# V2 Flash Toolkit — Product Specification & UI/UX Gui

Version: 1.0 · Date: 2025-09-19
Author: Project HANDHELD R35 · Owner: Pawel (Team Lead)

This document specifies the V2 Flash Toolkit — a Bash-based, TUI-driven utility for embedded/retro handheld workflows. It consolidates four wizards:
[1] Kernel Build, [2] Flash Image, [3] Boot.img / RootFS Adjuster, [4] Backup / Resto

Purpose
- Provide a safe, repeatable path to build, flash, adjust, and recover device images.
- Eliminate common failure modes (boot loops, panel/DTB mismatches, missing modules).
- Offer clear progress, logs, and guardrails for destructive actions.

Scope
- UX, flows, guardrails, architecture, file/folder structure, step-ID versioning, and strategy.
- Device family: RG35XX-H (Allwinner H700) — but design is extensible via device prof

# Table of Contents

# 1. Executive Summary

V2 Flash Toolkit is a portable, dependency-light TUI that codifies "Build → Flash → A
Restore".
It abstracts away low-level pitfalls (dd, mkbootimg, DTB alignment) into guided flows
guardrails,
so developers can iterate on custom kernels without bricking or wasting time.

Key Innovations
- Boot.img / RootFS Adjuster inserted between Flash and Restore to prevent boot loops
- Step-ID versioning per screen (e.g., 1-0, 1-1-1) for insert-anywhere evolution.
- Uniform logging, dry-run mode, and progressive disclosure of advanced options.

Outcomes
- Faster kernel iteration with visible progress and consistent logs.
- Safer flashing via mandatory backups and double-confirmation.
- Extensible architecture via profiles and YAML config.

## 2. Objectives & Success Criteria

Objectives
- Provide crystal-clear, low-friction flows for kernel build, flashing, boot image ac
and recovery.
- Reduce boot loop incidence to near-zero by enforcing stock header reuse, DTB mode c
and module sync.
- Maintain a predictable, auditable system: logs, reproducible steps, scripted tests.

Success Criteria
- 100% destructive actions preceded by double-confirm + backup recommendation.
- < 5 min to go from build artifacts to adjusted boot.img ready-to-flash (on a typica
dev box).
- Zero "unknown" mid-flow failures: every error printed with exact command context ar
pointer.

Non-Goals
- Not a GUI app; TUI only.
- Not a distro package — shipped as a self-contained repo with install script for dep
fzf, gum, etc.).

# 3. System Overview & Architecture

Architecture Stack
- Entry: flash-toolkit.sh
- Core libs: lib/*.sh (base, ui, disk, git, build, bootimg, modules, qemu, config)
- Wizards: wizards/*.sh orchestrators + per-step files in wizards/<wizard>/*
- Data/config: config/*.yml (repos, devices, safety, ui copy)
- Outputs: builds/, backups/, logs/, work/

Runtime Model
- Strict Bash mode + traps; UI wrappers (whiptail/gum/fzf) + progress sinks (pv/dd/ma
- State (CTX) per wizard persisted to .state.json for resume and crash safety.
- Router discovers step files by ID, sorts numerically by segments, and executes with

Data Flow (high level)
build → artifacts → (optional flash system image) → adjust stock boot → repack new-bo
sync modules → flash boot partition → (optional QEMU) → reboot → fallback via backups
needed.

## 4. Folder Structure & Naming Conventions

Top Level

flash-toolkit.sh, README.md, .env.example, .gitignore, .state.json

config/

- repos.yml, devices.yml, ui-copy.yml, safety.yml

lib/

- base.sh, ui.sh, disk.sh, git.sh, build.sh, bootimg.sh, modules.sh, qemu.sh, config.

wizards/

- kernel_build.sh, flash.sh, bootimg_adjust.sh, backup_restore.sh
- kernel-build/* per-step files, etc.

templates/ (config_patch.example, cmdline.example, ui-theme.example)

builds/, backups/, logs/, work/, test/, tools/

Naming

- Per-step files: <ID>__<slug>.sh, e.g., 1-0__choose_kernel_from_gh.sh
- Wizard screen IDs: numeric segments (1-0, 1-1-1). Sort numerically per segment.
- Logs: logs/<YYYYmmdd-HHMMss>-<wizard>.log
- Backups: disk-<model>-<size>-<ts>.img.zst; boot-<ts>.img; boot-<ts>-preflash.img

## 5. UI/UX Principles & Global Rules

Global UX
- Footer hint: ↑/↓ select · Enter confirm · ESC cancel · F1 Help · F4 View log
- Progress always visible; spinner for indeterminate; gauge for known %.
- Double confirmation with device model/size for destructive ops.
- Dry-run mode labels every CTA as [DRY RUN].

Microcopy & Theming
- Strings in config/ui-copy.yml, themes in templates/ui-theme.example.
- Consistent titles, danger colours, and safe defaults.

Accessibility & Clarity
- One decision per screen; advanced toggles grouped under "Advanced".
- Logs always one keystroke away (F4).

# 6. Main Menu & Cross-Wizard Flow

Menu Order (fixed)
[1] Kernel Build → [2] Flash Image → [3] Boot.img / RootFS Adjuster → [4] Backup / Re
[5] Exit

Cross-Wizard CTAs
- Build completion offers "Go to Flash" or "Open Adjuster".
- After Flash, nudge to Adjuster for boot.img correctness and modules sync.
- Adjuster offers QEMU smoke test and returns to menu; Backup/Restore always availabl

State Passing
- CTX persists repo, ref, artifact paths, device selection, kernel version, etc.
- Resume from last step via .state.json after unexpected exit.

# 7. Wizard 1 — Kernel Build (Steps & Specs)

Steps (per-step file pattern: wizards/kernel-build/<ID>__<slug>.sh)

- 1-0__choose_kernel_from_gh: curated list + custom URL input; set CTX[repo_url].
- 1-1__pick_branch_tag: fetch refs via ls-remote; set CTX[ref].
- 1-1-1__repo_health_check (optional): shallow clone metrics, last commit age; warn i
- 1-2__apply_config_patch: toggle; choose patch; run merge_config.sh + olddefconfig.
- 2-0__build_settings: jobs, arch, outdir; defaults from env.
- 2-1__build_progress: make -jN; feed to gauge via pv -l; tee log.
- 3-0__artifacts_summary: list Image/DTB/modules; CTA to Flash/Adjuster.

Validation & Errors

- Missing patch file when enabled → inline error.
- Make failure → show last lines of log + "Open full log".

Outputs

- builds/<repo>/<ts> with artifacts; logs/<ts>-build.log

## 8. Wizard 2 — Flash Image (Steps & Specs)

Steps

- 1-0__pick_artifact: select system.img / new-boot.img / Image*.
- 2-0__select_sd_card: lsblk JSON; highlight boot partition ~64 MB; hard-block system
- 3-0__backup_first: scope (disk vs partition), compression (zstd/gzip/none).
- 4-0__flash_progress: pv | dd of=/dev/sdX bs=4M conv=fsync; gauge + log.
- 5-0__verify_done: optional hash/cmp; CTA to Adjuster.

Safety

- Double confirmation shows device path, model, size.
- Default path proposes backups/<label>-<ts>.img.zst

## 9. Wizard 3 — Adjuster (Purpose & Value)

Why this exists

- Stock boot.img encodes page size, base, and cmdline; reusing it prevents boot loops
- Different devices/bootloaders require either --dt dtb.img or kernel+dtb concatenati
- Kernel upgrades need matching /lib/modules/<ver> on rootfs to avoid missing drivers

Design

- Compare stock vs new kernel; allow DTB mode selection; editable cmdline; enforce si checks.
- Auto-backup boot partition; optional QEMU smoke test. Logs everything for auditabil

## 10. Adjuster Steps & Anti–Boot Loop Playbook

Steps
- 1-0__choose_stock_bootimg: extract from /dev/sdX4 or browse; backup stock.
- 1-1__extract_summary: page_size, base, cmdline; files present.
- 2-0__select_new_kernel: Image/zImage; detect version.
- 3-0__dtb_mode: with-dt (mkbootimg --dt) vs catdt (cat Image+dtb).
- 3-1__select_dtb_variants: choose DTBs; build dtb.img.
- 4-0__cmdline_editor: quick toggles + freeform edit.
- 5-0__repack_bootimg: mkbootimg using stock header values; size ≤ boot partition.
- 6-0__modules_sync: copy modules to rootfs; depmod -a -b <mnt> <ver>.
- 7-0__flash_boot_partition: backup small partition; dd new-boot.img → /dev/sdX4.
- 8-0__qemu_smoke_test (optional).
- 9-0__done.

Anti–Boot Loop Checklist
- Always reuse stock page_size/base/cmdline.
- Pick the right DTB mode & variant (panel mapping).
- Ensure /lib/modules/<ver> exists on rootfs.
- Validate new-boot.img size ≤ boot partition; never truncate.
- Always backup before flashing the boot partition.

## 11. Device Profiles, DTB Modes & Heuristics

File: config/devices.yml (example excerpt)

```yaml
rg35xxh:
  boot_partition_hint: /dev/*4
  boot_partition_size_mb: 64
  page_size: 2048
  base: 0x40000000
  dtb_mode_default: with-dt
  dtb_variants:
    - sun50i-h700-anbernic-rg35xx-h.dtb
    - sun50i-h700-anbernic-rg35xx-h-rev6-panel.dtb
    - rg40xx-h.dtb
```

Heuristics
- Suggest boot partition by size (~64 MB) and label signature.
- Prefer "with-dt" first; fall back to "catdt" if black screen persists.
- Surface a "Try alternate DTB" CTA if no framebuffer after boot.

## 12. Wizard 4 — Backup / Restore (Steps & Specs)

Steps
- 1-0__choose_action: backup/restore; disk/partition.
- 2-0__pick_source_target: device/file + compression.
- 3-0__run_progress: pv/dd pipeline; gauge + logs.
- 4-0__done: outputs folder link; optional verify.

Pipelines
- Backup disk: dd if=/dev/sdX bs=4M status=progress | pv | zstd -T0 > backups/disk-<model>-<ts>.img.zst
- Restore disk: pv backups/disk-<ts>.img.zst | zstd -d | dd of=/dev/sdX bs=4M conv=fs

# 13. Step-ID Versioning Pattern & Router

Pattern
- Files: wizards/<wizard>/<ID>__<slug>.sh where ID is numeric segments: 1-0, 1-1-1, e
- Router scans, parses, and sorts by numeric segments (lexical with zero-pad per segm

Anchors & Safety
- config/safety.yml lists required anchors (e.g., 1-0). Router ensures presence or in
placeholder.

Context
- Global associative array CTX (Bash) shared across steps; persisted to .state.json.
- Navigation contract: steps set NEXT_ID/GOTO_ID/BACK/SKIP; router enforces one actio

Dev Tools
- --list (print ordered steps), --goto <ID>, scaffolder (new step), linter (IDs/ancho

## 14. UI Toolkit & Disk Safety

UI Toolkit (lib/ui.sh)
- ui_menu(title, var, items...), ui_input(title, placeholder, default), ui_confirm(ti
danger)
- ui_gauge(title, feed_fn), ui_spinner(title, cmd)
- Use whiptail/dialog for menus/forms/gauges; gum for prompts/spinners; fzf for long

Disk Safety (lib/disk.sh)
- Parse lsblk -J -o NAME,SIZE,TYPE,FSTYPE,LABEL,MODEL,PATH; render model/size groups.
- Hard block if selected device is system root; require 'YES' typed confirm for disk
- Mark Suggested boot partition (~64 MB), show partition map and labels.

## 15. Build, Boot Image & Modules Pipelines

Build (lib/build.sh)

- defconfig → (merge_config.sh -m) → olddefconfig → make -jN with pv -l estimate.

Boot image (lib/bootimg.sh)

- abootimg -x stock.img → extract kernel/ramdisk/dtb/header.
- Repack with mkbootimg using stock page_size/base/cmdline.
- Modes: with-dt (mkbootimg --dt dtb.img) or catdt (cat Image+dtb → Image_dtb).

Modules (lib/modules.sh)

- Detect kernel <ver>; mount rootfs; copy /lib/modules/<ver>; depmod -a -b <mnt> <ver
- Validate critical drivers presence as per devices.yml (optional).

## 16. Quality Gates: Lint, Tests, Preflight, CI

Preflight (on startup)

- require_cmd git make pv dd lsblk blkid fzf (whiptail|dialog) gum abootimg mkbootimg

Lint/Format

- shellcheck (no warnings), shfmt (consistent style)

Tests (bats-core)

- disk_parsing.bats, bootimg_header.bats, mkbootimg_args.bats
- fixtures/lsblk-sample.json used for predictable results

CI

- Run preflight in container, execute unit tests, produce artifacts with dry-run samp

# 17. Configuration Schemas (YAML Examples)

config/repos.yml

```yaml
repos:
  - name: anbernic/h700-linux
    url: https://github.com/anbernic/h700-linux
  - name: community/h700-kernel-tuned
    url: https://github.com/community/h700-kernel-tuned
```

config/ui-copy.yml

```yaml
titles:
  main: "V2 Flash Toolkit"
warnings:
  destructive: "This action writes to a block device. Proceed only if you have a back
```

config/safety.yml

```yaml
anchors:
  kernel-build: ["1-0"]
  flash: ["1-0"]
  bootimg_adjust: ["1-0"]
```

## 18. Roadmap, Risks, and Glossary

Roadmap
- v1.1: per-step linter, QEMU smoke test automation, and i18n polish.
- v1.2: device profile wizard + telemetry-free crash report dumper.
- v1.3: modular plugins for other SoCs (Rockchip, Qualcomm).

Risks & Mitigations
- Tool availability across distros → tools/install-deps.sh + preflight.
- Wrong target device chosen → double confirm + system root hard block.
- Boot loops due to DTB/cmdline → Adjuster guidance + decision tree + QEMU.

Glossary
- DTB/DTBO: Device Tree Blob/Overlay.
- Ramdisk/initrd: early userspace image in boot flow.
- mkbootimg/abootimg: tools to pack/unpack Android-style boot images.
- pv/dd: pipe viewer and disk writer utilities for progress & reliability.