On Tap — DSP Manual

Carbonated Audio · v1.0.0 Tempo-synced sidechain ducker for modern mix engineers.

This document explains the signal processing inside On Tap from the top down: what each control does mathematically, why it's there, and how to reason about it on a mix. It's written for engineers and producers who want a real understanding of what the plug-in does, not just which knobs make things louder.

---

1. Overview

On Tap is a **tempo-synced amplitude modulator** — a sidechain ducker that doesn't need a sidechain. Instead of detecting a kick drum and reacting to it, On Tap runs a mathematically-defined **ducking curve** against the host's transport or an incoming MIDI trigger, pumping the signal on a rhythmic grid you choose.

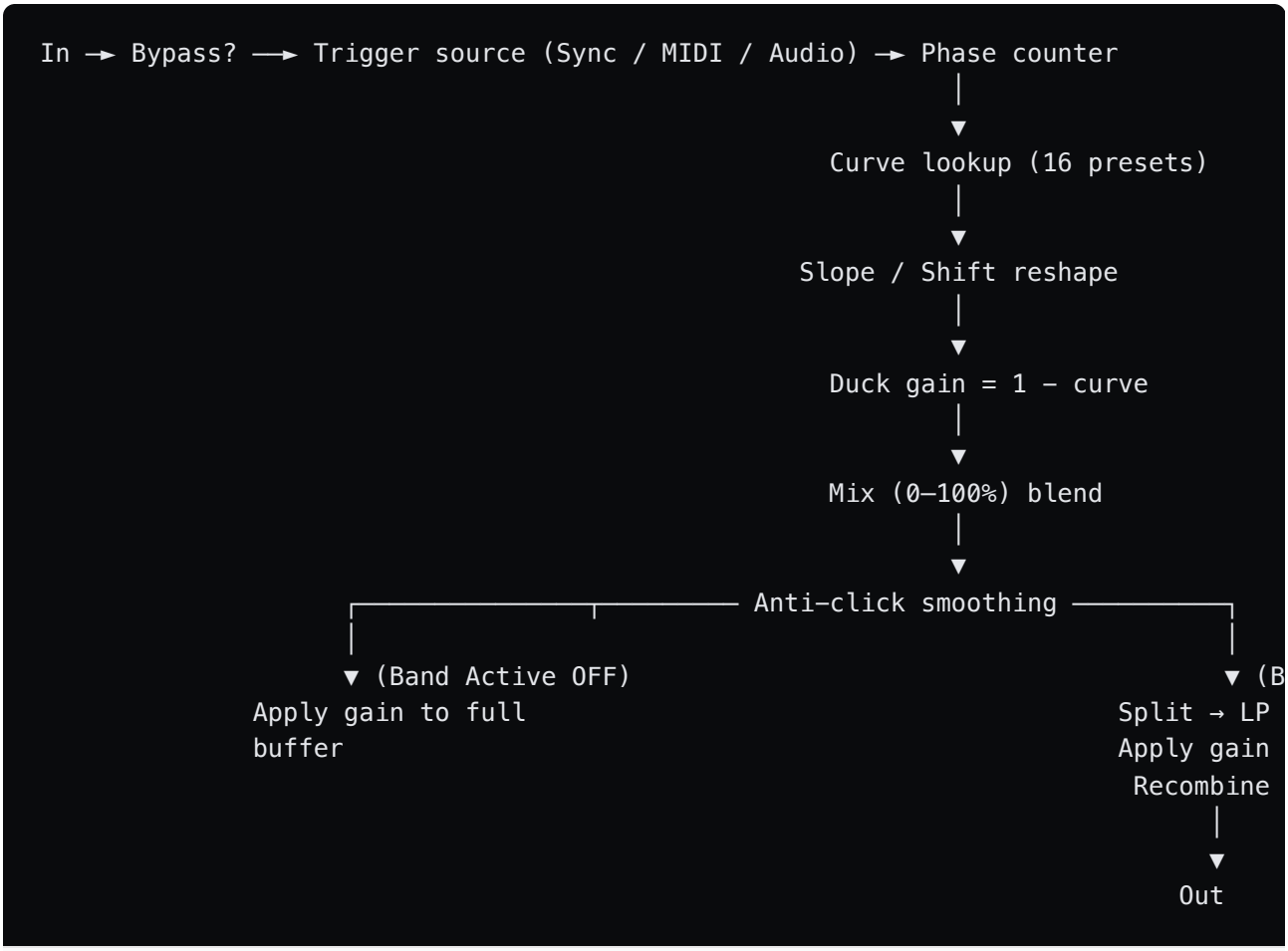
Everything in On Tap reduces to three core ideas:

1. **Pick a trigger** — the host's PPQ clock, an incoming MIDI note-on, or (in the future) a sidechain audio trigger.
2. **Sample a curve** — a pre-computed envelope shape that describes how the gain should rise and fall across one cycle.
3. **Apply the curve** — multiply the audio by the resulting gain, optionally on a single band if you only want to duck the low end.

There's no detector, no threshold, no attack/release to tune. The "sidechain envelope" is whatever curve you pick, stretched and shifted by the Slope and Shift controls, looped at the tempo division you set.

---

2. Signal Flow



Every stage runs **per sample** in the gain-application loop, so automation and knob tweaks are click-free when Anti-Click Smoothing is on.

---

3. Trigger Modes

On Tap supports three ways of knowing when to start a ducking cycle:

Sync (default)

The plug-in reads the host's `ppqPosition` (position in quarter notes since the start of the transport) and computes a phase value from 0.0 to 1.0:

```
phase = (ppqPosition mod lengthBeats) / lengthBeats
```

where `lengthBeats` is determined by the Length control (see Section 4). This means the ducking cycle is **locked to your DAW's grid** — start the transport and every cycle aligns exactly to a beat, bar, or bar-division.

When transport is stopped, phase is held at 0 (cycle does not advance).

MIDI

The plug-in listens for MIDI note-on messages on its input bus. Any note-on sets `triggered = true` and resets the phase to 0. The phase then advances at a fixed rate derived from a nominal BPM of 120 and the Length setting:

```
phaseIncrement = (120/60) / (sampleRate × lengthBeats)
```

When phase reaches 1.0, the cycle either loops (if **Loop** is on) or latches at 1.0 (silent, no ducking) and waits for the next note-on.

Use this when you want a ducking cycle gated to specific beats — e.g. feeding it from a kick MIDI clip to get chops on kick hits only.

Audio

Reserved for future revision. In v1.0 the mode is present but not active — phase advances based on the MIDI-style trigger state.

---

4. Length — Cycle Division

Options: 1/8, 1/4, 1/2, 1/1 — default 1/4.

Sets the length of one full ducking cycle in musical beats:

Setting	Beats per cycle
1/8	0.5 (eighth note)
1/4	1.0 (quarter note — default)
1/2	2.0 (half note)
1/1	4.0 (whole note / one bar in 4/4)

Shorter values give you per-beat chops; longer values give you slow, swelling fades over a whole bar. In Sync mode, phase resets exactly on the grid — the first duck starts exactly on beat 1.

---

5. Curve Presets

On Tap ships with **16 pre-computed envelope curves**, lookup-table driven for zero-overhead per-sample access. Each is a 1024-point table stored as `tables[preset][position]`, where position is the current cycle phase (0.0 to 1.0).

Each curve value means "how much signal to let through at this point in the cycle" — 1.0 is full volume, 0.0 is silent. The final duck gain is the inverse (1 – curve), so a curve value of 1.0 at phase 0 means **no ducking at the top of the cycle**, and dropping toward 0 means **increasing ducking**.

#	Name	Shape
0	Sharp Kick Duck	Very fast attack, exponential release. `curve(x) = exp(−8x)·0.9` after a 5% flat head.
1	Medium Duck	Exponential release, `exp(−4x)`.
2	Slow Duck	Gentler exponential, `exp(−2x)`.
3	Linear Ramp Down	`1 − x`. The most predictable shape.
4	S-Curve (default)	Logistic sigmoid, `1 − 1/(1 + exp(−12(x − 0.5)))` — the classic "sidechain pump" shape.
5	Punchy	Fast dip then half-amplitude exponential recovery.
6	Half Duck	Exponential for the first half of the cycle, silent for the second half.
7	Smooth Cosine	Raised cosine, `(1 + cos(πx))/2` — the most symmetric, pillowy shape.
8	Double Pump	`|sin(2πx)| × exp(−3x)` — two attenuations per cycle for rapid gating.
9	Sub Bass Smooth	Linear ramp, `x`. Opens into the cycle rather than ducking — useful for "swell" effects.
10	Stepped	Three-stage ladder: 1.0 → 0.5 → 0.0 at thirds of the cycle.
11	Gentle Decay	Slow exponential, `exp(−1.5x)`.
12	Aggressive	2% flat head then very fast exponential — reverse-chain style.
13	Wobble	Exponentially-decayed cosine, `exp(−3x)(0.5 + 0.5·cos(6πx))`.
14	Flat Top Decay	Holds 1.0 for 30% of cycle, then exponential decay.
15	Quick Release	Fast linear dip, long quiet tail.

The preset bank is generated on construction and never recomputed, so changing presets is a pointer swap — no audio artefact.

---

6. Slope Offset

Range: –1.0 to +1.0, default 0.0.

Stretches or compresses the sloped portion of the curve by applying a power remapping to the position value **before** the lookup:

```
power = 2^(−slopeOffset × 2)
position' = position^power
```

At slopeOffset = 0, power = 1, position is unchanged (neutral). At slopeOffset = +1, power = 0.25 — the curve is "stretched" so the slope happens earlier. At slopeOffset = –1, power = 4 — the curve is "compressed" so the slope happens later.

Musically this gives you a **single-knob attack/release shaper** — push positive for a faster initial dip, pull negative for a lazier one, with the total cycle length unchanged.

---

7. Shift

Range: –1.0 to +1.0, default 0.0.

Phase-shifts the curve inside its cycle. Applied before the slope remap:

```
position' = (position + shift + 10) mod 1.0
```

The `+10` is there to guarantee a positive value before `mod` for negative shift values. The net effect: the curve starts at a different point of the cycle, letting you dial in where the duck lands relative to the beat. Shift = +0.25 rotates the curve a quarter of a cycle to the right; Shift = –0.5 flips it to the opposite half.

Use Shift to "delay" or "pre-duck" against the beat without touching Length.

---

8. Mix

Range: 0% to 100%, default 100%.

Parallel dry/wet blend on the ducked gain. Computed per sample:

```
targetGain = 1 − (Mix × (1 − duckGain))
```

At Mix = 100%, the full duck is applied (targetGain = duckGain). At Mix = 50%, the duck amount is halved — a 20 dB dip becomes ~10 dB. At Mix = 0%, the plug-in is effectively inactive (targetGain = 1.0 constant).

This lets you dial in subtle pumping without picking a different curve preset, and is the right control to automate if you want the ducking to "build" into a chorus.

---

9. Loop

Boolean, default ON.

Only relevant in MIDI (and future Audio) trigger modes.

- **Loop ON** — after each cycle completes, phase wraps back to 0 and the curve repeats until a new trigger arrives.
- **Loop OFF** — the cycle runs once per trigger, then phase latches at 1.0 (full volume, no ducking) until the next MIDI note-on.

In Sync mode, Loop is effectively always on — the host's PPQ doesn't stop advancing until you stop the transport.

---

10. Anti-Click Smoothing

Boolean, default ON.

Controls whether the gain-change smoother runs across sample changes.

- **Smoothing ON** — a one-pole lowpass smoother sits on `smoothedGain`, with a time constant of ~6 ms (computed as `6/1000 × sampleRate` samples). This guarantees any abrupt curve transition (e.g. the stepped preset crossing a boundary, or a shift-automation movement) fades smoothly across the change.
- **Smoothing OFF** — gain is applied as-is, sample-accurate. You get sharper transients on stepped curves, at the cost of possible clicks.

Default to ON unless you specifically want the stepped behaviour (preset 10) to click on every boundary.

---

11. Band Split (Low-Band Only Ducking)

Band Active: Boolean, default OFF. **Band Frequency:** 20 Hz to 5,120 Hz, default 320 Hz, logarithmic skew.

When Band Active is on, On Tap splits the incoming audio into **two bands** using a Linkwitz-Riley crossover, ducks **only the low band**, and recombines:

1. **Split** — input runs through both a 2nd-order Linkwitz-Riley lowpass and a matched highpass at the Band Frequency. (JUCE's `juce::dsp::LinkwitzRileyFilter` — the split is phase-coherent, so recombining at unity is flat.)
2. **Duck the low band** — the per-sample gain curve is applied to the low band only.
3. **Recombine** — low + high summed back into the output.

Why? Full-range ducking is great for creating space for a kick on a bus, but on a full mix it kills your cymbals, vocal sibilance, and reverb tails. Band Split lets you chop just the sub-kick energy without touching the top end — modern "side-chain only the bass" approach used on modern EDM and pop mixes.

When to use: - **Active OFF** — full-range ducking. Good for buses, drum loops, pads sitting behind kicks. - **Active ON, ~100 Hz** — just the sub rumble ducks out of the way of the kick fundamental. - **Active ON, ~300 Hz** — ducks kick + bass body, leaves guitars, vocals, cymbals, snare all unaffected. - **Active ON, ~800 Hz** — aggressive, affects most of the low midrange. Less musical; creative tool.

---

12. Bypass

Global bypass, checked first in the process loop. When true, the audio passes through untouched at unity gain and none of the DSP is executed.

---

13. Latency and CPU

On Tap is **zero-latency**. No FFT, no lookahead, no oversampling.

CPU cost is dominated by: - A single LUT lookup + linear interpolation per sample (always cheap) - One multiply per sample for the gain application - Two 2nd-order biquads per channel when Band Split is active (slight increase)

Expect roughly the cost of a gate or a tempo-synced LFO. Safe to stack dozens of instances in a session.

---

14. Channel Configuration

- **Main I/O:** stereo in, stereo out. Mono supported.
- **MIDI input:** accepts note-on messages for MIDI trigger mode (the plug-in declares `NEEDS_MIDI_INPUT = TRUE` in its CMake manifest).
- **Sample rates:** all sample rates supported. Crossover coefficients recompute when sample rate changes.
- **Block sizes:** arbitrary.

---

15. Credits and References

- **Linkwitz-Riley crossover** — S. Linkwitz, *Active Crossover Networks for Noncoincident Drivers*, JAES 1976. The reason the two-band split recombines flat.
- **Robert Bristow-Johnson**, *Cookbook formulae for audio EQ biquad filter coefficients* — source for the filter math JUCE's `dsp::IIR::Coefficients` uses internally.

Questions, bug reports, or sound-design notes: mixedbysoda@gmail.com.