# **Pre-Coding Essentials (Component: Makefile, Version/FormulaID: VM-ENGINE v0) — 4/89**

## **1) Goal & Success**

Goal: One-command, deterministic build/test/package pipeline, mirroring Docs 3–6 constraints (offline, locked deps, canonical artifacts).

Success: make ci runs fmt→lint→build→test→determinism checks; make dist produces reproducible CLI bundles; no network at build/test time.

## **2) Scope**

In scope: Developer/CI targets (fmt, clippy, build, test, run fixtures, hash/verify, dist).

Out of scope: Per-crate code, report templates, schema content.

## **3) Inputs → Outputs**

Inputs: Rust toolchain pin (rust-toolchain.toml), Cargo workspace, fixtures under fixtures/annex\_b/\*, schemas, .cargo/config.toml.

Outputs:

Build: target/{debug,release}/…

Artifacts: dist/vm\_cli-<os>-<arch>.zip (reproducible zip/tar)

Test logs: artifacts/test/…

Result files & canonical hashes for fixture runs (optional artifacts/results/\*.json + .sha256)

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

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

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

(Make targets; no Rust functions.)

## **7) Algorithm Outline (targets & order)**

**fmt** → cargo fmt --all -- --check

**lint** → cargo clippy --all-targets -- -D warnings

**build** → cargo build --locked --profile $(MK.PROFILE)

**test** → cargo test --locked --profile $(MK.PROFILE)

**fixtures** (small Annex B set) → run vm\_cli run --manifest … per fixture; save result.json & run\_record.json; compare winners/labels to expected.

**verify** (determinism smoke test) → run same manifest twice with --rng-seed $(MK.SEED); compare RES: and RUN: IDs byte-for-byte.

**hash** → compute SHA-256 over canonical bytes of outputs; write \*.sha256.

**dist** → strip (if supported), bundle vm\_cli + LICENSE + README into reproducible archive (sorted entries, fixed mtime/uid/gid).

**clean** → remove target/ and artifacts/ (keep Cargo.lock).

*All build-like targets export*: CARGO\_NET\_OFFLINE=$(MK.OFFLINE); optional RUSTFLAGS for reproducibility only if needed.

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

ci meta-target := fmt → lint → build → test → fixtures → verify → hash.

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

Determinism anchored by: --locked, pinned toolchain, offline mode, stable sort in packaging (e.g., tar/zip with fixed mtime/owner, sorted file order).

No numeric policy here; core rules enforced in engine.

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

If vendor/ missing for first build, CARGO\_NET\_OFFLINE=1 will fail; document cargo fetch && cargo vendor step outside make.

Windows shells: prefer sh (Git Bash) for Make; avoid PowerShell-only syntax in recipes.

Cross-compile bundles only when required toolchains/targets installed; otherwise skip gracefully.

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

make ci succeeds on Linux/macOS/Windows (with sh).

make fixtures validates at least VM-TST-001/002/003 outcomes and labels.

make verify shows identical RES:/RUN: IDs on two runs with same seed & inputs.

make dist archives are byte-identical across hosts given same toolchain and inputs.