RhythML is more focused on voltage control and modular synthesis concepts, offering a wider range of input formats and direct voltage specification that traditional trackers don't support. RhythML also lacks some of the more advanced playback and audio processing features found in tracker software, which are usually intended to be what we would today call a DAW.

Glossary of Terms

1v/octave standard: A standard in analog synthesizers where a one-volt change in the control voltage corresponds to a one-octave change in pitch.

Arpeggio / Arpeggiator: A musical technique where notes of a chord are played in sequence, rather than simultaneously.

Audio rate: The rate at which audio signals are processed, typically much faster than control signals. Like the "frame rate" but for sound, and sounds need a much higher frame rate to be audible; in most digital systems, this is 44.1kHz or higher.

Cents: A unit of measurement for musical intervals. There are 100 cents in a semitone.

Chord voicings: Different ways of arranging the notes of a chord, often by changing their octave or order.

Clock pulses: Regular timing signals used to synchronize different parts of a modular system or to set the tempo of a sequence.

Control Voltage (CV): An electrical signal used to control parameters in a modular synthesizer, such as pitch, volume, or filter cutoff.

DAW (Digital Audio Workstation): Software used for recording, editing, and producing audio files.

Envelope generator: A module that creates a varying voltage over time, often used to shape the volume or timbre of a sound.

Eurorack: A popular format for modular synthesizers, characterized by specific physical dimensions and electrical standards.

Filter cutoff: The frequency at which a filter starts to attenuate (reduce) the signal passing through it.

Gate signals: On/off signals in a modular system, often used to trigger events like note starts and stops.

Hertz (Hz): A unit of frequency, equivalent to one cycle per second.

LFO (Low-Frequency Oscillator): An oscillator that typically operates below the range of human hearing, used for modulation effects.

Microtonal: Music using intervals smaller than a semitone, or pitches between the notes of a standard scale.

Modular synthesis: A method of sound synthesis using separate modules to generate and process sound, connected in various configurations.

Oscillator: An electronic circuit or module that produces a repeating waveform, forming the basis of many synthesized sounds.

Patch: In modular synthesis, a specific configuration of connections between modules to create a desired sound or effect.

Phasor: A type of oscillator that produces a rising sawtooth wave, often used for timing or modulation purposes by “syncing to its phase”, as an alternative to using clocks.

Pitch bend: A technique for smoothly changing the pitch of a note up or down from its original frequency.

Polyphonic cables: In VCV Rack, cables that can carry multiple independent signals (up to 16) simultaneously.

Polymeters: The simultaneous use of two or more meters (time signatures) in a piece of music.

Polyrhythms: The simultaneous use of two or more rhythms that are not readily perceived as deriving from one another, usually by scaling two different rhythmic patterns of different lengths to fit within the same total duration, creating the feeling that they “overlap”.

Pulse width: In a square wave or pulse wave, the ratio of the time the signal spends high versus low in each cycle.

Rising edge: The transition of a signal from a low state to a high state. Often used to trigger events in modular systems; this is what most modules are actually detecting when they “detect” a trigger or gate.

Semitones: The smallest musical interval commonly used in Western tonal music, equal to a half step.

Sequencer: A device or software that can store a sequence of musical events (like notes or control changes) and play them back in order. Really, anything that lets you describe and use a “pattern”, meaning there is a huge variety of things which could be called “sequencers”.

Single-cycle waveforms: Waveforms that complete one full cycle in a specific number of samples, used as the basis for many digital synthesis techniques.

Tracker: A type of music sequencer software that displays note and instrument data in a grid format. Trackers originated in the 1980s and became popular in early computer music composition, especially in the demoscene. They typically represent music as numeric and letter codes in a vertical, top-to-bottom format, with each column representing a separate instrument or sound channel. This format allows for precise control over individual notes, effects, and timing, making trackers particularly suited for electronic and chiptune music styles. The compact, text-based nature of tracker formats inspired aspects of RhythML’s design.

Trigger signals: Short pulses used to initiate events in a modular system, such as starting an envelope or advancing a sequencer.

VCA (Voltage-Controlled Amplifier): A module that controls the amplitude (volume) of a signal based on a control voltage input.

VCO (Voltage-Controlled Oscillator): An oscillator whose frequency can be controlled by an input voltage.

VCV Rack: A free, open-source software that emulates modular synthesizers on a computer.

VST (Virtual Studio Technology) plugin: Software that integrates with audio editing and processing applications to provide additional functionality, often in the form of virtual instruments or effects.

Wavetable synth: A type of synthesizer that uses stored tables of waveforms as its sound source, allowing for complex and evolving timbres.