Froth Implementation Roadmap

Feb 25 → End of Spring Break (initial plan)

Froth Project

2026-02-26

Contents

1 Overview

This document is **non-normative**. It is an implementation plan intended to guide development of a first reference system and thesis demos.

It is intentionally maintained **separately** from the language and platform specifications to keep the spec stable while the schedule evolves.

2 Roadmap

This roadmap is an **implementation-oriented addendum** to the spec. It is intentionally practical and time-boxed. It assumes you want a credible, demo-ready Froth by the end of Spring Break, without overbuilding the compiler/tooling.

Calendar assumption: Feb 25, 2026 → Sun, Mar 15, 2026.

2.1 Guiding principles for the first build

1) **Build the smallest thing that can blink.** A REPL that can bind a GPIO primitive and run a loop is already a proof.
2) **Keep the kernel tiny; move features into libraries.** If a feature can be expressed in Froth, do it there.
3) **Make errors recoverable early.** `catch/throw` and "prompt stays alive" are non-negotiable for rapid iteration.
4) **Defer performance tiers.** Start with QuoteRef interpretation; add DTC/native once correctness is proven.

2.2 High-level milestones (what you will show)

By the end of Spring Break, aim to demonstrate:

- **REPL-first on-device development** (bare serial, no host tool required).
- **Coherent redefinition** (hot-patch behavior live).
- **FFI for hardware** (GPIO, time, UART minimal).
- **Ctrl-C style interrupt** (CAN sets interrupt flag, throws cleanly).
- **Snapshot overlay** (`save`, power cycle, `autorun`, safe boot escape).
- Optional: **Link Mode** for file pushes.

2.3 Day-by-day plan

2.3.1 Feb 25 (Wed) — Repo + VM skeleton

Implement - Data stack (DS) + minimal call stack for quotations (CS). - Slot table with stable identity (intern by name). - Linear heap allocator for QuoteRef/PatternRef. - Minimal console I/O (UART read/write).

Proof - Hardcode a tiny quote `[ 1 2 ]` and execute it.

2.3.2 Feb 26 (Thu) — Reader + tokenization

Implement - Tokenization: numbers, identifiers, `'name`, `[ ... ]`, `p[ ... ]`. - A simple interpreter loop for token streams.

Proof - REPL reads `1 2` and prints stack `[1 2]`.

2.3.3 Feb 27 (Fri) — Core call/def + basic execution

Implement - `call`, `def`, `get`. - Execute QuoteRefs. - Basic error codes for underflow/type/undef.

Proof - `' inc [ 1 + ] def  41 inc` works (once `+` exists).

2.3.4 Feb 28–Mar 1 (Sat–Sun) — FROTH-Base arithmetic + I/O

Implement - `+ - * /mod` - comparisons and bitwise ops - `emit key key?`

Proof - You can write simple interactive math and a tiny echo loop.

2.3.5 Mar 2 (Mon) — `perm` + `pat` + canonical shuffles

Implement - PatternRef encoding from `p[ ... ]` and/or `pat`. - `perm` correctly rewires the top `n` DS items. - Define `dup swap drop over` in Froth as library words.

Proof - A short `perm` test suite passes (swap, over, nip, rot).

2.3.6 Mar 3 (Tue) — `choose` + `while`

Implement - `choose` primitive - `while` primitive with stack discipline rules - Define `if := choose call` in Froth stdlib

Proof - Non-recursive loops run indefinitely and can be interrupted later.

2.3.7 Mar 4 (Wed) — `catch/throw` + "prompt never dies"

Implement - `catch` installs handler frame (DS depth snapshot at minimum). - `throw` unwinds to nearest catch and restores DS depth. - REPL wraps each top-level evaluation in implicit catch.

Proof - Deliberate errors return to prompt with stack restored.

2.3.8 Mar 5–Mar 6 (Thu–Fri) — FFI Stage 1 + LED blink demo

Implement - `seed_pop_cell/seed_push_cell` equivalent (`froth_pop_cell`, etc.). - `FROTH_FN`, `FROTH_FN_TRY`, `FROTH_PRIM`. - Bind: `gpio.mode`, `gpio.write`, `ms` (or vendor equivalents).

Proof - `: blink ( pin -- ) ... ;` runs from REPL and blinks LED.

2.3.9 Mar 7 (Sat) — Ctrl-C / interrupt flag

Implement - CAN (0x18) sets VM interrupt flag. - VM checks flag at safe points; throws `ERR.INTERRUPT`.

Proof - Infinite loops can be stopped without reset.

2.3.10 Mar 8–Mar 9 (Sun–Mon) — Introspection essentials

Implement - `.s`, `words`, `see` (token dump is fine) - `info` banner: version, heap free, snapshot status

Proof - You can inspect definitions and recover from mistakes quickly.

2.3.11 Mar 10–Mar 12 (Tue–Thu) — Snapshot overlay persistence

Implement - A/B snapshot region, header + CRC, generation selection. - Serialize/restore overlay (QUOTE-only) using name table and object IDs. - `save`, `restore`, `wipe`. - Boot restores snapshot and runs `autorun` under catch.

Timebox note (informative): flash persistence is the highest-risk milestone. If platform edge cases consume too much time, prioritize:

1) correctness of the *format* and restore logic (round-trip serialize/restore in RAM), then
2) a minimal flash implementation for the demo (e.g., single-image + backup), and only then
3) hardening into full A/B atomicity and wear considerations.

Keep the spec's A/B model as the end goal; treat any simplified demo persistence as an interim implementation.

Proof - Define `autorun`, `save`, power cycle, it runs. - Safe boot escape exists (pin or CAN window).

2.3.12 Mar 13–Mar 15 (Fri–Sun) — Link Mode + host tool (optional)

Implement - `FROTH-LINK` handshake, STX/ETX framing. - `#ACK/#NAK` textual replies. - Minimal `froth-link` tool: send file line-by-line or frame-by-frame.

Proof - Edit a `.froth` file, push to device, it updates; Direct Mode still works.

2.4 Deliverable checklists

2.4.1 Kernel “definition of done”

- No GC.
- No implicit allocation during primitive execution.
- Coherent redefinition works.
- Errors recover to prompt.
- `perm` passes tests.

2.4.2 Demo “definition of done”

- LED blink from bare terminal.
- Interrupt stops runaway loop.
- `save` survives power loss.
- `wipe` returns to base-only state.

2.5 Deferred (post–spring break)

- DTC/native promotion (FROTH-Perf)
- Named frames compiler pass (FROTH-Named)
- Checked kinds/contracts as a selectable build profile (FROTH-Checked)
- Richer `see` (pretty printing, source retention policies)