# **Pre-Coding Essentials (Component: crates/vm\_core/src/variables.rs, Version/FormulaID: VM-ENGINE v0) — 25/89**

## **1) Goal & Success**

Goal: Define **typed variables** (VM-VAR-###) and a Params struct with **defaults + domain validation**, independent of I/O.

Success: Params::default() matches spec defaults; validate\_params(&Params) enforces **ranges/enums/conditionals**; no cross-artifact checks here; optional serde derives behind feature.

## **2) Scope**

In scope: enums for each family (ballot, allocation, gates, weighting, frontier, ties, MMP), Params with typed fields, default constants, domain validation.

Out of scope: schema parsing/JSON (in vm\_io), pipeline semantics (state machine, gating math), Formula ID hashing (Annex A lives elsewhere).

## **3) Inputs → Outputs**

Inputs: None at runtime; callers provide either defaults or values (from vm\_io).

Outputs: Params (typed snapshot), accessors like is\_random\_ties(), frontier\_enabled().

## **4) Entities/Tables (minimal)**

## **5) Variables (only ones used here)**

## **6) Functions (signatures only)**

rust

CopyEdit

pub struct Params { /\* typed fields for all VM-VARs \*/ }

impl Default for Params { fn default() -> Self } // spec defaults

pub fn validate\_params(p: &Params) -> Result<(), VarError>; // domain (ranges/enums/iff)

pub fn is\_frontier\_enabled(&self) -> bool;

pub fn is\_random\_ties(&self) -> bool;

pub fn pr\_threshold(&self) -> Option<u8>; // normalized helper

// (When serde feature on)

#[cfg(feature = "serde")]

pub fn to\_var\_map(&self) -> BTreeMap<String, serde\_json::Value>;

#[cfg(feature = "serde")]

pub fn from\_var\_map(m: &serde\_json::Map<String, Value>) -> Result<Params, VarError>;

## **7) Algorithm Outline (module layout)**

**Enums** per family (derive Copy, Clone, Eq, PartialEq, Debug, Ord, PartialOrd, Hash; plus serde with rename\_all="snake\_case" when feature on).

**Defaults**: const DEF\_\* for every field; impl Default for Params assembles them.

**Validation** (validate\_params):

**Ranges**: all % in 0..=100; specific caps: pr\_threshold ≤ 10, topup\_share ≤ 60.

**Iff rules**:

BallotType::Score ⇒ scale\_min < scale\_max; allow/deny normalization per enum.

BallotType::RankedCondorcet ⇒ condorcet\_rule present.

BallotType::RankedIrv ⇒ IrvExhaustion == ReduceContinuingDenominator.

AllocationMethod::MixedLocalCorrection ⇒ require 013–017 set with valid ranges.

DoubleMajority=On ⇒ require PartitionBasis and either non-empty PartitionFamily when ByList or a valid tag basis when ByTag.

TiePolicy::Random ⇒ rng\_seed is 64-hex.

FrontierMode != None ⇒ bands non-empty and each min ≤ max (non-overlap left to pipeline).

**Consistency**: DeterministicOrderKey must equal OptionOrderIndex when TiePolicy::DeterministicOrder.

**Helpers**: boolean predicates and small normalizers (e.g., clamp functions are **not** used—reject instead).

## **8) State Flow (very short)**

vm\_io builds Params from JSON → validate\_params → vm\_pipeline consumes to drive step order and algorithm switches.

## **9) Determinism & Numeric Rules**

All numeric fields are **integers**; no floats.

No RNG here beyond holding a **seed string**; algorithms consume it deterministically.

## **10) Edge Cases & Failure Policy**

Missing mandatory knobs for chosen mode (e.g., MMP without 013–017) ⇒ **VarError::MissingField**.

Bad hex or wrong length for seed ⇒ **VarError::BadSeed**.

Frontier bands empty when mode ≠ None ⇒ \*\*VarError::InvalidBands`.

Setting GateDenominatorMode to anything but ValidBallots ⇒ \*\*VarError::Unsupported` (locked by spec).

This module does **not** enforce WTA magnitude=1; that’s a pipeline validation.

## **11) Test Checklist (must pass)**

Params::default() values match spec defaults exactly.

Score mode: min<max passes; min>=max fails.

IRV: any exhaustion other than ReduceContinuingDenominator fails.

Random ties without 64-hex seed fails; with valid seed passes.

MMP: missing any of 013–017 fails; valid ranges pass.

Frontier: mode=None with bands present fails; mode≠None with empty bands fails; bands with min≤max pass domain check (overlap caught later).

Serialization (when serde on): round-trip to\_var\_map/from\_var\_map preserves values and enums.