343B

User Manual Tahera Guide

ULTIMATE OPERATIONS MANUAL

Includes Full Coverage:

- Driver instructions and control mappings
- The Tahera Sequence, Auton Planner, Image Selector, Basic Bonkers
- Repository navigation, build/upload, release, and troubleshooting
- Mac and Windows Tahera app workflows

Credits

Ethan Baur, Dixon Dean, Lorenzo DiIorio, William Nikodem, George Reed

St. Albans 343B Robotics

Generated February 14, 2026

Manual Use

How to Use This Manual

This guide is written as the full operating reference for the 343B stack. Follow sections in order for first-time setup, then use individual sections during build days, testing, and competition.

- If you are new: read Setup, Slot Map, Driver Controls, and Build/Upload first.
- If you are tuning autos: focus on Auton Planner, slot files, and Field Replay sections.
- If you are preparing a release: use Git flow, authentication, and release checks sections.
- If something fails: jump to Troubleshooting and the command checklist appendix.

> This manual describes current repository behavior based on source code in the PROS projects and Tahera apps. When code changes, regenerate this manual so procedures stay accurate.

System Overview

Primary architecture

- V5 Brain programs are built in PROS and uploaded into fixed slots.
- Tahera desktop apps (macOS and Windows) streamline build, upload, replay, SD, and release tasks.
- Basic Bonkers records controller activity to microSD logs.
- Field Replay visualizes those logs as path overlays on the digital field image.
- Auton Planner captures and edits movement steps, then saves to three slot files on microSD.

Current slot assignment

```
Slot 1: The Tahera Sequence (Pros projects/Tahera_Project)
Slot 2: Auton Planner (Pros projects/Auton_Planner_PROS)
Slot 3: Image Selector (Pros projects/Jerkbot_Image_Test)
Slot 4: Basic Bonkers (Pros projects/Basic_Bonkers_PROS)
```

Hardware Checklist

Hardware and Field Checklist

- V5 Brain with battery, radio, and USB cable.
- V5 controller paired to brain (wireless or tethered).
- microSD card inserted in brain before logging, planning, or image selection.
- Drive motors wired exactly to active port map.
- IMU on port 11 and GPS on port 10 for Tahera and Auton Planner features.
- Mac or Windows machine with repository cloned and build tools installed.

Recommended pre-drive checks

- Calibrate IMU at startup and wait for calibration to finish.
- Verify SD is readable by checking logs or save operations on-brain.
- Confirm correct slot is selected before each run.
- Clear field around robot before any autonomous run.

Software Setup

- Install PROS CLI and ensure the ‘pros‘ command is available in terminal.
- Install Swift toolchain/CommandLineTools for macOS Tahera builds.
- Install .NET SDK for Windows Tahera builds.
- Install Git and GitHub CLI (‘gh‘) for tag/release actions.
- Authenticate GitHub CLI with ‘gh auth login‘ before release operations.

Build commands

```
# macOS Tahera app
cd 'Mac Aplications/Tahera'
./build_app.sh

# Windows Tahera app
cd 'Windows Port/TaheraWin'
powershell -ExecutionPolicy Bypass -File build_windows.ps1
```

Repository Navigation

Repository Navigation

Use this map to find the right location quickly during coding, uploading, and troubleshooting.

```
2026-Vex-V5-Pushback-Code-and-Desighn-/
  README.md
  Pros projects/
    Tahera_Project/
    Auton_Planner_PROS/
    Jerkbot_Image_Test/
    Basic_Bonkers_PROS/
  Mac Aplications/Tahera/
  Windows Port/TaheraWin/
  Windows Port/BonkersFieldReplayWin/
  Developer Extras/Designs/
  tools/bonkers_log_to_field.py
  tools/convert_images_to_bmp.sh
```

Key files by purpose

- Robot behavior: each project ‘src/main.cpp‘ under ‘Pros projects/...‘.
- Tahera mac app logic: ‘Mac Aplications/Tahera/Sources/Tahera/Models/TaheraModel.swift‘.
- Tahera mac sections/UI: ‘Mac Aplications/Tahera/Sources/Tahera/Views/‘.
- Tahera windows UI: ‘Windows Port/TaheraWin/MainWindow.xaml‘ and ‘.cs‘ code-behind.
- Image assets: ‘Developer Extras/Designs/‘ and app resource folders.

Program Matrix and Purpose

Slot 1 - The Tahera Sequence

- Primary driver code with D-pad override, optional GPS heading assist, and 6WD toggle.
- Reads active auton slot file and runs GPS or BASIC plan from SD during autonomous.
- Reads optional ‘controller_mapping.txt‘ to remap actions to controller buttons.

Slot 2 - Auton Planner

Program Matrix

- On-brain touch UI to create/edit steps and save to three selectable auton slots.
- Can record tank drive motions while driving and convert them into TANK_MS steps.

Slot 3 - Image Selector

- Lets users browse BMP images on microSD and assign SPLASH/AUTON/DRIVER images.
- Writes ‘ui_images.txt‘ used by The Tahera Sequence for runtime visuals.

Slot 4 - Basic Bonkers

- Records every controller axis sample and button press events to SD log files.
- Shows recent button entries on-screen and stops when touch SAVE is pressed.

Driver Controls Quick Reference

The Tahera Sequence default mapping

```
Drive (tank): ANALOG_LEFT_Y -> left side, ANALOG_RIGHT_Y -> right side
D-pad override: UP/DOWN/LEFT/RIGHT
GPS drive toggle: A = enable, B = disable
6-wheel toggle: Y = on, X = off
Intake control: L1 = intake in, L2 = intake out
Outake control: R1 = outake out, R2 = outake in
```

- D-pad commands override sticks when any D-pad direction is pressed.
- With GPS drive enabled, D-pad headings hold cardinal targets (0/90/180/270).
- With GPS drive disabled, D-pad behaves as direct fixed-speed drive/turn commands.

> Controller mapping can be customized from the Tahera desktop app and saved to repo or SD. The brain loads that mapping file at initialize.

The Tahera Sequence

The Tahera Sequence - Full Guide

Drive behavior

- Outer motors are grouped and middle motors are controlled separately for 6WD toggling.
- If 6WD is disabled, middle motors brake while outer motors continue running.
- Intake and outake are independent motors on ports 7 and 8.

On-brain touch UI

- Buttons on screen: GPS, BASIC, and RUN.
- SLOT indicator reflects active slot from ‘auton_slot.txt‘ (1 to 3).
- Tap RUN to execute selected autonomous path when in operator control.

Files read by The Tahera Sequence

```
auton_slot.txt                # active slot number
auton_plans_slot1.txt         # slot 1 plan
auton_plans_slot2.txt         # slot 2 plan
auton_plans_slot3.txt         # slot 3 plan
controller_mapping.txt        # optional custom mapping
ui_images.txt                # splash/auton/driver image selection
```

Auton Planner - Full Guide

What it does

- Edits GPS and BASIC autonomous plans from touch UI.
- Saves and loads three slot files on SD card.
- Records live tank inputs into TANK_MS steps at fixed sample intervals.

Touch controls

```
Top row: GPS | BASIC | SAVE | S1 | S2 | S3
Step nav: PREV | NEXT
Step edit: TYPE, V1-, V1+, V2-, V2+, V3-, V3+
Record controls: REC/STOP and CLEAR
```

- SAVE writes both [GPS] and [BASIC] sections to the currently selected slot file.
- S1/S2/S3 selects active storage slot and updates ‘auton_slot.txt‘.

Auton Planner

- REC starts recording current drive mode (GPS or BASIC) into plan steps.
- CLEAR resets the active plan list to EMPTY steps.

Driver controls in Auton Planner opcontrol

- Tank drive uses left and right Y axes.
- L1/L2 control left intake motor; R1/R2 control right intake motor.
- A forces GPS mode; B forces BASIC mode.

Auton Step Types and File Format

- Supported step types: EMPTY, DRIVE_MS, TANK_MS, TURN_HEADING, WAIT_MS, INTAKE_ON, INTAKE_OFF, OUTTAKE_ON, OUTTAKE_OFF.
- Each row stores type and up to three numeric values.
- Plans are grouped into [GPS] and [BASIC] sections.

```
[GPS]
DRIVE_MS,60,1200,0
TURN_HEADING,90,0,0
WAIT_MS,250,0,0

[BASIC]
TANK_MS,50,50,1000
OUTTAKE_ON,0,0,0
OUTTAKE_OFF,0,0,0
```

Tahera Sequence and Auton Planner must agree on this format. If custom scripts modify files, keep section names and CSV field counts intact.

Image Selector - Full Guide

- Scans ‘/usd/Images‘ for BMP files and presents a touch UI browser.
- PREV/NEXT cycles through available BMP assets.
- SPLASH, AUTON, and DRIVER assign current image to each role.
- SAVE writes selections to ‘ui_images.txt‘.
- REFRESH rescans SD card in case files were added/removed.

Config file written by Image Selector

Image Selector

```
SPLASH=/usd/Images/loading_icon.bmp
AUTON=/usd/Images/jerkbot.bmp
DRIVER=/usd/Images/driver.bmp
RUN=/usd/Images/jerkbot.bmp
```

- Use 24-bit or 32-bit BMPs; avoid PNG/JPEG on brain runtime screens.

Basic Bonkers - Logger Guide

What is recorded

- Every loop sample logs AXIS1, AXIS2, AXIS3, AXIS4 values.
- Button press transitions log entries in format ‘TYPE : ACTION‘.
- File path format: ‘/usd/bonkers_log_<millis>.txt‘.
- A touch press logs ‘SCREEN_TAP : SAVE‘ and ends recording.

On-brain display behavior

- Shows logger status and output path while recording.
- Displays recent button events and stops adding once history area is full.

```
AXIS1 : 0
AXIS2 : -54
AXIS3 : 62
AXIS4 : 0
BTN_L1 : INTAKE_IN
BTN_R1 : OUTTAKE_OUT
SCREEN_TAP : SAVE
```

Field Replay on Mac and Windows

- Purpose: replay Bonkers logs as movement lines over the digital field image.
- Open a ‘bonkers_log_*.txt‘ file from SD or copied local folder.
- Use play controls to inspect path and directional behavior across the run.
- Replay uses joystick axis interpretation for approximate path visualization.

Workflow

Field Replay

```
1) Record a run in Basic Bonkers and tap SAVE.
2) Move log file from microSD to computer if needed.
3) Open Tahera -> Field Replay -> Open File.
4) Select log and run replay.
5) Compare path against expected route and adjust code/plans.
```

SD Card Workflow

- Mount path used by Tahera on macOS: ‘/Volumes/MICROBONK‘.
- Store runtime images in ‘/usd/Images‘ on the card.
- Keep plan files and mapping files at SD root for automatic detection.
- Use BMP format for brain-rendered images; remove duplicate non-BMP versions.

Core SD files

```
auton_slot.txt
auton_plans_slot1.txt
auton_plans_slot2.txt
auton_plans_slot3.txt
controller_mapping.txt
ui_images.txt
bonkers_log_*.txt
Images/*.bmp
```

Port Map and Wiring

The Tahera Sequence and Auton Planner

```
Left outer drive: 1(reversed), 3(reversed)
Left middle drive: 2
Right outer drive: 4, 6
Right middle drive: 5(reversed)
Intake motor: 7
Outake motor: 8
GPS: 10
IMU: 11
```

- If wiring changes, update both PROS code and Tahera Port Map section together.
- Reversed signs in code (‘-port‘) matter for correct drive direction.

Port Map

- After rewiring, run a low-speed direction test before full drive.

Controller Mapping Customization

- Tahera app section allows action-to-button remapping.
- Save mapping to repository file and/or SD card file.
- Brain loads mapping on initialize from ‘controller_mapping.txt‘.

Supported actions

```
INTAKE_IN
INTAKE_OUT
OUTAKE_OUT
OUTAKE_IN
GPS_ENABLE
GPS_DISABLE
SIX_WHEEL_ON
SIX_WHEEL_OFF
```

Example mapping file

```
# Tahera controller mapping
# Format: ACTION=BUTTON
INTAKE_IN=L1
INTAKE_OUT=L2
OUTAKE_OUT=R1
OUTAKE_IN=R2
GPS_ENABLE=A
GPS_DISABLE=B
SIX_WHEEL_ON=Y
SIX_WHEEL_OFF=X
```

Build and Upload from Tahera App

- Open Tahera and go to Build & Upload section.
- Choose target project and verify slot number.
- Use auto brain detection and verify connected port.
- Run build/upload action and monitor output log for completion.
- Repeat for each project when doing full brain refresh.

Recommended full upload order

Build and Upload

```
1) The Tahera Sequence -> slot 1
2) Auton Planner       -> slot 2
3) Image Selector      -> slot 3
4) Basic Bonkers       -> slot 4
```

Build and Upload from PROS CLI

Use CLI when GUI is unavailable or when scripting uploads in a terminal workflow.

```
cd 'Pros projects/Tahera_Project' && pros make && pros upload --slot 1
cd 'Pros projects/Auton_Planner_PROS' && pros make && pros upload --slot 2
cd 'Pros projects/Jerkbot_Image_Test' && pros make && pros upload --slot 3
cd 'Pros projects/Basic_Bonkers_PROS' && pros make && pros upload --slot 4
```

- If multiple brains are attached, choose the intended device at upload prompt.
- Keep microSD inserted for programs that read/write files.

Tahera Desktop Apps

macOS

```
cd 'Mac Aplications/Tahera'
./build_app.sh
# output: Mac Aplications/Tahera/build/Tahera.app
```

Windows

```
cd 'Windows Port/TaheraWin'
powershell -ExecutionPolicy Bypass -File build_windows.ps1
# output: Windows Port/TaheraWin/build/Tahera.exe
```

- Both apps include sections for build/upload, port map, controller mapping, replay, SD, README, and repository settings.
- Repository Settings section is protected by password in-app: 56Wrenches.782

Git and Release

Git Commit, Tag, and Release

- Use Tahera Repository Settings section to run commit/push/tag/release operations.
- Release log should show each phase and final success/failure message.
- If release fails, address authentication or CLI path issues before retry.

Authentication check

```
gh auth status
gh auth login
gh release list
```

Command-line fallback release

```
git add -A
git commit -m 'release prep'
git push origin main
git tag -a 2.0 -m 'Release 2.0'
git push origin 2.0
gh release create 2.0 --title '2.0' --notes 'Release notes'
```

Troubleshooting Guide

Brain and upload issues

- Brain not detected: reconnect USB, power cycle brain, re-run detection.
- Upload fails: verify project compiles with ‘pros make‘ first.
- Wrong slot behavior: check upload command used correct ‘--slot‘ value.

SD and file issues

- No logs or plans: confirm SD inserted and writable.
- Images not drawing: ensure ‘.bmp‘ format and valid path under ‘/usd/Images‘.
- Slot not switching: verify ‘auton_slot.txt‘ contains 1, 2, or 3 only.

Release issues

- ‘gh: No such file or directory‘: install GitHub CLI and restart app.
- ‘not authenticated‘: run ‘gh auth login‘ and verify with ‘gh auth status‘.
- Infinite loading: rely on release log; enforce command timeout and retry after fix.

Troubleshooting

> When debugging anything critical, isolate one subsystem at a time: wiring -> brain upload -> SD file presence -> controller behavior -> desktop integration.

Autonomous Tuning Workflow

- Use Auton Planner to create a baseline plan in the correct slot.
- Run autonomous from The Tahera Sequence with that same selected slot.
- Record a Basic Bonkers log during driver validation runs.
- Replay the path in Tahera Field Replay and compare against expected field path.
- Adjust TURN_HEADING and DRIVE/TANK durations incrementally, then save and retest.

Tuning loop

```
1) Edit plan step values (Auton Planner)
2) SAVE to current slot
3) Run auton from Slot 1 (The Tahera Sequence)
4) Record and replay if movement is off
5) Repeat until route stability is acceptable
```

Windows Operations and Portability

- Tahera Windows app is under ‘Windows Port/TaheraWin‘.
- Build output is a runnable ‘Tahera.exe‘ in the ‘build‘ folder.
- Bonkers replay companion app is under ‘Windows Port/BonkersFieldReplayWin‘.
- Keep assets, slot files, and mapping files identical between Mac and Windows workflows.

Windows build commands

```
cd 'Windows Port/TaheraWin'
build_windows.bat
# or build_windows.ps1

cd 'Windows Port/BonkersFieldReplayWin'
powershell -ExecutionPolicy Bypass -File build_windows.ps1
```

Developer Workflow

Developer Workflow and Code Update Path

- Update PROS source first, then validate behavior on-brain.
- Reflect any port-map or control changes in both Tahera apps immediately.
- Rebuild Mac and Windows apps after UI or model changes.
- Run one upload cycle for each slot after major refactors.
- Commit with specific messages that describe the robot behavior change.

Recommended change sequence

```
1) Edit PROS project code
2) Build + upload to test slot
3) Validate controls and auton behavior
4) Update Tahera app models/views if needed
5) Rebuild macOS and Windows Tahera apps
6) Commit, push, tag, release
```

Backup and Recovery Procedures

- Store critical logs and slot files after every major test session.
- Keep a copy of working SD card files in a dated folder in the repo or external backup.
- If files are deleted locally, restore by pulling from the last known good commit.
- If slot behavior is wrong, reupload all four programs in slot order.

Fast restore checklist

```
git fetch origin
git checkout <known-good-commit>
# rebuild apps
# reupload slots 1 through 4
# restore SD files: auton_slot, slot plans, mapping, ui_images
```

FAQ

FAQ - Common Team Questions

- Q: Can I drive while Auton Planner records? A: Yes, REC captures tank samples while driving in opcontrol.
- Q: Are Auton Planner ports the same as The Tahera Sequence? A: Yes, shared drive and sensor assumptions are maintained.
- Q: Why is replay path not exact? A: Replay is axis-based approximation, useful for relative path checks.
- Q: What if release does not appear on GitHub? A: Check gh auth, tag push, and release log output.
- Q: Where do I change button mapping? A: Tahera app -> Controller Mapping section, then save to repo/SD.

Competition Day Checklist

Pre-match

- Verify slot assignments (1 Tahera Sequence, 2 Auton Planner, 3 Image Selector, 4 Basic Bonkers).
- Run a short drivetrain direction test and intake/outake test.
- Confirm active auton slot and expected path data on SD.
- Confirm controller mapping file is correct for current drivers.

During testing rounds

- Record one Bonkers log for each major tuning change.
- Replay each log in Tahera Field Replay before locking match strategy.
- If autonomous drifts, adjust plan values and resave via Auton Planner.

Post-match

- Archive new logs and plans in dated folders.
- Commit high-value changes and tag stable states.

Appendix A

Appendix A - File Examples

auton_slot.txt

```
2
```

auton_plans_slot2.txt

```
[GPS]
DRIVE_MS,70,900,0
TURN_HEADING,90,0,0
DRIVE_MS,55,600,0

[BASIC]
TANK_MS,70,70,900
TANK_MS,-40,40,350
WAIT_MS,250,0,0
```

ui_images.txt

```
SPLASH=/usd/Images/loading_icon.bmp
AUTON=/usd/Images/jerkbot.bmp
DRIVER=/usd/Images/driver_overlay.bmp
RUN=/usd/Images/jerkbot.bmp
```

Appendix B - Fast Path Reference

```
Top-level readme: README.md
Tahera PROS code: Pros projects/Tahera_Project/src/main.cpp
Auton Planner code: Pros projects/Auton_Planner_PROS/src/main.cpp
Image Selector code: Pros projects/Jerkbot_Image_Test/src/main.cpp
Basic Bonkers code: Pros projects/Basic_Bonkers_PROS/src/main.cpp
Mac app model: Mac Aplications/Tahera/Sources/Tahera/Models/TaheraModel.swift
Windows app UI: Windows Port/TaheraWin/MainWindow.xaml
Field replay helper: tools/bonkers_log_to_field.py
Image conversion script: tools/convert_images_to_bmp.sh
```

End of ultimate manual. Keep this document synchronized with code after port map changes, control remapping changes, slot behavior updates, and release workflow updates.