Firmware user manual for the Ornament & Crime eurorack module.

This documentation is for v1.3 firmware.

Documentation for the previous **production** firmware release, which is v1.2, is here (/user-manual-v1_2/).

Documentation for earlier versions can be found here (https://github.com/mxmxmx/O_C/wiki/Ornaments-and-Crimes-extended-firmware).

Overview

- the **Ornament & Crime** firmware is a collaborative open-source (/licensing/) project by Patrick Dowling (aka pld), Max Stadler (aka mxmxmx) and Tim Churches (aka bennelong.bicyclist). It (considerably) extends the original firmware for the Ornament & Crime (o_C) DIY eurorack module, designed by mxmxmx.
- the original o_C module was designed to perform a single function: a digital, quantising version of the classic **analogue shift register** (ASR).
- there is still a **quantising ASR** (analogue shift register) function in the current Ornament & Crime firmware, now named *CopierMaschine*, **but several other "apps" have been added**, incl. quantisers, sequencers, LFOs, random/chaotic CV generators, and so on. These apps are selectable **on-the-fly**, without having to reboot the module or toggle the power.

- the o_C module, and the firmware for it, break new ground as a polymorphic module: a generic set of inputs and outputs are provided, and the textual OLED display is used to permit re-mapping of these inputs and outputs for each app, and in some cases, for each of the four channels within each app. The o_C module does not purport to be the paradigmatic pinnacle of polypurpose — in fact, as noted above, it was originally designed to fulfil just one purpose (ASR) — but together with the much-expanded firmware for it, we hope it provides an interesting and useful early step in the evolution of such multi-purpose modules.

The apps currently available in Ornaments & Crimes are:

- *CopierMaschine* is an enhanced version of the original quantising digital emulation of a four stage **analogue shift register** (ASR).
- *Harrington 1200* provides basic neo-Riemannian **Tonnetz transformations** of triadic chords, triggered by the digital (gate/trigger) inputs.
- *Automatonnetz* combines Tonnetz transforms with a **“vector” sequencer** - it can be both a chord sequencer and a melody sequencer, but not of the usual kind.
- *Quantermain* is a **quad pitch quantiser** for external voltages, with editable scales; it can do clocked (trigger-driven) quantising, or continuous quantising, with a latency of under 100 microseconds; it also features quad Turing Machines, May-Verhulst logistic maps or byte beats as optional, semi-random, internally generated CV sources.
- *Meta-Q* is a dual-channel quantiser, similar to *Quantermain*, but also offering **scale and note mask sequencing**.
- *Quadraturia* is a wavetable **quadrature LFO**, based on the “Easter egg” in the Mutable Instruments Frames (http://mutable-instruments.net/modules/frames) module.
- *Low-rents* is a dual **Lorenz and Rössler** (strange attractor) modulation generator, partially based on the “Easter egg” in the Mutable Instruments Streams (http://mutable-instruments.net/modules/streams) module.
- *Piqued* is a quad voltage-controlled **envelope generator**, based on envelope generator code from the Mutable Instruments Peaks (http://mutable-instruments.net/modules/peaks) module, but extending it with voltage control, additional envelope types, including re-triggering (looping) envelopes, additional segment shapes, adjustable trigger delays, and a unique Euclidean “trigger filter” which turns the app into a Euclidean rhythm generator which can output envelopes, not just gate or trigger pulses.
- *Sequins* is a dual-channel **step sequencer** offering 4 “tracks” of up to 16 steps each; tracks can themselves be sequenced.
- *Dialectic Ping Pong* is a quad bouncing ball **envelope generator**, based on a hidden mode of the Mutable Instruments Peaks (http://mutable-instruments.net/modules/peaks) module.
- *Viznutcracker, sweet!* is a quad **“byte beat” equation generator**, which can be used as an audio source to generate curious but often interesting 8-bit noises and tunes, or which can be clocked by an external source to produce “byte beat” control voltage sequences. “Byte beats” were first described (http://countercomplex.blogspot.com/2011/10/algorithmic-symphonies-from-one-line-of.html) in 2011 by viznut (aka Ville-Matias Heikkilä).
- *Acid Curds* is both a chord quantiser (sometimes called a “harmonic quantiser” for external pitch voltages), and a **chord progression sequencer**.
- *References* is an utility app that outputs specific **reference voltages** on each channel to help tune or calibrate VCOs and other modules. It also includes a high-precision **frequency meter** and note tuner, a high-precision BPM (beats per minute) **tempo meter**, and a **closed-loop calibration** mode.

For information about the o_C module **hardware**, including DIY build details, please start here (/hardware-basics/).

Operational principles

Startup

Immediately after power-on, the start-up phase is indicated by a line at the bottom of the display which progressively shrinks right-to-left. By default, following start-up phase, the module boots into the last saved app, unless one of the encoders is held down:

- hold down the Left encoder: Boot into calibration mode (unchanged from the original o_C firmware - please see the calibration procedure here (/calibration/)
- hold down the Right encoder: Enter the app selection menu

App Selection

The **app selection menu** can be reached either when the module is first powered on, as described above, or by depressing the right encoder for two seconds or longer (a long press) during normal use.

- Turn the right encoder to select an app.
- Press the right encoder to switch to the selected app.
- Pressing the Up button (the upper button to the right of the display) while the App Selection screen is displayed toggles encoder acceleration on and off (there is no visual indication of this). Encoder acceleration increases the amount by which values are incremented or decremented on each encoder rotational click when you are rapidly spinning the encoder in value edit mode, rather than advancing it slowly one click at a time. This speeds up sweeping across the full range of values, while still retaining full precision when moving the encoder one step at a time.

Save Settings / Module State

The settings for each o_C app are **NOT** saved automagically. To save the current state/settings, enter the **app selection menu** (see above), then **long press** the right encoder (> 2s) **a second time** to save the module state and make the selected app the default at the next boot. A confirmatory expanding quadrilateral will briefly be displayed. Note that the current settings for **all apps** are saved each time you save the settings, not just the settings for the currently selected app.

- **Note:** the operation of the module is briefly **paused** while the settings are saved. (That's intentional). Normally this pause is of no consequence, but you may need to take it into account if/when you are using your o_C in a live performance setting. The module does not pause when switching from one app to another, only when settings are saved.
- The settings are saved permanently in the Teensy 3.2 EEPROM storage. (EEPROM memory can only be written to a certain number of times before it becomes unreliable. The EEPROM storage is good for at least 100,000 write cycles, and probably a lot more. Thus, you are very unlikely to wear out the EEPROM memory in the Teensy/MK20 processor in normal use. In the unlikely event that you do wear it out, just replace the Teensy at a cost of about US$25.)

Reset app settings to default

To **reset** the app settings to their default state, push the up and down buttons **simultanously** during start-up (the 'splash screen'). (This won't reset the calibration data, of course).

General operation of the apps

Each app has two display pages, a settings mode and a “screensaver” mode. The apps drop into screensaver mode after a short period of inactivity (that is, no user interaction); the timeout period can be set in the calibration menu (/calibration/#4-screensaver-timeout-period). Alternatively, **long press** (> ~ 1.5 sec) the `up button` to **invoke** the screensaver.

- In the background, the module functions the same regardless of whether settings are displayed or the screensaver is displayed. When the screensaver is displayed, clicking or rotating either of the two encoders, or pressing the Up or Down buttons will immediately swap back to the settings display.
- In settings mode, for most of the apps, the right encoder is used to scroll up and down the list of available settings, and clicking on the right encoder will toggle edit mode, indicated by one or two small up/down triangles next to the setting. Turning the right encoder when in edit mode will increment (increase) or decrement (decrease) the relevant setting. Clicking again on the right encoder toggles back to settings selection mode.
- In multi-channel apps, the left encoder selects which of the four channel (A to D) settings are currently displayed (and editable). All channels remain active in the background, all the time, even while editing settings. The channel can be changed even when value edit mode is active, making it easy to edit the same setting on each of the four channels.
- In other apps, the left encoder is used to set a root note or a transposition, or to set frequency or rate, or a scale. Its exact behaviour is noted below in the documentation for each of the apps.
- The two buttons to the right of the display (the **Up** and **Down** buttons) either transpose notes up or down an octave, or increase or decrease frequency or speed in steps of 32 (with a few exceptions for some apps, as noted below).

Scales, custom-scales, and non-octaval (non-1V/oct) tunings.

- o_C comes with > 100 predefined scales (modifiable in the source code) plus 4 user-defined scales editable directly through the module user interface; for details on the scale-editor, see *Quantermain* below; for more information on custom scales see here (/custom-scales/); for a list of the predefined scales, see here (/predefined_scales/).
- many of the predefined scales are microtonal i.e. they have more than 12 notes per octave, or they are subsets of tunings with more than 12 notes per octave. o_C supports scales with up to 16 notes per octave (but those 16 notes are arbitrary and themselves can be subsets of scales with many more than 16 notes per octave).

Non-octaval tunings for the xenharmonically obsessed

- in addition, it’s possible to select from a choice of alternative non-octaval tunings (per channel):
    - Wendy Carlos (https://en.wikipedia.org/wiki/Wendy_Carlos) alpha scale (https://en.wikipedia.org/wiki/Alpha_scale)
    - Wendy Carlos beta scale (https://en.wikipedia.org/wiki/Beta_scale)
    - Wendy Carlos gamma scale (https://en.wikipedia.org/wiki/Gamma_scale)

    - Tritaval tuning (as used by the Bohlen-Pierce macrotonal scale (https://en.wikipedia.org/wiki/Bohlen–Pierce_scale)), and
    - Quartertone scale (https://en.wikipedia.org/wiki/Quarter_tone) (which essentially downscales to 0.5V/oct)
- some of these non-octaval tunings are designed to work with particular scales eg the Carlos tunings work well with normal 12-TET scales, while the tritaval tuning is intended to work specifically with the Bohlen-Pierce scales and related harmonic scales — see the full list here (/predefined_scales/#tritaval-scales).
- the alternative tunings can be accessed by **long-pressing** the left encoder in the scale-editor. (Select the channel turning the right encoder; the left encoder is used to select from the choice of tunings). Use of alternative tunings is indicated by a **dashed line** underneath the top menu row (the line is solid when using the default 1V/oct).

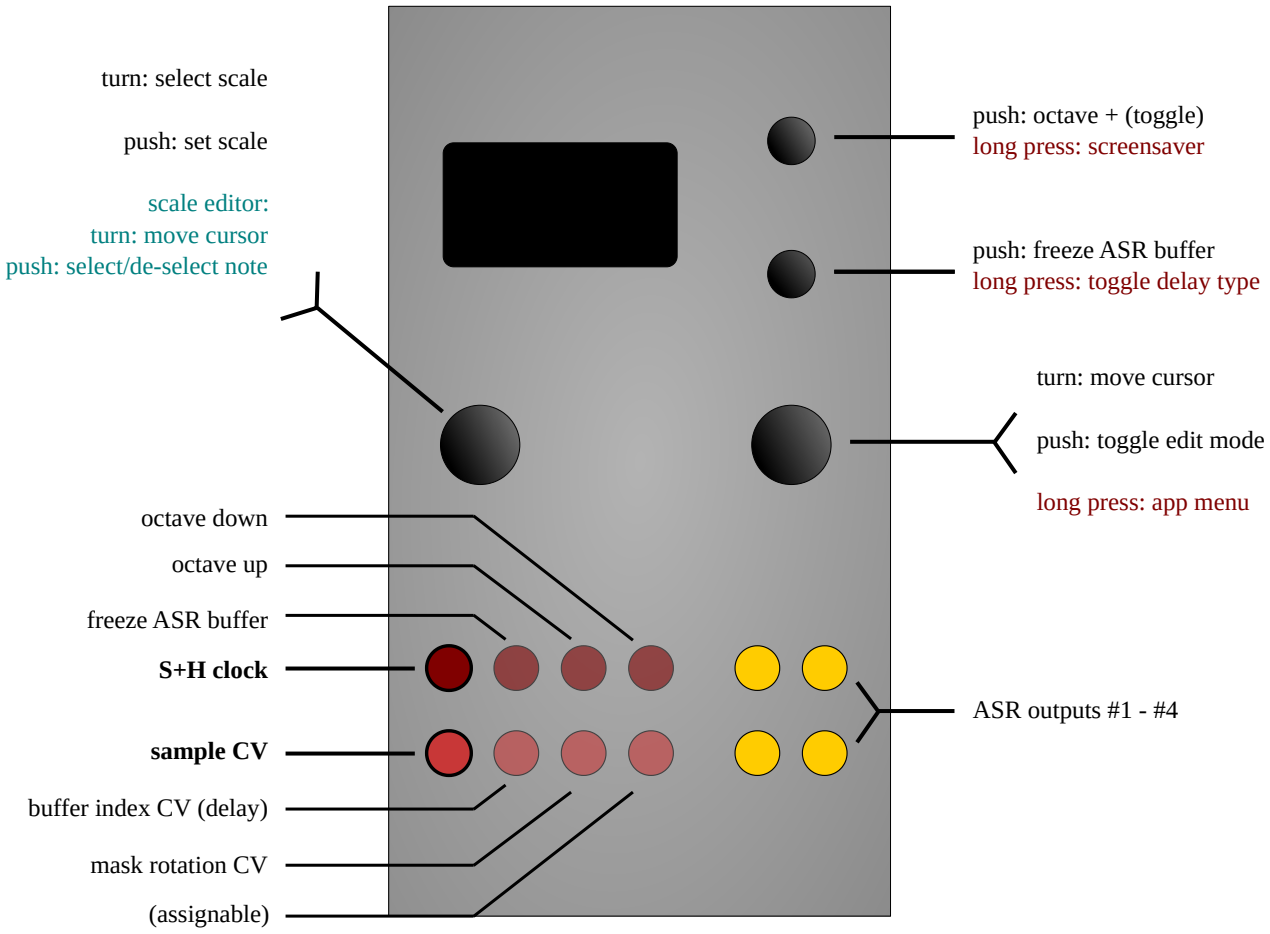
The Apps

CopierMaschine

This app works much the same as the original o_C / quantising **ASR** firmware, except that it now incorporates some of the new/improved quantiser features, including a larger selection of **(editable) preset scales** as well as advanced user-scale-edit and (non-1V/oct) tuning options (for details, see the Quantermain “app” below and the info re: custom scales/tunings above).

In essence, then, ASR mode works as a cascaded, four-stage sample-and-hold module (see here (http://www.cyndustries.com/synapse/synapse.cfm?pc=35&folder=sept1976&pic=19) for an exposition on the classic ASR implementation):

- Feed a pulse (clock, trigger) into the left-most digital input (**TR1**), and a CV signal (LFO, ADSR, etc) into the leftmost CV input (**CV1**): on receiving a/the clock, the DAC outputs will be updated, **ASR-style**: the sampled value will be present at output A, the previous sample values shifted down the remaining outputs B, C, and D.



- The ASR mode features additional parameters, including
  - a **delay** (= controlled via the `buf. index` parameter (CV2))
  - **scale 'mask'** rotation (CV3)
  - **hold** (which "freezes" the sample buffer) (TR2, down button)
  - CV over **transposition**, **scale mask rotation**, **root**, **buffer size** (assignable via CV4)
- Please also see the discussion of the `Trigger delay` menu selection in the *Quantermain* app documentation below — the same considerations apply to the `trigger delay` setting in *CopierMaschine*.

buffer index (delay)

- The `index` parameter works as a delay, sort of: internally, the ASR is a ring-buffer (buffer size = 256), and (to simplify things) by default outputs the sampled values S[x] stored at the buffer locations **index * output-stage**, ie A = S[i * 1], B = S[i * 2], C = S[i * 3], and D = S[i * 4].
- The default index setting (`buf. index`) is 0 (internally i = 1), in which case things boil down to standard ASR behaviour:

  A = S[1], **B = S[2]**, **C = S[3]**, and **D = S[4]**
- If the index parameter was instead set to, say, i = 8, the ASR in that case would output the values stored in the buffer at locations S[8], S[16], S[24], and S[32], thus delaying output A by 8 clocks, B by 16 clocks, and so on. Thus, modulating the `index` parameter doesn't just delay the output on channels B to D, but also allows different patterns to be created (based on the contents of the buffer).

- **Alternatively**, you can change the way the delay behaves by long-pressing the down-button once. (Two little dots will appear next to the clock indicator at the top of the menu, on the right). In this case, the value of `buf. index` will simply be **added** to the buffer locations, ie A = S[1 + i], B = S[2 + i], C = S[3 + i], and D = S[4 + i]. The resulting behaviour is that of a regular delay line. Another long-press on the down button will toggle back to the first `index` mode.

hold (‘freeze’)

- Pressing the **down button** will toggle ‘freeze’ mode. Freeze mode is also activated while the TR2 (= hold) input is held **high** (using a gate or the like). In freeze mode, **no further samples will be acquired**; when clocked, the four outputs then simply cycle through what’s already the buffer. If ‘freeze’ is engaged, two little dots will appear next to the scale name, at the top of the menu, indicating the buffer is currently frozen.
- The size of the hold buffer is determined by the `hold (buflen)` parameter, which goes from 4 to 63. While the buffer is ‘frozen’, all the various modulation options (buffer length, transposition, scale change, etc) are still available / operate on the frozen buffer contents.
- ‘Freeze’ mode effectively turns *CopierMaschine* into a clocked, quantised CV recorder, with four output taps and variable playback parameters.

Inputs and outputs

I/O	Function
TR1	Clock input
TR2	Hold (= freeze ring buffer)
TR3	Transpose: Octave up, when high
TR4	Transpose: Octave down, when high (overridden by TR3)
CV1	Sample in
CV2	Index: ring buffer index (= “delay”)
CV3	Mask: rotate scale mask
CV4	assignable: octave, root, transpose (by scale-degrees), buffer-length, or CV input scaling

A, B, C, DASR outputs 1-4

Controls

Control	Function
Left encoder (turn)	Select scale in main menu; move cursor in scale edit menu
Left encoder (press)	Activate scale in main menu; add/remove note in scale edit menu
Right encoder (turn)	Navigation mode: move up and down through the menu items. Edit mode: increase or decrease the value being edited; move scale “mask” in scale edit menu
Right encoder (press)	Toggle between menu navigation (selection) mode and value editing mode
Right encoder (long press)	App selection menu
Up button	toggle octave up
Up button (long press)	screensaver shortcut
Down button	freeze ASR input buffer
Down button (long press)	toggle delay type: multiplying / basic

Available settings

Setting	Meaning
`scale`	Current scale (active scale labelled with a little dot)
`root`	Root note for scale
`mask`	“scale mask” / active note pattern in the selected scale
`octave`	octave
`buf. index`	ring buffer index (= “delay”) amount
`hold (buflen)`	length of buffer (when freezing the ASR): `4 - 63`
`trigger delay`	sets the TR1-input-to-processing latency (for details see *Quantermain* below)
`input gain`	CV “gain”, multiplies incoming CV1 sample value by selected value (range: 0.05 - 2.00 in steps of 0.05). Note that this setting also affects the range of the ‘internal’ CV sources (LFSR, bytebeats and integer sequences, see below).
`CV4 dest. -->`	parameter assigned to CV4 input
`CV source`	sets the source of the sample voltage, either an external voltage (CV1), or an ‘internal’ source (details see below)

Scale edit:

see here (user-scales are shared across apps).

Screensaver display

Four little “Arabesque” patterns, representing the pitch CV output on each of the four channels.

CV sources

Four settings of the `CV source` parameter are available:

- `CV1` — voltages on the CV1 input are quantised, according to the `scale` and `Active notes` settings.
- `LFSR` — “Linear Feedback Shift register” (also referred to as a “Turing Machine” or “TM” elsewhere in this documentation). This is the same as the `Turing` source in the *Quantermain* app - please see the discussion of the Turing Machine source in the *Quantermain* section for further details of operation. The available settings for the Turing Machine source in *CopierMaschine* are shown in the table below.
- `ByteB` — “bytebeat” equations used to generate semi-fractal note values, rather than audio signals (which is what byte beat equations are usually used for). Please see the *Viznutcracker, sweet!* app for details of the byte beat equations available.
- `IntSq` — integer sequences - several classes of random and fractal integer sequences, used as note values.

LFSR source settings in *CopierMaschine*

Note: `LFSR` is used equivalently to “Turing Machine” here. (see *Quantermain* for details).

Setting	Meaning
`LFSR length`	Length of the linear feedback shift register, in bits, range 4 to 32
`LFSR p`	Probability that the least significant bit will be flipped when it to copied, range 0 to 255 (0 means *p*= 0, 255 means *p*= 1)
`LFSR CV1`	The Turing Machine parameter to which any voltage input on CV1 will be directed. Choices are `rng`, `len` and `p` (ie, range, length, and probability)

Note that the `LFSR range` setting in previous versions has been subsumed by the `input gain` setting, which now also affects the 'internal' CV sources such as LFSR, bytebeats and integer sequences.

Byte beats source settings in *CopierMaschine*

Setting	Meaning
`BB eqn`	sets the byte beat equation used as the source. See the *Viznutcracker, sweet!* app documentation for more details of the currently available equations.
`BB P0`	Parameter 0 for the byte beat equation - see See the *Viznutcracker, sweet!* app documentation for more details.
`BB P1`	Parameter 1 for the byte beat equation - see See the *Viznutcracker, sweet!* app documentation for more details.
`BB P2`	Parameter 2 for the byte beat equation - see See the *Viznutcracker, sweet!* app documentation for more details.
`BB CV1`	The byte beat parameter to which the input on CV1 is directed. Possible destinations are "igain" (input gain), "eqn" (equation), "P0", "P1", "P2". See `LFSR CV1` above for details of input voltage ranges.

Note that compared to the bytebeat source in *Quantermain*, the `Bytebeat range` parameter is missing from *CopierMaschine*. The reason is that the `input gain` setting has the same effect in *CopierMaschine* when the `ByteB` source is used as the `Bytebeat range` setting does in *Quantermain*.

Integer sequence source settings in *CopierMaschine*

Setting	Meaning
`IntSeq`	sets the integer sequence used as the source. See below for a list of available integer sequences and their characteristics.
`IntSeq modul`	sets the modulus for the integer sequence. The value of the integer from the integer sequence is divided by the modulus and the remainder is used. For example, if the modulus is 8 and the current integer value from the sequence is 19, then the remainder of 19 - (2 x 8) i.e. 3 is used as the value. In other words, values "wrap around" at the modulus setting — it sets a maximum note range for the integer sequence, similarly to the `M/A` setting, but `M/A` compresses of expands the range of notes for a given integer value from the sequence, whereas the modulus wraps the values around.
`IntSeq start`	sets the start point in the stored integer sequence. The stored sequences are 128 steps long, and the maximum start point is 126 to ensure a minimum sequence length of 2.
`IntSeq len`	sets the length of the integer sequence. Thus a length of 16 will use just 16 values from the stored 128 step sequence, starting at the step specified by `IntSeq start`.
`IntSeq dir`	sets whether the integer sequence loops back to the beginning when it gets to the end, or whether it swings back like a pendulum and plays in reverse when it reaches the end. The "end" is the last step in the sequence, as defined by the sequence start plus the sequence length settings.

Setting	Meaning
`Fractal stride`	Several of the sequences are fractal or semi-fractal in nature (i.e. they are self-similar), and the “stride” setting sets how many steps are advanced on each trigger input. This also works well with the non-fractal sequences and provides additional variation, particularly if the stride is not an exact divisor of the sequence length.
`IntSeq CV1`	The integer sequence parameter to which the input on CV1 is directed. Possible destinations are (parameter names in brackets): `M/A` (mult/att), `seq` (IntSeq), `strt` (IntSeq start), `len` (IntSeq len), `strd` (Fractal stride) and `mod` (IntSeq modul). See `LFSR CV1` above for details of input voltage ranges.

Integer sequences available in *CopierMaschine* and *Quantermain*

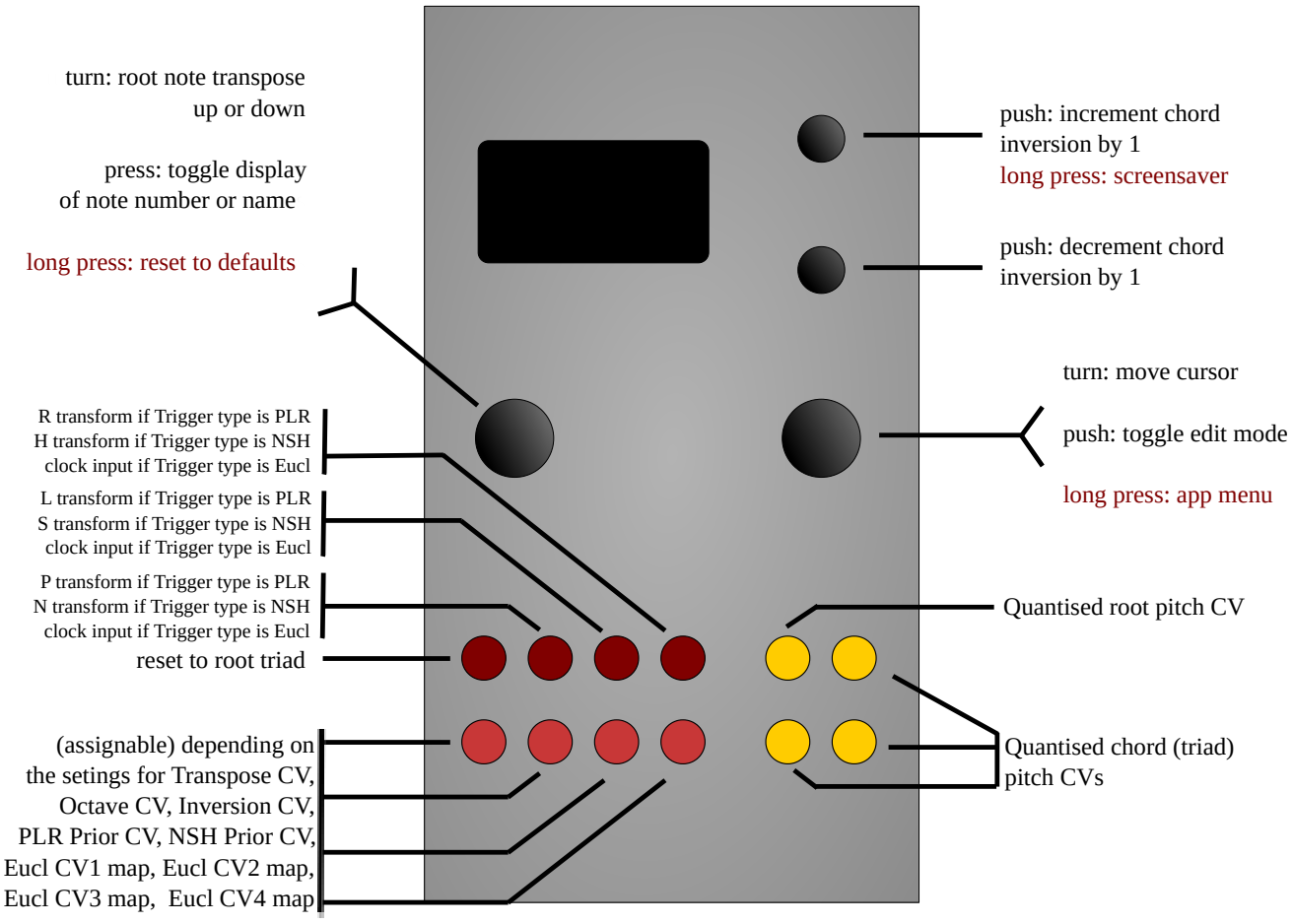
Menu name	Description
`pi`	The first 128 digits of π (https://en.wikipedia.org/wiki/Pi)
`vnEck`	The first 128 integers in van Eck’s sequence (https://oeis.org/A181391).
`ssdn`	The first 128 integers in the sequence of the sum of squares of the digits of *n* (https://oeis.org/A003132).
`Dress`	Dress’s sequence (https://oeis.org/A001316).
`PNinf`	Per Nørgård’s infinity series (http://www.pernoergaard.dk/eng/strukturer/uendelig/ukonstruktion03.html)
`Dsum`	Digital sum (i.e. sum of digits) of *n*; also called digsum(n) (https://oeis.org/A007953).
`Dsum4`	Digital sum (i.e. sum of digits) of *n* written in base 4 (https://oeis.org/A053737).
`Dsum5`	Digital sum (i.e. sum of digits) of *n* written in base 5 (https://oeis.org/A053824).
`CDn2`	Fractal sequence: count down by 2’s from successive integers (https://oeis.org/A122196).
`Frcti`	Fractal sequence of the interspersion A163253 (https://oeis.org/A163256).

See the acknowledgements section for additional references for some of these integer sequences.

Note that there isn’t really anything magical or mystical about the digits of transcendental numbers such as π — they are just a convenient source of sequences of digits. However, if you use them to make music, then you can post videos like this (https://www.youtube.com/watch?v=YOQb_mtkEEE), or this (https://www.youtube.com/watch?v=OMq9he-5HUU). The fractal and semi-fractal sequences can produce some magical melodies, however.

Harrington 1200

This is a relatively straight-forward implementation of neo-Riemannian transformations for generating triad (three note chord) progressions (see the acknowledgements (http://ornament-and-cri.me/acknowledgements/) section for more details on neo-Riemannian music theory).



- In settings mode, the top line of the display shows, from left to right:
  - the current **root note** for the root (initial) chord
  - the **musicological mode** (major or minor) for the root chord
  - the three notes comprising the **current triad** being output. Clicking on the left encoder toggles between note display and display of semitone offset from the root note.
- The root note can be changed using the `Transpose` setting in the menu. The left encoder can also be used to change this setting at any time (such as during live performance). The voltage input on CV1 also changes the root note (i.e. the transposition).
- The musicological mode of the root chord is set by `Root mode` in the menu to either major or minor. Chord inversion is similarly set using the `Inversion` menu item. Note that these settings are not immediate - they will take effect when the next transformation trigger input is received. The top line of the display indicates the current triad, not the current menu settings for these parameters. Note that the voltage input on CV4 also changes the inversion.
- The pitch voltage (scaled to 1V/octave) for the root note appears on output A. The pitch voltages for the three notes of the current triad appear on outputs B, C and D. Thus, to pproduce chords, you should feed the B, C and D outputs into the 1V/octave pitch inputs of three VCOs.
- Trigger inputs TR2, TR3 and TR4 are used to apply the atomic P, L or R transformations, or the compound N, S or H transformations (see below), depending on the `Trigger type` setting. Trigger input TR1 resets the current triad back to the root chord for all settings of `Trigger type1` .

- If multiple triggers are received, the reset input (TR1) always has priority, then all triggered transforms are applied. The order in which they are applied can be set in the menu by the `PLR Priority` and `NHS Priority` settings.
- The neo-Riemannian transformations themselves are quite simple, and “reversible” i.e. applying them twice returns the original triad. The following basic (atomic) tranformations are provided:
    - P (Parallel): Moves the third up or down a semitone, thus P(Cmaj) = Cmin, P(Cmin) = Cmaj.
    - L (Leittonwechsel): Converts a major triad to a minor by shifting the root down a semitone and making the third the root, or from minor to major by moving the fifth up a semitone to become the root.
    - R (Relativ): Converts a major triad to its relative minor, or a minor triad to its relative major.
- Alternatively, secondary transformations can be used:
    - N (Nebenverwandt): Exchanges a major triad for its minor subdominant, and a minor triad for its major dominant (e.g. C major and F minor). The “N” transformation is the same as applying R, L, and P successively.
    - S (Slide): Exchanges two triads that share a third (e.g. C major and C♯ minor); it is the same as applying L, P, and R successively, in that order.
    - H (Hexatonic): Exchanges a triad for its hexatonic pole (e.g. C major and A♭ minor); it is the same as applying L, P and L transformations successively.
- The implementation computes these N, S and H transformations in a single step however, not sequentially.
- Note that in trigger modes `PLR` and `NSH`, only PLR or NSH transformations can be applied, but not mixtures of both. However, in `Eucl` (Euclidean) trigger mode (see below), mixtures of both PLR atomic transformations and NSH secondary or compound transformations can be applied.
- Internally, the triad is stored in a neutral form (basically just offsets), thus the chord voicing is preserved and can be shifted easily to the quantised root note, and inversions created on-the-fly. For an alternate way of implementing these transformations, see the documentation of the Tonnetz Sequent (http://www.noiseengineering.us/tonnetz-sequent/).

Screensaver display

The current triad (output as pitch CVs on the B, C and D sockets) is shown graphically on the left. The circumference of the circle has 12 points on it, one point for each semitone in an octave, with C at the 12 o’clock position. The current triad is indicated by the three dots, joined to form a triangle. The middle section of the screensaver display shows the last four transformations applied, with the most recent one at the top. The rightmost section indicated the current triad being output, with numbers showing the octave for each note.

Tip: for a pretty display, patch the output of an LFO into CV1 in order to rapidly rotate the root note up and down.

Settings

Setting	Meaning
`Transpose`	Shift root/triad in semitones, range -24 to 24 (i.e. two octaves up or down)
`Transpose CV`	Selects CV input used to shift root/triad in semitones, up (positive voltages up to about 6.5V) or down (negative voltages down to about -3.5V). Transposition should be scaled at 1V/octave. Choices are `None`, `CV1`, `CV2`, `CV3`, `CV4`.
`Octave`	Shift root/triad in octaves, range -3 to 3.

Setting	Meaning
`Octave CV`	Selects CV input used to shift root/triad in semitones, up (positive voltages up to about 6.5V) or down (negative voltages down to about -3.5V). Scaling is **not** 1V/octave. Choices are `None`, `CV1`, `CV2`, `CV3`, `CV4`.
`Root mode`	Mode of root triad, either `maj` or `min`
`Inversion`	Chord inversion, range is from -3 to 3.
`Inversion CV`	Selects CV input used to shift the chord inversion, up (positive voltages up to about 6.5V) or down (negative voltages down to about -3.5V). Choices are `None`, `CV1`, `CV2`, `CV3`, `CV4`.
`PLR Priority`	Order in which the P, L and R transforms are applied if multiple triggers on TR2, TR3 and TR4 are received simultaneously
`PLR Prior CV`	Selects CV input used to shift the PLR priority amongst the available choices, up (positive voltages up to about 6.5V) or down (negative voltages down to about -3.5V). Choices are `None`, `CV1`, `CV2`, `CV3`, `CV4`.
`NSH Priority`	Order in which the N, S and H transforms are applied if multiple triggers on TR2, TR3 and TR4 are received simultaneously
`NSH Prior CV`	Selects CV input used to shift the NSH priority amongst the available choices, up (positive voltages up to about 6.5V) or down (negative voltages down to about -3.5V). Choices are `None`, `CV1`, `CV2`, `CV3`, `CV4`.
`CV sampling`	Selects how sampling of CV1, CV2, CV3 and CV4 inputs is done: either continuously (`Cont`) or on a sample-and-hold basis (`Trig`) which is triggered by a trigger on any or all of the four trigger inputs.
`Output mode`	Output mode, with the default being `chord`, or use `tune` to output the quantised root on all four output channels
`Trigger type`	Sets the trigger behaviour, either P. L and R transforms triggered by trigger inputs on TR2, TR3 and TR4 respectively, or N, S or H transforms, also triggered bt TR2, TR3 or TR4 respectively, or `Eucl`, which enable Euclidean trigger masks - see below for details.
`Trigger delay`	sets the TR1-input-to-processing latency (for details see the setting with the same name in *Quantermain* below)

Controls

Control	Function
Left encoder (turn)	Root note transpose up or down
Left encoder (press)	Toggle display of note numbers (semitone offsets from the root note) or names (note: the names are the simplest possible mapping, thus there are no enharmonic substitutions)
Left encoder (long press)	Reset to defaults
Right encoder (turn)	Navigation mode: move up and down through the menu items. Edit mode: increase or decrease the value being edited.
Right encoder (press)	Toggle between menu navigation (selection) mode and value editing mode
Right encoder (long press)	App selection menu
Up button (long press)	Invoke screensaver display
Up button (press)	Increment chord inversion by 1

Control	Function
Down button (press)	Decrement chord inversion by 1

Inputs and outputs

I/O	Function
TR1	Reset to root triad
TR2	P transform (if `Trigger type` is `PLR`), or N transform (if `Trigger type` is `NSH`), or clock input (if `Trigger type` is `Eucl`)
TR3	L transform (if `Trigger type` is `PLR`), or S transform (if `Trigger type` is `NSH`), or clock input (if `Trigger type` is `Eucl`)
TR4	R transform (if `Trigger type` is `PLR`), or H transform (if `Trigger type` is `NSH`), or clock input (if `Trigger type` is `Eucl`)
CV1	Mapped according to `Transpose CV`, `Octave CV`, `Inversion CV`, `PLR Prior CV` and `NSH Prior CV` settings (see table above), as well as according to the `Eucl CV1 map`, `Eucl CV2 map`, `Eucl CV3 map` and `Eucl CV4 map` settings (see below) - each CV input can be used for multiple internal destinations.
CV2	Ditto
CV3	Ditto
CV4	Ditto
A	Quantised root
B, C, D	Transformed & inverted triad (also quantised)

Euclidean trigger masks

Euclidean trigger mask mode in the *Harrington 1200* app is enabled by setting the `Trigger type` setting to `Eucl`. Twenty-two additional settings will appear in the menu when the `Eucl` trigger type is selected. In this mode, a simple, single, regular clock input can be used (into TR2, TR3 or TR4) to trigger P, L, R, N, S or H chord transformations. Each of the six transformation types has its own “Euclidean mask” or “Euclidean filter” — by varying the parameters on each of these for each transformation type, very complex “polyrhythms” of chord transformations can be derived from as single regular clock input. Furthermore, all of the Euclidean trigger mask parameters, for each transformation type, can be placed under external voltage control. This permits external voltages to influence complex, evolving patterns of chord transformations driven by a single external clock.

Each type of transformation (P, L, R, N, S and H) has three settings controlling the Euclidean trigger mask (Euclidean filter) for it (thus 18 controls all together): the length of the Euclidean pattern (named `X EuLeng`, where X is the type of transformation, one of P, L, R, N, S or H), the fill amount (`X EuFill`) for that length of pattern, and the offset or rotation (`X EuOffs`) of the pattern. These all operate in the same way as the Euclidean trigger filters for the envelope generators in the *Piqued* app operate - see the discussion in the *Piqued* section for further details.

The settings `Eucl CV1 map` through to `Eucl CV4 map` permit the four external voltage inputs to be mapped to four of the 18 available Euclidean trigger mask settings. The same input voltages can also be used to simultaneous control transposition, octave, inversion and trigger priorities (see above), if desired.

Tips

Try using a rhythm generator, such as the Mutable Instruments Grids (http://mutable-instruments.net/modules/grids) module, or the ALM/Busy Circuits Pamela's Workout (http://busycircuits.com/alm001/) module, or the Rebel Technology Stoicheia (http://www.rebeltech.org/products/stoicheia/) module, or some combination of clock division and logic modules, or even a second o_C module running the *Piqued* app with Euclidean trigger filters engaged, to trigger the P, L and R transformations (via trigger inputs TR2, TR3 and TR4) in varying patterns to create a variety of chord progressions that may or may not be musically pleasing, or interesting, or horrifying.

Update: in v1.2, you can use the built-in Euclidean trigger masks to achieve the same thing, and more, with just a single external regular clock input (into TR2, TR3 or TR4).

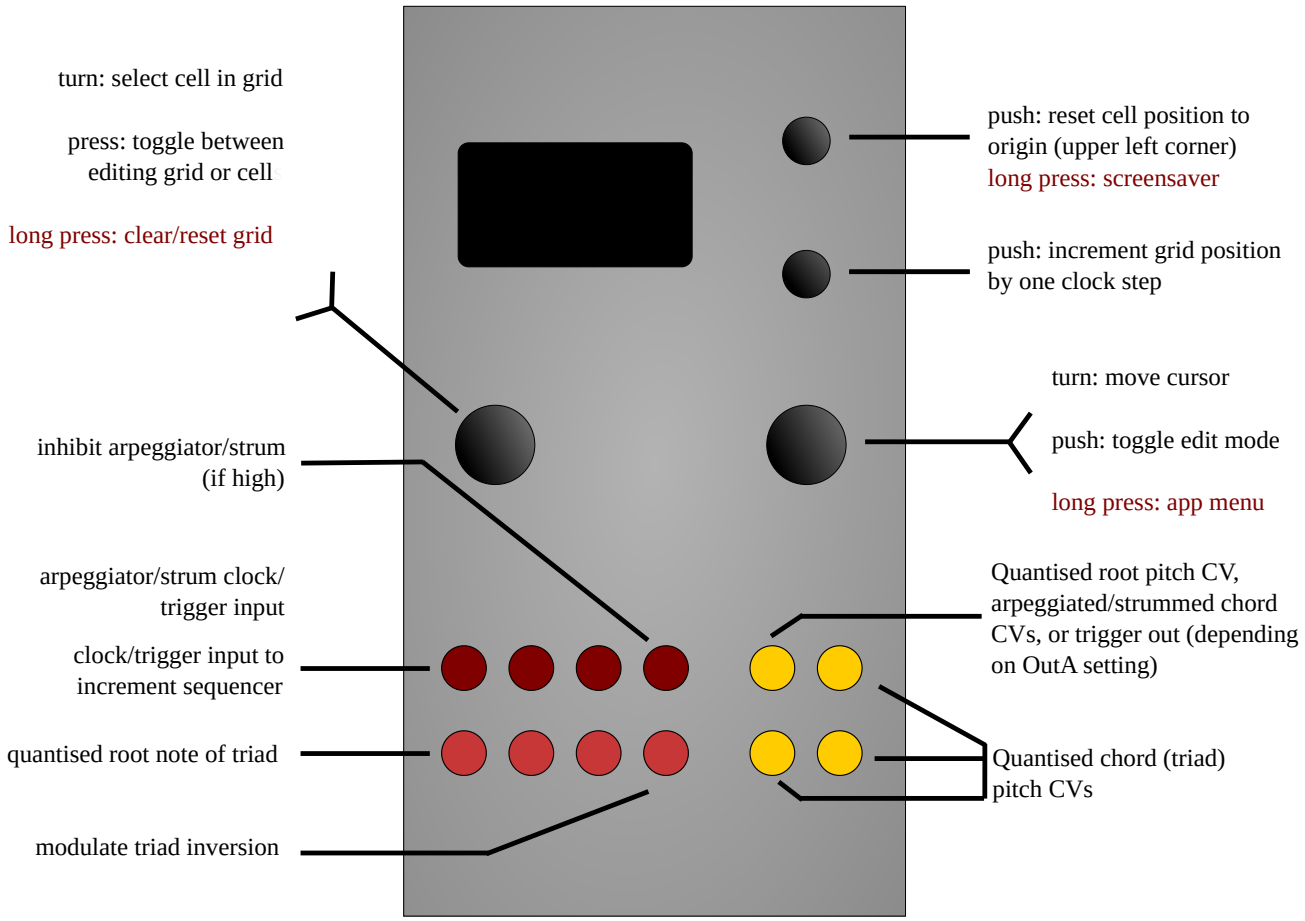
Automatonnetz

This app uses the neo-Riemannian transformations implemented in the Harrington 1200 app (see above), but with the sequence of transformations determined not through triggers for each type of transformation, but rather by navigating a 5x5 grid of cells. On each clock input the `dx` (delta x) and `dy` (delta y) — hence "vector" sequencer — values are added to the current position on the grid to determine the next cell. The position simply wraps around when it reaches the edge of the grid, and "backwards" motion is also possible. The position and movement can also be fractional, allowing for clock divisions and all kinds of patterns.

Each cell of the grid can contain a neo-Riemannian transformation, or a reset, as well as other parameters. In this app, there are three additional transforms available, which can be represented as a combination of the basic three neo-Riemannian transforms:

- S (Slide): *LPR*, example S(Cmaj) = C#min
- H (Hexatonic): *LPL*, example H(Cmaj) = A-min
- N (Nebenverwandt): *RLP*, example N(CMaj) = Fmin

The implementation computes these in a single transform step however, not sequentially.



Controls

Control	Function
Left encoder (turn)	Select cell in the grid — the 25 cells are accessed sequentially, row-wise.
Left encoder (press)	Toggle between editing overall grid settings or current cell settings
Left encoder (long)	Clear/reset grid (the results of this action depends on `Clr` setting)
Right encoder (turn)	Navigation mode: move up and down through the menu items. Edit mode: increase or decrease the value being edited.
Right encoder (press)	Toggle between menu navigation (selection) mode and value editing mode
Right encoder (long press)	App selection menu
Up button	Reset current cell position to grid origin (top left-hand corner of grid)
Down button	Increment grid position by one clock step

Grid settings

Setting	Meaning
`dx`	Amount of movement along x-axis (horizontal) per clock input
`dy`	Amount of movement along y-axis (vertical) per clock input
`Mode`	Musicological mode of root triad, either `maj` or `min`
`Oct`	Move outputs up/down in octave steps
`TrDly`	sets the TR1-input-to-processing latency (for details see the `Trigger Delay` setting in *Quantermain* below)

Setting	Meaning
`OutA`	Switch output mode of channel A: `root` outputs root note, `trig` outputs a trigger whenever the triad output on B, C and D is transformed, `arp` arpeggiates the current triad, `strm` (strum) arpeggiates the triad once only as soon as the triad transformation has taken place (tip: very useful with the Mutable Instruments Elements or Rings modules, or Mutable Instruments Braids in “PLUK” mode)
`Clr`	Sets how the grid is cleared on a long-press of the left encoder. `zero` clears the grid, `rT` fills with random transforms, `rTev` sets each cell’s event to `randT`

Per-cell settings

Setting	Meaning
`Trfm`	Determines the transform which is applied when this cell is active; special values are `@` (reset) and `*` (no transform)
`Offs`	Offset in semitones applied while this cell is active
`Inv`	Inversion of the transformed triad
`Muta`	Mutation event that is applied when the cell is left (i.e. on the next clock after the cell’s transform is applied). Valid values are shown in the table below. Note that this setting makes the grid self-modifying as the current cell traverses it!

`Muta` setting	Action
`none`	nothing happens
`rT__`	The transformation for this cell is set to a random value.
`r_O_`	The transposition for this cell is set to a random value.
`rTO_`	The transformation and the transposition for this cell is set to a random value.
`r__I`	The inversion for this cell is set to a random value.
`r_OI`	The transposition and the inversion for this cell is set to a random value.
`rTOI`	The transformation, the transposition and the inversion for this cell is set to a random value.

Input/output assignment

I/O	Function
TR1	Clock/trigger input to increment steps in the sequencer
TR2	Arpeggiator clock (if mode is `arp` or `stem`)
TR3	
TR4	If high, inhibits arpeggiator clock
CV1	The voltage on this input is quantised to the root note of triad (before transform) - that is, it provides external voltage control of the root note (same as Harrington 1200)
CV2	
CV3	
CV4	Modulate triad inversion (same as Harrington 1200)
A	Depending on the `OutA` setting: pitch CVs for quantised root note, arppegio/strum, or trigger out
B, C, D	Pitch CVs for the triad after transformation

Screensaver display

Similar to Harrington 1200 app, the current triad (output as pitch CVs on the B, C and D sockets) is shown graphically on a pitch circle on the left. On the right, the last few vector moves are shown as a “snake”. The current output triad is also displayed.

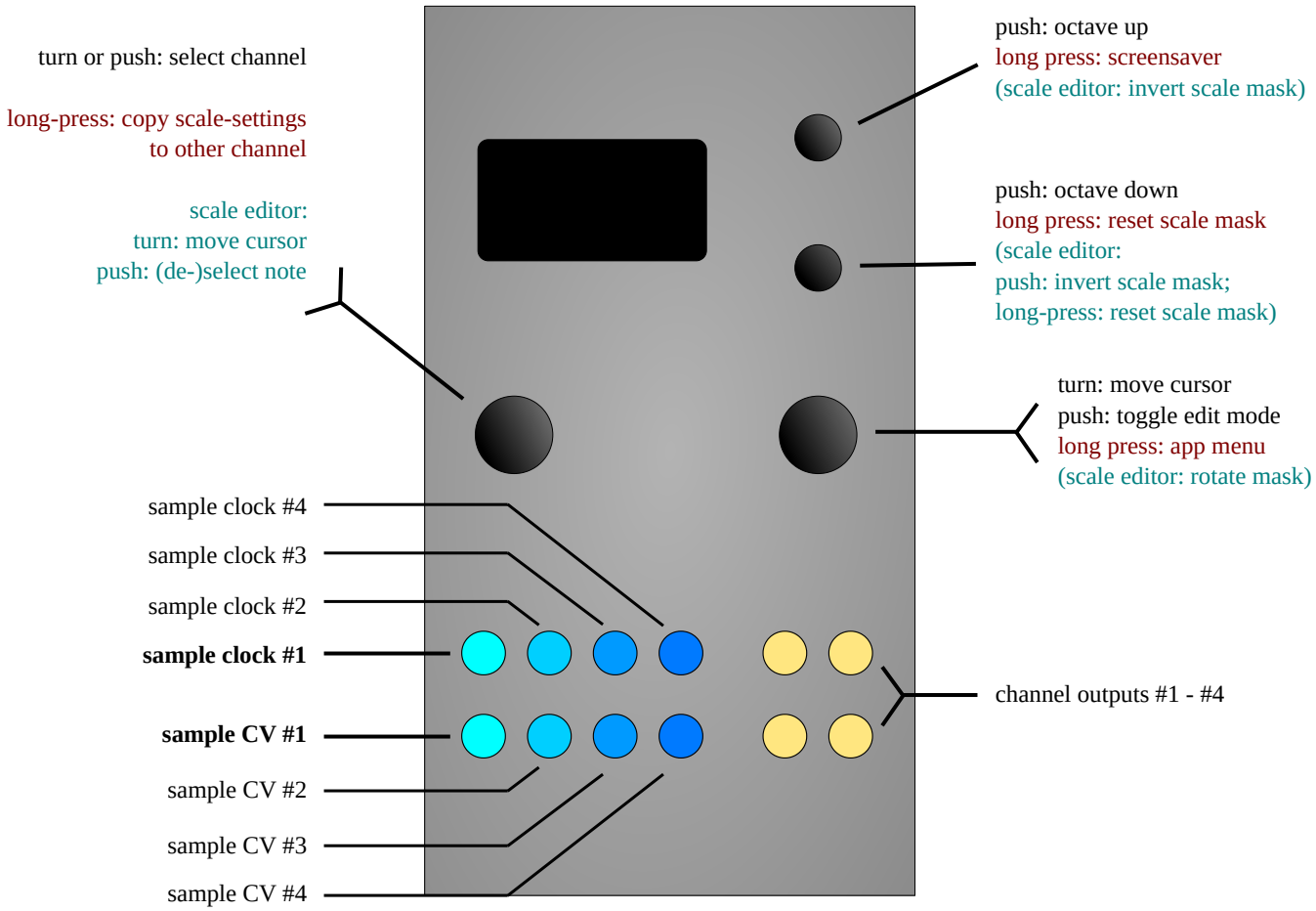
Tips

If you wish to use the vector sequencer to play melodies, then set the TRFM (transform) value for every cell in the grid to * (null transform), and set the `Offs` value for each cell to a specific note offset from the root note that you want to appear in your sequence. Then, as the current cell is moved around the grid by clock/trigger inputs on TR1, the note defined for that cell will be output on output B (with transpositions of the same note sequence on outputs C and D).

Quantermain

4 channel quantizer, integrates the quantiser from the Mutable Instruments Braids (https://github.com/pichenettes/eurorack/blob/master/braids/quantizer.h) module, but expands it with **interactive scale-edit** functionalities (99 pre-defined scales (editable in the source code - see instructions here (/custom-scales/) - are included, plus 4 fully user-definable scales (with a maximum of 16 notes per octave and additional finetune/microtonal edit options)).

- for further information re **custom scales, microtonal scales and tunings** see here (http://ornament-and-cri.me/custom-scales/) and here.
- if properly, calibrated, the **output accuracy** should be fairly excellent (is fairly excellent) due to the use of the precision TI 16-bit DAC; the **trigger-to-quantised-output latency** is also very decent: < 100 microseconds.
- the four channels are fully independent, but can be slaved to the same clock and/or track the same CV sources, if desired. here’s the default mapping:



Input/output assignment

I/O	Function
TR1, TR2, TR3 & TR4	Mappable as the trigger input for each channel A to D, independently (except when `Trigger source` is set to `cnt+` or `cnt-` — see below)
CV1, CV2, CV3 & CV4	Mappable as the external CV input to be quantised for each channel A to D, independently, or can be mapped to control various parameters when internal CV “sources” (e.g. built-in Turing Machines) are used.
A	Output voltage for channel A (quantised if quantisation is enabled)
B	Output voltage for channel B (quantised if quantisation is enabled)
C	Output voltage for channel C (quantised if quantisation is enabled)
D	Output voltage for channel D (quantised if quantisation is enabled)

Controls

Control	Function
Left encoder (turn)	Select channel, or note in scale in scale edit mode
Left encoder (press)	Increment channel by one
Right encoder (turn)	Navigation mode: move up and down through the menu items. Edit mode: increase or decrease the value being edited.
Right encoder (press)	Toggle between menu navigation (selection) mode and value editing mode
Right encoder (long press)	App selection menu

Control	Function
Up button	Transpose up one octave for currently displayed channel
Down button	Transpose down one octave for current displayed channel
Up button long	screen saver short cut
Down button down	reset scale mask

Available settings (per-channel)

Setting	Meaning
`scale`	Current scale
`Root`	Root note for scale
`Active notes`	“scale mask” / active note pattern in the selected scale
`CV Source`	external CV source (CV1 to CV4) or internal value sources (Turing Machine or Lgstc (logistic map) or `ByteB` (byte beats) or `IntSq` (integer sequences) for the current channel
`CV aux >`	auxiliary CV destination: `root` , `oct` , `trns` , or `mask` (enabled only when using a **non-default** CV setting for `CV Source` is used, e.g. if `CV2` is selected as the source for channel A, then `CV1` can be re-mapped using `CV aux` . `CV aux` is not enabled if any of the internal pseudo-CV sources (Turing Machine/LFSR, byte beats etc) are selected).)
`Trigger source`	Trigger input source (TR1 to TR4, or `cnt+` or `cnt-` for continuous quantisation, see below for discussion of continuous quantisation) for the current channel
`Clock div`	Clock divider for the trigger input selected for the current channel (note: the same trigger source may have different clock dividers set in each channel)
`Trigger delay`	sets the delay between receiving a trigger or gate (rising-edge) on the chosen trigger input for the current channel, and the input voltage being sampled. It defaults to Off (no delay beyond the normal latency, which is less than 100 microseconds, typically about 50 microseconds), but available values are 120us, 240us, 360us, 480us, 1ms, 2ms and 4ms. (†)
`Transpose`	Transpose up or down of output CV for current channel (in scale-degrees)
`Fine`	Fine tuning control, adjusts the pitch CV for the current channel up or down in tiny increments.

- (†) o_C is very fast, processing signals in the audiorate, which (in triggered/clocked quantisation mode) can easily produce a **mismatch** between the voltage that you *intend* to sample and the voltage that **actually** is being sampled (that is, used as an input to the quantizer). For example, if a sequencer or some other stepped voltage source is being sent to the quantizer, then that voltage may still be slewing to its new intended voltage when o_C acquires the samples; or if this source is digital, there may be a few tens of microseconds delay before the voltage source reacts to the trigger signal (assuming it is the same trigger signal that has been sent to o_C) and starts to slew to the new voltage. If this is the case, the *Quantermain*, *CopierMaschine*, *Harrington 1200*, *Automatonnetz*, *Meta-Q*, *Sequins* or *Acid Curds* apps will be sampling/quantizing the **previous** note / step, or they may be sampling the CV as it is slewing to the new value. In such cases, increase the `Trigger delay` setting to compensate for these differences in timing. NB: These considerations only apply when using *Quantermain* in trigger mode and to the *CopierMaschine* app, which can only be used with an external trigger) — they do not apply when `Trigger Source` is set to `cnt+` or `cnt-` (= continuous quantization) in *Quantermain*.

Method of operation

- When used as a voltage quantiser, for each of the four independent channels, the Quantermain app reads an input voltage from the CV source set for that channel and quantises it to the scale and notes set for that channel, and then applies any post-quantisation transpositions that have been specified. Voltages are read an a new quantised voltage is output when the trigger source specified for that channel fires (rising edge of the trigger, clock or gate signal), unless `Trigger source` is set to `cnt+` or `cnt-`, in which case quantisation is performed continuously on the input voltage. The latency from receipt of a trigger to output of a newly quantised note is under 100 microseconds, and is typically about 50 microseconds. In continuous quantisation mode, the input voltage is read at an effective rate of about 1 kHz, thus quantisation of the input occurs approximately once every millisecond.
- When `CV source` is set to an input channel **other** than the default/nominal input (e.g. channel #2 is set to `CV source = CV4`), the default input can be re-mapped to a different channel parameter. See `CV aux >`.
- When the `CV source` is set to `Turing`, `Lgstc`, `ByteB` or `IntSq`, instead of reading a voltage on one of the CV inputs (`CV1` to `CV4`), an internally generated value is used instead (for that channel - you can use the internal sources on some channels and external CV sources on others if you wish). Details of each of these internal sources follow.

Note: the Turing Machine, Logistic Map, byte beat and integer sequences sources all require external clock inputs to make then step, and thus they do not operate when `Trigger source` is set to `cnt+` or `cnt-`.

The internal Turing Machine source

The `Turing` source uses a linear-feedback shift register (LFSR) randomly seeded with habit pattern, which is right-shifted by one bit at each clock step (read from the `Trigger source` for that channel). There is a settable probability that the least-significant bit is randomly each time the pattern is shifted by one. This arrangement was popularised in modular synthesis as the Richter Noise Ring and the Music Thing Turing Machine. The name of the latter module has been borrowed here (see acknowledgements (http://ornament-and-cri.me/acknowledgements/)).

If a Turing Machine has been selected as as the `CV Source`, then the following additional settings are made available:

Setting	Meaning
`LFSR length`	length of the linear feedback shift register, in bits, range 2 to 32
`LFSR modulus`	sets the modulus for the integer value output from the LFSR. The value of the integer from the LFSR is divided by the modulus and the remainder is used. For example, if the modulus is 8 and the current integer value from the LFSR is 19, then the remainder of 19 - (2 x 8) i.e. 3 is used as the value. In other words, values “wrap around” at the modulus setting value — it sets a maximum note range for the LFSR, similarly to the `LFSR range` setting, but `LFSR range` compresses or expands the range of notes for a given integer value from the sequence, whereas the modulus wraps the values around. **Note**: the `LFSR modulus` setting and any external voltage input as set by `LFSR mod CV src` have **no effect** on the LFSR is no `scale` is selected.
`LFSR range`	the range or span of notes available to the LFSR (Turing Machine) from which a note is selected
`LFSR prb`	probability that the least significant bit will be flipped when it to copied

Setting	Meaning
`LFSR p CV src`	(v1.1 or later only) The control voltage source (none, or CV1, CV2, CV3 or CV4 inputs) used to control the probability. The CV value is added to the probability value set via the `LFSR p` menu item (see above).
`LFSR mod CV src`	(v1.2 or later only) The control voltage source (none, or CV1, CV2, CV3 or CV4 inputs) used to control the modulus of the values output by the LFSR (Turing Machine) from which a note is selected. The CV value is added to the LFSR modulus value set via the `LFSR modulus` menu item (see above).
`LFSR rng CV src`	(v1.1 or later only) The control voltage source (none, or CV1, CV2, CV3 or CV4 inputs) used to control the range or span of notes available to the LFSR (Turing Machine) from which a note is selected. The CV value is added to the range value set via the `LFSR range` menu item (see above).

The internal Logistic Map source

If a Logistic Map has been selected as as the `CV Source`, then the following additional settings are made available:

Setting	Meaning
`Logistic r`	The ***r*** coefficient for the logistic map equation (https://en.wikipedia.org/wiki/Logistic_map)
`Logistic range`	The range or span of notes available to the Logistic Map from which a note is selected.
`Log r CV src`	(v1.1 or later only) The control voltage source (none, or CV1, CV2, CV3 or CV4 inputs) used to control the r coefficient for the logistic map equation (https://en.wikipedia.org/wiki/Logistic_map). The CV value is added to the ***r*** value set via the `Logistic r` menu item (see above).
`Log rng CV src`	(v1.1 or later only) The control voltage source (none, or CV1, CV2, CV3 or CV4 inputs) used to control the range or span of notes available to the Logistic Map from which a note is selected. The CV value is added to the range value set via the `Logistic range` menu item (see above).
`Logistic seed`	The seed value to initialise the logistic map (has very little effect, but different values result in different sequences)

The internal Byte Beats source

If `ByteB` (byte beats) has been selected as as the `CV Source`, then the following additional settings are made available:

Setting	Meaning
`Bytebeat eqn`	sets the byte beat equation used as the source. See the *Viznutcracker, sweet!* app documentation for more details of the currently available equations.
`Bytebeat range`	The range or span of notes available to the byte beat equation source from which a note is selected.
`Bytebeat P0`	Parameter 0 for the byte beat equation - see the *Viznutcracker, sweet!* app documentation for more details.
`Bytebeat P1`	Parameter 1 for the byte beat equation - see the *Viznutcracker, sweet!* app documentation for more details.
`Bytebeat P2`	Parameter 2 for the byte beat equation - see the *Viznutcracker, sweet!* app documentation for more details.
`Bb eqn CV src`	The CV input source to vary the byte beat equation used.

Setting	Meaning
`Bb rng CV src`	The CV input source to vary the byte beat range.
`Bb P0 CV src`	The CV input source to vary byte beat equation parameter 0.
`Bb P1 CV src`	The CV input source to vary byte beat equation parameter 1.
`Bb P2 CV src`	The CV input source to vary byte beat equation parameter 2.

The internal Integer Sequences source

The `IntSq` source in *Quantermain* behaves similarly to the `IntSq` source in *CopierMaschine* — full details are given in the *CopierMaschine* section above. The same integer sequences are available, but in *Quantermain*, up to four channels of them can be used independently and simultaneously.

Setting	Meaning
`IntSeq`	sets the integer sequence used as the source. See the *CopierMaschine* section for a list of available integer sequences and their characteristics.
`IntSeq modul`	sets the modulus for the integer sequence. The value of the integer from the integer sequence is divided by the modulus and the remainder is used. For example, if the modulus is 8 and the current integer value from the sequence is 19, then the remainder of 19 - (2 x 8) i.e. 3 is used as the value. In other words, values “wrap around” at the modulus setting — it sets a maximum note range for the integer sequence, similarly to the `IntSeq range` setting (see next row of this table), but `IntSeq range` compresses or expands the range of notes for a given integer value from the sequence, whereas the modulus wraps the values around.
`IntSeq range`	sets the span or range of notes created by the integer sequence. It essentially compresses or expands the mapping from integer values to notes. Unlike `IntSeq modul`, it does cause the note values to “wrap around”. Both `IntSeq range` and `IntSeq modul` can be used together.
`IntSeq dir`	sets whether the integer sequence loops back to the beginning when it gets to the end, or whether it swings back like a pendulum and plays in reverse when it reaches the end. The “end” is the last step in the sequence, as defined by the sequence start plus the sequence length settings.
`IntSeq start`	sets the start point in the stored integer sequence. The stored sequences are 128 steps long, and the maximum start point is 126 to ensure a minimum sequence length of 2.
`IntSeq len`	sets the length of the integer sequence. Thus a length of 16 will use just 16 values from the stored 128 step sequence, starting at the step specified by `IntSeq start`.
`IntSeq FS prob`	sets the probability (0 means *p*=0, 255 means *p*=1) that the `Fractal stride` value will be randomly altered at the next step. See `Fractal stride` in this table.
`IntSeq FS rng`	sets the range of the probabilistic shift in the fractal stride controlled by `IntSeq FS prob`. The shift range is from 0 to 5.
`Fractal stride`	Two of the sequences are fractal in nature, and the “stride” setting sets how many steps are advanced on each trigger input. This also works with the non-fractal sequences and provides additional variation, particularly if the stride is not an exact divisor of the sequence length.
`IntSeq CV`	configures external CV control over the integer sequence used. The available choices are `None`, `CV1`, `CV2`, `CV3` or `CV4`.
`IntSeq mod CV`	configures external CV control over the integer sequence modulus. The available choices are `None`, `CV1`, `CV2`, `CV3` or `CV4`.

Setting	Meaning
`IntSeq rng`	configures external CV control over the integer sequence range. The available choices are `None`, `CV1`, `CV2`, `CV3` or `CV4`.
`Frctl stride CV`	configures external CV control over the integer sequence fractal stride. The available choices are `None`, `CV1`, `CV2`, `CV3` or `CV4`.
`IntSeq reset`	configures external trigger control over reset of integer sequence pointer. The available choices are `None`, `TR1`, `TR2`, `TR3` or `TR4`. A positive trigger (or rising edge) received on the selected trigger input will cause the integer sequence counter to return to the start position of the sequence (as determined by `IntSeq start`).

Active note (scale mask) and scale editing

To invoke the active note and scale editor, click the right encoder when the *Active notes* menu option is selected.

- turn the left encoder to move the note cursor, click the left encoder to toggle currently selected note on or off
- Up/Down buttons invert the current note mask
- turning the right encoder note-shifts the pattern left or right (= rotate scale mask)
- click the right encoder to exit

Using the note editor (user-scales 1-4):

- use the right encoder to scroll to the top of the menu, and click the right encoder to select the scale. Choose one of the 4 user-editable scales (`USER1`, `USER2`, etc) at the beginning of the list of scales. Click the right encoder again to return to menu scrolling mode.
- Scroll down to *Active notes* and click the right encoder to invoke the note editor.
- turn the left encoder to move to the note you wish to edit. **Hold down the left encoder, and use the right encoder to edit the pitch value for that note**.
- **the length of the scale** can be set by navigating to the final note in the scale, and turning the right encoder to shorten or lengthen the scale. In this way, scales with **4 up to 16 microtonal notes** are possible.
- click the right encoder to exit the note editor
- user-defined scales are saved along with all the other settings, when the save settings procedure is used

Note

The user-defined scales only save the note values in a scale - that is, the number of notes in the scale and the pitch value for each of those notes. The user scales don't save the note masks applied to those scales. Note masks **are** stored when you save settings, but they are not stored with the user-defined scales because note masks also apply to the pre-defined scales.

The user-defined scales were really intended for defining microtonal or non-equally-tempered tunings, where the notes are not spaced 100 cents apart (or at multiples of 100 cents).

However, you can also use the user-defined scales for defining any subset of, say, the "normal" 12-tone equally-tempered (12-TET) chromatic scale. It is a bit fiddly, but doable. In the note mask editor, with a user scale selected, first note down with a pen and paper the pitch values for the 12-TET notes you want in your scale. Note that the values are not in cents, they are in 1/1536 divisions of the octave. Then adjust the number of notes in the user-defined scale using the lozenge at the extreme right of the scale in the editor. Then go to each note in turn in your user-defined scale and adjust the pitch value for it to match the values you wrote down. The notes must be in

ascending order! Exit the note mask editor, save you settings and then the user-defined scale should be that subset of notes you just defined. You can still apply note masks on top of that to further subset the notes in the scale if you wish.

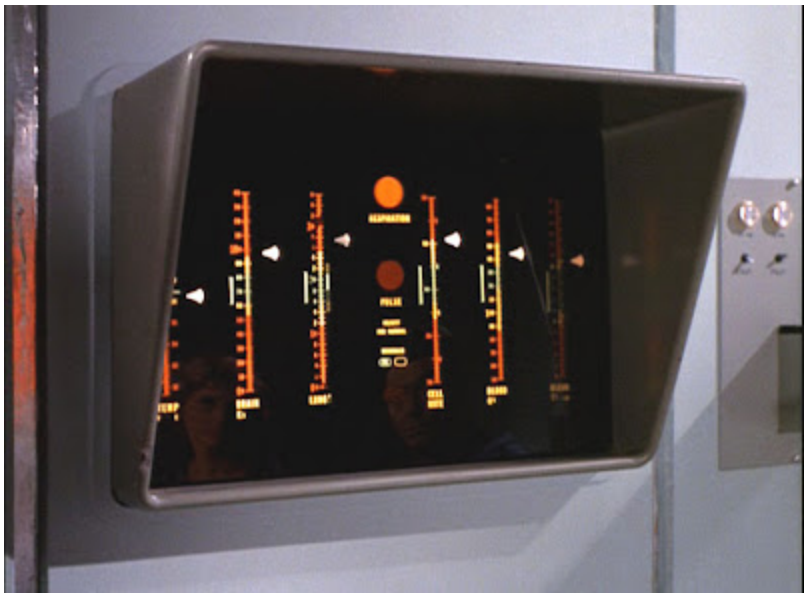
Continuous (non-triggered) quantisation modes

When `Trigger source` for a channel is set to `cnt+` or `cnt-`, quantisation is performed on that channel continuously (well, about 16,667 times per second). In addition, the trigger input which ordinally corresponds to that channel (i.e. TR1 for channel A, TR2 for channel B, TR3 for channel C and TR4 for channel D) can then be used as a gate to transpose the output for that channel up by one octave (if `cnt+` is set) or down by one octave if `cnt-` is set. In other words, for channel B, if the `Trigger source` for it is set to `cnt+`, then if TR2 input is high, the output for that channel will be transposed up by one full octave, while the TR2 input remains high. Likewise, if `Trigger source` is set to `cnt-`, then channel B output will be transposed down one octave while TR2 is held high. Note, however, that the octave transpositions only take effect when there is a note change i.e. when the CV input for that channel has changed sufficiently that it quantises to a new note value. That way, octave transpositions co-incide with note changes, which sounds much better.

Note also that although the trigger inputs used by each channel can be mapped independently, the `cnt+` and `cnt-` octave transposition behaviour cannot be mapped - it is hard-coded, so that only TR1 can be used for this purpose for channel A, only TR2 for channel B and so on. Note that TR1 can still also be used as a trigger for any other channel, but not for the `cnt+` and `cnt-` octave transposition behaviour on other channels.

Screensaver display

The screensaver for *Quantermain* is inspired, possibly on an unconscious level, by the Sick Bay vital signs display in the original *Star Trek* series.



There are four “lanes”, one for each channel. In each lane, short lines representing the quantised pitch (on a semitone scale) scroll leftwards. The small triangles to their right move up and down to indicate the octave for that channel. Triggers (a vertical bar) and input voltages (a horizontal bar, left-going for negative voltages, right-going for positive) are indicated above each channel lane (replaced by a bit-pattern display when the sources are set to LFSR (Turing Machine) or logistic map, rather than external CV).

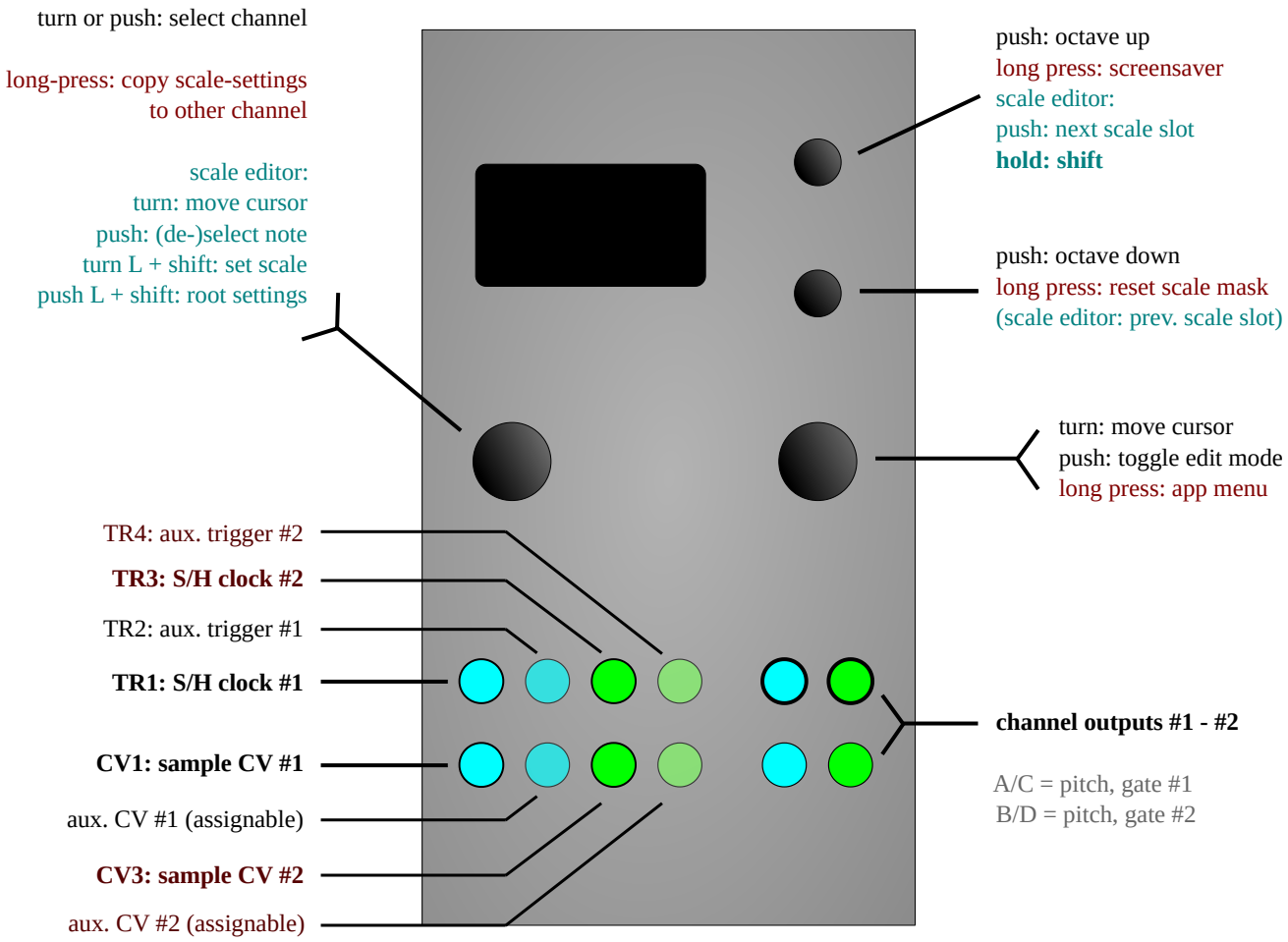
Tips

You can disable quantisation on each channel by setting the scale to `Off`. When in clocked mode (that is, `Trigger source` is set to TR1, TR2, TR3 or TR4), then that channel acts as a traditional sample-and-hold (S&H), without any voltage droop, of course. Thus, Quantermain can act as a quad S&H if you want. The only limitation is that the effective maximum sample rate is about 1 kHz, even if the clock/trigger signal you give it is at a higher frequency. This is because the ADCs are only read at an effective rate of about 1 kHz.

Disabling quantisation when using a Turing Machine, Logistic map, byte beats or integer sequence internal source allows the output for that channel to act as a semi-random modulation source.

Meta-Q

Meta-Q is a dual-channel quantiser that's basically similar to *Quantermain*, except, of course, there are only two channels. It has fewer internal CV sources (currently, only LFSR. See *Copiermaschine*/*Quantermain* for details), but comes with four 'scale slots' per channel — a package of scale, mask, root and transpose values —, which themselves can be **sequenced** to provide a lot of additional, structured variation to the input CV sequences (or the LFSR internal CV source).



scale 'slots' / seq mode:

- Each scale ‘slot’ (= `scale #`) can be mapped to any of the available preset and user scales and/or scale masks (via the main menu or scale editor). The scale slots can also be assigned their own root and transpose settings.
- The scale editor in *Meta-Q* thus is slightly more complex than in the other modes: it allows you to edit the four scales-slots (not just one scale) from within the editor, ie both scale as well as the root and transpose values. In basic use, it works much the same as in *Quantermain* or *Copiermaschine*; notably, the `up` button behaves slighlty differently, however: in the *Meta-Q* scale editor it assumes a ‘shift’ functionality to access the (slot-specific) root and (diatonic) transpose settings (details see below).
- What is more, said four scale-slots can be sequenced (or just toggled) by using a clock signal and/or by modulating the slot-parameter with a control voltage:
    - Use the auxiliary trigger inputs **TR2** resp. **TR4** to cycle through either 2, 3, or 4 *adjacent* scale slots: see the `seq mode` setting. Available settings are **TR+1** (= plus next scale), **TR+2** (= plus next two scales), **TR+3** (= plus next three scales). Things wrap around slot #4. Thus, for example, when selecting `scale #` = 3 and `seq mode` = TR+2 (= scale #3 plus next two scales), when clocked via the aux. trigger input, the scale slots will step through the following sequence: #3, #4, #1, #3, #4, #1, #3, #4, #1, #3, #4, #1, #3 …
    - Setting the `CV aux.` parameter to `scl#` (= scale #) allows modulation of the slot-parameter with a CV signal applied to CV2 (channel #1) or CV4 (channel #2). (Alternatively, **CV aux.** can be routed to root, mask (= scale mask), trns (= transpose/scale degrees), or oct (transpose/octaves)).
    - **NB**: inputs TR2 and CV2 are hardcoded to service channel #1, TR4 and CV4 to service channel #2.

aux. outputs (C, D):

- when `trigger source` = TR1 - TR4, the **aux. outputs** (C, D) simply pass through the main trigger signal, with adjustable pulse-width (`--> pw`); when set to continuous (`cnt`), the aux. output goes high if/when the note **changes** (= trigger-on-note-change).
- the aux. outputs can alternatively output a transposed copy of the main channel CV (aux.output = `copy`), or output said CV, delayed by one clock (aux.output = `asr`).

Inputs and outputs

I/O	Function	-
TR1	clock input #1	-
TR2	aux. / scale sequencer clock #1	-
TR3	clock input #2	-
TR4	aux. / scale sequencer clock #2	-
CV1	sample in # 1	-
CV2	(mappable)	-
CV3	sample in # 2	-
CV4	(mappable)	-
A, B	CV outputs #1, #2	-
C, D	aux outputs #1, #2 (default to gate output)	-

Available settings (per-channel)

Setting	Meaning
`scale`	selected scale
`--> edit`	edit scale mask (for details see *Quantermain* and below)
`seq mode`	number of slots (-, 2, 3, or 4) to advance scale sequencer before reset (via `TR2` resp. `TR4`)
`scale #`	selected scale slot (`#1 - #4`)
`root #n`	root note for scale slot n
`transpose #n`	offset (in scale degrees) for scale slot n
`octave`	offset in octaves (for all scale slots)
`CV source`	sample input, internal/external (`CV1 - CV4`, `LFSR`)
`CV aux.`	auxiliary CV input destination: `scale #`, `root`, `transpose`, `octave`, `mask`
`trigger source`	main trigger source: `TR1 - TR4`, `cnt+`, `cnt-` (†)
`--> latency`	trigger-to-quantization latency (default: `0` = 60 us). Note: this is equivalent to the `Trigger delay` setting in several of the other apps, please see the *Quantermain* app documentation for a discussion of this setting.
`aux.output`	aux channel output: `gate`, `copy`, `asr` (††)
`--> pw`	pulse-width of triggers at C/D outputs (`gate` mode)
`--> aux +/-`	pitch offset at C/D outputs (in octaves) (`copy` and `asr` modes)
`> LFSR length`	length of shift register (when in LFSR mode)
`> LFSR p`	probability of flipping (when in LFSR mode)
`> LFSR range`	output range (when in LFSR mode)
`> LFSR CV`	destination of `CV1` resp. `CV3` (when in LFSR mode)
`> LFSR TRIG`	aux channel output (when in LFSR mode): `echo`, `lsb`, `chng`: clock through, track LSB, and output on note-change

- notes:
  - (†) `cnt+` and `cnt-` = continuous quantization; in this case, a gate applied to `TR1` resp. `TR3` will shift the pitch up (`cnt+`) or down (`cnt-`) by one octave.
  - (††) `copy` simply duplicates the main channel output to the aux channel output. `asr` does the same, but delayed by one clock.

Controls:

main menu

Control	Function
Left encoder (turn)	select channel
Left encoder (press)	toggle channel
Left encoder (long press)	copy selected scale/mask to *all* channels/slots
Right encoder (turn)	Navigation mode: move up and down through the menu items. Edit mode: increase or decrease the value being edited
Right encoder (press)	Toggle between menu navigation (selection) mode and value editing mode

Control	Function
Right encoder (long press)	app selection menu
Up button	Transpose up: one octave (parameter menu)
Down button	Transpose down: one octave (parameter menu)
Up button (long press)	invoke screensaver
Down button (long press)	reset scale mask

scale editor

- works **mostly** the same (as in the other quantising modes). Notably, the `up` and `down` buttons behave slighlty differently, however: pushing `up` will point the editor to the next scale-slot, pushing `down` will point it to the previous scale-slot. This way, you can edit all the scale-slots more easily from within the editor.
- In the editor, you can also change the scale-type, root, and transpose settings: to do so, **hold** the `up` button (= 'shift') while turning the **left** encoder: that'll change the scale, just like in the regular menu. to edit the root and transpose values, **hold** the `up` button, then push the left encoder (= 'shift' + push left): that'll open a new window, displaying the slot's root and transpose values. you can select which parameter to edit by using the left encoder; adjust the value by turning the right encoder; or use the up / down buttons to advance to a different slot. push the left encoder again to return to the basic scale editor.

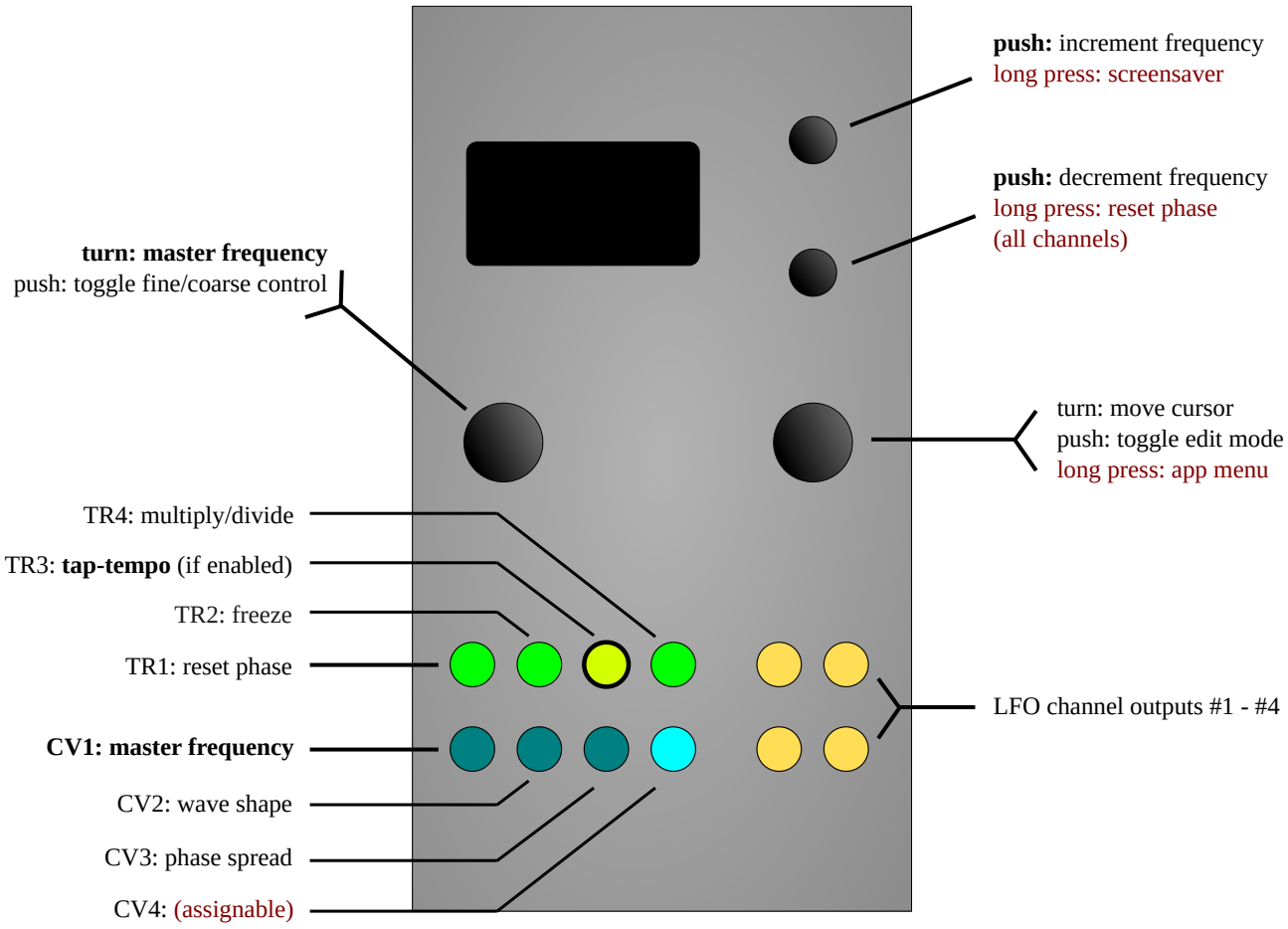
Control	Function
Left encoder (turn)	select note
Left encoder (push)	activate/de-active note
Right encoder (turn)	rotate mask
Right encoder (push)	exit editor
Up button (push)	**go to next scale slot**
Down button (push)	**go to previous scale slot**
Up button (hold) + Left encoder (turn)	**shift: select scale**
Up button (hold) + Left encoder (push)	**shift: toggle root/transpose view**
Right encoder (long press)	– (app selection menu)
Up button long press	– (screensaver)

Quadraturia

Quadraturia is a port of of the quadrature wavetable LFO that is available as the "Easter egg" in the Mutable Instruments Frames (https://github.com/pichenettes/eurorack/tree/master/frames) module. However, *Quadraturia* adds voltage control over three of the four LFO parameters (as well as CV control of frequency/rate of the LFO, as in Frames). Because the background ISR of o_C runs at 16.7KHz instead of 32KHz as in Frames, the behaviour of *Quadraturia* may not be identical to the Frames Easter egg, but it should be very close, and is nonetheless a very useful and flexible source of modulation voltages. For more details of how the Frames Easter egg LFO behaves, see the relevant section of the Frames Manual (http://mutable-instruments.net/modules/frames/manual).

Quadraturia also incorporates the "predictive" tap-tempo capability from the Tap LFO mode in the Mutable Peaks module. Details are given in the **Tap tempo** section below.

Internally, there are four LFOs (LFO1 to LFO4), with LFO2 to 4 running at some ratio of the (master) frequency of LFO1 (by default, that ratio is one, so all the LFOs run at the same frequency). The wave shape, phase of frequency modulation of LFO2 to 4 can be changed, relative to those parameters for LFO1, using the Shape spread, phase/frequency spread and coupling controls (see table below).



I/O	Function
TR1	Reset phase (all LFOs)
TR2	Freeze (all LFOs stop in their tracks and hold current value while TR2 is high)
TR3	Tap tempo lock or trigger/gate input when used in **Tap tempo** mode.
TR4	Divide/Multiply frequency (while TR4 is high — see `TR4: MULT`)
CV1	Master frequency
CV2	Wave shape
CV3	Phase/frequency spread
CV4	(mappable) : coupling, shape spread, range, offset, amplitude modulation (a > b, b > c, or c > d)

A, B, C, DLFO channel outputs

Controls

Control	Function
Left encoder (turn)	Increase or decrease frequency of master LFO
Left encoder (press)	Toggle frequency control between Coarse and Fine control.
Right encoder (turn)	Navigation mode: move up and down through the menu items. Edit mode: increase or decrease the value being edited.
Right encoder (press)	Toggle between menu navigation (selection) mode and value editing mode

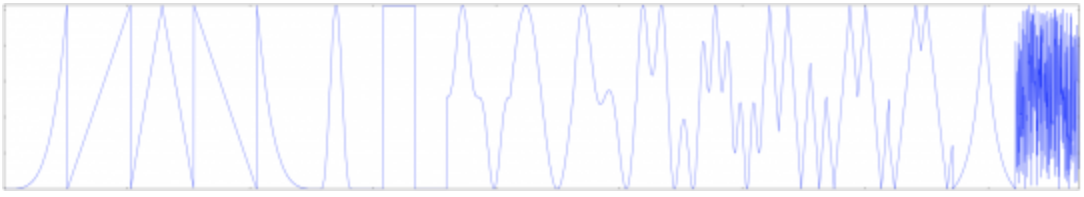
Control	Function
Right encoder (long press)	App selection menu
Up button	Increment frequency by 32
Up button long press	screen saver short cut
Down button	Decrement frequency by 32
Down button long press	reset waveform phase to zero on all channels

Settings

Setting	Meaning
`Coarse` or `Fine` (frequency)	When in normal mode (**not tap tempo mode**), frequency is adjused with the left encoder. The frequency is displayed above the line in the menu. Clicking on the left encoder toggles between (C)oarse and (F)ine frequency adjustments. There are 256 steps for coarse adjustment, and each unit of the coarse adjustment is divided into 256 steps for fine adjustment. If **tap tempo** mode is enabled, the frequency adjustment using the left encoder is disabled.
`Tap tempo`	enables or disables **tap tempo** mode. The tap tempo trigger/clock input is TR3.
`Shape`	controls the shape of the primary waveform (LFO1), as shown in the preview waveform. See below for a diagram of all the waveforms in the wavetable.
`Shape spread`	sets the difference in wavetable position between each channel (and thus the differences in the wave shape for chennels B, C and D.
`Phase/frq spread`	sets the phase or frequency offset between each channel. Values greater than zero cause the phase to be offset incrementally in channels B, C and D with respect to (wrt) channel A. When `Phase/frq spread` is set to its maximum value of 127, the phase shift is 90 degrees on channel B, 180 degrees on channel C and 270 degrees on channel D. Values for `Phase/frq spread` of less than zero cause a progressive frequency shift (detune) across channels B, C and D wrt channel A, rather than a phase shift. **Note:** when frequency division or multiplication is set on channels B, C or D (see `B freq ratio` etc below), only 90 degrees of phase shift is available on each of channels B, C and D.
`Coupling`	sets the degree of phase-modulation “bleed” between each successive channel.
`Output range`	sets the overall output range for all channels. The range of this settings goes from 0 (no output) to 230, which equates to a nominal output range from about -3.5V to +6V. By default, with the `Offset` setting (see below) at zero, the output range is asymmetric. By reducing the `Output range` and then setting a positive `Offset` , the output can be shifted so that it is unipolar or otherwise offset at the level desired.
`Offset`	shifts the output on all channels up or down by up to several volts. The range is -128 to 127, with a default of zero. Note that this is an internal offset and the absolute output voltage range is constrained by the hardware to about -3.5V to +6V. Nonetheless, by using it in conjunction with reduced settings of the `Output range` parameter, it is possible to make the output entirely unipolar, or even entirely negatively unipolar, if desired. Note that the `Output` value is added directly to the value sent to the DAC, and therefore positive or negative offsets (that is, non-zero offsets) without any reduction in the `Output range` setting will result in the 16-bit values sent to the DAC overflowing and wrapping around. This will cause waveform deformations, which may be useful or interesting. To remove any such wrap-around deformation, reduce the `Output range` setting when using non-zero offsets. If `Output range` and `Offset` are left at their defaults of 230 and zero respectively, no waveform deformation will occur.

Setting	Meaning
`Freq range`	sets the frequency range for the quadrature LFOs, with self-explanatory settings of `cosm` (cosmological), `geol` (geological), `glacial` (glacial), `snail`, `sloth`, `lazy` (very lazy), `lazy`, `vslow` (very slow), `slow`, `med` (medium), `fast` and `vfast` (very fast). The faster settings extend into audio range. The slowest period for one cycle of the LFO on the `cosm` setting exceeds 18 hours. Note that Quadraturia does not, and is not intended to track 1V/octave.
`B freq ratio`	sets the frequency ratio at which LFO2 (channel B) runs with respect to the master LFO (LFO1, channel A). The default is `unity`, so that it runs at the same frequency as channel A. Available ratios are 16/1, 15/1, 14/1, 13/1, 12/1, 11/1, 10/1, 9/1, 8/1, 7/1, 6/1, 5/1, 4/1, 3/1, 5/2, 2/1, 5/3, 3/2, 5/4, 1/1 (unity), 4/5, 2/3, 3/5, 1/2, 2/5, 1/3, 1/4, 1/5, 1/6, 1/7, 1/8, 1/9, 1/10, 1/11, 1/12, 1/13, 1/14, 1/15 and 1/16. 16/1 means the the frequency is multiplied by 16. 1/16 means the frequency is divided by 16, and so on.
`C freq ratio`	does the same as `B freq ratio`, except for the channel C LFO.
`D freq ratio`	*likewise*
`B XOR A`	if enabled causes the 16 bits of the channel B LFO current value to be bit-wise XORed with the bits of the channel A LFO current value before being output on channel B. Available settings are `Off`, or 1 to 8, where the number refers to the number of bit of right-shift that are carried out before bit-XORing. bit-XORing can create digital noise at audio rates, but at slower modulation rates, all sorts of interesting “toothed” or other geometric patterns may emerge - but with very sharp transitions in amplitude. Try feeding it into the cut-off of a LPF (a technique shown in a Bastl video somewhere).
`C XOR A`	if enabled causes the 16 bits of the channel C LFO current value to be bit-wise XORed with the bits of the channel A LFO current value.
`D XOR A`	if enabled causes the 16 bits of the channel D LFO current value to be bit-wise XORed with the 16 bits of the channel A LFO current value.
`B AM by A`	set the level of cross-channel amplitude modulation (AM), deaulting to none (zero), up to 127. Note that the amplitude modulation is inverted, so that higher values of the waveform in channel A result in lower amplitudes of the waveform in channel B. If the relative frequency of channel B is set at unity or lower, then a type of waveshaping occurs.
`C AM by B`	as for `B AM by A` except that the amplitude of channel C is modulated by the current output value of channel B.
`D AM by B`	as for `B AM by A` except that the amplitude of channel D is modulated by the current output value of channel C.
`CV4: DEST`	CV4 destination: `cplg` (coupling), `sprd` (shape spread), `rng` (range), `offs` (offset), `a -> b` (B AM by A), `b -> c` (C AM by B), or `c -> d` (D AM by B)
`TR4: MULT`	gated frequency division/multiplication factor (TR4): `/8`, `/4`, `/2`, `x2`, `x4`, `x8`

Waveforms in the wavetable



Graphic courtesy of Mutable Instruments (https://mutable-instruments.net/modules/frames/manual/).

Tap tempo mode

The “predictive” tap tempo facility used in the Tap LFO mode in the Mutable Peaks (https://mutable-instruments.net/modules/peaks/) module has been added to *Quadraturia* in v1.3. This allows the period of the LFO waveform output on channel A to be locked to the period of “taps” or pulses (or, in fact, the rising edge of any regular input signal of more than about 1V amplitude) input on TR3. This will synchronise to both regular clock inputs as well as regular rhythms with unequal spacing of the taps. However, it **cannot** predict the future, and thus it will **not** be able to sync to irregular rhythms or randomly-timed pulses.

The frequency/period of the output on channels B, C and D will be the same as channel A if the frequency ratio setting for those channels is 1.0 (unity). Otherwise, it will be some multiple or division of the channel A frequency. In other words, the frquency ratio settings for channels B, C and D are honoured in **tap tempo** mode as well as in normal mode.

Screensaver display

The screen is divided into quadrants, each showing a rolling display of the output values on each of channels A to D.

Tips

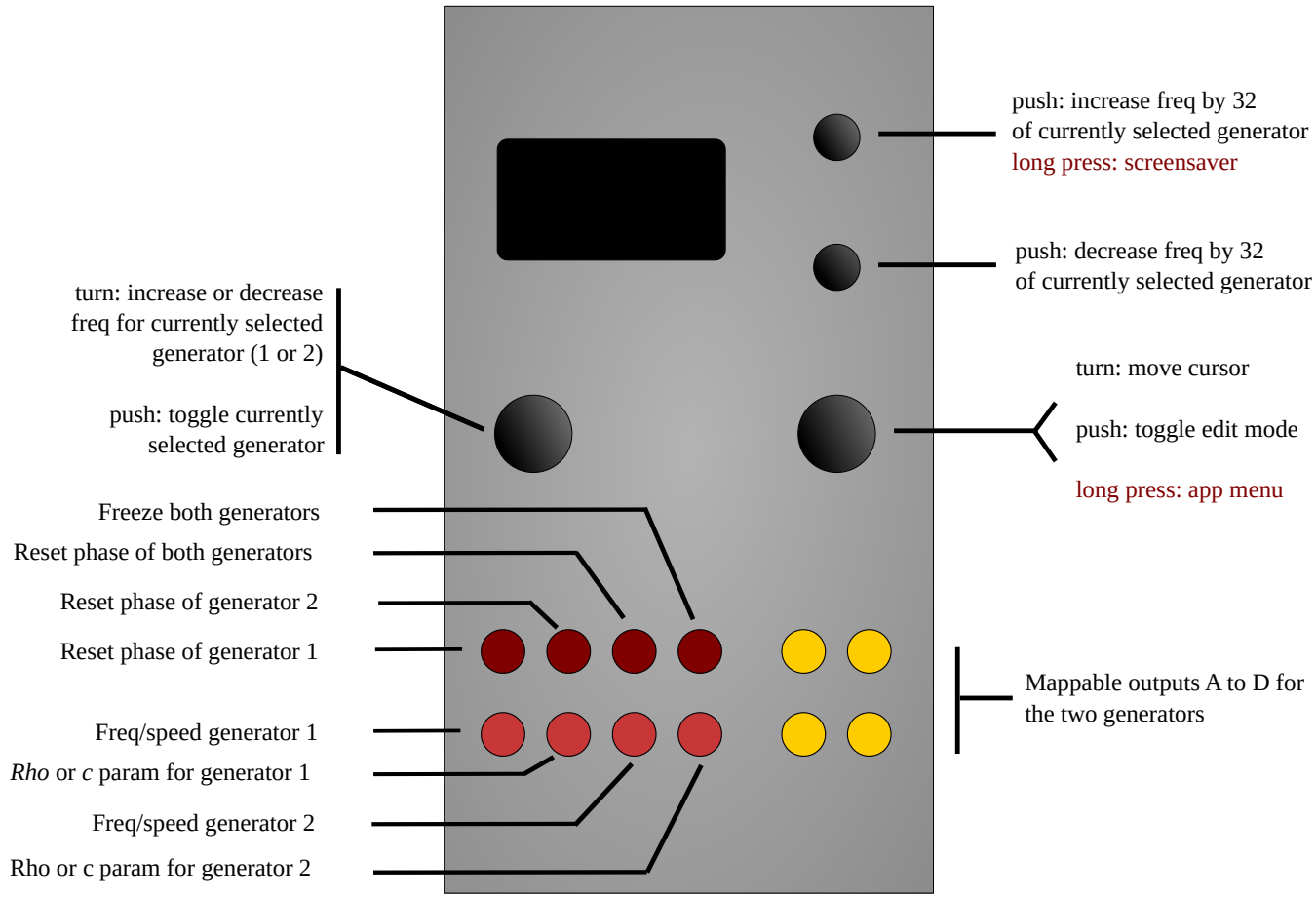
- To achieve the classic quadrature LFO patch (incremental 90° phase offset on outputs), set the phase/frq spread to +127.
- Try mixing the some of the outputs with a DC-coupled mixer (such as the Mutable Instruments Shades or Links modules) in order to create even more complex waveforms.

Low-rents

Low-rents is a port of the Lorenz attractor (https://en.wikipedia.org/wiki/Lorenz_system) modulation generator from the “Easter egg” in the Mutable Instruments Streams (http://mutable-instruments.net/modules/streams) module, to which Rössler attractors (https://en.wikipedia.org/wiki/Rössler_attractor) have been added.

Two independent function generators are provided (referred to here as Generator 1 and Generator 2), with each generator calculating *both* the Lorenz and Rössler functions simultaneously, using the same phase accumulator, but with the rate/speed of each generator independently settable. Both the Lorenz and the Rössler functions output three values (*x*, *y* & *z*), and various combinations of these can be mapped to the four output channels. The chaotic strange attractors work best as slow modulation functions.

Note that the output voltage range of the o_C module is asymmetrical (about -3V to +6V) — because it was designed to process pitch CVs. Therefore the output of the *Low-rents* app is not centred about 0V.



I/O	Function
TR1	Reset phase of generator 1
TR2	Reset phase of generator 2
TR3	Rest phase of both generators
TR4	Freeze (both generators stop in their tracks and hold current value while TR4 is high)
CV1	Frequency/speed of generator 1
CV2	Rho or c parameter for generator 1
CV3	Frequency/speed of generator 2
CV4	Rho or c parameter for generator 2

A, B, C, DMappable outputs from the two generators (see table below)

Controls

Control	Function
Left encoder (turn)	Increase or decrease frequency of currently selected generator (1 or 2)
Left encoder (press)	Toggle currently selected generator frequency control
Right encoder (turn)	Navigation mode: move up and down through the menu items. Edit mode: increase or decrease the value being edited.
Right encoder (press)	Toggle between menu navigation (selection) mode and value editing mode
Right encoder (long press)	App selection menu
Up button (long press)	invoke screensaver

Control	Function
Up button (press)	Increment frequency of currently selected generator by 32
Down button (press)	Decrement frequency of currently selected generator by 32

Settings

Setting	Meaning
`Freq 1`	Frequency/speed of generator 1, range is 0-255
`Freq 2`	Frequency/speed of generator 2, range is 0-255
`Rho/c 1`	Rho (for Lorenz attractor) or c (for the Rössler attractor) parameters for generator 1
`Rho/c 2`	Rho (for Lorenz attractor) or c (for the Rössler attractor) parameters for generator 2
`LFreq 1 Rng`	frequency/speed range of Lorenz and Rössler generator 1, ranges from ‘sloth’ to ‘fast’.
`LFreq 2 Rng`	frequency/speed range of Lorenz and Rössler generator 2, ranges from ‘sloth’ to ‘fast’.
`out A`	output mapping for output A. Available choices shown in the table below.
`out B`	output mapping for output B. Available choices shown in the table below.
`out C`	output mapping for output C. Available choices shown in the table below.
`out D`	output mapping for output D. Available choices shown in the table below.

Output mapping value	Meaning
`Lx1`	Generator 1 Lorenz attractor *x* value
`Ly1`	Generator 1 Lorenz attractor *y* value
`Lz1`	Generator 1 Lorenz attractor *z* value
`Lx2`	Generator 2 Lorenz attractor *x* value
`Ly2`	Generator 2 Lorenz attractor *y* value
`Lz2`	Generator 2 Lorenz attractor *z* value
`Rx1`	Generator 1 Rössler attractor *x* value
`Ry1`	Generator 1 Rössler attractor *y* value
`Rz1`	Generator 1 Rössler attractor *z* value
`Rx2`	Generator 2 Rössler attractor *x* value
`Ry2`	Generator 2 Rössler attractor *y* value
`Rz2`	Generator 2 Rössler attractor *z* value
`Lx1+Rx1`	Sum of Generator 1 Lorenz attractor *x* value and Generator 1 Rössler attractor *x* value
`Lx1+Rz1`	Sum of Generator 1 Lorenz attractor *x* value and Generator 1 Rössler attractor *z* value
`Lx1+Ly2`	Sum of Generator 1 Lorenz attractor *x* value and Generator 2 Lorenz attractor *y* value
`Lx1+Lz2`	Sum of Generator 1 Lorenz attractor *x* value and Generator 2 Lorenz attractor *z* value
`Lx1+Rx2`	Sum of Generator 1 Lorenz attractor *x* value and Generator 2 Rössler attractor *x* value
`Lx1+Rz2`	Sum of Generator 1 Lorenz attractor *x* value and Generator 2 Rössler attractor *z* value
`Lx1xLy1`	Bit-wise XOR of Generator 1 Lorenz attractor *x* value and Generator 1 Lorenz attractor *y* value
`Lx1xLx2`	Bit-wise XOR of Generator 1 Lorenz attractor *x* value and Generator 2 Lorenz attractor *x* value
`Lx1xRx1`	Bit-wise XOR of Generator 1 Lorenz attractor *x* value and Generator 1 Rössler attractor *x* value
`Lx1xRx2`	Bit-wise XOR of Generator 1 Lorenz attractor *x* value and Generator 2 Rössler attractor *x* value

The Rho and c parameters for the Lorenz and Rössler attractors respectively determine the degree of variability in the chaotic generator system. Note that the values have been constrained so that the functions do not collapse, but some combinations of extreme settings may cause the generator functions to collapse completely. If this happens, change the Rho/c setting and send a rest pulse to the relevant trigger input to reset the function generator.

Screensaver display

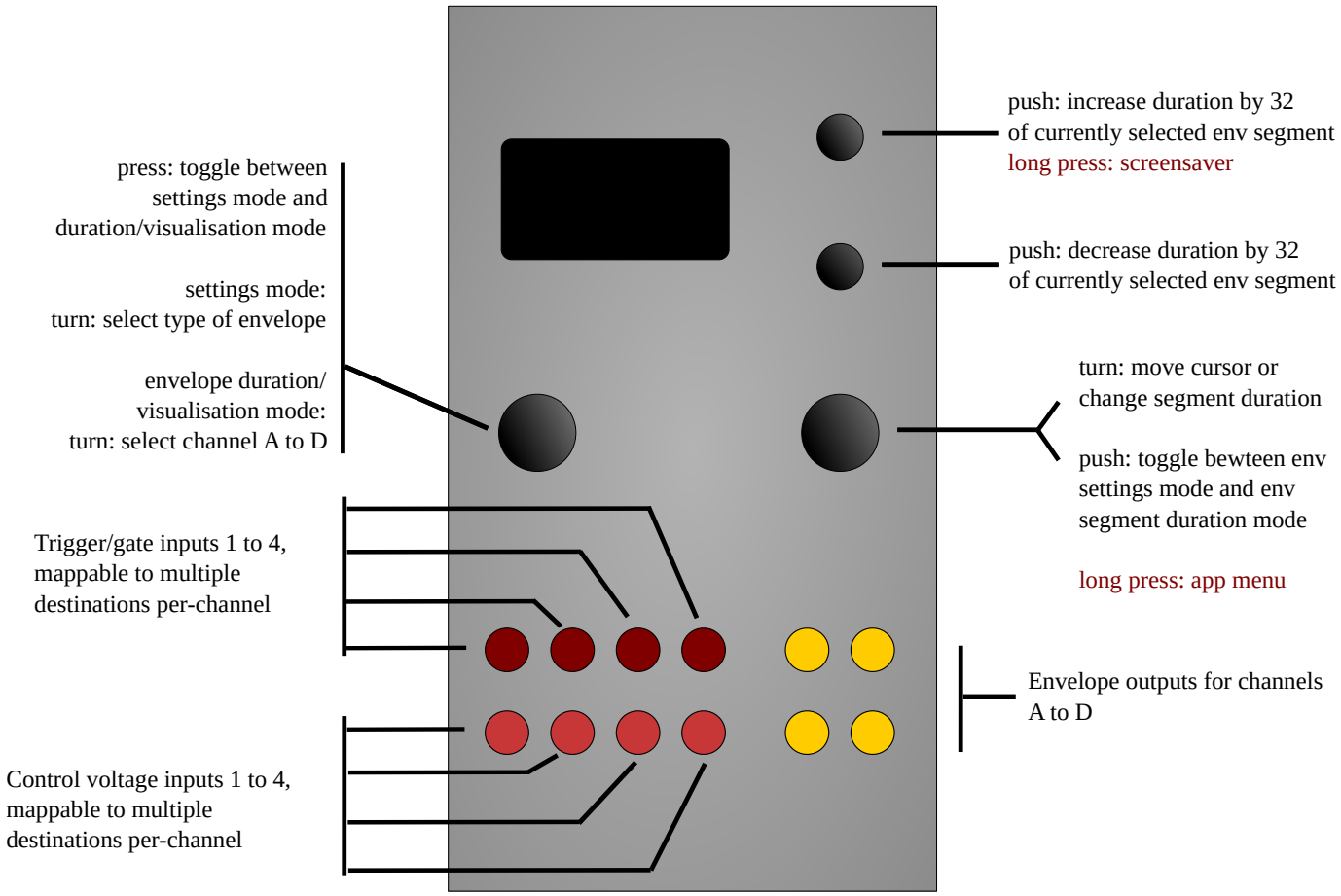
The screensaver show the A and B outputs in a vectorscope (X/Y) display on the left half of the screen, and the C and D outputs as a vectorscope display on the right half of the screen.

Tips

If you have an oscilloscope capable of displaying X/Y (vectorscope) signals, try patching pairs of the *x*, *y* and *z* outputs from either type of generator into it to observe the classic strange attractor patterns.

Piqued

Piqued is a a port of the envelope generator function from the open-source Mutable Instruments Peaks (http://mutable-instruments.net/modules/peaks) module. *Piqued* provides four independently-triggerable envelopes on output channels A to D, with independently mappable voltage-control via the CV1 to CV4 inputs over envelope duration and/or other parameters for each segment of each envelope. Triggers for each of the four envelopes can also be mapped from any of the four trigger inputs (TR1 to TR4). Segment shape (curves) can be set for each segment of each of the four envelopes. A variety of envelope types are available, also independently settable for each envelope, including repeating (looping) envelope types. The shape of each envelope can be visualised while setting parameters. There is also a “Euclidean trigger filter” included, which turns the *Piqued* app into a quad-channel Euclidean polyrhythm generator, that can output envelopes, not just gate/trigger signals. See the o_C videos page (/videos/) for some demonstrations of *Piqued*.



Controls

Piqued presents quite a rich UI (user interface), which is harder to describe in words than it is to use. The following explanations should make sense as soon as you see the UI in action. Once experienced, the interface becomes quite intuitive, we think.

Control	Function
Left encoder (turn)	In menu settings mode, elect the type of envelope. In envelope visualisation mode, select channel A to D to edit (all channels always active)
Left encoder (press)	Toggle between menu settings and envelope visualisation and segment duration settings.
Right encoder (turn)	Navigation mode: move up and down through the menu items (when in menu setting mode), or move back and forth between envelope segments (envelope visualisation/segment duration setting mode). Edit mode: increase or decrease the value being edited (segment durations when in envelope visualisation mode).
Right encoder (press)	Toggle between menu navigation (selection)/envelope segment selection mode and value editing mode
Right encoder (long press)	App selection menu
Up button (long press)	invoke screensave

Control	Function
Up button (press)	Increase duration for currently selected envelope segment for currently displayed envelope generator (A to D) by 32 (when in envelope visualisation mode only)
Down button (press)	Decrease duration for currently selected envelope segment for currently displayed envelope generator (A to D) by 32 (when in envelope visualisation mode only)

Available settings (per-channel)

Setting	Meaning
(envelope type)	the type of envelope (segment nomenclature is standard, with additions: A=attack, D=decay, S=sustain (level), R=release, L=loop point, with the number indication how many enevelope segments immediately before the L will be looped over). Available envelope types are described in the table below.
`Trigger input`	specifies which trigger/gate input, TR1 to TR4, is used to trigger or gate the envelope on the channel currently displayed. In addition, the end-of-cycle (EOC) of an envelope in one of the three other channels in *Piqued* can be specified as a trigger source. Thus, if `Trigger input` for channel B is set to `A EOC`, then chanel B will trigger as soon as the channel A envelope reaches it's end-of-cycle. In this way, envelopes across the channels can be chained or sequenced, in potentially complex patterns. By such internal cross-channel triggering of an envelope with a gate shape at the end-of-cycle of another envelope, EOC gate outputs can be emulated.
`Tr delay mode`	sets the mode for the trigger delay. See notes below.If enabled, the trigger delay will postpone the "firing" of the envelope (that is, the commencement of the attack segment of the envelope) for the time set by `Tr delay msecs` and `Tr delay secs` (see below). Available trigger delay modes are `Off`, `Queue` and `Ring`. `Queue` means that subsequent triggers received while a delay period is active are added to a queue for later action, up to a maximum queue depth set by the `Tr delay count` setting (maximum 32). Further triggers during the delay period are ignored until the number of queued triggers falls below the value set by `Tr delay count`. `Ring` is similar except that triggers received after the queue is full will replace the final trigger in the queue.
`Tr delay count`	sets the number of trigger delays that will be stored or buffered for later processing. The maximum setting is 24 triggers that can be delayed.
`Tr delay msecs`	trigger delay in milliseconds (range 0 to 999 milliseconds). If you set a trigger delay of greater than zero, then the envelope for that channel will not "fire" (commence its attack segment) until the specified delay has elapsed. The delay in milliseconds and the delay in seconds are added together, allowing very fine control over the delay. The "countdown" time for the delayed trigger is shown as a fall bar on the righthand side of the trigger indicator for that channel (at the top of the display).
`Tr delay secs`	trigger delay in seconds (range 0 to 64 seconds) — see `Tr delay msecs` above.
`Eucl length`	sets the length of the Euclidean pattern (`Off` or range 2 to 32 in "beats", where each beat is a received trigger/gate pulse) used to filter triggers for that channel. For a detailed explanation of Euclidean patterns and their use in rhythm generation, see this paper (http://cgm.cs.mcgill.ca/~godfried/publications/banff.pdf) by Godfried Toussaint, or for a brief explanation, see this presentation (http://www.maths.usyd.edu.au/u/joachimw/talk2.pdf). `Eucl length` defaults to `Off`, which means there is no filtering of triggers.

Setting	Meaning
Euclidean pattern	is visible if `Eucl length` is not set to `off`. This displays the currently set Euclidean pattern. The current step is indicated by a small moving dash under the dashed line (one dash per step in the pattern, with the number of steps set by `Eucl length`). Pressing the R encoder invokes an "Euclidean editor". When this editor is displayed, rotating the L encoder sets the fill setting for the pattern, and rotating the R encoder sets the offset or rotation of the patter. Both encoders can be used simultaneously. The fill setting is the number of beats in the pattern that let triggers through to "fire" the envelope. If the fill number is equal to or greater than the Euclidean pattern length number, then every incoming trigger will pass the Euclidean filter and fire the envelope for that channel. If the fill number is zero, then none shall pass, to quote Gandalf. The combination of pattern length (`Eucl length`) and number of active beats (the fill setting) within that pattern length uniquely determines the Euclidean pattern, using the Bjorklund algorithm. For example, if `Eucl length` is set to 8 and fill is set to 5, and the offset parameter is set to the default of 0, then the pattern will be 10110110, where 1 is an active beat (triggers are allowed to pass) and 0 is inactive (triggers are blocked). By setting offset to 1, the pattern becomes 01101101, if set to 2 the pattern becomes 11011010 and so on - that is, the pattern is rotated.
`Eucl reset`	defaults to `-` (off), but when set to a trigger input (TR1 to TR4), a trigger or gate on that input will cause the current position in the Euclidean pattern to skip back to the first position ie, it resets the step in the Euclidean pattern to 1. **Note:** if you set `Eucl reset` to the same trigger input used to trigger the enevlope for a particular channel (i.e. the `Trigger input` setting), and if `Eucl reset div` (see below) is set to the deafult of 1, then the envelope will either never fire or will always fire, because you will be constantly resetiting the Euclidean pattern to step 1 each time the envelope receives a trigger – and step 1 will be either 1 (fire) or 0 (don't fire). In general, use a different trigger input for `Eucl reset`, and/or set `Eucl reset div` to greater than 1, or even better, to a value greater than the length of the Euclidean pattern.
`Eucl reset div`	is a divider for `Eucl reset`. For example, if set to 7, then the current step position in the Euclidean pattern is only reset to 1 after 7 triggers or gates have been received on the input specified by `Eucl reset`. It defaults to 1.
`CV1 ->`	sets the mapping from the CV1 input. The value on CV1 can be ignored (`None`) or sent to control the duration of one of: `Att` (ack), `Dec` (ay), `Sus` (tain) (level, not duration), `Rel` (ease), or `ADR` which affects the duration of all three of attack, decay and release simultaneously (†), or it can be set to control the Euclidean trigger filter parameters: `Eleng` (Euclidean pattern length), `Efill` (Euclidean pattern fill, `Eoffs` (Euclidean pattern offset/rotation), or the trigger delay time (`Delay`), or it can be set to control overall envelope amplitude (`Ampl`), or it can be set to `Loops` which controls the maximum number of enevelope loops (automatic re-triggers) for the looping envelope types. Note that when set to `Eleng` or `Efill`, negative voltages can be used to block all triggers. The input CV values are added to whatever is set for the duration/level or Euclidean parameters or trigger delay time or overall amplitude level or loop number via the respective menu settings.
`CV2 ->`	same as `CV1 ->`, but for the CV2 input
`CV3 ->`	same as `CV1 ->`, but for the CV3 input
`CV4 ->`	same as `CV1 ->`, but for the CV4 input

Setting	Meaning
Attack reset	specifies the behaviour when a new trigger or gate (i.e. rising edge of the gate) is received while the attack segment of an envelope is still in progress. The available values are `None` (new triggers are ignored while the attack segment is active), `SP` (reset Segment and Phase, which resets the segment to attack, which is was anyway, and restarts the segment at phase zero, but retains the current envelope level when the trigger was received, thus avoiding sudden jumps in envelope level and consequent audible clicks or pops, as far as possible), `SLP` (reset Segment, Level and Phase — this also resets the level back to zero — this may result in clicks or pops, as the change in level is instantaneous), `SL` (reset Segment and Level - this resets the level but not the phase - this also results in sudden level jumps, but of a different type), `P` (reset Phase only, which for the attack segment is the same as resetting both segment to attack and resetting phase). The default is `None`, and in general, the expected behaviour will probably result by setting it to `None` or `SP`. The other options are offered to permit experimentation. A dual-channel oscilloscope which allows simultaneous visualisation of the trigger pulses and the resulting envelope shapes is very useful in working out the exact behaviour, which can be complex when the period of a train of triggers is about the same as the period for the envelope being triggered. See also the `Att fall gt` setting, immediately below, which also affects the behaviour of the envelope types with a sustain segment when triggered by short-duration triggers or pulses.
Att fall gt	specifies the behaviour for the `ASR`, `ADSR` and `ADSAR` envelope types when a trigger or gate signal falls (i.e. when a falling edge is detected) during the attack segment of the envelope. The choices are: `Ignor` (ignore), the default - the falling edge of the trigger/gate is ignored, and the attack segment continues; or `Honor` - the envelope proceeds immediately to the segment immediately following the sustain segment (i.e. the release segment for `ASR` and `ADSR` types, and the second attack segment for the `ADSAR` type). `Honor` was the hard-coded behaviour prior to v1.3.
DecRel reset	specifies the behaviour when a new trigger or gate (i.e. rising edge of the gate) is received while the decay or release segments of an envelope are still in progress. The available values are `None` (new triggers are ignored while the decay or release segment is active), `SP` (reset Segment and Phase, which resets the segment to attack and restarts the segment at phase zero, but retains the current envelope level when the trigger was received, thus avoiding sudden jumps in envelope level and consequent audible clicks or pops, as far as possible), `SLP` (reset Segment, Level and Phase - this also resets the level back to zero - this may result in clicks or pops, as the change in level is instantaneous), `SL` (reset Segment and Level - this resets the level but not the phase - this also results in sudden level jumps, but of a different type), `P` (reset Phase only, which doesn't restart a new attack segment, instead, the current decay or release segment is restarted at phase 0 but at the current level, effectively extending it). The default is `SP`, and in general, the expected behaviour will probably result by setting it to `None` or `SP`. The other options are offered to permit experimentation. A dual-channel oscilloscope which allows simultaneous visualisation of the trigger pulses and the resulting envelope shapes is very useful in working out the exact behaviour, which can be complex when the period of a train of triggers is about the same as the period for the envelope being triggered.
Gate high	when set to `Yes`, forces the trigger/gate to high. This is useful when using the looping envelope types, which then just loop continuously, regardless of what is happen on their trigger/gate input, and thus they act as LFOs.
Attack shape	sets the shape of the attack segment for the currently displayed envelope. Available shapes are listed in the table below.
Decay shape	sets the shape of the decay segment for the currently displayed envelope. Available shapes are listed in the table below.

Setting	Meaning
`Release shape`	sets the shape of the release segment for the currently displayed envelope. Available shapes are listed in the table below.
`Attack mult`	sets the duration multiplier for the attack segment(s). Range is 1 to 8192, which allows for very, very slow envelopes if desired.
`Decay mult`	sets the duration multiplier for the decay segment. Range is 1 to 8192. Note that you can still have a short attack but a very, very long decay or release, if desired, by setting different duration multipliers for each of attack, decay and release.
`Release mult`	sets the duration multiplier for the release segment. Range is 1 to 8192.
`Amplitude`	sets the overall amplitude for the envelope, from 0 (no amplitude) to 127 (full range). Defaults to 127. Use it in conjunction with voltage control overall envelope amplitude to dynamically vary the envelope amplitude (that is, the maximum envelope level). See also the related `Sampled Ampl` setting, immediately below.
`Sampled Ampl`	toggles whether the overall amplitude value for the envelope is set continuously, or whether the overall amplitude value is sampled when a new trigger/gate is received and then remains at that sampled value until a new trigger/gate is received. It defaults to `off`.
`Max loops`	sets the maximum number of loops (automatic re-triggers) for the looping envelope types, range 0 to 127. A setting of 0 (zero) means “loop forever” (or at least while the gate input is high - see above), while numbers greater than zero mean loop for that number of times, then stop (at least until a new gate signal is received. New gate signals reset the loop counter. Note that there is settable CV control over this parameter. This allows these looping envelope modes to become CVable burst generators, with CV control of the number of repetitions in each burst of envelopes, as well as the shape, duration and amplitude of the envelopes that make up the burst. By self-patching, one or more other channels can be used to dynamically vary the nature of the burst (eg declining amplitude and increasing duration of each repetition in the burst, simulating an echo if used to drive a VCA etc).
`Inverted`	values are `Yes` or `No`, and `Yes` inverts the envelope output. The output voltage is shifted so that the inverted envelope drops below zero volts (to about -0.5V) at it’s peak (or rather, at it’s trough, or nadir), but this can be trimmed using the Amplitude setting so that it drops to exactly 0V if desired.

- (†) When the `ADR` CV input target is used with looping enevlope modes, each channel of Piqued effectively becomes a voltage-controlled LFO (but not with 1V/octave scaling), with settable waveshapes (via the envelope segment shape settings).

Envelope type	Description
`AD`	Attack-Decay: the attack segment commences on receipt of a trigger or on the rising edge of a gate signal, and the decay segment follows immediately after the attack segment has reached its peak, regardless of whether the gate or trigger signal is still high.
`ADSR`	Attack-Decay-Sustain-Release: just like every other ADSR envelope
`ADR`	Attack-Decay-Release: the attack segment commences on receipt of a trigger or on the rising edge of a gate signal, and the decay segment follows immediately after the attack segment has reached its peak. The sustain level is an inflection point for the decay - when the decay reaches the sustain level, the release segment immediately commences, regardless of whether the gate or trigger signal is still high. See the *Tips* section below on how to use the ADR mode as an AHR/AHD (attack-hold-release or attack-hold-decay) envelope generator with trigger signals.
`ASR`	Attack-Sustain-Release: the attack segment commences on receipt of a trigger or on the rising edge of a gate signal, and the envelope stays at maximum level for as long as the gate input for it remains high (sustain), and then commences the release segment as soon as the gate signal goes low.

Envelope type	Description
`ADSAR`	is like an ADSR envelope, except that the attack segment re-triggers as soon as the sustain segment is finished, before proceeding to the release segment.
`ADAR`	is like the ADR envelope, except that the attack segment re-triggers as soon as the decay segment has finished, before going into the release segment.
`ADL2`	is like the AD envelope, except that it automatically re-triggers (i.e loops) the entire envelope (i.e. both the A and D segments) for as long as the trigger/gate input for it is high (see also the `Gate high` and `Max loops` settings)
`ADRL3`	is like the ADR envelope, except that it automatically re-triggers (i.e loops) the entire envelope (i.e. the A, D and R segments) for as long as the trigger/gate input for it is high (see also the `Gate high` and `Max loops` settings)
`ADL2R`	is like the ADR envelope, except that it automatically re-triggers (i.e loops) the A and D segments for as long as the trigger/gate input for it is high (see also the `Gate high` and `Max loops` settings), before proceeding to the R segment
`ADAL2R`	is like the ADAR envelope, except that it automatically re-triggers (i.e loops) the D and second A segments for as long as the trigger/gate input for it is high (see also the `Gate high` and `Max loops` settings), before proceeding to the R segment
`ADARL4`	is like the ADAR envelope, except that it automatically re-triggers (i.e loops) the entire envelope for as long as the trigger/gate input for it is high (see also the `Gate high` and `Max loops` settings)

Illustrations of these envelope types can be found here (/envelopes/).

Segment shape	Description
`Lin`	Linear (a straight line, equation $x = t$ where t is time)
`Exp`	Exponential (equation $x = 1 - e^{-4t}$)
`Quart`	Quartic (equation $x = t^{3.32}$)
`Sine`	half a sine wave (equation $x = \sin(8 * \pi * t)$)
`Ledge`	almost a square wave, but with rounded corners, when used for attack, gives an immediate (punchy) rise, then a plateau. When used for decay or release, it has a plateau before falling.
`Cliff`	similar to `Ledge` , but when used for attack, has a delay before rising, when used for decay or release, it falls immediately.
`Gate` (v1.1)	is used when gate outputs are desired. The value rises immediately to maximum in the attack segment, and the value falls immediately to minimum in the decay and release segments. In other words, a pulse is output.
`BgDip`	Big dipper - has one large bump on the way up or down.
`MeDip`	Medium dipper - has a medium sized dip on the way up or down.
`LtDip`	Little dipper - has a little dip (more a ledge) on the way up or down.
`Wiggl`	Wiggles - lots of wiggles on the way up or down.

Inputs and outputs

Trigger input and CV1 to CV4 are mappable per-channel via the menu, as described above. Outputs for envelopes A to D appear on outputs A to D respectively.

Screensaver display

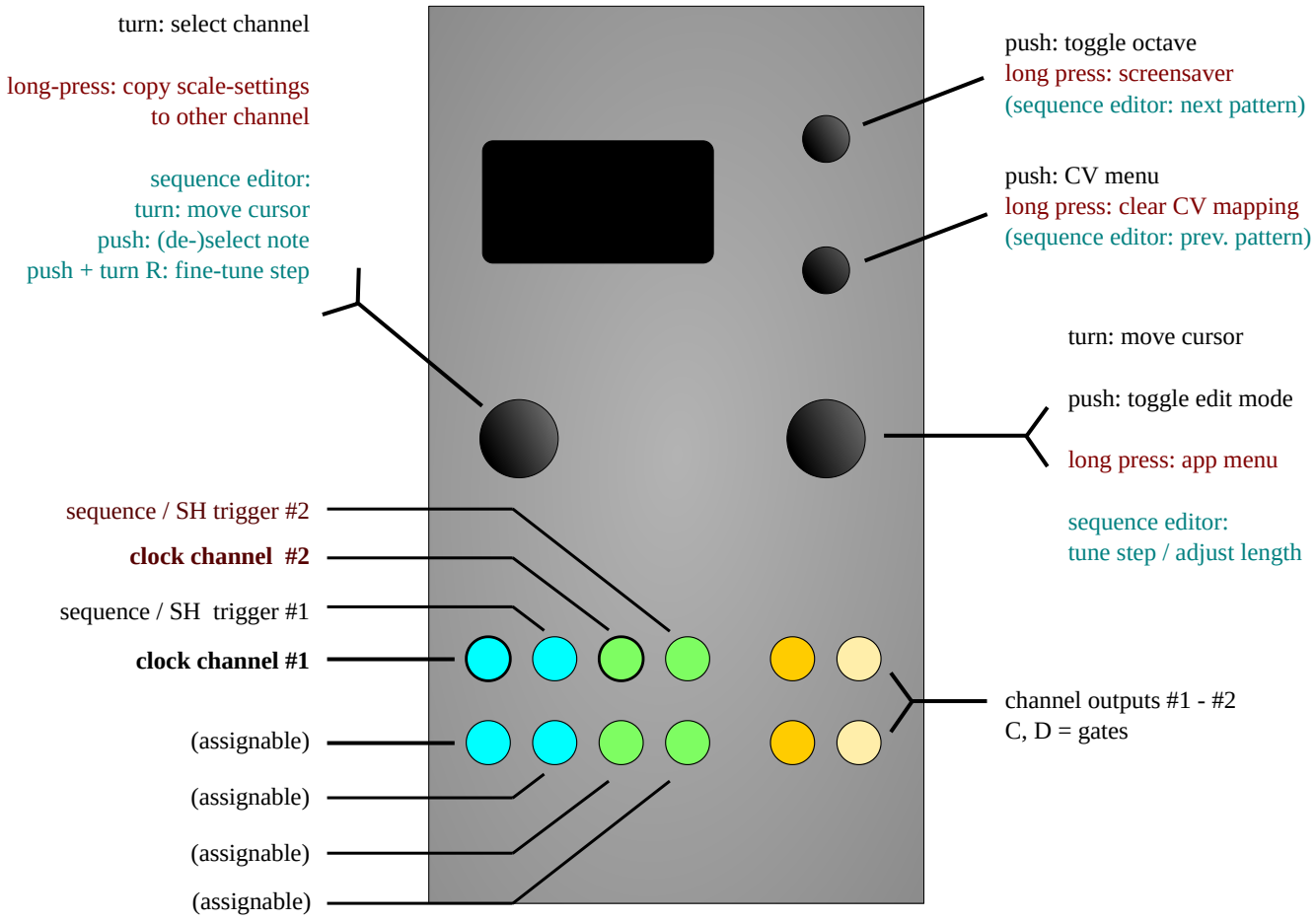
The screen is divided into quadrants, each showing a rolling display of the output values on each of channels A to D. Superimposed on this rolling value line is a representation of the envelope for that channel, as it progresses through its segments.

Tips and tricks

- the ADR mode can also be used as an AHR (attack-hold-release, sometimes also called AHD (attack-hold-decay)) envelope generator. AHR envelope are useful when you want to generate an envelope with a flat sustain period, which usually requires a gate input with some duration. If you only have trigger signals, that is, short pulses, then you can create an AHR envelope by choosing ADR mode, and setting the sustain inflection point to 255 (maximum). By doing that, the decay segment falls from maximum value to… maximum value - in other words, it's flat! The decay segment time/duration then sets the hold duration after a trigger is received. Also try setting the sustain level in ADR mode to something a little bit less than 255 (say 230), and set the decay shape to Wiggle. Now you have an AHR envelope with a wiggly and slightly downsloping plateau segment. Several variations on this theme are possible.

Sequins

Sequins is a basic, **dual-channel sequencer**. The app provides four 16-step sequences per channel, and CV-control over various channel parameters, including scale mask, octave, pulsewidth, clock multiplication and division. The four sequences can be **chained** in various ways, providing sequences from **4 to up to 64 notes**. The sequence data is stored along with the other settings (whenever settings are saved).



- the **sequence editor** works similarly to the scale editor in the quantizer modes:
  - in the main menu, select which sequence to edit by adjusting the `sequence #` parameter.
  - then right-click on the item displayed right below (`--> edit`) to **open the sequence-editor**.
  - **mute/unmute** notes by clicking the left encoder: the step value is displayed in a format x+y.yy, where x is the octave (above or below 0.00V) and y.yy gives the semitone offset. for instance `+2+2.23` = 2.23 semitones above octave #2 (=~ 2.000V).
  - adjust the sequence **length** by pointing the cursor to the far right, then turn the right encoder.
  - (**simultaneously** push down the *left* encoder to also **mute/clear** the steps that are being added while expanding the sequence).
  - to adjust the **pitch**, simply move the cursor to a given slot, then turn the right encoder: this will increase or decrease the pitch in **semitones** (= step size: `1.00`). note these values are **pre-quantization**, so the effect this has will depend on the chosen scale.
  - to **fine-tune** the pitch (= step size: `~ 0.01`), hold down the left encoder **while** turning the right encoder.
  - **note**: the number displayed when adjusting the note values is the actual (pre-quantization) DAC code (that's far from ideal, i know …). details on what those numbers mean can be found here (/custom-scales/).
  - in the sequence-editor, you can use the **up/down** buttons to select which sequence to edit (`#1 - #4`). (this makes it possible to edit sequences 'offline', ie make adjustments to, say, sequence #2 while sequence #1 is playing).
  - **copy+paste**: in the sequence-editor, long-press the left encoder to **copy** the selected sequence. select a different sequence slot, then long-press the down button to **paste** it into that slot. copy+paste

works across channels. (note: once copied (or pasted), sequence data will be stored in a temporary buffer for about 15 seconds, after which it'll expire.)

- the `playmode` and `direction` settings offer various ways of moving through the sequence(s):
    - `-` (default): advance by trigger, using the chosen sequence and direction settings.
    - `SEQ+1 - SEQ+3` : ditto, but **cycling** through 2, 3, or 4 adjacent sequences (= allows up to 64 notes).
    - `TR+1 - TR+3` : ditto, but **jumps** to the next sequence only *if/when* a trigger is received at the **aux.** trigger inputs.
    - `ARP` : **arpeggiate** the sequence.
    - `S+H#1 - S+H#4` : CV-address, when triggered (**sample and hold**).
    - `CV#1 - CV#4` : CV-address, **free-running**.
- note: when in CV-address mode, the `CV adr. range` setting adjusts the resolution of the chosen input (CV1-CV4), relative to the length of the sequence (ie, you'll either need `5V` or `10V` to move through the entire span of the sequence).
- the secondary/aux. clock inputs (TR2, TR4) can be used to either **cycle** through the sequences (see above), or they can be selected as a **reset/mute** signal (`reset/mute`). the options are: `RST2` , `RST4` (reset) and `=HI2` , `=LO2` , `=HI4` , and `=LO4` (mute when aux. clock goes high, respectively low.). NB: When using a clock divider or the like (or anything that will introduce latency relative to the main clock inputs TR1/TR3), it can/will make sense to increase the trigger-to-processing latency (see the `Trigger delay` setting). This will make sure the auxiliary trigger (or control voltage) is processed within the current update window. (Also see *Quantermain* above for an extended discussion of the trigger delay setting).

Inputs and outputs

I/O	Function	-
TR1	clock input #1	-
TR2	aux clock input #1 / reset/mute #1	-
TR3	clock input #2	-
TR4	aux clock input #2 / reset/mute #2	-
CV1	(mappable)	-
CV2	(mappable)	-
CV3	(mappable)	-
CV4	(mappable)	-
A, B	CV outputs #1, #2	-
C, D	aux outputs #1, #2 (default to gate output)	-

Available settings (per-channel)

Setting	Meaning
`scale`	current scale
`--> edit`	edit scale mask (for details see here)
`sequence #`	select sequence `#1 - #4`
`--> edit`	edit sequence
`playmode`	chain sequences (`SEQ+x`), advance by trigger (`TR+x`), or CV-address
`direction`	forward, reverse, pendulum1, pendulum2 (repeat first/last), random, Brownian
`-->brown prob`	(if `direction` = `brwn` (Brownian)) probability (*p*) that sequence direction (up or down will reverse on the next step, 0 means *p*=0, 255 means *p*=1)

Setting	Meaning
`mult/div`	set clock multiplier / divider
`octave`	offset octave
`aux. mode`	aux channel output: `gate` (see note below), `copy`, `AD`, `ADR`, `ADSR` (the last three are envelopes, see below for details)
`CV adr. range`	toggle `5V/10V` (in CV address mode)
`--> pw`	(if `aux. mode` = `gate`) pulse-width of triggers at C/D outputs
`--> aux +/-`	(if `aux. mode` = `copy`) offset at C/D outputs (in octaves)
`--> att dur`	(if `aux. mode` = `AD`, `ADR` or `ADSR`) duration of the attack segment of the envelope output on the auxiliary channels (C or D outputs)
`--> att shape`	(if `aux. mode` = `AD`, `ADR` or `ADSR`) shape of the attack segment of the envelope output on the auxiliary channels (C or D outputs). See the *Piqued* app for details of envelope segment shapes.
`--> dec dur`	(if `aux. mode` = `AD`, `ADR` or `ADSR`) duration of the decay segment of the envelope output on the auxiliary channels (C or D outputs)
`--> dec shape`	(if `aux. mode` = `AD`, `ADR` or `ADSR`) shape of the decay segment of the envelope output on the auxiliary channels (C or D outputs). See the *Piqued* app for details of envelope segment shapes.
`--> sus dur`	(if `aux. mode` = `ADSR`) duration of the sustain segment of the envelope output on the auxiliary channels (C or D outputs)
`--> sus level`	(if `aux. mode` = `ADR` or `ADSR`) level of the sustain segment of the envelope output (for `ADSR` envelopes), or the level at which the the decay segment transitions into the release segment (for `ADR` envelopes) on the auxiliary channels (C or D outputs).
`--> rel dur`	(if `aux. mode` = `ADR` or `ADSR`) duration of the release segment of the envelope output on the auxiliary channels (C or D outputs)
`--> rel shape`	(if `aux. mode` = `ADR` or `ADSR`) shape of the release segment of the envelope output on the auxiliary channels (C or D outputs)
`--> loops`	(if `aux. mode` = `AD` or `ADR` or `ADSR`) sets the number of loops for each envelope output on the auxiliary channels (C or D outputs), defaults to 1 (see also the Tip below)
`reset/mute`	select reset/mute source (mute being more like ‘pause’)
`clock src`	choose channel clock source (`TR1` or `TR3`)
`trigger delay`	trigger-to-processing latency (accessed via CV menu): details see *Quantermain* above

Note: There is a bug in the v1.3.3 firmware which prevents the gate auxiliary output from working unless the `mult/div` for that channel is set to `x2` or above. This bug was fixed in v1.3.3b firmware.

Controls

main menu

Control	Function
Left encoder (turn)	select channel
Left encoder (press)	re-sync channels (reset)
Left encoder (long press)	copy selected scale to other channel
Right encoder (turn)	Navigation mode: move up and down through the menu items. Edit mode: increase or decrease the value being edited
Right encoder (press)	Toggle between menu navigation (selection) mode and value editing mode

Control	Function
Right encoder (long press)	app selection menu
Up button	toggle octave up (parameter menu)
Down button	enter CV menu
Up button (long press)	invoke screensaver
Down button (long press)	toggle sequence-select behaviour (instant update vs. update at end-of-sequence)

CV menu

- **enter** the CV menu by pushing the down button:
  - use the right encoder to assign CV input channels 1-4 to a channel parameter (currently available parameters are: `transposition` (in octaves), `scale mask`, `sequence number` (1-4), `sequence length`, `direction`, `multiplier/divisor` and `pulsewidth`. The quantiser/sampling `trigger delay` (see notes for `Trigger delay` in *Quantermain*) and the `clock src` (clock source) can also be specified in this sub-menu, as well as CV input mappings to these envelope parameters: `att dur`, `dec dur`, `sus lvl`, `rel dur` and `env loops`. By using CV to control `env loops`, a sort of psuedo-ratchetting effect can be obtained. Other envelope behaviours can also be specified (see the *Piqued* app for details): `att rest`, `att fall gt` and `dec/rel reset`.
  - **return** to the main menu by either pressing the **up** or **down** buttons, or by moving the cursor to an empty field and then pressing the right encoder button.
  - **clear** all mappings (per channel) by long-pressing the down button (in CV menu)

Control	Function
Left encoder (turn)	select channel
Left encoder (press)	re-sync channels (reset)
Left encoder (long press)	copy selected scale to other channel
Right encoder (turn)	Navigation mode: move up and down through the menu items. Edit mode: increase or decrease the value being edited
Right encoder (press)	toggle editing mode, if applicable; return to main menu otherwise
Right encoder (long press)	app selection menu
Up button	return to main menu
Down button	return to main menu
Up button (long press)	screensaver
Down button (long press)	clear CV mappings

sequence editor

Control	Function
Left encoder (turn)	select step
Left encoder (press)	activate/de-active step

Control	Function
Right encoder (turn)	1) adjust pitch in semitones; **push left encoder switch** while turning to **fine-tune**. 2) adjust sequence **length** by pointing the cursor to the far right, then turn; **push left encoder switch** while turning to also **clear** the sequence mask (when expanding the pattern)
Right encoder (press)	exit editor
Up button	go to next sequence (edit ‘offline’)
Down button	go to previous sequence (edit ‘offline’)
Left encoder (long press)	**copy** selected pattern
Down button long press	**paste** previously copied sequence
Down button **and** Left encoder long press	**clear** sequence (including pitch data)
Right encoder (long press)	– (app selection menu)
Up button (long press)	– (screensaver)

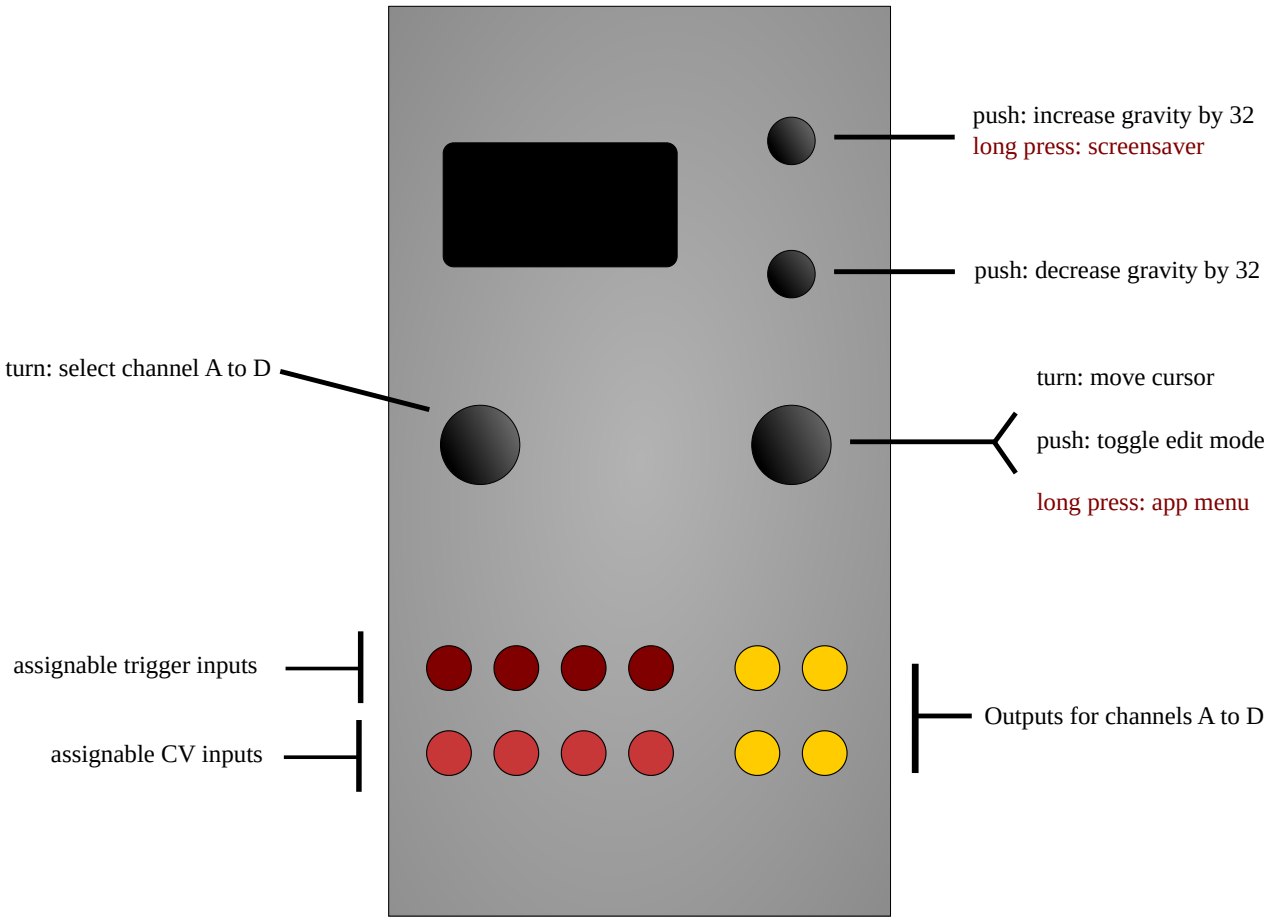
Tips:

- if identical sequences are set up on both sequencer channels, and the same clock input is used to drive both channels, then Steve Reich “Piano Phase” (https://en.wikipedia.org/wiki/Piano_Phase) type effcts can be acheived by setting the `mult/div` setting for one channel to, say, `/32` (divide by 32), and to `/31` on the other channel, and then using a fast clock. The two sequences will slowly move out of phase. By varying the division up and done in one channel, the two sequences can be moved in and out of phase as required.
- If you set the `aux. mode` to one of the envelope types (as an alternative to just gate or trigger out), then an additional setting `--> loops` will appear in the menu. This setting defaults to 1, but you can increase that, and then each time the envelope triggers, it will loop that set number of times. Tthat gives you a fixed ratchet-like effect, which might be useful in some circumstances.
    - However, in the CV assignment menu (see above), you can assign an input CV to the loops parameter using the `env loops ->` setting. If you do that, and also set `--> loops` to 1 in the main menu and you can then use an external voltage (which could be a gate signal, suitably attenuated) to vary the number of loops for each new step in the sequence between 1 and 127 (usually you would want between 1 and 3 or 4). Of course, the voltage to do that can come from the other channel of Sequins, so you can set the number of envelope loops at each step in channel A by a sequence in channel B, clocked from the same source, and cross-patched to one of the CV inputs configured to affect the loops parameter for channel A. Or you can use external voltages, of course, including controllers such as joysticks etc.
    - You can do something similar in the Piqued app, because there is a Loops parameter for the looping envelope types, and you can put that under external CV control.

Dialectic Ping Pong

Dialectic Ping Pong is a port of the bouncing ball envelope generators from the Mutable Instruments Peaks module source code (these are not exposed in the official Peaks firmware, but are available on Peaks with the Dead Man’s Catch (https://github.com/timchurches/Mutated-Mutables/releases) alternative firmware installed).

These generators implement a simple but effective simulation of the physics of a ball that is thrown into the air with a certain velocity, from a certain height, and which then returns to Earth (or a planet of your choice) under the influence of (configurable) gravity, and then bounces (with a settable “bounce loss” simulating how hard the ball is pumped up, if it is a basketball), before being pulled back to Earth and bouncing again, and so on.



Controls

Control	Function
Left encoder (turn)	Select channel A to D to edit (all channels always active)
Right encoder (turn)	Navigation mode: move up and down through the menu items. Edit mode: increase or decrease the value being edited.
Right encoder (press)	Toggle between menu navigation (selection) mode and value editing mode
Right encoder (long press)	App selection menu
Up button (long press)	invoke screensaver
Up button	Increase gravity by 32
Down button	Decrease gravity by 32

Available settings (per-channel)

Setting	Meaning
`Gravity`	*g*, the acceleration due to gravity, from 0 (no gravity) to 255 (gravity on a massive alien planet)

Setting	Meaning
`Bounce loss`	The amount of energy lost by the ball at each bounce (0 to 255). Higher values act like a deflated basketball that goes “splat!” instead of bouncing.
`Amplitude`	Initial amplitude (height) of the ball when the envelope is triggered, 0 to 255.
`Velocity`	Initial velocity of the ball when the envelope is triggered - the size of the kick or impulse imparted to the ball, if you like. Note that high values will cause the ball to bounce off the roof of the miniature gymnasium inside the module.
`Trigger input`	Trigger input source (TR1 to TR4) for the current channel
`Retrigger`	retriggers the ball after *x* bounces, where *x* is the value of this setting. There is configurable CV control over this parameter. This retriggering after *x* bounces makes the bouncing ball into a strange sort of LFO…
`CV1 ->`	Mapping of the CV1 input to a parameter for the selected channel. Values are “off”, “grav” (gravity), “bnce” (bounce loss), “ampl” (initial amplitude), “vel” (initial velocity) and `retr` (number of bounces before retriggering).
`CV2 ->`	As for `CV1->`, but for CV2.
`CV3 ->`	As for `CV1->`, but for CV3.
`CV4 ->`	As for `CV1->`, but for CV4.
`Hard reset`	If set to on, the envelope will instantly restart at the currently set initial amplitude, rather than starting from the height that the ball happens to be at the time when the trigger is received.

Inputs and outputs

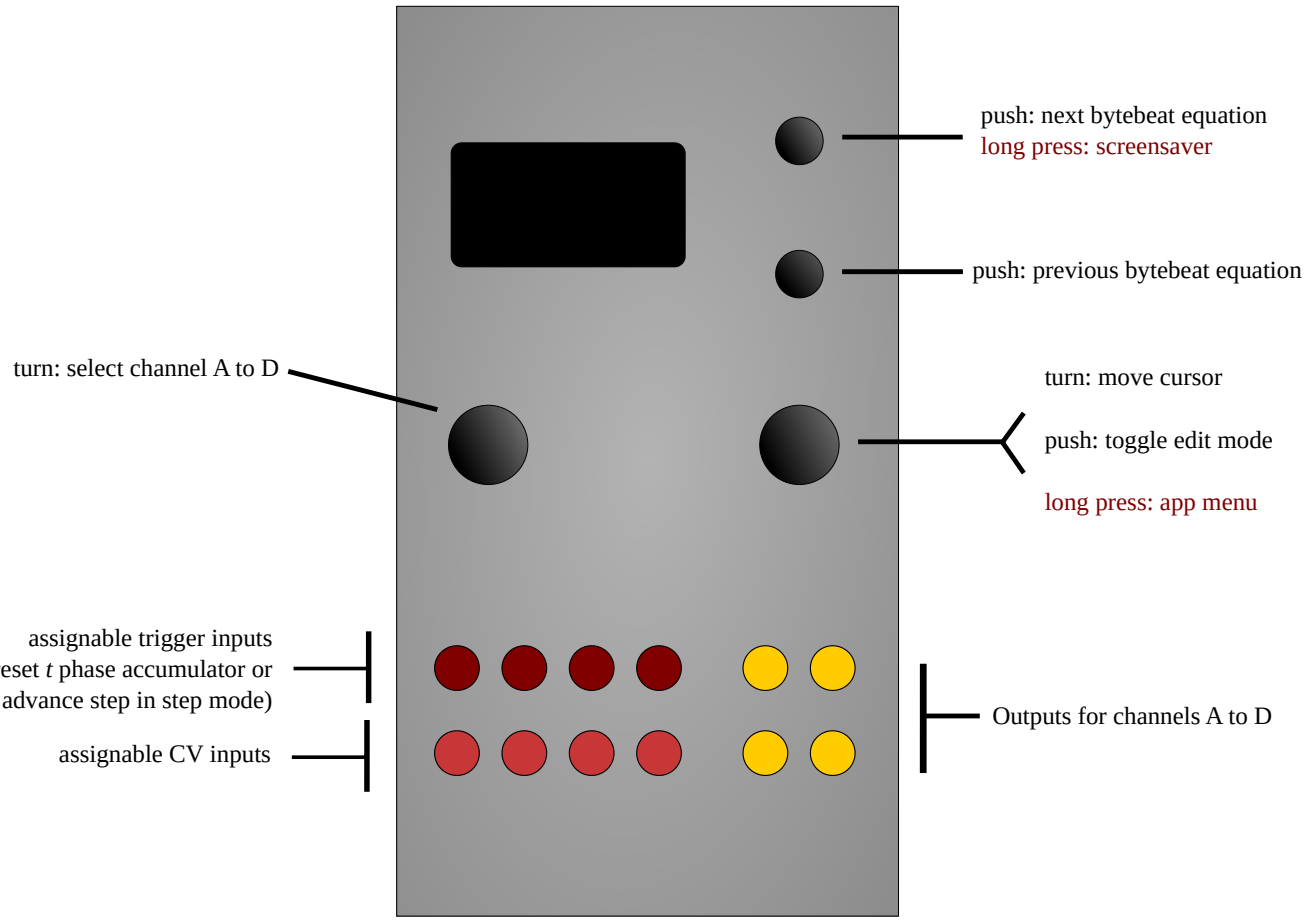
Trigger inputs and CV1 to CV4 are mappable per-channel via the menu. Outputs for channels A to D appear on outputs A to D respectively.

Screensaver display

The screen is divided into quadrants, each showing a rolling display of the output values on each of channels A to D.

Viznutcracker, sweet!

This is a experimental implementation of several bytebeats signal generators. “bytebeats” are equations (actually, recursive functions), expressed usually as a single line of programme code, typically involving various bit-level operators, which when evaluated with an incrementing phase value at audio rates produce all manner of harsh digital noises, some of which sound musical, or at least, interesting. bytebeats were first described (http://countercomplex.blogspot.com/2011/10/algorithmic-symphonies-from-one-line-of.html) in 2011 by viznut (aka Ville-Matias Heikkilä).



The output, if used as an audio signal, usually needs to be fairly heavily filtered through a low-pass filter to remove at least some of the unpleasant digital “screech” due to high-frequency aliasing and other effects which are characteristics of bytebeats. This digital aliasing is a fundamental characteristic of the way bytebeats work, and isn’t due to any hardware limitations of the o_C module.

The *Viznutcracker, sweet!* app current provides access to 16 different byte beat equations. The app provides four independent byte beat generators, on channels A to D, which all run independently. The equation, speed/frequency and three equation parameter values (p0, p1 and p2) can be set via the menus and/or voltage-controlled for each generator via mappable CV inputs.

Perhaps uniquely amongst byte beat generator modules, the *Viznutcracker, sweet!* apps permits the byte beat generators to be run at very slow rates, and because the o_C module outputs are DC-coupled, they can therefore be used as sources of stepped control voltages. For example, the outputs can be fed into a quantiser (such as another o_C module) to create potentially interesting pitch sequences (possibly even melodies…). Furthermore, the app allows each byte beat equation to be incremented by an external clock/trigger input, so that these stepped voltages can be generated in synchrony with other external processes.

Controls

Control	Function
Left encoder (turn)	Select channel A to D to edit (all channels always active)
Right encoder (turn)	Navigation mode: move up and down through the menu items. Edit mode: increase or decrease the value being edited.
Right encoder (press)	Toggle between menu navigation (selection) mode and value editing mode

Control	Function
Right encoder (long press)	App selection menu
Up button (long press)	invoke screensaver
Up button	Next bytebeat equation
Down button	Previous bytebeat equation

Available settings (per-channel)

Setting	Meaning
`Equation`	Byte beat equation (see table below)
`Speed`	0 to 255. 255 equates to a 16.7 kHz sample rate (i.e. the phase accumulator advances 16,666 times per second). There is a rough quadratic scaling of lower rates, meaning that 0 is quite slow.
`Pitch`	1 to 255. This is a very rough pitch setting, which behaves differently with each equation. Increasing values of `Pitch` will increase the pitch (frequency) of some elements of the equation output, but not necessarily all components, so it may not sound like a normal pitch increase or decrease. However, its effect does sound different to the `Speed` parameter. (shrugs shoulders)
`Parameter 0`	The first adjustable parameter in the chosen equation. Range 0 to 255, but some parameter settings in some equations do not produce any output, or do not produce output for all values of the phase accumulator (be patient!)
`Parameter 1`	ditto for the second adjustable parameter in the equation
`Parameter 2`	ditto for the third adjustable parameter in the equation
`Loop mode`	Enables loop mode, in which the phase accumulator is constrained to loop between specific start and end values, instead of between 0 and 4,294,967,296.
`Loop begin ++`	Coarse loop begin point, range 0-255
`Loop begin +`	Medium loop begin point, range 0-255
`Loop begin`	Fine loop begin point, range 0 - 255
`Loop end ++`	Coarse loop end point, range 0-255
`Loop end +`	Medium loop end point, range 0-255
`Loop end`	Fine loop end point, range 0 - 255
`Trigger input`	Specified which of the 4 trigger inputs (TR1 to TR4) is used for the trigger input for `Step mode` , or when `Step Mode` is off, which trigger input is used to reset the phase accumulator for that channel.
`Step mode`	When set to on, the phase accumulator is incremented when a trigger or clock pulse is received on the digital (trigger) input specified by the `Trigger input` setting. When set to off, a trigger (or rising edge of a pulse or clock) received on the trigger input specified by the `Trigger Input` setting will reset the phase accumulator (the *t* variable in byte beat equations), which has the effect of resetting the byte beats “tune” or “melody” back to its beginning, or back to the start of the loop start point if loop mode is enabled.

Setting	Meaning
`CV1 ->`	specifies which parameter CV1 is mapped to for this channel. Choices are `off`, `eqn`, `spd`, `p0`, `p1`, `p2`, `beg++`, `beg`, `end++`, `end` and `pitch`.
`CV2 ->`	ditto for CV2
`CV3 ->`	ditto for CV3
`CV4 ->`	ditto for CV4

Equation name	Source of equation
`hope`	“atmospheric, hopeful” via royal paw (http://royal-paw.com/2012/01/bytebeats-in-c-and-python-generative-symphonies-from-extremely-small-programs/)
`love`	the equation by stephth via here (https://www.youtube.com/watch?v=tCRPUv8V22o) at 3:38
`life`	the second equation listed here (http://xifeng.weebly.com/bytebeats.html)
`age`	“Arp rotator” via Microbe Modular (http://microbemodular.com/products/equation-composer/overview) Equation Composer Ptah bank (https://github.com/clone45/EquationComposer/blob/master/EquationBankPtah.cpp)
`clysm`	“BitWiz Transplant” via Microbe Modular (http://microbemodular.com/products/equation-composer/overview) Equation Composer Ptah bank (https://github.com/clone45/EquationComposer/blob/master/EquationBankPtah.cpp)
`monk`	“Vocaliser” via Microbe Modular (http://microbemodular.com/products/equation-composer/overview) Equation Composer Khepri bank (https://github.com/clone45/EquationComposer/blob/master/EquationBankKhepri.cpp)
`NERV`	“Chewie” via Microbe Modular (http://microbemodular.com/products/equation-composer/overview) Equation Composer Khepri bank (https://github.com/clone45/EquationComposer/blob/master/EquationBankKhepri.cpp)
`Trurl`	“Tinbot” via Microbe Modular (http://microbemodular.com/products/equation-composer/overview) Equation Composer Sobek bank (https://github.com/clone45/EquationComposer/blob/master/EquationBankSobek.cpp)
`Pirx`	“My Loud Friend” via Microbe Modular (http://microbemodular.com/products/equation-composer/overview) Equation Composer Ptah bank (https://github.com/clone45/EquationComposer/blob/master/EquationBankPtah.cpp)
`Snaut`	""A bit high-frequency, but keeper anyhow” via Microbe Modular (http://microbemodular.com/products/equation-composer/overview) Equation Composer Khepri bank (https://github.com/clone45/EquationComposer/blob/master/EquationBankKhepri.cpp)
`Hari`	“The Signs” via Microbe Modular (http://microbemodular.com/products/equation-composer/overview) Equation Composer Ptah bank (https://github.com/clone45/EquationComposer/blob/master/EquationBankPtah.cpp)
`Kris`	“Light Reactor” via Microbe Modular (http://microbemodular.com/products/equation-composer/overview) Equation Composer Ptah bank (https://github.com/clone45/EquationComposer/blob/master/EquationBankPtah.cpp)
`Tichy`	“Alpha” via Microbe Modular (http://microbemodular.com/products/equation-composer/overview) Equation Composer Khepri bank (https://github.com/clone45/EquationComposer/blob/master/EquationBankKhepri.cpp)
`Bregg`	“Hooks” via Microbe Modular (http://microbemodular.com/products/equation-composer/overview) Equation Composer Khepri bank (https://github.com/clone45/EquationComposer/blob/master/EquationBankKhepri.cpp)
`Avon`	“Widerange” via Microbe Modular (http://microbemodular.com/products/equation-composer/overview) Equation Composer Khepri bank (https://github.com/clone45/EquationComposer/blob/master/EquationBankKhepri.cpp)

Equation name	Source of equation
Orac	“Abducted” via Microbe Modular (http://microbemodular.com/products/equation-composer/overview) Equation Composer Ptah bank (https://github.com/clone45/EquationComposer/blob/master/EquationBankPtah.cpp)

Inputs and outputs

Trigger inputs and CV1 to CV4 are mappable per-channel via the menu. Outputs for channels A to D appear on outputs A to D respectively.

Screensaver display

My God, it’s full of stars!

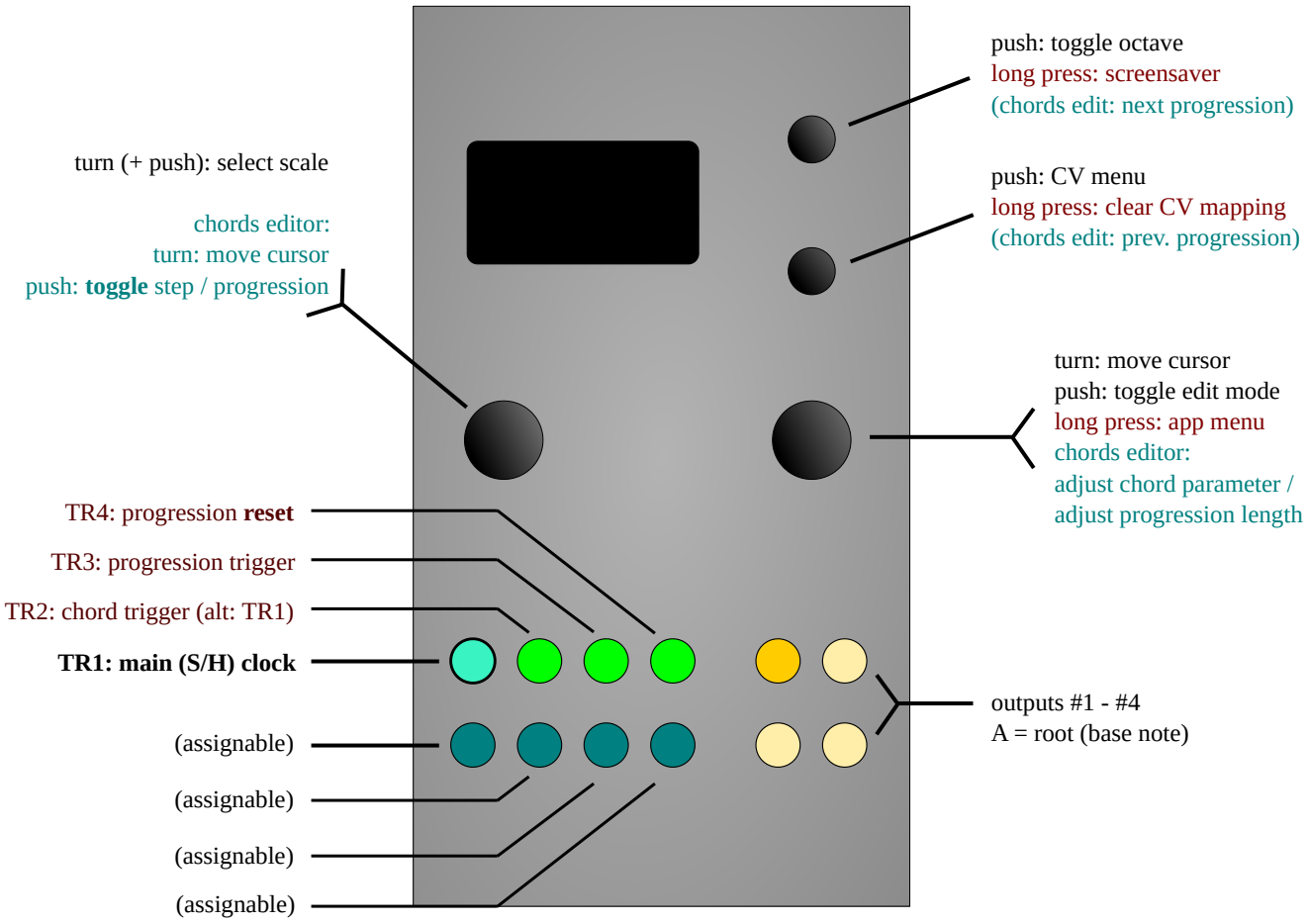
Tips

- as noted above, use the byte beat generators at very slow rates to generate DC control voltages. Feed these to a quantiser to produced pitch sequences, or pass them through a portamento or slew-limiting module to create interesting smooth modulation signals, or use them as-is, to modulate filters etc.
- process the byte beats outputs running at audio rates through a low-pass or shelf filter to remove some of the high-pitched screech
- process the output through band-pass filters (such as the Mutable Instruments Shelves (http://mutable-instruments.net/modules/shelves) filter), or other complex filtering arrangements subject to slow modulation (by another o_C module, perhaps)
- process the output through a VCA and/or filter with a percussive envelope to produce interesting, well, percussive sounds. Use the same trigger or gate signal used to fire the envelopes also reset the phase on the byte beat generator. Use slow modulation to slowly vary the frequency/rate, and/or to “scrub” loop start and end points in order to vary the nature of the percussive sound.

Acid Curds

Acid Curds is a basic chord sequencer. The app provides **four 8-step chord progressions** (in total), and CV-control over various parameters, including chord type, progression length, direction, voicing, inversion, etc. The four chord progressions can be **chained** in various ways, providing progressions from **1 to up to 32 chords**. The chords/sequence data is stored along with the other settings (whenever settings are saved).

- *Acid Curds* can be used as either a basic quantizer / sample-and-hold type thing, in which case the chords will be formed on the basis of the incoming root CV, or it can be used as a step-sequencer, in which case the chord base-notes and chord properties will be fixed notes (chosen from a given scale) respectively features (inversion, voicing, etc); it’s also possible to combine sequencer and S+H type behaviors. Here is the basic i/o mapping; the CV inputs are freely assignable:



- the **chords editor** works much like the scale and sequence editors in Copiermaschine, Sequins, etc:
  - in the main menu, select which progression (#1-#4) to edit by adjusting the `progression` parameter.
  - then right-click on the item `chords -->` to **open the chords-editor**:
    - adjust the progression **length** by pointing the cursor to the far right (using the left encoder), then turn the right encoder.
    - turn the **right** encoder to change the selected chord-parameter (highlighted w/ white background).
    - turn the left encoder to either select which chord (step) to edit, or to select which step-feature to edit.
    - pushing the left encoder **toggles** between chord-select and feature-select.
    - push the right encoder to close the editor again.
- a chord (or step) consists of five features: quality/type (`Q`), voicing (`V`), inversion (`I`), base note (`B`), and the register/octave (`O`). most of the parameter values should be fairly self-explanatory.
- the 'base note' (or root note) of a chord can be set to either `CV` , in which case a chord will be formed based on the voltage present at input CV1 (TR1 in that case is used as the S+H clock, the chords progression will advance depending on the `chords trg src` setting); or, it can be set to a fixed value: `#1` , `#2` , `#3` , etc (these values are given in scale degrees, e.g. choosing `#3` = the third note of a given scale).
- the `playmode` and `direction` settings offer various ways of moving through the sequence(s):
  - `-` (default): advance by trigger, using the chosen direction settings.
  - `SEQ+1 - SEQ+3` : ditto, but **cycling** through 2, 3, or 4 adjacent progressions (= chain up to 32 chords).

- `TR3+1 - TR3+3` : ditto, but **jumps** to the next progression only *if/when* a trigger is received at the **TR3** trigger inputs.
- `S+H#1 - S+H#4` : CV-address, triggered by **TR3** (= **sample and hold**).
- `CV#1 - CV#4` : CV-address, **free-running**.

Inputs and outputs

I/O	Function	-
TR1	main clock / S+H input	-
TR2	chord advance trigger (if selected via `chords trg src`)	-
TR3	playmode trigger `TR3+1 - TR3+3`, CV-address trigger (`S+H#1 - S+H#4`)	-
TR4	progression reset	-
CV1	(mappable)	-
CV2	(mappable)	-
CV3	(mappable)	-
CV4	(mappable)	-
A, B, C, Dchord outputs		A is the base note

Available settings

Setting	Meaning
`--> scale`	edit current scale
`root`	scale root
`progression`	select progression `#1 - #4`
`chords -->`	edit progression/chords
`playmode`	chain sequences (`SEQ+x`), advance by TR3 (`TR3+x`), or CV-address
`direction`	forward, reverse, pendulum1, pendulum2 (repeat first/last), random, brownian
`transpose`	transpose (in scale-degrees)
`octave`	transpose in octaves
`CV source`	CV source (`CV1`, `CV2`, `CV3` or `CV4`
`chords trg src`	chords-trigger: `TR1` or `TR2`
`TR1 delay`	TR1 trigger-to-processing latency - see the discussion of the `Trigger delay` setting in the *quantermain* app for more details

Controls

main menu

Control	Function
Left encoder (turn)	select scale
Left encoder (press)	activate scale
Left encoder (long press)	-
Right encoder (turn)	Navigation mode: move up and down through the menu items. Edit mode: increase or decrease the value being edited
Right encoder (press)	Toggle between menu navigation (selection) mode and value editing mode

Control	Function
Right encoder (long press)	app selection menu
Up button	toggle up/down by one octave
Down button	go to CV menu
Up button (long press)	screensaver
Down button (long press)	clear CV mapping

CV menu

- **enter** the CV menu by holding down the **down** button.
  - use the right encoder to assign CV input channels 1-4 to a channel parameter (currently available parameters are: `root` (scale/global), `scale mask`, `transpose`, `octave`, `voicing`, `inversion`, `progression #`, `direction`, and `progression length`.
  - **return** to the main menu by either pressing the **up** or **down** buttons.
  - **clear** all mappings (per channel) by long-pressing the down button.

chords editor

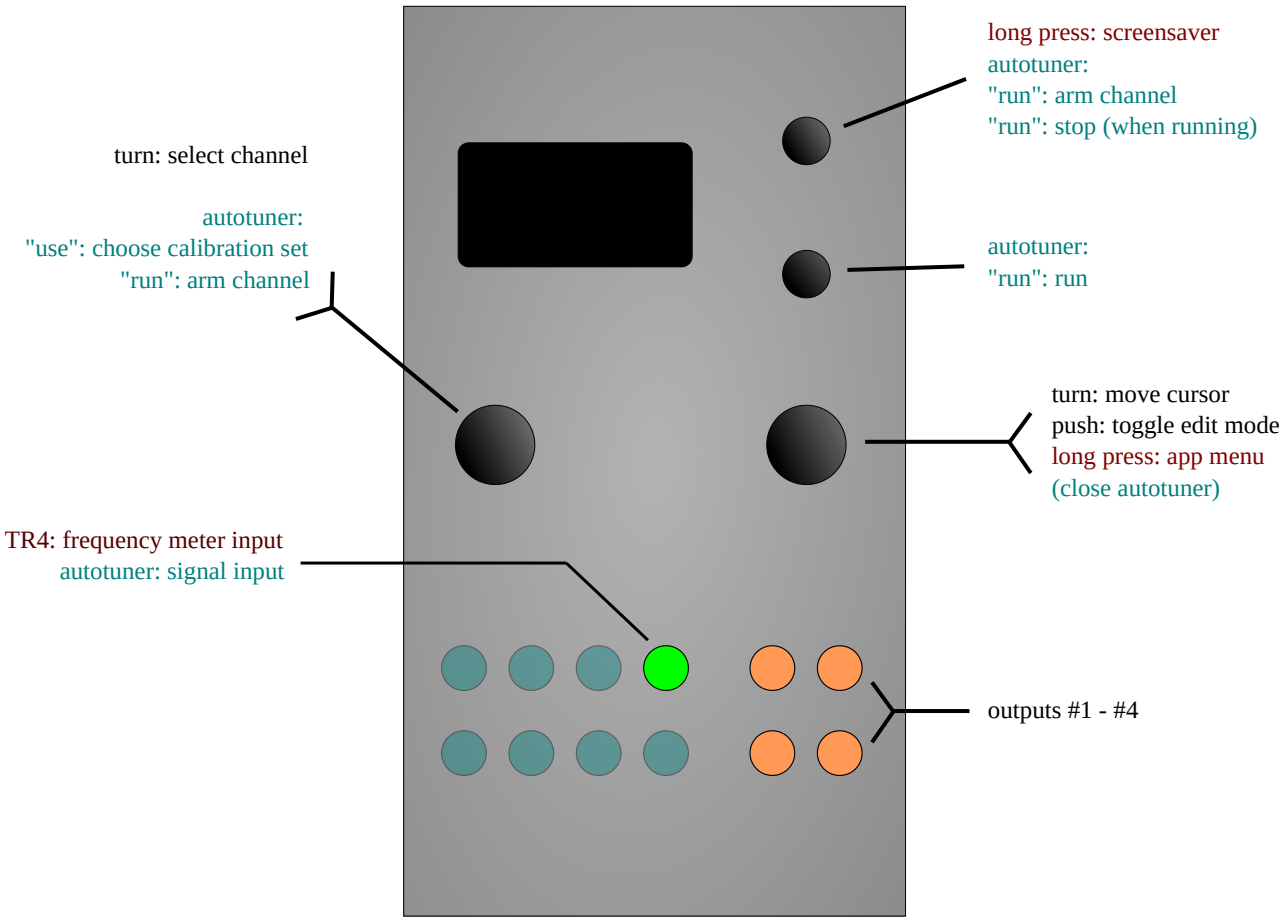
Control	Function
Left encoder (turn)	select chord-step, or select chord-feature
Left encoder (press)	toggle step-select / feature-select
Right encoder (turn)	1) adjust chord feature; 2) adjust progression **length** by pointing the cursor to the far right, then turn
Right encoder (press)	exit editor
Up button	go to next progression (edit ‘offline’)
Down button	go to previous progression (edit ‘offline’)
Left encoder (long press)	-
Down button long press	-
Right encoder (long press)	– (app selection menu)
Up button long press	– (screensaver)

Tips

- usually what trips people up is that by default there’s two clocks involved:
  - TR1 = S+H
  - TR2 = “chord trigger”
- a pulse into TR1 will make the module sample the CV inputs (which won’t do anything if you choose a static ‘base’ note)
- a pulse into TR2 will actually step to the next chord in the progression (if any)
- you can map both types of behaviour to TR1, see the `chords trg src` parameter above

References

References is a tuning-utility app. It comes with a frequency/BPM meter (via the TR4 input, or TR1 if the module is being used in upside-down mode) and a **closed-loop calibration** mode, as well as outputs fixed **reference voltages**. These are handy when calibrating other modules, such as VCOs.



- Most of these functions should be fairly self-explanatory.
  - To use the frequency/BPM meter, simply patch a signal into **TR4** (see note below) and long-press the up-button to invoke the screensaver, where the measurement results will be displayed:
    - the frequency of the audio or clock signal input into TR4 is displayed in Hertz, to three decimal places (and it really is accurate to at least three decimal places!).
    - the `Note/BMP` menu setting determines whether notes tuning or BPM is displayed below the frequency reading
  - To use the closed-loop calibration:
    - 1) patch the output channel (A-D) that you want to calibrate into the V/oct input of your VCO.
    - 2) patch output of the VCO (ideally, a square wave) into **TR4**.
    - 3) then right-click on `autotune -->`, and turn right encoder to select `run -->`.
    - 4) press the up-button to “arm” the tuner. you should see a number displaying the frequency (at 0V); this shouldn’t jump around erratically (in which case the procedure is bound to fail); tune the initial frequency to somewhere in the 50Hz-200Hz range.
    - 5) then push the down-button to set off the procedure, which will take a couple of seconds. If everything goes well, it’ll save the calibration data into the slot `use --> .auto`, which can be selected/de-selected via the menu item `use -->`.

Note: the frequency meter/BPM and closed-loop calibration input is **TR1** if the module is being used in upside-down mode.

Autotune, make it so! (https://www.youtube.com/watch?v=VjWxTyj2kFw&cc_load_policy=1&vq=hd720)

Controls

Control	Function
Left encoder (turn)	Select channel A to D to edit (all channels always active)
Right encoder (turn)	Navigation mode: move up and down through the menu items. Edit mode: increase or decrease the value being edited.
Right encoder (press)	Toggle between menu navigation (selection) mode and value editing mode
Right encoder (long press)	App selection menu
Up button (long press)	invokes the screensaver display immediately
Down button	

Available settings (per-channel)

Setting	Meaning
`Octave`	sets the octave range. Note that the octave numbering is such that C for octave 0 is 0V, C for octave 1 is 1V and so on.
`Semitone`	sets the semitone offset within each octave range. One semitone increment is 83.33mV.
`Mod range oct`	sets the number of octaves to automatically jump up or down. This can be useful when adjusting trimpots on VCOs.
`Mod rate (s)`	sets the rate (as a period in seconds) at which the reference voltage jumps up or down by the number of octaves set by `Mod range oct`.
`autotune -->`	opens the autotuner-submenu
`Notes/BPM`	toggle displayed values (for frequency meter) - frequency in Hertz or beats per minute (bpm)
`A above mid C`	Sets the frequency in Hertz (Hz) of the A above middle C used for the tuner calculations (range is 400 Hz to 480 Hz). It defaults to 440 Hz, which is now standard Western concert pitch, but it can be set to other values.
`> mantissa`	sets the fractional part of the `A above mid C` value, in tenths of a Hertz (0.1 Hz) increments.
`> ppqn`	sets the number of pulses per quarter note (ppqn) used for the bpm display (available settings are 1, 2, 4, 8, 16, 24, 48, 64 and 96 ppqn)

Closed-Loop Calibration

Control	Function
Right encoder (turn)	Navigation mode: move up and down through the menu items.
Left encoder (turn)	item: `use -->` : choose calibration-set for selected channel (when options are available): default calibration autotuned
Up button	item: `run-->` : arm autotuner (TR4, or TR1 if upside-down), or stop procedure if already running
Down button	item: `run-->` : run autotuner
Right encoder (press)	close

Control	Function
Right encoder (long press)	App selection menu

Inputs and outputs

TR4: frequency-meter input, inputs TR1-TR3 are unused. Reference voltages for channels A to D appear on outputs A to D respectively.

Note: the reference voltages output by the References app are only as accurate as the calibration of the module. However, the module is capable of a precision of better than +/- 0.5mV, if calibrated with a precision multimeter. Furthermore, the `autotune -->` setting for each channel must be set to `default`, not `auto` in order for the output voltages output to reflect the indicated voltage. If set to `auto`, then correction factors for specific VCOs will be applied, and the output voltage will not match the indicated voltage.

Note: the indicated output voltages are only valid if 1V/oct output scaling (the default) is selected for each channel. If a dotted line appears under the menu, then an alternative output scaling is active. You may need to set this back to 1V/oct before using the References app. As at v1.3, you can only access the alternative output scaling settings via the scale editor in the *CopierMaschine*, *Quantermain*, *meta-Q*, *Sequins* or *Acid Curds* apps. We apologise for this slight inconvenience.

Screensaver display

The screen is divided into four channel lanes (columns), with the same layout as the *Quantermain* screensaver, except that the output voltage (rounded to the nearest millivolt) is shown as the top of each lane (column). Note that the minus signs for negative voltages are quite small due to the limited display space, and can be quite hard to see.

Below the channel lanes, the TR4 (or TR1 if run upside-down) frequency measurement is displayed. This will blank itself out ofter about a minute if no signal is being received on TR4 (or TR1 if flipped). Below that is the note/tuner display, or the BPM (beats per minute) display, depending on the `Notes/BPM` setting. Note that the note tuner only operates down to about 16 Hz.

Firmware CHANGELOG

- Changes between v1.3.2 and v1.3.3 (/firmware-changelog/#changes-between-v132-and-v133)
- Changes between v1.3.1 and v1.3.2 (/firmware-changelog/#changes-between-v131-and-v132)
- Changes between v1.3.0 and v1.3.1 (/firmware-changelog/#changes-between-v130-and-v131)
- Changes between v1.2 and v1.3 (/firmware-changelog/#changes-between-v12-and-v13)
- Changes between v1.1 and v1.2 (/firmware-changelog/#changes-between-v11-and-v12)
- Changes between v1.0 and v1.1 (/firmware-changelog/#changes-between-v10-and-v11)

Known Issues

- **sample rate**: limited because DAC and OLED sit on the same data bus. for its (originally) intended purpose — as a module that quantizes/generates pitch CVs — the (resulting) **16.6 KHz** DAC update rate is not a problem; in fact, it's pretty good, hence the low latency (~60us). however, a 16.6 KHz update rate means that audible digital aliasing is present when the output is used as an audio source — which is why the only app intended as an audio source is *Viznutcracker, sweet!*, ie the byte beat generator, in which case audio

quality isn't a huge priority. That said, several of the other apps can produce signals in the audio range, but you may need to use them with a low-pass filter to remove some of the aliasing noise. **Anyway, just be aware that o_C is intended to be used as a pitch CV generator, and as a slow modulation CV source, not as audio oscillator or audio signal generator.**

- In theory, some of the digital aliasing may "bleed through" into the audio domain even when the o_C is used as a modulation CV source (eg as an envelope generator or LFO modulating a VCF cut-off or VCA gain). We have not found that to be a problem in practice, but if it does become apparent, then you can always interpose a low-pass filter between the o_C and whatever is being modulated. WMD make a module exactly for this purpose: the Quad Anti-aliasing Filter (QAAF) (https://www.wmdevices.com/collections/eurorack-modules/products/quad-anti-aliasing-filter-qaaf). However, this is unlikely to be required.
- **output range**: asymmetrical (-3V/+6V), as a concession to typical pitch voltage ranges (re: quantizer). when used as an LFO (that is, the *Quadraturia* app), the waveform is thus **not symmetrically bipolar**. This can be corrected by adding -1.5V to each output, so that the output range is shifted from -3V to +6V, to -4.5V to +4.5V. The Mutable Instruments Shades (http://mutable-instruments.net/modules/shades) module is excellent for doing such level shifting (and voltage scaling), but there are many other utility modules which can do the same. The Lorenz and Rössler functions output by the *Low-rents* app are similarly asymmetrical — they also range from -3V to +6V, due to the hardware design of o_C. (note: in v1.2, output offset and attenuation parameters were added to *Quadraturia* to permit unipolar output, albeit with reduced peak-to-peak amplitude). alternatively, you could **mod** the output stage (/hardware-basics/#output).

24678 words

ornament & crime is developed and maintained by mxmxmx, Patrick Dowling and Tim Churches.