Pieplate Reference Manual

2026-03-29

Pieplate Voice Router: Reference Manual

Version 1.0.0

---

1. Overview

Pieplate is a Max for Live MIDI voice router designed to sit between a single MIDI input and up to 16 satellite synthesizer voices. It receives note messages on a single channel and distributes them across numbered voice outputs according to a configurable routing algorithm.

The core problem Pieplate solves: when you play a chord on a keyboard, each note needs to reach a separate synthesizer instance so that per-voice effects, panning, and timbral variation are possible. Rather than hardwiring this routing, Pieplate offers 15 different strategies for deciding which note goes to which voice, from simple polyphony to chord memory to minimal-movement optimization.

Signal chain position:

```
MIDI Input -> Pieplate -> Satellite Voice 1
                                   -> Satellite Voice 2
                                   -> ...
                                   -> Satellite Voice N
```

For each note, Pieplate sends a voice number, pitch, and velocity to the corresponding satellite. Outlet 1 emits the number of currently active voices. Pieplate also broadcasts pitch bend and CC messages to all voices (1 through `maxvoices`).

---

2. Routing Modes

Pieplate provides 15 routing modes (numbered 0-14). Each mode represents a fundamentally different strategy for assigning incoming notes to satellite voices.

Terminology

- **Event-routing modes** assign each note-on to a voice at the moment the note arrives. The voice assignment is fixed for the lifetime of that note.
- **Stable-voicing modes** continuously re-evaluate voice assignments as notes are added and removed. When you add a third note to a two-note chord, all three notes may be repositioned across voices to maintain a consistent layout.
- **Voice stealing** occurs when all voices are occupied and a new note arrives. The oldest voice (by assignment time) is silenced and reused.

Mode 0: Simple Poly

What it does: Standard polyphonic voice allocation. Each incoming note is assigned to the first available voice. When all voices are in use, the oldest voice is stolen.

Voice assignment: First-available scan, with oldest-voice stealing as fallback.

Musical use case: General-purpose polyphonic playing. Use this when you want each note to stay on whichever voice it was assigned to, with no repositioning.

Legato retrigger interaction: When `legato_retrigger` is OFF and the same pitch is pressed again while still held, the duplicate press is ignored (no re-attack). When ON, duplicate presses generate fresh attacks. Additionally, when `maxvoices` is 2 and `legato_retrigger` is ON, Simple Poly switches to a special duophonic bass/lead behavior (see Section 6).

Gotchas: Voice stealing can cause audible clicks with sustaining sounds. This is by design. Pieplate does not perform crossfading.

Mode 1: Poly Shape Stable

What it does: A chord-aware stable allocator that always maps the lowest held pitch to voice 1, the next lowest to voice 2, and so on. As the chord changes, voices are repositioned to maintain this pitch-ordered layout.

Voice assignment: All held notes are sorted by pitch (lowest first). The sorted list is mapped 1:1 to voice slots. When notes are added or removed, the entire layout is recalculated.

Musical use case: Stable pads and chords where you want a predictable relationship between pitch register and satellite voice. For example, voice 1 always handles bass, voice 4 always handles the top note. This lets you apply different effects per register with confidence.

Legato retrigger interaction: Not directly affected by `legato_retrigger`. Instead, use `stable_retrigger_policy` to control whether same-pitch restrikes cause re-attacks.

Gotchas: Without a settle window, fast chord building can cause brief voice swaps as each note re-sorts the layout. Use `settle_window_ms` of 5-15ms to batch rapid changes (see Section 4). Poly Shape always repositions notes on both note-on and note-off events to maintain the lowest-to-voice-1 contract. The `voice_reassign_mode` parameter does not affect this mode. Use Minimal Movement if you need voice-stable behavior with configurable reassignment.

Mode 2: Stack All

What it does: Every incoming note is sent to ALL voices simultaneously, creating a unison stack. All satellites play the same pitch.

Voice assignment: Every note-on is duplicated to voices 1 through `maxvoices`.

Musical use case: Massive unison sounds, especially when each satellite has slightly different detuning, effects, or timbre. Also useful as a mono-synth substitute when paired with `legato_retrigger`.

Legato retrigger interaction: When `legato_retrigger` is ON, Stack All becomes a last-note-priority monophonic note stack. Only one pitch sounds at a time (on all voices). Pressing a new note switches all voices to that pitch. Releasing it restores the previously held note. This is classic mono-synth behavior with time-ordered priority (not pitch-ordered).

Gotchas: Without `legato_retrigger`, holding two notes means all voices received both pitches in sequence. The second note-on steals from the first on every voice. Turn on `legato_retrigger` for clean mono-synth behavior.

Mode 3: Chord Position

What it does: Stateful chord voicing where each note's voice assignment reflects its position within the currently held chord. The first note (by pitch or play order) goes to voice 1, the second to voice 2, and so on. When notes are added or removed, the entire chord is re-voiced.

Voice assignment: Held notes are sorted according to `chord_order` (pitch order or play order), then mapped sequentially to voice slots. Re-voicing occurs on every note-on and note-off.

Musical use case: Chord-based arrangements where voice position must track the musical role of each note. Unlike Poly Shape, Chord Position supports play-order sorting, which preserves the performer's voicing intent rather than forcing pitch order.

Legato retrigger interaction: Not directly affected by `legato_retrigger`. Uses `stable_retrigger_policy` for restrike behavior.

Gotchas: Re-voicing on every event means that pressing C2 after C3 will shift C3 from voice 1 to voice 2 (because C2 is now the lowest). This is intentional and required for correct stateful voicing. Use a settle window to batch rapid chord changes.

Mode 4: Round Robin Forward

What it does: Assigns each new note to the next voice in sequence, cycling forward through voices 1, 2, 3, …, N, 1, 2, 3, …

Voice assignment: A counter advances by 1 on each note-on, wrapping at `maxvoices`.

Musical use case: Arpeggiated or sequenced patterns where you want each successive note to hit a different satellite, creating a predictable cycling texture. Useful for hocket-style effects or distributing a melody across differently-processed voices.

Legato retrigger interaction: When `legato_retrigger` is OFF, duplicate presses of a still-held pitch are ignored.

Gotchas: The counter does not reset between phrases. If you play 3 notes, pause, then play 3 more, the second phrase starts at voice 4 (not voice 1). Use `panic` to reset the counter if needed.

Mode 5: Round Robin PingPong

What it does: Assigns notes in a bouncing pattern: 1, 2, 3, 4, 3, 2, 1, 2, 3, 4, … The direction reverses when reaching either end of the voice range.

Voice assignment: A counter moves forward until it reaches `maxvoices`, then reverses direction. With 1 voice, it stays on voice 1.

Musical use case: Spatial effects where notes sweep back and forth across a stereo or surround field mapped to voice indices. More organic than strict forward cycling.

Legato retrigger interaction: Same as Round Robin Forward.

Gotchas: With 2 voices, PingPong is identical to Forward (1, 2, 1, 2, …).

Mode 6: Round Robin Reverse

What it does: Assigns notes in reverse order: N, N-1, N-2, …, 1, N, N-1, …

Voice assignment: A counter decrements from `maxvoices` down to 1, then wraps back to `maxvoices`.

Musical use case: Same applications as Forward, but starting from the highest-numbered voice. Useful when your spatial layout puts a particular voice at the "end" and you want the sequence to start there.

Legato retrigger interaction: Same as Round Robin Forward.

Mode 7: Key/Velocity Zones

What it does: Assigns notes to voices based on a combination of pitch and velocity. Low pitch + low velocity routes to voice 1; high pitch + high velocity routes to the highest voice. The mapping is continuous and deterministic.

Voice assignment: Pitch (0-127) and velocity (0-127) are each normalized to 0.0-1.0, averaged, then multiplied by `maxvoices` to select a voice index. The same pitch/velocity combination always routes to the same voice.

Musical use case: Timbral layering where soft bass notes should use one satellite and bright high notes another. Creates natural register/dynamic splits without manual zone configuration.

Legato retrigger interaction: When `legato_retrigger` is OFF, duplicate presses of a still-held pitch are ignored.

Gotchas: The split points depend on `maxvoices`. Changing voice count shifts all zone boundaries.

Mode 8: Random Pure

What it does: Assigns each note to a uniformly random voice. Any voice is equally likely, including the same voice as the previous note.

Voice assignment: `Math.random()` scaled to `maxvoices`.

Musical use case: Generative textures where unpredictability is the goal. Each note could land on any satellite, creating chaotic timbral variation.

Legato retrigger interaction: When `legato_retrigger` is OFF, duplicate presses of a still-held pitch are ignored.

Gotchas: With few voices, consecutive notes frequently land on the same voice (voice stealing). Use Random No-Repeat if you want guaranteed variation.

Mode 9: Random No-Repeat

What it does: Random voice assignment that guarantees a different voice from the previous note. The selection is always a single draw with no retries.

Voice assignment: A random value selects from the available voices excluding the previously used voice. With only 1 voice available, repetition is unavoidable.

Musical use case: Randomized textures that always move between voices, ensuring every note triggers a different satellite than the last.

Legato retrigger interaction: Same as Random Pure.

Gotchas: With only 1 voice, repetition is unavoidable (always the same voice).

Mode 10: Random Low Bias

What it does: Random assignment biased toward lower-numbered voices. Voice 1 is selected much more frequently than the highest voice.

Voice assignment: A uniform random value is raised to the power of 2.0, then scaled to `maxvoices`. This squares the distribution, heavily favoring the low end.

Musical use case: When you want randomization but with a "home base" satellite. Voice 1 gets most of the action; higher voices are occasional accents.

Legato retrigger interaction: Same as Random Pure.

Mode 11: Random High Bias

What it does: Random assignment biased toward higher-numbered voices. The highest voice is selected most frequently.

Voice assignment: Same power-curve as Low Bias, but mirrored so the highest voice index is favored.

Musical use case: The inverse of Low Bias. Useful when your highest-numbered satellite has the "default" sound and lower voices are for occasional variety.

Legato retrigger interaction: Same as Random Pure.

Mode 12: Chord Memory

What it does: Captures a chord shape (as a set of intervals from a root note), then expands any single note-on into the full chord across multiple voices. Press one key, hear a chord.

Voice assignment: Each root note-on allocates one voice per chord interval, using Simple Poly allocation for each constituent pitch. When the chord has more tones than available voices, the overflow selection algorithm keeps the outer voicing (see Section 5).

Musical use case: One-finger chord playing. Capture a voicing once, then trigger it from any root. Ideal for live performance where you want complex chords from simple input, or for layering chord stabs across satellites.

Legato retrigger interaction: When `legato_retrigger` is ON, Chord Memory becomes last-note-priority: only one root chord sounds at a time. Pressing a new root silences the previous chord and plays the new one. Releasing a root restores the previously held root's chord. This is monophonic chord behavior.

Gotchas: You must capture a chord template first (see Section 5). Without a template, Chord Memory falls back to Simple Poly for single notes. Use `chord_clear` to reset the template and release any hanging chord notes.

Mode 13: Minimal Movement

What it does: A stable voicing algorithm that minimizes the total pitch movement when reassigning notes to voices. Notes that are already sounding on a voice stay there; new notes are assigned to voices that require the least pitch change.

Voice assignment: Two-phase algorithm: (1) Lock voices that are already playing a desired pitch. (2) Assign remaining notes to remaining voices using a bitmask dynamic programming solver that minimizes total semitone distance. For more than 8 free voices, a greedy fallback is used to avoid latency.

Musical use case: Smooth chord transitions in pad-style playing. When moving from C-E-G to C-E-A, only the voice playing G needs to change. The C and E voices stay untouched. This eliminates unnecessary re-attacks and creates the smoothest possible transitions between chords.

Legato retrigger interaction: Uses `stable_retrigger_policy` for restrike behavior. Not directly affected by `legato_retrigger`.

Gotchas: Legacy overflow modes (Wrap, Clamp) are automatically converted to Lowest-N for this mode, since wrap/clamp semantics are not meaningful for a movement-minimization algorithm. Respects `voice_reassign_mode`.

Mode 14: Legato Duophonic

What it does: Dedicated two-voice mode where voice 1 always tracks the lowest held note (bass) and voice 2 always tracks the highest held note (lead). A single held note plays on bass only; the lead voice activates only when two or more notes are held.

Voice assignment: Held notes are sorted by pitch. The lowest goes to voice 1, the highest to voice 2. All notes in between are ignored (they do not sound). `maxvoices` is automatically bumped to at least 2 when entering this mode.

Musical use case: Bass-and-lead splits for two-hand playing, organ-style bass pedal + melody, or any context where you want exactly two independent pitch-tracked voices without chordal fill.

Legato retrigger interaction: When `legato_retrigger` is ON, any change to the bass or lead pitch causes a retrigger (note-off then note-on). When OFF, retriggers only occur on new note-on events; releasing a note that causes the bass or lead to shift does not retrigger.

Gotchas: Voices beyond 2 are unused in this mode (they never receive notes). Same-pitch restrikes respect `stable_retrigger_policy`.

---

3. Parameters Reference

routing_mode

Property	Value
Range	0-14
Default	0 (Simple Poly)

Selects the active routing algorithm. Changing modes mid-play flushes all active voices (sends note-offs) and resets internal state to prevent stuck notes.

See Section 2 for detailed behavior of each mode.

---

maxvoices

Property	Value
Range	1-16
Default	4

The number of satellite voices available for routing. Changing this value flushes all active voices and reinitializes voice state.

Musical effect: More voices allow richer polyphony but require more satellite instances. Fewer voices force more voice stealing or overflow handling.

Interactions: In Legato Duophonic mode, `maxvoices` is automatically bumped to 2 if set to 1. Key/Velocity Zones distributes the MIDI range across this many zones. Round Robin modes cycle through exactly this many voices.

---

legato_retrigger

Property	Value
Range	0 (off) or 1 (on)
Default	0 (off)

Controls whether overlapping notes of the same pitch generate fresh attacks, and enables special legato behaviors in certain modes.

When OFF: In Simple Poly and sequence modes (RR, Random, Key/Vel Zones), pressing a key that is already held does not generate a new note-on. This prevents accidental double-attacks from overlapping MIDI events.

When ON: Every note-on generates an attack regardless of held state. Additionally: - Stack All becomes a last-note-priority monophonic note stack. - Simple Poly with 2 voices becomes duophonic bass/lead. - Chord Memory becomes last-note-priority monophonic chord mode.

See Section 6 for full legato behavior details.

---

settle_window_ms

Property	Value
Range	0-50ms
Default	0 (disabled)

A timing window that batches rapid note events before finalizing voice assignments. Applies to: Poly Shape Stable, Chord Position, Minimal Movement, Legato Duophonic, and Chord Memory.

Musical effect: When playing chords on a keyboard, individual fingers do not press keys at exactly the same instant. Without a settle window, each finger press triggers a full revoice. A settle window of 5-15ms collects all the near-simultaneous presses and revoices once, producing a cleaner chord attack.

Interactions: Behavior depends on `settle_style`. Setting to 0 disables the window and causes immediate revoicing on every event. See Section 4 for a full explanation.

---

settle_style

Property	Value
Range	0 (Soft) or 1 (Hard)
Default	0 (Soft)

Controls how notes behave during the settle window.

Soft (0): Notes sound immediately using a provisional voice assignment, then the layout is corrected after the settle window expires. You hear every note as soon as it is pressed, but some notes may briefly appear on the "wrong" voice before settling.

Hard (1): No notes sound until the settle window expires. The chord is completely silent during the window, then all notes attack simultaneously with the final voice layout. This produces the cleanest possible chord attack at the cost of added latency.

Interactions: Works in conjunction with `settle_window_ms` and `hard_settle_mute_existing`. See Section 4.

---

hard_settle_mute_existing

Property	Value
Range	0 (off) or 1 (on)
Default	0 (off)

Only relevant when `settle_style` is Hard. When enabled, any currently sounding voices are muted (silenced) as soon as a new note-on begins the hard settle window. The voices re-attack with the final chord layout when the settle window expires.

Musical effect: Without this, previously held notes continue sounding while you build the new chord during the settle window. With it, there is a brief silence as the old chord cuts off and the new chord assembles. This creates cleaner chord-to-chord transitions at the expense of a momentary gap.

Interactions: Only active during hard settle. Has no effect when `settle_style` is Soft or when `settle_window_ms` is 0.

---

chord_order

Property	Value
Range	0 (Pitch) or 1 (Play Order)
Default	0 (Pitch)

Controls how notes are sorted before being assigned to voice slots in Chord Position mode.

Pitch (0): Notes are sorted lowest to highest by MIDI pitch. Voice 1 always gets the lowest note.

Play Order (1): Notes are sorted by the time they were pressed. Voice 1 gets the first note you played, voice 2 the second, and so on.

Interactions: Only affects Chord Position mode (mode 3). Poly Shape Stable always uses pitch order. Minimal Movement does not use sorting for assignment (it optimizes for movement instead).

---

chord_overflow

Property	Value
Range	0-4
Default	0 (Wrap)

Controls what happens when more notes are held than voices are available, in Chord Position, Poly Shape, and Minimal Movement modes.

Wrap (0): Excess notes wrap around to voice 1 and overwrite earlier assignments. The last note to wrap onto a voice wins.

Clamp (1): All excess notes are stacked onto the last voice. Earlier voices are unaffected.

Lowest-N (2): Only the N lowest-pitched notes are kept; higher notes are dropped.

Highest-N (3): Only the N highest-pitched notes are kept; lower notes are dropped.

Keep Edges (4): The lowest and highest notes are always kept. Remaining voice slots are filled from the middle of the chord. This preserves the outer voicing (bass and soprano) while dropping inner notes.

Interactions: When `settle_window_ms` is active, Wrap and Clamp are automatically coerced to Lowest-N for stability (wrap/clamp can produce inconsistent results when note order changes during the settle window). Minimal Movement always coerces Wrap and Clamp to Lowest-N regardless of settle state.

---

voice_reassign_mode

Property	Value
Range	0-2
Default	1 (On Note-On)

Controls when notes are allowed to move between voices during revoicing. Only affects Minimal Movement mode. Poly Shape and Chord Position always reposition on both note-on and note-off events to maintain their positional contracts (lowest pitch on voice 1, etc.).

Dynamic (0): Notes can move between voices on both note-on and note-off events. This always produces the optimal layout but may cause held notes to re-attack when other notes are released.

On Note-On (1): Notes can reposition on note-on events, but not on note-off events. When you release a note, the remaining notes stay on their current voices. This prevents the distracting re-attack that can occur when releasing one note from a chord.

Locked (2): Notes never reposition. Once a note is assigned to a voice, it stays there until released. New notes are placed on available voices. This is the most stable option but may result in non-optimal voice assignments.

Interactions: The settle window's finalize step tracks whether any note-on occurred during the window. If only note-offs occurred, the finalizer respects this parameter by treating the finalize as a note-off context.

---

stable__retrigger__policy

Property	Value
Range	0-2
Default	1 (On Restrike)

Controls whether pressing a key that is already held on the same voice causes a re-attack in stable-voicing modes and Legato Duophonic mode.

None (0): Same-pitch restrikes never cause re-attacks. The voice continues sustaining silently.

On Restrike (1): Any restrike of a sounding pitch causes a note-off/note-on pair, producing a fresh attack. This gives a more "alive" feel for repeated notes.

On Velocity Change (2): Restrikes only cause re-attacks if the velocity difference exceeds `stable_retrigger_vel_thresh`. Subtle velocity variations are ignored; only significant dynamic changes trigger a new attack.

Interactions: Works with `stable_retrigger_vel_thresh` when set to mode 2.

---

stable__retrigger__vel__thresh

Property	Value
Range	0-127
Default	3

The minimum velocity difference required to trigger a re-attack when `stable_retrigger_policy` is set to 2 (On Velocity Change).

Musical effect: At 0, any velocity change triggers a re-attack (equivalent to On Restrike). At higher values, you can restrike a note softly without interrupting the sustaining voice. Only significantly harder or softer restrikes will re-trigger.

---

chord__capture

Property	Value
Range	0 (off) or 1 (on)
Default	0 (off)

Toggles chord capture mode for the Chord Memory system.

When enabled: All currently sounding voices are flushed. Subsequent note-on events are recorded as interval offsets from the first note pressed (the root), but no sound is produced. Note-offs are ignored during capture.

When disabled: Normal routing resumes. The captured intervals are stored as the chord template used by Chord Memory mode.

See Section 5 for the full chord capture workflow.

---

4. Settle Window

The Problem: Micro-Timing Jitter

When a musician plays a chord on a keyboard, the fingers do not land simultaneously. A four-note chord might arrive as four separate note-on messages spread across 3-12 milliseconds. In event-routing modes (Simple Poly, Round Robin, etc.) this is not a problem. Each note is independently assigned and stays put.

In stable-voicing modes (Poly Shape, Chord Position, Minimal Movement), every note-on triggers a full revoice of the entire chord. Without a settle window, a four-note chord causes four revoices in rapid succession. Each intermediate revoice is "correct" for the notes held at that instant, but the voice assignments may shuffle as each new note shifts the layout. This can produce audible artifacts: brief re-attacks, notes jumping between voices, and inconsistent chord attacks.

How the Settle Window Works

The settle window introduces a short delay between receiving a note event and finalizing the voice layout. When a note-on arrives and `settle_window_ms` is greater than 0:

1. A timer starts (or restarts if already running) for the configured duration.
2. Each subsequent note event during the window resets the timer.
3. When the timer expires (no new events for the full window duration), the voice layout is finalized based on all currently held notes.

This "rolling window" behavior means the settle period extends as long as notes keep arriving. Once there is a gap longer than the window, the layout locks in.

Soft Settle

With `settle_style` set to 0 (Soft):

- Notes sound immediately using a provisional voice assignment.
- The new note is placed according to the current stable-voicing algorithm.
- Already-sounding held notes are not repositioned during the provisional assignment (only the new note triggers an attack).
- When the settle window expires, a final revoice corrects any layout inconsistencies.

Trade-off: Notes are always audible with zero latency, but you may hear a brief voice swap when the settle finalizes. In practice, with windows of 5-15ms, this swap is rarely perceptible.

Hard Settle

With `settle_style` set to 1 (Hard):

- New notes do NOT sound during the settle window.
- When `hard_settle_mute_existing` is ON, currently sounding voices are silenced immediately on the first note-on.
- When the settle window expires, all held notes attack simultaneously with the correct final layout.
- Note-offs during a hard settle are handled with a fast finalize (5ms or the settle window, whichever is shorter) to keep releases feeling responsive.

Trade-off: The chord attack is perfectly clean and simultaneous, but there is perceivable latency equal to the settle window. This can feel "sticky" for fast melodic runs. Hard settle is best suited for pad-style chords where a 10-20ms delay is acceptable.

When to Use Each Style

Scenario	Recommended Style
Fast melodies over held chords	Soft, 5-10ms
Slow pad chords, clean attacks	Hard, 10-20ms
Rapid chord stabs	Soft, 5ms or 0
Chord transitions with gap needed	Hard + `hard_settle_mute_existing` ON
Monophonic/duophonic modes	Usually not needed (0ms)

Interaction with voice_reassign_mode

The settle window's finalize step inherits the note-on/note-off context from the events that occurred during the window. If only note-offs occurred during the window, the finalize respects `voice_reassign_mode` as a note-off context (meaning "On Note-On" mode will not reposition notes). If any note-on occurred during the window, the finalize allows repositioning according to the configured mode.

Parameter Change Safety

If you change `settle_window_ms` or `settle_style` while a settle window is active, Pieplate immediately cancels the pending task and reconciles the audible state. For hard settle, this means any muted voices are restored. For soft settle, the current provisional layout becomes final. This prevents stuck-mute or double-play bugs when adjusting parameters during performance.

---

5. Chord Memory

Overview

Chord Memory lets you capture a chord shape and then trigger that entire chord from a single key press. The captured shape is stored as a set of intervals (in semitones) relative to a root note. When you play any note in Chord Memory mode, Pieplate expands it into the full chord by adding those intervals to the played pitch.

How to Capture a Chord Template

1. Enable `chord_capture` (set to 1). All sounding voices are flushed.
2. Press the root note of your chord. This becomes interval 0.
3. Press additional notes. Each is recorded as an interval offset from the root. For example, if you press C3 first, then E3, then G3, the template becomes [0, +4, +7] (a major triad).
4. Disable `chord_capture` (set to 0). The template is stored.
5. Switch to Chord Memory mode (routing_mode 12) if not already active.

Now any single note-on will produce a three-note chord (root + the captured intervals) across three voices.

During capture: Note-offs are completely ignored. You can release keys between presses without losing the template. No sound is produced.

How Expansion Works

When a note arrives in Chord Memory mode with an active template:

1. The incoming pitch is treated as the root.
2. Each interval in the template is added to the root pitch.
3. Resulting pitches outside 0-127 are discarded.
4. If more chord tones exist than `maxvoices`, the selection algorithm keeps the lowest, highest, and fills inward (preserving the outer voicing).
5. Each chord tone is assigned a voice using Simple Poly allocation.

6. The set of voice assignments is tracked as an "instance" tied to the root pitch, so that releasing the root correctly releases all its chord tones.

Legato vs Non-Legato Chord Memory

Without legato_retrigger: Each root note-on creates an independent chord instance. You can play multiple roots simultaneously (polyphonic chord memory). Releasing a root releases only that root's chord tones. Voices are shared across instances via the standard Simple Poly allocator.

With legato_retrigger: Only one chord sounds at a time (monophonic chord memory). Pressing a new root silences the previous chord and triggers the new one. Releasing the current root restores the previously held root's chord. This is last-note-priority behavior, identical to the Stack All legato concept but applied to full chords.

Voice Allocation and Overflow

When the chord template produces more pitches than `maxvoices`:

- The selection algorithm always keeps the lowest pitch.
- If more than 1 voice is available, it also keeps the highest pitch.
- Remaining slots are filled with inner pitches, from low to high.

This means a 7-note chord template with 4 voices will play the root, the highest interval, and two middle intervals. The outer voicing is always preserved.

Settle Window with Chord Memory

When `settle_window_ms` is active: - **Soft:** Chords are played immediately. The settle window has no effect on the chord attack itself (since each chord is triggered by a single root), but it batches rapid root changes. - **Hard:** The chord does not sound until the settle window expires. If the root key is released during the window, no chord plays (ghost-chord prevention).

Multiple roots pressed during a single settle window are queued and all play when the window expires (for non-legato mode).

chord_clear

Sending `chord_clear` to Pieplate: - Releases all currently sounding chord memory allocations (prevents hung notes). - Clears the chord template. - Cancels any pending chord memory settle tasks.

Use this to reset chord memory state without a full panic.

---

6. Legato Behavior

The `legato_retrigger` parameter has different effects depending on the active routing mode. This section describes each interaction in detail.

Stack All Legato (Last-Note-Priority Note Stack)

When `legato_retrigger` is ON and routing mode is Stack All:

- All voices play the same pitch (unison).
- Only one pitch sounds at a time.
- Pressing a new note immediately switches all voices to the new pitch.
- Releasing the current note restores the previously held note (if any).
- Priority is by time (last note pressed), NOT by pitch height.

Example sequence: 1. Press C3: all voices play C3. 2. Press E3 (hold C3): all voices switch to E3. 3. Press G3 (hold C3, E3): all voices switch to G3. 4. Release G3: all voices restore E3 (previous in stack). 5. Release E3: all voices restore C3. 6. Release C3: all voices release.

This is a note-count-safe implementation: pressing C3 twice and releasing once does not silence the note (the held count tracks outstanding note-ons).

Simple Poly Duophonic (2-Voice + Legato)

When `legato_retrigger` is ON, routing mode is Simple Poly, and `maxvoices` is exactly 2:

- Voice 1 always plays the lowest held pitch (bass).
- Voice 2 always plays the highest held pitch (lead).
- A single held note plays on voice 1 only; voice 2 is silent.
- This is functionally identical to Legato Duophonic mode but activated implicitly.

Chord Memory Legato (Last-Note-Priority Root)

When `legato_retrigger` is ON and routing mode is Chord Memory with an active template:

- Only one root chord sounds at a time.
- Pressing a new root silences the previous chord and plays the new one.
- Releasing the current root restores the previously held root's chord.
- Same time-based priority as Stack All legato.

Legato Duophonic Mode

Legato Duophonic (mode 14) has its own legato behavior independent of the `legato_retrigger` parameter:

- Voice 1 (bass): always tracks the lowest held pitch.
- Voice 2 (lead): always tracks the highest held pitch (only when 2+ notes are held).
- `legato_retrigger` ON: retrigger occurs whenever the bass or lead pitch changes, regardless of whether the change was caused by a note-on or note-off.
- `legato_retrigger` OFF: retrigger occurs only on note-on events. When a note-off causes the bass or lead to shift, the transition is smooth (note-off for old pitch, note-on for new pitch, but no re-attack of pitches that did not change).

Same-pitch restrikes in Legato Duophonic are governed by `stable_retrigger_policy`, not by `legato_retrigger`.

Event-Routing Modes (Simple Poly, RR, Random, Key/Vel Zones)

For these modes, `legato_retrigger` controls duplicate-press behavior:

- **OFF:** If a pitch is already held and the same pitch arrives again, the duplicate note-on is silently ignored. The corresponding note-off is also suppressed (count-safe). This prevents double-attacks from overlapping MIDI data.
- **ON:** Every note-on is routed normally, even if the pitch is already sounding on another voice.

---

7. Voice Overflow

When Overflow Occurs

Overflow happens when the number of distinct held pitches exceeds `maxvoices` in a stable-voicing mode (Poly Shape, Chord Position, Minimal Movement). In event-routing modes, overflow is handled by voice stealing instead.

The 5 Overflow Policies

Mode	ID	Behavior
Wrap	0	Excess notes wrap around to voice 1 and overwrite. The highest-indexed overflow note wins each voice slot.
Clamp	1	All excess notes are assigned to the last voice. The last voice plays the highest overflow note.
Lowest-N	2	Only the N lowest pitches are kept. All higher pitches are silently dropped.
Highest-N	3	Only the N highest pitches are kept. All lower pitches are silently dropped.
Keep Edges	4	The lowest and highest pitches are always kept. Remaining slots are filled from the bottom of the middle range upward. Inner-high notes are dropped first.

Which Modes Use Overflow

- **Poly Shape Stable** (1): Uses the configured overflow policy. With settle window active, Wrap and Clamp are coerced to Lowest-N.
- **Chord Position** (3): Uses the configured overflow policy. Same settle-window coercion.
- **Minimal Movement** (13): Always coerces Wrap and Clamp to Lowest-N (wrap/clamp semantics are incompatible with movement minimization).
- **Chord Memory** (12): Has its own overflow handling that preserves outer voicing (lowest + highest + inner fill).
- **Event-routing modes** (0, 2, 4-11): Do not use overflow. Instead, they use voice stealing (oldest voice is replaced).

Practical Guidance

- **Lowest-N** is the safest general-purpose overflow policy. Bass notes are always preserved.
- **Keep Edges** is ideal for open voicings where the soprano and bass lines matter most.
- **Highest-N** is useful for lead-focused playing where bass can be sacrificed.
- **Wrap** and **Clamp** are legacy options retained for backward compatibility. They can produce surprising results with settle windows and are automatically overridden in those contexts.

---

8. MIDI Broadcast

Pitch Bend

Pitch bend messages are broadcast to all voices (1 through `maxvoices`), regardless of whether each voice is currently active. The bend value range is 0-16383, with 8192 as center (no bend). Every satellite receives the same bend amount simultaneously.

Control Change (CC)

CC messages are broadcast identically to all voices (1 through `maxvoices`), regardless of active state. Any CC number (0-127) and value (0-127) is forwarded to every satellite.

Mod Wheel

A convenience shortcut for CC 1. Calling `modwheel(value)` is equivalent to `ctlin(1, value)`. It broadcasts CC 1 to all voices.

Implications

Because MIDI messages are broadcast to ALL voices, per-voice expression is not possible through Pieplate's broadcast mechanism. If you need independent pitch bend or CC per voice, route those messages directly to individual satellites outside of Pieplate.

---

9. Panic and Safety

Panic

Sending `panic` (or `panic_button` or `panic!`) to Pieplate triggers a full emergency reset:

1. All pending settle tasks are canceled (stable modes, legato duo, chord memory).
2. An all-notes-off message (velocity 0) is sent for every MIDI pitch (0-127) on every possible satellite voice (1-16), regardless of the current `maxvoices` setting. This is intentionally aggressive to guarantee no stuck notes on any satellite.
3. The chord template is cleared.
4. Chord capture mode is disabled.
5. All internal state is reinitialized (voices, note stacks, round robin counters, random state, legato tracking).

When to use panic: Stuck notes after a crash, after USB MIDI disconnection, or whenever the audible state does not match what you expect. Panic is the nuclear option. It resets everything.

chord_clear

A lighter-weight reset that only affects Chord Memory state:

- Releases all sounding chord memory allocations.
- Clears the chord template.
- Cancels pending chord memory settle tasks.
- Does NOT affect other modes, voices, or the settle system.

Mode Change Flush

Changing `routing_mode` automatically flushes all active voices (sends note-offs) and resets internal state. This prevents stuck notes when switching between modes mid-performance. You do not need to manually panic when changing modes.

maxvoices Change Flush

Changing `maxvoices` also flushes all voices and reinitializes state, for the same reason.

---

10. Tips and Recipes

Mono Synth with Portamento Feel

Mode: Stack All (2) | **Voices:** 4-8 | **legato_retrigger:** ON

With legato retrigger enabled, Stack All becomes a last-note-priority monophonic unison. All satellites play the same pitch. Configure each satellite with slight detuning and portamento/glide. The legato note stack ensures smooth pitch transitions: press and hold a new note to glide; release to glide back. Multiple satellites with different glide times create a rich, evolving unison.

Stable 4-Voice Pad

Mode: Poly Shape Stable (1) | **Voices:** 4 | **settle_window_ms:** 10 | **settle_style:** Soft | **voice_reassign_mode:** On Note-On (1)

The settle window batches your chord press into a single revoice, and "On Note-On" reassignment means releasing one note from the chord will not cause the remaining notes to shuffle and re-attack. Each satellite can have a different filter, reverb, or pan setting for a wide, immersive pad.

One-Finger Chord Player

Mode: Chord Memory (12) | **Voices:** 4-6 | **legato_retrigger:** ON

Capture a jazz voicing (e.g., root, minor 3rd, 7th, 9th) with chord capture. Enable legato retrigger so only one chord sounds at a time. Now play a bass line with one hand. Each note triggers the full voicing, and legato transitions create smooth chord movement. Each satellite handles one chord tone with its own timbre.

Bass + Lead Split

Mode: Legato Duophonic (14) | **Voices:** 2 | **legato_retrigger:** ON

Voice 1 always tracks your lowest note (left hand bass), voice 2 tracks your highest note (right hand lead). Route satellite 1 to a bass sound and satellite 2 to a lead synth. For a similar effect with Simple Poly, set maxvoices to 2 and enable legato_retrigger. The duophonic behavior activates automatically.

Randomized Texture

Mode: Random Low Bias (10) | **Voices:** 8 | **legato_retrigger:** OFF

Configure 8 satellites with different timbres, filters, or spatial positions. Low bias means most notes land on the lower-numbered satellites (your "primary" sounds), with occasional notes hitting higher-numbered satellites (your "accent" sounds). The result is a mostly-consistent texture with unpredictable timbral flickers.

Arpeggiated Cycling

Mode: Round Robin Forward (4) | **Voices:** 4-6

Each successive note hits the next satellite in order. If your satellites have different delay times, filter settings, or spatial positions, a monophonic melody becomes a cycling texture that rotates through timbres. Combine with an arpeggiator before Pieplate for automated cycling through all satellites.

Smooth Chord Transitions

Mode: Minimal Movement (13) | **Voices:** 4 | **settle_window_ms:** 8 | **settle_style:** Soft | **voice_reassign_mode:** On Note-On (1)

Minimal Movement keeps notes on their current voices whenever possible, minimizing re-attacks during chord changes. Moving from Cmaj to Fmaj (C-E-G to C-F-A) only re-triggers the two voices that change pitch. The C voice sustains through. Use this for ambient or cinematic textures where smooth voice leading matters.

Clean Chord Stabs

Mode: Chord Position (3) or Poly Shape Stable (1) | **Voices:** 4 | **settle_window_ms:** 15 | **settle_style:** Hard | **hard_settle_mute_existing:** ON

Hard settle with mute-existing creates a brief silence between chord stabs, ensuring each chord attacks cleanly as a unit. The 15ms window is long enough to capture a full chord press but short enough to feel responsive. All four notes attack simultaneously when the window expires.

Key-Split Layering

Mode: Key/Velocity Zones (7) | **Voices:** 4

With 4 voices, the MIDI range is divided into four zones based on combined pitch and velocity. Low/soft notes go to satellite 1 (a warm pad), mid-range to satellites 2-3 (different lead sounds), and high/loud notes to satellite 4 (a bright stab). The zones are automatic and require no manual split-point configuration.