Intellijel — Plog

- Manual PDF

Manual PDF - Intellijel Plog rev 1.0

Using the Intellijel Plog to Create Melodic Components

The **Intellijel Plog** is a digital logic and flip-flop module, so it does not generate pitch CV by itself. But in a Eurorack system, it is extremely useful for creating **melodic structure** by manipulating clocks, gates, triggers, and binary states that drive sequencers, quantizers, sample-and-holds, switches, and envelopes.

What the Plog gives you musically

The module contains:

- **Two 3-input logic sections**: A and B
  Logic types:
- AND
- OR
- NOR
- XOR
- NAND

- XNOR

- **One toggle flip-flop (T)**

- **One data flip-flop (D)**
- **CV over logic type** for sections A and B

- **Tap tempo mode** on the trigger button

Important internal normaling

These normals are what make the module especially useful as a melodic utility:

- **Z is normalled to Y**
  So each 3-input logic block can behave like a 2-input logic block if only X and Y are patched.
- **X and Y of A are normalled to X and Y of B**
  So one pair of rhythm sources can feed both logic channels at once.
- **Logic B can be normalled to the toggle input via rear jumper**
  This lets logic B directly drive the T flip-flop.
- **OUT T is normalled to CLK of the D flip-flop**
- Default flip-flop behavior effectively creates:
- **/2 at OUT T**
- **/4 at OUT D** from an incoming clock or logic rhythm

This means Plog is very good at building **derived rhythmic control signals** from one or two master streams.

Best way to think about it for melody

Plog is not a melody source in the traditional sense. It is a **melodic organizer**.

It helps create:

- note timing
- phrase resets

- octave jumps
- alternating note choices
- stepped binary patterns
- rhythmically gated pitch changes
- structured randomness

If you pair it with modules like:

- sequencers
- quantizers
- sample & hold
- switches
- clock dividers
- sequential switches
- precision adders
- envelope generators
- VCAs

then it becomes very powerful for melody generation.

Core melodic use cases

1. Create note trigger patterns for a sequencer voice

Use Plog to derive interesting gate streams from simple clocks.

Patch idea

- Clock 1 -> X of A
- Clock 2 or Euclidean rhythm -> Y of A
- OUT A -> trigger input of envelope or sequencer advance

Musical results by logic type

- **AND**: sparse, only coincident hits
  Great for reducing density and making selective note events.
- **OR**: denser melodic rhythm
- **XOR**: syncopated, “difference” rhythm
- **NOR/NAND/XNOR**: inverted and more unusual gate behavior

Why this helps melody

If your sequencer only advances when OUT A fires, then the logic block becomes a **melodic rhythm composer**.

---

2. Use A and B to create two related trigger streams

Because A and B share normalled X and Y inputs, you can derive **two different gate patterns** from the same source material.

Patch idea

- Master clock pattern -> X of A
- Second rhythm or gate sequence -> Y of A
- Set:
- A = XOR
- B = AND
- OUT A -> sequencer clock
- OUT B -> envelope reset, accent, or second sequencer clock

Musical use

This creates two musically related but different streams: - one drives pitch movement - the other drives accents, octave shifts, ratchets, resets, or a second voice

This is excellent for making melody and accompaniment feel connected.

---

3. Use the T flip-flop as an alternating note selector

The **toggle flip-flop** changes state each time it receives a trigger.

Patch idea

- Steady trigger stream -> TOGGLE
- OUT T -> sequential switch control / precision adder gate / VCA gate
- Two different pitch voltages -> switch inputs
- Switch output -> quantizer or oscillator 1V/oct

Result

Every trigger alternates between two pitch sources: - note A / note B / note A / note B...

Musical applications

- alternating bass notes
- root/fifth patterns
- call-and-response between two melodic cells
- octave switching every other note

This is one of the most directly melodic uses of Plog.

---

4. Use T and D together as binary phrase structure

By default: - **OUT T = divide by 2** - **OUT D = divide by 4**

So from one clock, you get slower square-wave states.

Patch idea

- Clock -> TOGGLE
- OUT T -> switch between two pitch rows
- OUT D -> transpose up an octave every 4 steps

Result

You get binary phrase architecture such as: - steps 1–2 one note set - steps 3–4 alternate set - every four steps add a transposition or change source

This creates very stable melodic form with minimal patching.

Example uses

- OUT T controls a sequential switch between 2 melodic CV sources
- OUT D opens a VCA that adds +1V through a precision adder
- or OUT D resets a sub-sequencer every 4 beats

---

5. Clock a sample-and-hold only on specific logic events

This is one of the strongest melodic patches with a logic module.

Patch idea

- Random CV or noise -> Sample & Hold input
- Plog OUT A -> Sample & Hold clock
- Sample & Hold out -> quantizer -> oscillator pitch

Set the inputs

- X = steady pulse train
- Y = slower gate pattern
- A = AND, XOR, or XNOR

Result

The random pitch only updates when the logic condition is met.

Why it works musically

Instead of random notes on every beat, Plog gives you **conditional note changes**: - only when two rhythms coincide - only when one rhythm differs from another - only on alternating structural states

That creates melodic phrases that feel intentional rather than chaotic.

---

6. Use logic outputs to control when pitch changes vs when notes repeat

A great melodic trick is separating: - **when a note is heard** from - **when pitch is updated**

Patch idea

- Main clock -> envelope trigger
- Plog OUT A -> sequencer advance or S&H clock
- oscillator always plays from main clock
- pitch source only changes on OUT A events

Result

Some note triggers repeat the same pitch, while others advance to a new one.

Musical feel

This produces: - repeated notes - motifs - held tones - irregular phrase lengths

Plog becomes a phrase engine rather than just a gate combiner.

---

7. CV-scan between logic types for evolving melodic rhythm

Each logic block has **CV control over logic type**.

That means the relationship between X/Y/Z and the resulting rhythm can change over time.

Patch idea

- Clock 1 -> X
- Clock 2 -> Y
- Slow LFO or stepped random voltage -> TYPE A CV
- OUT A -> sequencer clock or S&H clock

Result

The generated trigger pattern evolves between AND / OR / NOR / XOR / NAND / XNOR.

Why this is good for melody

Instead of manually changing rhythm logic, you get: - evolving note density - phrase mutation - variation without losing sync - semi-generative melodic timing

This is one of the most distinctive Plog features.

How the flip-flops help melody specifically

Toggle flip-flop (T)

Receives triggers and flips state each time.

Useful for: - alternating between two notes - alternating between two sequencers - creating even/odd step behavior - generating octave up/down every other note - creating slower phrase gates from faster clocks

Data flip-flop (D)

Latches incoming DATA value only when CLK rises.

Useful for: - sampling gate states into phrase memory - holding binary decisions - locking in transposition states - creating structured, clocked variation

Patch idea

- Random gate or logic signal -> DATA
- Clock -> CLK
- OUT D -> enable transposition / switch melody source / reset sequencer

This lets you create a decision that only updates on a clock edge, which is very musical.

Melodic patch examples

Patch 1: Quantized random melody with rhythmic intelligence

Modules needed

- Plog
- noise/random CV
- sample & hold
- quantizer
- oscillator
- envelope/VCA

- two clock sources

Patch

- Clock 1 -> X of A
- Clock 2 -> Y of A
- Set A = AND
- OUT A -> S&H clock
- Random CV -> S&H input
- S&H out -> quantizer -> oscillator pitch
- Main clock or OUT A -> envelope trigger

Result

The melody changes only at rhythmic coincidences, producing a structured random line.

Patch 2: Two-note alternating bassline

Patch

- Master clock -> TOGGLE
- Two fixed voltages or two sequencer rows -> switch inputs
- OUT T -> switch control
- Switch output -> quantizer or oscillator pitch
- Master clock -> envelope trigger

Result

Alternating bass intervals: - root / fifth - root / octave - low / high variation

Very effective for Berlin-school or electro patterns.

Patch 3: Phrase transposition every 4 beats

Patch

- Clock -> TOGGLE
- OUT D -> gate a precision adder adding +1V or +0.583V etc.
- Main sequencer CV -> precision adder -> oscillator pitch

Result

The same melodic pattern transposes every fourth cycle or phrase segment.

This makes a simple sequence sound composed.

Patch 4: Logic-driven sequencer advance and reset

Patch

- Clock -> X
- Slow clock -> Y
- A = XOR
- B = AND
- OUT A -> sequencer advance
- OUT B -> sequencer reset or stage select

Result

The sequence advances irregularly but resets at related moments, creating looping melodic cells.

Patch 5: Binary melody router

Patch

- Two melodic CV sources -> switch inputs
- OUT T -> switch control
- OUT D -> second switch or transpose gate
- Clock into TOGGLE
- Optional logic B drives TOGGLE via rear jumper

Result

A compact two-bit phrase system: - T decides between source A/B - D decides whether phrase is transposed or not

This gives simple but highly musical 4-step/8-step structural behavior.

Particularly strong pairings with other module types

With a quantizer

Plog is excellent before a quantizer when used to decide: - when random CV is sampled - when transposition occurs - when sequencers advance

With sample & hold

This is probably one of the best pairings. Plog makes random melodic generation feel clocked and intentional.

With sequential switches

Very strong. The flip-flops create binary control states that are ideal for switching between: - two note rows - two transpositions - two modulation sources

With precision adders

Use T or D outputs as gates that add interval offsets: - octave jumps - fifths - modal shifts

With sequencers

Use logic outputs to: - clock them - reset them - select rows - enable glide/accent/transposition

---

Musical character of each logic type for melody

AND

- sparse
- selective
- “only when both agree”
- great for controlled melodic changes

OR

- active
- dense
- useful for busier melodic movement

XOR

- syncopated
- surprising
- excellent for generative patterns

NOR

- inverted silence logic
- useful for negative space and off-beat events

NAND

- mostly active except on coincidences
- good for “dropout” phrasing

XNOR

- high when inputs match
- useful for locked/paired rhythm behaviors

Practical voltage notes from the manual

Inputs expect: - **Logic/triggers/clocks: 0–5V** - **CV for type: +/-5V**

The logic/trigger inputs use comparators with about a **3V threshold**, so non-square signals can still work if they cross that threshold.

That means: - square LFOs work well - many envelopes or triangles can work as logic sources - audio-rate signals may also produce interesting results if they exceed threshold

---

One especially useful built-in melodic trick

The manual notes that with no extra patching, the flip-flop section behaves as: - **divide by two** - **divide by four**

This is very useful musically because even a plain clock can immediately become: - every beat - every other beat - every fourth beat

Those slower binary layers are perfect for: - phrase changes - octave toggling - note source switching - transposition timing

---

Bottom line

The **Intellijel Plog** is best used for melody as a **control-logic brain**, not a pitch generator.

It excels at:

- generating note trigger patterns
- deciding when pitch changes occur
- alternating between pitch sources
- building phrase-level binary structure
- creating transpositions on fixed rhythmic divisions
- making random melodies feel intentional
- evolving melodic timing through CV-controlled logic selection

If you combine it with: - a quantizer - sample & hold - one or two sequencers - a switch or precision adder

you can build very sophisticated melodic systems from very little material.
