
Mustang 2 Firmware Changelog (Teensy 4.1 sketch)

Covers only changes to the actual Teensy 4.1 firmware source — new/changed PiLink wire messages, new virtual-encoder/button ids, and changes to what the sketch itself sends or how it computes a value. **Every entry here requires reflashing the Teensy** before it takes effect; confirm you're on the right build by checking the on-screen VERSION string (`SDT.h`), not just whatever used to be broken. Companion-display/web-remote-only changes (no reflash needed) are tracked separately in the Pi companion display changelog.

V1.0.34 (2026-10-04) — released

- **Built and posted on the website** (SerialMIDIAudio and DualSerial .hex, plus source ZIP). Includes every change from V1.0.26 through V1.0.33 below.
- Version number only: V1.0.33 with the 42-byte `CW_DECODE` message (CW settings page values), renumbered so it can be told apart from the first V1.0.33 build, which sent only the 24-byte text part (the Pi and remotes then show “-” on the CW page). The first V1.0.33 build was flashed from the `MUSTANG_2_V1.0.32` copy made at 11:11.

V1.0.33 (2026-10-04)

- **New `PILINK_MSG_CW_DECODE (0x0E, 24 bytes)`**: active (CW receive with the decoder on - same test as `MorseCharacterDisplay()`), locked, decoded WPM, a running count of decoded characters, and the T41's 15-character decode window. 10 Hz while decoding, 2 Hz otherwise.
- `DSP_CWProcessing`: `MorseCharacterAdd()` counts characters; new `PeekMorseCharacters()` copies the window and count without clearing the “updated” flag the T41's own display uses.
- VERSION “Mustang 2 V1.0.33”. Includes V1.0.29-V1.0.32.
- (V1.0.33, later the same day) `PiLinkCWDecodeMsg` extended to 42 bytes with the CW settings page values: SNR, AT Grace, Manual mode, key type, keyer WPM, Dit(ms), average dit/dah, update age, Coeff - same “-” gates as `DrawCWSettingsPane()`. WPM now uses the same gate as the T41's WPM: (CW receive + decoder on + locked).

V1.0.32 (2026-10-04)

- **New `PILINK_MSG_WATERFALL (0x0D)`**: 512 bytes, the T41's own waterfall colour index per column (`230 - pixelold[x]`, clamped 0..116 - the same “test1” rule `ShowSpectrum()` uses for `gradient[]`). Sent at the SPECTRUM rate, half an interval after it so the two 518-byte frames never share the 1 KB TX buffer; skipped during SSB transmit like the T41's own waterfall. Lets the Pi colour its waterfall exactly like the T41 (the noise blue was much darker on the Pi).
- VERSION “Mustang 2 V1.0.32”. Includes V1.0.29-V1.0.31.

V1.0.31 (2026-10-04)

- **Power up in DATA mode.** New `#define STARTUP_MODE_DATA` in `Config.h` (on by default). Right after `ModeSm_start()` in `Globals.cpp`, if the mode is SSB receive it sends `ModeSm_EventId_TO_DIGITAL_MODE` (the same event the Mode button uses) and updates the RF hardware. Only compiled into builds with `AUDIO_INTERFACE`. Comment the define out to power up in SSB as before.
- VERSION “Mustang 2 V1.0.31”. Includes everything in V1.0.29 and V1.0.30.

V1.0.30 (2026-10-04)

- **New PiLink telemetry** for the Transmit IQ, Frequency and Power calibration screens: `PILINK_MSG_TXIQ_CAL (0x0A, 168 bytes)`, `PILINK_MSG_FREQ_CAL (0x0B, 18 bytes)`, `PILINK_MSG_POWER_CAL (0x0C, 212 bytes)`. Sent at 10 Hz while that screen is showing, 2 Hz otherwise (`PI_LINK_CAL_ACTIVE/IDLE_INTERVAL_MS`). Accessors added for values that were file-local: `GetTXIQIncrement()`, `GetTXIQAttenuation()`, `GetFrequencyCorrectionIncrement()`, `GetPowerIncrement()`, `IsPowerCalTransmitting()`.
- **Exit without saving** on the Frequency, Transmit IQ and Power calibration screens: `MAIN_MENU_UP` (button 1) leaves the screen like Home but skips `SaveDataToStorage()` (Transmit IQ/Power still restore `XAttenCW`). Button 1 was unused on all three screens; DFE (14) already has a job on Transmit IQ and Power.
- **Transmit IQ Auto Stop**: `DECODER_TOGGLE` (button 13) on the Transmit IQ screen calls the new `RequestTXIQAutoStop()`, which ends a running Auto sweep through the normal last-band exit (so PTT is released). Only armed while Auto is running.
- **Receive IQ Stop fixes**: `RequestRXIQAutoStop()` is now ignored unless Auto is running (a press while idle used to abort the next Auto run straight away), and a Stop on any band before GEN no longer jumps to GEN and calibrates it.
- **Power calibration “Transmit:” readout fix**: it tested a Transmit IQ state and always read Off; it now shows On in `CALIBRATE_POWER_MARK / CALIBRATE_OFFSET_MARK`.
- VERSION “Mustang 2 V1.0.30”. (V1.0.29's Spectrum fill power-up default of Off is included.)

V1.0.29 (2026-10-03)

- **Spectrum fill off at power-up.** `uint8_t spectrumFillTraceEnabled` default changed from 1 to 0 in SDT.h, so the spectrum always starts as an outline trace. Menu > Display > Spectrum Fill > On turns the fill on until the next power cycle (this setting isn't saved to storage).
- VERSION string "Mustang 2 V1.0.29".

V1.0.28 (2026-10-01)

Added

- **New uplink `PILINK_MSG_MENU_SELECT (0x14)`, payload `[primary, secondary]`:** jumps straight to a Menu item via new `JumpToMenuItem()` (MainBoard_DisplayMenus.cpp). It dispatches the same `UISm` events the buttons do (`MENU`, `SELECT`, `SELECT`) after setting `primaryMenuIndex/ secondaryMenuIndex`, so variable items open their adjust screen and action items run exactly as if picked by hand. All three events happen before the next redraw, so no menu list is drawn on the way. It only acts from the Home screen and is refused while transmitting or for an out-of-range index.
- Fixes the Pi/web Menu dropdown visibly scrolling through every item in between (e.g. EEPROM to Display > Spectrum Floor). Same complaint the V1.0.16 Zoom fix answered.
- VERSION "Mustang 2 V1.0.28".

V1.0.27 (2026-10-01)

Changed

- **Audio Equalizer "Exit" (no save):** button 14 on the equalizer screen now restores both equalizers to the levels they had when the screen was entered and returns Home without saving (`RestoreEqualizerSnapshot()`, snapshot taken on the screen's first draw in `MainBoard_DisplayEqualizer.cpp`; new `case 14` in `Loop.cpp`'s `UISm_StateId_EQUALIZER` button switch). Home still saves and exits. The T41's own Instructions pane lists the new button.
- **"Load:" now measures DSP work only:** `ReceiveProcessing()` (DSP.cpp) adds the block time to `elapsed_micros_sum` *before* `PlayBuffer()` instead of after. `AudioPlayQueue::playBuffer()` waits while the output queue is full, so Load: was counting that wait, and the wait depends on the queue alignment set at power-up. That's the likely reason Load: read 60-79% at power-on and fell to ~20% as soon as DATA mode (176.4 kbps) restarted the audio chain. Expect a steady, lower reading in every mode. No change to the audio itself.
- VERSION "Mustang 2 V1.0.27".

V1.0.26 (2026-10-01)

Added

- **`PILINK_MSG_EQUALIZER (0x09)` downlink message** (`PiLinkEqualizerMsg`, 32 bytes: `active`, `rxTxSelection`, `cellSelection`, `increment`, `rx[14]`, `tx[14]`), sent at 10 Hz (`PI_LINK_EQUALIZER_INTERVAL_MS`) by the new `SendEqualizer()` in `PiLink.cpp`. Feeds the Audio Equalizer screen (Menu > Audio Options > "Adjust Equalizers") on the Pi and both web remotes. `active` is 1 while `uiSm.state_id == UISm_StateId_EQUALIZER`; the levels are sent continuously regardless, same convention as `RXIQ_CAL`. Each level is clamped to 0..100 (the same range `Increment/DecrementEqualizerValue()` already enforce) so it fits a `uint8_t`.
- **`GetEqualizerRxTxSelection()`, `GetEqualizerCellSelection()`, `GetEqualizerIncrement()`** read-only accessors in `MainBoard_DisplayEqualizer.cpp` for that screen's file-local state (same pattern as `GetRXIQIncrementIndex()`).
- No new uplink message: the screen's controls are the existing `VIRTUAL_BUTTON` (15 BEARING = RX/TX, 16 SPARE = increment, 17 HOME_SCREEN = save & exit) and `VIRTUAL_ENCODER` (Volume = cell, Filter = level) primitives, routed by `Loop.cpp`'s existing `UISm_StateId_EQUALIZER` switches.

Version

- `VERSION` (SDT.h) bumped to "Mustang 2 V1.0.26".

V1.0.25 (2026-10-01)

Added

- **`PILINK_MSG_BIT_RESULTS (0x08)` downlink message**, feeding a new I2C BIT (Built-In Test) results screen ported to the Pi and both web remotes, at the user's request ("the I2C BIT Test needs to be ported over the the Pi and both remotes"). New `PiLinkBITMsg` struct (`active`
 - one boolean per board - front panel, BPF, RF, RF SI5351, K9HZ LPF - plus `dualVfo` to pick which SI5351 address to show, same as `DrawBIT()`'s own `HasDualVFOs()` branch). New `SendBITResults()` in `PiLink.cpp`, called from `UpdatePiLinkTelemetry()` at a new `PI_LINK_BIT_INTERVAL_MS` (500ms/2Hz) cadence - slower than `PILINK_RXIQ_CAL`'s 10Hz since `bit_results` is only ever written once at boot (each board's own `Init*()`) and never changes while running. Like Receive IQ calibration, entering/exiting this screen needed no new UPLINK message: "I2C BIT test" was already a normal `functionOption` in the Pi's/both web remotes' menu mirror, so the existing generic menu-select sequence already reaches `GotoDisplayBIT()` -> `SetInterrupt(iBITDISPLAY)` exactly like a physical `MENU_OPTION_SELECT` press would, and the screen's only control (`HOME_SCREEN`, to exit - see `UISm_StateId_BIT`'s switch in `Loop.cpp`) is the same `PILINK_MSG_VIRTUAL_BUTTON` every other screen already uses. The AD7991 SWR-sense board's result is deliberately left off this message - this build doesn't define `V12_LPF_SWR_AD7991`, so the Teensy's own screen never

shows that row either; the Pi/web screens stay a faithful mirror of what the Teensy actually displays.

V1.0.24 (2026-09-30)

Added

- **Receive IQ calibration screen: “Stop” and “Exit” buttons**, at the user’s follow-up request after trying V1.0.23’s screen (“It appears that the GEN calibration never stops, ‘Val’ keeps rolling data. Need to add a ‘Stop’ button to stop the calibration when in Auto Mode if possible. And an ‘Exit’ button, keep the ‘Save & Exit’ button, it can mimic the Home - Home Screen dropdown”). Both are repurposed buttons scoped ONLY to `UISm_StateId_CALIBRATE_RX_IQ`’s own switch block in `Loop.cpp` - neither changes what that button id does on any other screen.
 - **RequestRXIQAutoStop() (new, HardwareSm_ReceiveIQCalibration.cpp/ HardwareSm.h)**: sets a flag checked at the top of `ReadRXIQDelta()`. When set, instead of continuing the normal step-by-step sweep, it marks the current band complete, forces `currentBand = LAST_BAND`, and dispatches the ordinary `MIN_EXIT` event - the exact same completion path `AdjustRXIQBand()` already takes when the real last band finishes, so the state machine returns to `STANDBY` cleanly with no changes to the autogenerated `ReceiveIQCalSm` state machine itself (a direct external `AUTO_COMPLETE` dispatch would silently no-op outside the brief instant `AdjustRXIQBand()` itself runs - see that function’s comment for why). Wired to the `DECODER_TOGGLE` button id (repurposed only within this screen) - “Stop” aborts an in-progress Auto sweep early; harmless no-op if Auto isn’t running.
 - **“Exit”**: wired to the `DfE` button id (repurposed only within this screen) - plain `SetInterrupt(iCALIBRATE_EXIT)`, no `SaveDataToStorage()` call, mirroring the ordinary Home > Home Screen behavior elsewhere. Distinct from this screen’s existing `HOME_SCREEN` case (“Save & Exit”, unchanged).
 - **Root cause note**: a full Auto sweep is bounded (fixed step counts per band, not delta-dependent) but does sweep all `NUMBER_OF_BANDS` bands in turn, taking on the order of a couple of minutes - on GEN (the last band) this can look hung while its noisy IIR-filtered readings keep updating until the sweep naturally finishes. “Stop” is the fix requested rather than a deeper investigation into GEN specifically, since nothing band-specific in the auto-tune algorithm singles it out.

Version

- `SDT.h` `VERSION` bumped to “Mustang 2 V1.0.24”.

V1.0.23 (2026-09-30)

Added

- **PILINK_MSG_RXIQ_CAL (0x07) / PiLinkRXIQCalMsg (PiLink.h/PiLink.cpp)**: new downlink telemetry message feeding the Pi/web remotes’ new Receive IQ calibration screen (Menu > Calibration > Receive IQ), at the user’s request (“Need to add the Calibration Screen from T41 to the PI... Yes the PI needs tpo control calibraaion the same as the T41 would”). Sent at 10Hz (`PI_LINK_RXIQ_CAL_INTERVAL_MS`, same cadence as `STATUS/ PARAMETER`) from `SendRXIQCal()`, called from `UpdatePiLinkTelemetry()`. Carries whether `uisM.state_id == UISm_StateId_CALIBRATE_RX_IQ` right now, the band currently being calibrated, the current Amp/Phase step size index (`GetRXIQIncrementIndex()`, see below), and all 13 bands’ `ED.IQAmpCorrectionFactor/ED.IQPhaseCorrectionFactor/ GetRXDeltaVals()` values (scaled to `int16` - `x1000/x1000/x10` - 81 bytes total).
- **GetRXIQIncrementIndex() (MainBoard_DisplayCalibration_RXIQ.cpp)**: small new public accessor for `ChangeRXIQIncrement()`’s existing file-local `incindex` (toggles the Amp/Phase adjustment step between 0.01 and 0.001) - the only other piece of this screen’s state that wasn’t already readable from outside the file. No behavior change to `ChangeRXIQIncrement()` itself.

Notes

- **No new uplink message was needed**. Full remote control of this screen (Amp/Phase via Volume/Filter encoders, step-size toggle via the BEARING button, auto-tune via SPARE, band cycling via `BAND_UP/BAND_DN`, Save & Exit via `HOME_SCREEN`) routes entirely through the existing generic `PILINK_MSG_VIRTUAL_BUTTON/PILINK_MSG_VIRTUAL_ENCODER` messages, which `Loop.cpp`’s `UISm_StateId_CALIBRATE_RX_IQ` switch blocks already handle identically to a physical knob/button press - the same mechanism already used for the Pi’s menu navigation and Vol/BW/ Band/Mode/Demod dropdowns. Entry into the screen also needed no new mechanism: the existing Menu > Calibration > “Receive IQ” item (already mirrored on the Pi/web `_T41_MENU/T41_MENU` tables) already drives `StartRXIQCal()` via the normal menu-select sequence.
- Scope for this pass is Receive IQ calibration only, at the user’s request (“Lets start with Receive IQ and once thats done we’ll move to the other 3”) - Frequency, Transmit IQ, and Power calibration screens are deferred to future passes.
- Full detail on the Pi/web side of this feature (the new screen, its controls, and the deliberate omission of the live two-sideband spectrum mini-plot on the web remotes) is in `CHANGELOG_pidisplay.md`.

Version

- `SDT.h` `VERSION` bumped to “Mustang 2 V1.0.23”.

V1.0.22 (2026-09-30)

Added

- **PiLinkSettingsMsg.spectrumFillTraceEnabled (PiLink.h/PiLink.cpp)**: sends `ED.spectrumFillTraceEnabled` (the Menu > Display > “Spectrum fill on/off” setting added in V1.0.21) to the Pi over PiLink, at the user’s request, so the Pi’s own spectrum pane and both

web remotes can mirror it instead of always drawing filled regardless of what the T41 itself is set to - purely cosmetic, unrelated to the choppy-audio fix itself (already confirmed to be T41-side only). 23rd field in the Settings frame (58 bytes total, up from 57) - see `PiDisplay/pilink_client.py` for the matching struct-format change. **Requires reflashing the Teensy** (until then the whole Settings pane/New Line 2 strip on the Pi and web remotes goes stale, same as every prior field addition to this struct - see that file's history comment).

Version

- `SDT.h` `VERSION` bumped to "Mustang 2 V1.0.22".

V1.0.21 (2026-09-30)

Fixed

- **Choppy audio, root cause confirmed and fixed:** V1.0.20's test build (`SPECTRUM_FILL_TRACE` forced to 0) confirmed the filled/graded spectrum trace's draw time was delaying `loop()` (`Loop.cpp`) from getting back to `PerformSignalProcessing()` often enough, overflowing the continuously-running audio input - on the Teensy's own local speaker output as much as anywhere downstream (Pi, web remotes) - and invisible to the "Load:" reading the whole time (it only measures time inside `PerformSignalProcessing()`, never display-drawing time). User confirmed on V1.0.21 (via the new live toggle below): fill ON = choppy on every output (T41 local, Pi, both web remotes) with "Load:" reading a LOW 0-24%; fill OFF = clean audio everywhere with "Load:" reading a HIGHER 60-67%. **That "Load:" going UP when audio got FIXED is not a contradiction - it's the confirming signature of exactly this bug:** `elapsed_micros_sum` (`SDT.h`) only accumulates once per block that makes it all the way through `ReceiveProcessing()` to completion (see the end of that function, `DSP.cpp`) - a block dropped earlier by a full/overflowing audio queue (because `loop()` was stuck drawing and never got back around to service it) never reaches that point and is never counted at all. With the fill on, most blocks were being silently dropped rather than processed, so few completions were left to average into "Load:", making the DSP chain look artificially idle even as the audio was falling apart. With the fill off, every block gets serviced on time, none are dropped, and "Load:" finally reads what this receiver's DSP chain actually, normally costs (60-67%) - a healthy number for a full decimate/filter/AGC/demod/NR chain at this sample rate, not a new problem. If "Load:" alone is ever checked again for a similar report, remember a suspiciously LOW reading during a choppy-audio complaint can mean blocks are being dropped before they're ever measured, not that the DSP chain is idle.

Added

- `ED.spectrumFillTraceEnabled` (`SDT.h`) + **Menu > Display > "Spectrum fill on"/"Spectrum fill off"**: turns the diagnostic test build into a permanent, live setting instead of a compile-time `#define` — so the filled-trace look and this exact failure mode (should it ever resurface on a strong/wide enough signal) can be traded off from the radio itself, no reflash needed either way. Defaults to on (1), matching the shipped look before V1.0.20's test. `BuildFrozenBackdrop()` (the Fine-Tune frozen-trace backdrop, `#ifdef MUTE_ON_RAPID_TUNE`) was updated to match the live setting too, so the frozen trace's look doesn't drift from the live one.

Version

- `SDT.h` `VERSION` bumped to "Mustang 2 V1.0.21".

V1.0.20 (2026-09-30) — TEST BUILD, diagnostic only

Changed

- `SPECTRUM_FILL_TRACE` (`MainBoard_DisplayHome.cpp`) **temporarily set to 0 (outline-only spectrum trace) instead of 1 (filled trace)**. The "choppy audio" report continued after V1.0.19's `ApplyRFGain()` fix, and after confirming both DSP Gain and Noise Reduction/Auto-Notch make no difference — with the "Load:" reading (Menu > Diagnostics / Page 1) holding steady at a normal 0-25% throughout, ruling out CPU starvation *inside* `PerformSignalProcessing()`. But "Load:" only measures time inside that one call (`elapsed_micros_sum` is only accumulated around it) — it says nothing about `DrawDisplay()`, which `loop()` (`Loop.cpp`) calls immediately afterward, every single pass, on the same core. The filled-trace code this flag controls (`FillSpectrumColumnGraded()`) already carried its own comment noting it scales with a strong/broadband signal on screen; if a slow sweep's draw time delays `loop()` from getting back to `PerformSignalProcessing()` in time, the continuously-running I2S/DMA audio input has nowhere to go but overflow — on the Teensy's own local output as much as anywhere downstream, since both come from the same stalled pipeline. Setting this to 0 isolates that one variable for testing. **This is a diagnostic test build, not a shipped fix** — revert to 1 once the result is confirmed either way (see `CHANGELOG.md` for the outcome once known).

Version

- `SDT.h` `VERSION` bumped to "Mustang 2 V1.0.20".

V1.0.19 (2026-09-30)

Changed

- `ApplyRFGain()` (`DSP.cpp`) **caches its computed linear gain instead of recomputing `pow(10, dB/20)` on every audio block**. Found while chasing a "sound cutting out very rapidly" report that was traced step by step all the way back to the Teensy's own local audio output (confirmed independent of USB, the Pi, and the web remotes — a raw `sd.rec()` capture on the Pi and the Teensy's own

speaker/headphone output were BOTH choppy). `ApplyRFGain()` runs on every single audio block regardless of mode (called from all 3 of `ProcessIQData()`'s demod-mode branches), but `rfGainAllBands_dB/bandGain_dB` only actually change when the user adjusts `Menu > RF Options > RX DSP Gain` or switches bands — both rare next to the block rate. Two `pow()` calls (a real cost on a microcontroller with no hardware support for it) were being paid every block for values that are usually identical block after block; now only recomputed when the dB input actually changes. **This read of `ED.rfGainAllBands_dB` has been “live every audio block” since at least V1.0.6** (see V1.0.17's entry below), so this isn't a newly introduced cost — if the choppiness persists after this, it wasn't the (or the only) cause, and the rest of the DSP chain (filters/AGC/noise reduction) is the next place to look.

Version

- `VERSION` (`SDT.h`) bumped to `"Mustang 2 V1.0.19"`.

V1.0.18 (2026-09-30)

Added

- `PiLinkSettingsMsg` gained `spectrumFloor/spectrumFloorAuto`** (`PiLink.h/PiLink.cpp`; struct grew 54 → 57 bytes, appended right after V1.0.17's `spectrumScale`) — `ED.spectrumNoiseFloor[currentBand]` (the manual “Spectrum floor” menu value, a raw screen-pixel baseline shift, -100..100, NOT a dB value) and `ED.spectrumFloorAuto[currentBand]` (whether “Auto spectrum floor” is active, in which case the manual value isn't what's actually driving the trace). At the user's follow-up request, right after the same-day `spectrumScale` addition. Backs a new “Floor:” reading on the Pi/web remotes' New Line 2 strip, right after Scale: — shows “Auto” while auto mode is on, the signed manual value otherwise (see `CHANGELOG_pidisplay.md/CHANGELOG_remote.md/CHANGELOG_remote_mobile.md` for the Pi-side detail). **Requires reflashing the Teensy together with re-syncing `PiDisplay/`** — same “entire Settings pane/New Line 2 strip goes stale until both sides agree on the new byte layout” caveat as V1.0.17's `spectrumScale`.

Version

- `VERSION` (`SDT.h`) bumped to `"Mustang 2 V1.0.18"`.

V1.0.17 (2026-09-30)

Added

- `RFSet[]` (`MainBoard_DisplayMenus.cpp`) gained a live “RX DSP Gain” entry** — `Menu > RF Options > RX DSP Gain`, -5.0 to 20.0 dB in 0.5 dB steps, using the gain `VariableParameter` already bound to `&ED.rfGainAllBands_dB` (defined since at least V1.0.6, when `DSP Gain:` first appeared read-only on the Settings pane) but never wired into a menu — its `RFSet[]` line sat commented out as `"[__RX DSP Gain]"`. Found at the user's report “I don't see a way to change RF gain on any menu options” while diagnosing a weak-signal spectrum-display complaint. `postUpdateFunc` is `NULL`: `ApplyRFGain()` (`DSP.cpp`) already reads `ED.rfGainAllBands_dB` live every audio block, so no separate “apply” step is needed, unlike `RX Attenuation's UpdateRatten()` (which writes to actual attenuator hardware over I2C). `"TX Attenuation(CW)"` (`"[__TX Attenuation(CW)]"`) stays commented out/hidden - not part of this change.
- `PiLinkSettingsMsg` gained `spectrumScale`** (`PiLink.h/PiLink.cpp`; struct grew 53 → 54 bytes, appended at the end) — `ED.spectrumScale[currentBand]`, the index into `displayScale[]` (`MainBoard_DisplayHome.cpp`) that `Menu > Display > “Spectrum scale”` sets. Backs a new “Scale:” reading in the Pi/web remotes' New Line 2 strip, right after AGC: (see `CHANGELOG_pidisplay.md/CHANGELOG_remote.md/CHANGELOG_remote_mobile.md` for the Pi-side detail - a new `SPECTRUM_SCALE_TEXT`-style literal copy of `displayScale[].dbText`, same convention the AGC/Noise label tables already use). **Requires reflashing the Teensy together with re-syncing `PiDisplay/`** — `pilink_client.py`'s `SETTINGS_SIZE` check will keep failing on the old 53-byte frame, and the Pi's *entire* Settings pane/New Line 2 strip goes stale (not just this one field) until both sides agree on the new 54-byte layout.

Version

- `VERSION` (`SDT.h`) bumped to `"Mustang 2 V1.0.17"` — covers both items above, neither of which had been reflashed yet when this bump was made.

V1.0.16 (2026-09-27)

Added

- `PILINK_MSG_VIRTUAL_BUTTON_REPEAT (0x13)`**, a new uplink message with payload `[button_id, count]`. `DispatchFrame()` (`PiLink.cpp`) queues `count` back-to-back `SetButton(button_id) + SetInterrupt(iBUTTON_PRESSED)` calls in one shot instead of one press per frame. Fixes the user-reported “why does it have to go through all the levels” complaint for `Zoom/Band/ Mode/Demod/Noise/Course Inc/Fine Inc/Vol/BW` selection: the Pi side used to send one `PILINK_MSG_VIRTUAL_BUTTON` per step, paced 300ms apart (needed only to stop a *different* button id from clobbering the single-slot `ButtonPressed` mailbox before `ConsumeInterrupt()` drains the previous press's FIFO marker — a real risk between different ids, but pointless when every press in a run is the same id). With the new message, the Pi asks for N presses of the same button at once and the Teensy queues them without any inter-press delay. `ConsumeInterrupt()/HandleButtonPress()` themselves are unmodified — this reuses the exact same per-control logic (DSP reconfiguration, etc.), it just removes the artificial pacing between repeats of the same button. Note this does not add a reverse direction to any button: `Zoom` (and `Mode/Demod/Noise/ Course Inc/Fine Inc/Vol/BW`) genuinely have no “down” counterpart in `Loop.cpp`'s `HandleButtonPress()`, so going 8x back to 1x zoom still passes through 16x internally — it just does so instantly instead of visibly, ~300ms per step. **Requires a Teensy reflash, and it must be reflashed together with re-syncing `PiDisplay/`** — the Pi's `radio_ui.py` no longer falls back to

the old one-press-at-a-time path, so an un-reflashed Teensy won't respond correctly to Zoom/Band/Mode/ Demod/Noise/Course Inc/Fine Inc/Vol/BW selection until both sides are updated. See `CHANGELOG_pidisplay.md` for the Pi-side detail (`pilink_client.send_virtual_button_repeat()`, `radio_ui.py`'s `_cycle_button_to()`).

V1.0.15 (2026-09-26) — version bump confirming the Menu-index reflash

Version bump only, so the Teensy's own screen and the Pi's Settings pane carry a visible marker that the reflash below actually took, rather than relying on the button behavior alone to prove it.

V1.0.14 (2026-09-25)

Added

- **PILINK_ENCODER_TUNE / PILINK_ENCODER_FINETUNE**: the main VFO tuning knob and the fine-tuning knob now have virtual-encoder ids, wired to the same `iCENTERTUNE_INCREASE/DECREASE/iFINETUNE_INCREASE/DECREASE` events their own physical ISRs post. Previously PiLink only exposed Filter/Volume remotely, so nothing (Pi touchscreen or browser) could move the VFO at all.

Fixed

- **Live-hardware PiLinkSettingsMsg wire-size mismatch (source reconciliation still open)**: the Pi-side fix landed in `pilink_client.py` (reverting `SETTINGS_FMT` back to the 53-byte, 19-field format the real hardware actually sends), but the underlying issue is that the firmware source files checked in on the Mac (`PiLink.h/PiLink.cpp`) don't match what's actually flashed — they show only 17 fields/51 bytes, missing `audioVolume/` `secondScreenActive`. **Open item**: those source files still need reconciling against whatever's actually flashed (find/recover the real `PiLink.h/.cpp`, or hand-add the two fields back with a confirmed guess at their types and verify against a reflash) before any future firmware change touches this struct.
- **Permanent fix for remote Menu selection landing back on Home instead of sticking on the picked setting's adjustment screen**: `primaryMenuIndex/secondaryMenuIndex` (`MainBoard_DisplayMenus.cpp`) added to `PiLinkStatusMsg` (`PiLink.h`) as two persistent `uint8_t` globals, sent every `SendStatus()` call — continuously, in the already-frequent `STATUS` frame, not gated to any particular screen state — so the Pi's copy of the Menu's real position is never more than one `STATUS` frame stale, closing out the whole class of bug (there's no longer any guessed/shadowed state for a remote Menu click to compute from). Confirmed with the version bump in V1.0.15 above.

V1.0.13 (2026-09-25)

Added

- **Frequency-scale ruler telemetry**. `PILINK_MSG_STATUS` gained `lowerFreqHz/upperFreqHz` (the visible spectrum span's edges — `GetLowerFreq_Hz()/GetUpperFreq_Hz()`), not the same as the already-sent `frequencyHz`, which differs by `ED.fineTuneFreq_Hz`) and `bandLowHz/bandHighHz` (the current band's legal limits, `bands[ED.currentBand].fBandLow_Hz/.fBandHigh_Hz`) — the data behind the Pi's own frequency-scale ruler (tick marks, labels, red out-of-band coloring) between the trace box and the waterfall.

V1.0.12 (2026-09-25)

Added

- **Secondary (“Toggle Layout”) home-screen telemetry**. `PILINK_MSG_SETTINGS` gained `secondScreenActive` (plus `audioVolume`, always `ED.audioVolume` regardless of which function the volume knob currently multiplexes to) — lets the Pi/web remote mirror whichever of the Teensy's own 11-pane primary or 5-pane secondary home-screen layouts is currently active, following a screen switch made at the front panel too, not just one requested remotely.

V1.0.11 (2026-09-25)

Fixed

- **Menu > Display changes (“Spectrum Scale”, “Spectrum floor”) weren't reaching the Pi's spectrum/waterfall pane**. `PiLink.cpp`'s `SendSpectrum()` was quantizing every bin against a fixed -140..-20dBm window, completely independent of `ED.spectrumScale[]/ED.spectrumNoiseFloor[]`. `SendSpectrum()` now reuses the Teensy's own `pixelnew()` (the exact per-bin calculation `ShowSpectrum()` draws the real trace from) plus the noise-floor shift, normalized against the same on-screen box height via a new `GetSpectrumHeightPx()` accessor — so both settings now affect the Pi's display the same way they affect the Teensy's own.

V1.0.10 (2026-09-25)

Added

- **PILINK_MSG_VIRTUAL_ENCODER extended from a 1-byte to a 2-byte payload** (`[encoder id, direction]`), with new `PILINK_ENCODER_FILTER/` `PILINK_ENCODER_VOLUME` ids — lets one message drive either the Filter or the Volume encoder's interrupt,

backing the Pi's new Vol dropdown (cycles Vol -> AGC -> Mic -> STn via the real VOLUME_BUTTON cycle, Loop.cpp) and the mouse wheel adjusting whichever of those four is currently selected.

V1.0.9 (2026-09-25)

Added

- **PILINK_MSG_PARAMETER**, sent at 10Hz while `uiSM.state_id == UISm_StateId_UPDATE` — backs the Menu parameter-adjustment overlay (the red-boxed “Label: value” readout, mirroring the Teensy's own `DrawParameter()`) and lets a remote mouse wheel adjust the value while that box is showing, via one virtual Filter-encoder click per tick — exactly like turning the real encoder.

V1.0.8 (2026-09-25)

Added

- **PILINK_MSG_VIRTUAL_ENCODER** (new uplink message), injecting the same `iFILTER_INCREASE/iFILTER_DECREASE` interrupt the physical encoder's ISR posts — needed so a remote Menu item pick (which needs Filter-encoder turns, not button presses) can select a specific menu item. Backs the Pi's Band dropdown (160M...4M, from the real `bands[]/BAND_UP/BAND_DN` cycle) and eight new toolbar dropdowns built straight from firmware source: Mode (`TOGGLE_MODE`'s real cycle), Demod (`DEMODULATION`), Noise (`NOISE_REDUCTION`), Course Inc/Fine Inc (the real `incrementValues[]/selectFT[]` tables), Decode (now backed by a new `decoderOn` Settings field), Home (`HOME_SCREEN/SPARE`'s `ToggleHomeScreen()`), and a real two-level Menu ported verbatim from `MainBoard_DisplayMenus.cpp`'s `primaryMenu[]`.

V1.0.7 (2026-09-25)

Fixed

- **TX/RX status pane showed “XMT” while the radio was actually receiving.** `SendStatus()` was reading the raw PTT hardware pin (`digitalRead(PTT)`), which can read spuriously high with no relation to whether the radio is actually transmitting. Now derived from `modeSM.state_id` — the exact same source the Teensy's own on-screen REC/XMT indicator (`DrawTXRXStatusPane()`) uses.

V1.0.6 (2026-09-25)

Added

- **PILINK_MSG_SETTINGS**, sent at 2Hz — backs full Settings-pane parity with the T41's own Settings page 1 (Vol/AGC/Mic/STn slot, AGC mode, Tune Increment, Fine-Tune Increment, RX Attenuation, Auto-Notch, TX Attenuation, Noise Reduction, DSP Gain, Antenna Selection, Spectrum Zoom, Sample Rate, CPU Temp, CPU Load, and the running firmware version string).

V1.0.5 (2026-09-25)

Added

- **PILINK_MSG_AUDIO_SPECTRUM** — backs the genuinely live magenta audio-FFT bar display mirroring `ShowSpectrum()`'s `audioYPixel[]` plot in the audio-spectrum pane.

V1.0.4 (2026-09-25)

Fixed

- **Spectrum/waterfall showed outlines but no data.** The `Serial1` TX buffer was far smaller than a spectrum frame (518 bytes), so `SendFrame()`'s non-blocking guard silently dropped every spectrum frame, forever. Fixed with an explicit 1024-byte `addMemoryForWrite()` buffer in `InitializePiLink()`.

V1.0.3 (2026-09-25)

Fixed

- **Spectrum/waterfall data was pinned to full-scale** (a flat line, solid-colored waterfall): `SendSpectrum()` was sending the raw `psdnew[]` FFT magnitude directly into `QuantizeDbm()`, which expects real dBm. Fixed by converting via `dBm = psdnew*10 + RECEIVE_POWER_OFFSET`, the same conversion the Teensy's own trace (`pixelnew()`) uses.

V1.0.2 and earlier

Baseline PiLink protocol: STATUS/SMETER/SPECTRUM messages and the virtual-button uplink — predates this changelog’s start and isn’t individually dated.