WSJT-X / QMAP-improved

Beta-tester guide — N6NU fork

Document built 2026-05-07.

This is a beta build of WSJT-X with K1JT's **QMAP** extended to support wideband (256 kHz) spectrum processing, multi-instance side-by-side operation, and a family of seven **SDR bridges** that feed the wideband data into QMAP. What follows is the testers' overview: what's new, how to install, why the bridges exist, and how to set each one up. A technical reference is at the end for those who want it.

Author: **Andreas Junge, N6NU** <n6nu@n6nu.org>. Bug reports + feedback go to the GitHub issue tracker on the individual bridge repos (`n6nu/<bridge>`) or directly to N6NU. Underlying QMAP-improved branch is `n6nu/qmap-improved` on `github.com/dg2ycb/wsjtxi`.

1. What's new for testers

If you've used DG2YCB's stock WSJT-X-improved before, this build is a drop-in replacement that adds the items below. All existing 96 kHz workflows keep working unchanged — wide mode is opt-in.

Wider waterfall (up to 256 kHz)

QMAP's WideGraph window can now display up to **256 kHz** of spectrum (vs. the upstream's fixed 96 kHz). That's enough to see the full Q65 EME calling segment plus surrounding activity in one view, on bands where DX is spread across the spectrum.

How to use: open Settings (`Ctrl`+S) → Linrad input → "Sample rate mode" combo → pick `96/128/192/256 kHz` or leave on `Auto` (recommended — let the bridge tell QMAP what rate to expect, and the WideGraph auto-fits the span). The WideGraph itself has a separate "FreqSpan" combo with `Auto / 96 / 256 / User` options for fine-grained zoom control.

Multi-instance side-by-side QMAPs

You can now run two (or more) QMAPs at once on the same machine, each with its own settings, listening on different UDP ports. Useful for monitoring two bands at once, or running a 96 kHz reference QMAP alongside a 256 kHz wide one while you A/B compare decode performance.

```
# Instance A — 96 kHz reference
qmap.exe --config qmap-A.ini

# Instance B — 256 kHz wide-mode (different bridge, different ports)
qmap.exe --config qmap-B.ini
```

Each `--config` INI sets its own `UDPport`, `tcp_port`, sample rate, etc. The two QMAPs are fully independent — closing one doesn't affect the other.

Settings dialog: editable host + ports

DevSetup (Settings dialog) gained editors for the Linrad host (default `127.0.0.1`) and the Linrad TCP parameter-server port (default `49812`). Useful if you run the bridge on a different machine on the LAN, or if you're running multiple bridge/QMAP pairs that need separate ports.

Auto-relaunch on bridge rate change

If the bridge changes its IQ rate while QMAP is running (common if you switch a bridge from 96 to 256 kHz), QMAP detects the new rate via the Linrad TCP probe, posts a quick yellow banner in the WideGraph, and relaunches itself with the new rate baked in. Saves you the dance of restarting QMAP manually.

No-stream + probe-failed diagnostics

If the bridge isn't running, the WideGraph rate label shows "`probe failed at 127.0.0.1:49812`" instead of silently falling back to baseline 96 kHz with no indication. The most common gotcha — a stale `tcp_port` in qmap.ini left over from multi-instance testing — used to require digging into the INI; now it's visible at a glance.

DC blocker default-off for RF-direct radios

Bridge-side DC blocker (the IIR high-pass that strips the centre spike) defaults to **OFF** on RF-direct receivers (HackRF, RTL-SDR, SDRplay, Pluto, AirSpy) since their hardware DC correction handles it upstream. The checkbox stays editable for diagnostic A/B. Sound-card-IQ sources (FunCube Pro+ V2, FlexRadio DAX-IQ, Malachite via iq-rx-bridge) keep DC blocker default-ON because they have no hardware DC mitigation. Setting persists across launches via INI.

2. Installing the beta

Two pieces to install: **WSJT-X (with the improved QMAP)** and the **SDR bridge for your hardware**.

WSJT-X with improved QMAP

Provided as a Windows installer separately by N6NU. Installs to `C:\WSJTX` alongside (not over) any stock WSJT-X you may already have. Existing user data and logs at `%LOCALAPPDATA%\WSJT-X` are preserved.

SDR bridge for your hardware

Pick the bridge for your SDR from the table below, download the v1.1.4 installer, run it, follow the per-bridge setup in section 4.

Your SDR	Bridge	Download URL
HackRF One	hackrf-rx-bridge	https://github.com/n6nu/HackRF-RX-Bridge/releases/latest
RTL-SDR (R820T2, V3+)	rtlsdr-rx-bridge	https://github.com/n6nu/rtlsdr-rx-bridge/releases/latest
SDRplay (RSP1A, RSPduo, RSPdx)	sdrplay-rx-bridge	https://github.com/n6nu/sdrplay-rx-bridge/releases/latest
AirSpy R2	airspy-rx-bridge	https://github.com/n6nu/airspy-rx-bridge/releases/latest
ADALM-Pluto / Pluto+ (RX only)	pluto-rx-bridge	https://github.com/n6nu/pluto-rx-bridge/releases/latest
ADALM-Pluto / Pluto+ (TX as well)	pluto-wsjtx-bridge	https://github.com/n6nu/pluto-wsjtx-bridge/releases/latest
FunCube Pro+ V2, Malachite, etc.	iq-rx-bridge	https://github.com/n6nu/iq-rx-bridge/releases/latest

3. Why a bridge?

QMAP doesn't talk to your SDR directly. By design — and carried over from the original Linrad / MAP65 stack on which QMAP is built — it expects a wideband I/Q stream on a specific UDP wire format ("Linrad packets"). The bridges provide that translation:

SDR / Source	→	Bridge	→	QMAP / WSJT-X
HackRF, RTL-SDR, SDRplay, AirSpy, Pluto, FunCube...	→	bridge-specific SDR API + resampler + Linrad UDP encoder	→	UDP 50004 wideband I/Q (QMAP) + UDP 2237 dial freq / TX state (WSJT-X)

What the bridge does for you:

• **Talks to your SDR** using its native API (HackRF API, RTL-SDR API, SDRplay API, libairspy, libiio, sound-card IQ). You don't need a separate SDR app — the bridge IS that app.

• **Resamples to the QMAP rate** (96 / 128 / 192 / 256 kHz) from whatever the SDR's native rate is (typically 2 Msps or 4 Msps).

• **Re-emits as Linrad UDP** on port 50004 — the wire format QMAP listens on. Plus a TCP parameter server on 49812 for QMAP to query the actual rate.

• **Listens to WSJT-X** on UDP 2237 for dial-frequency changes and TX state. When WSJT-X retunes, the bridge retunes the SDR. When WSJT-X transmits, the bridge mutes the SDR's audio path so it doesn't shoot itself in the foot.

• **Demodulates SSB to your real audio path** (typically VB-Cable Line 1 → WSJT-X RX) so WSJT-X gets normal narrow-band audio for FT8/FT4 alongside QMAP's wideband Q65 decoding.

Your *real* radio (IC-905, IC-705, FT-991A, FlexRadio, etc.) keeps doing TX duty. The SDR-via-bridge handles wideband RX. That's why the family is mostly `-rx-bridge` — they're RX-only complementary feeders, not full transceivers.

Network ports used

Port	Protocol	Direction	What
50004	UDP	bridge → QMAP	Linrad wideband I/Q
49812	TCP	bridge ← QMAP	Linrad parameter probe (rate / centre freq)
2237	UDP	WSJT-X → bridge	Dial freq, TX/RX state, mode
4533+	TCP	WSJT-X ← bridge (opt)	Hamlib rigctld CAT for Doppler tracking

All on `127.0.0.1` by default. The Linrad host + TCP port are now editable in QMAP's DevSetup if you split bridge and QMAP across machines.

4. Per-bridge setup

Each bridge ships as a standalone Windows app with its own GUI. Common pattern:

• Connect the SDR to USB. Confirm it appears in Device Manager.
• Launch the bridge. Pick the right device and gain in Settings.
• Set the bridge's IQ output rate to match QMAP — start with 96 kHz, move to 256 kHz once you've confirmed the chain works.
• Start QMAP. The auto-detect probe should pick up the bridge rate automatically. WideGraph rate label should show `Linrad: 96 kHz (auto)` or similar.
• Start WSJT-X. Confirm dial-freq changes propagate to the bridge (waterfall centre shifts).

Per-bridge specifics follow.

hackrf-rx-bridge — HackRF One

Hardware: HackRF One. 8-bit ADC, 2 Msps native, RX-only path here.

Drivers: requires **WinUSB** (not the default WCID driver). The installer bundles `zadig.exe` — run it once after first plug-in to bind WinUSB to the HackRF interface. Restart the bridge after Zadig.

Settings to set: LNA gain (default 24 dB is fine for most), VGA gain (default 24 dB), Amp on/off (off for above 100 MHz, on for HF if signal is weak). Sample rate combo: 96/128/192/256 kHz.

Known quirks: at 256 kHz wide mode the HackRF's 8-bit ADC starts limiting dynamic range — you may see strong stations modulate the noise floor at adjacent offsets. Acceptable for Q65 EME at 2 m / 70 cm; less so for crowded HF bands. The bridge's DC blocker is default-OFF (the chip handles DC) — keep it that way unless you see a residual centre spike, in which case toggle it on for diagnostic A/B.

rtlsdr-rx-bridge — RTL-SDR (R820T2 / V3+)

Hardware: any R820T2-based RTL-SDR dongle. RTL-SDR.com V3+ preferred for HF (Q-channel direct-sampling). 8-bit ADC, max 2.4 Msps native.

Drivers: WinUSB via `zadig.exe` (bundled). Bind to the dongle's bulk interface. Don't bind to the DAB+ TV interface — that's a different driver.

Settings: Tuner gain (auto or fixed 30-40 dB), AGC on off (default off), bias-tee on/off (on for V3+ feeding an LNA), PPM correction (calibrate against a known reference). For HF below 25 MHz the bridge auto-flips to direct sampling Q-channel.

Known quirks: R820T2 has known LO leakage at the centre frequency. DC blocker is default-OFF (hardware DC correction handles it via the API). G3WDG reported a 100 Hz residual centre spike when the software HP was enabled in v1.1.2 — v1.1.3+ default-off avoids it; v1.1.4 lets you enable for diagnostic comparison.

sdrplay-rx-bridge — SDRplay (RSP1A / RSPduo / RSPdx)

Hardware: any sdrplay_api-supported device. RSPduo + RSPdx preferred for VHF/UHF EME (better front-end). 14-bit ADC, various sample-rate ladders depending on model.

Drivers: install SDRplay's `API 3.x` service from the SDRplay website BEFORE running the bridge. The bridge talks to the service via shared memory, doesn't bind the USB device directly. No Zadig needed.

Settings: Device antenna port (RSPduo: tuner 1A vs tuner 1B vs tuner 2; RSPdx: HF / VHF), IF mode (Zero-IF for wide-mode 256 kHz; Low-IF works for narrow modes but adds DC handling), gain reduction (0-59 dB; lower = more gain), RF / IF gain. 256 kHz wide mode requires Zero-IF.

Known quirks: first-restart-fails on RSP-series — the sdrplay_api device handle isn't released cleanly when the bridge exits, so re-launching within ~5 s of close fails the first time. Wait 10 s or kill the SDRplay service. Documented in the bridge's own About box.

airspy-rx-bridge — AirSpy R2

Hardware: AirSpy R2 (12-bit ADC, 10 Msps native, 24-1750 MHz). Mini and HF+ Discovery work too with reduced bandwidth.

Drivers: WinUSB via `zadig.exe` (bundled). Bind once after first plug-in.

Settings: linearity / sensitivity gain mode, mixer gain, VGA gain, IF gain. Default mode "linearity" with gain index 10-14 is a reasonable starting point. R2's 10 Msps decimates down to QMAP rate cleanly via the bridge's resampler.

Known quirks: v0.99.3 was the long-standing beta; v1.0.0 promotion happened after R2 tester confirmation. Should be stable in v1.1.4. Watch for stalls on USB hubs that share bandwidth with USB 2.0 audio devices — direct-to-host USB 3.0 preferred.

pluto-rx-bridge — ADALM-Pluto (RX only)

Hardware: ADALM-Pluto (Analog Devices) or Pluto+ knockoff. 12-bit ADC, AD9363 transceiver, 70 MHz - 6 GHz. RX path only here — your real radio still does TX (use pluto-wsjtx-bridge instead if you want Pluto to TX as well).

Drivers: install Analog Devices' **libiio** (`https://wiki.analog.com/university/tools/pluto/drivers/windows`). Sets up USB-Ethernet adapter on Windows; Pluto appears at `192.168.2.1` by default. Bridge talks via libiio over USB or LAN.

Settings: RF gain mode (manual or AGC), RF gain (0-71 dB), RX bandwidth, sample rate. The Pluto has wider native bandwidth than HackRF/RTL-SDR so 256 kHz wide mode is comfortable.

Known quirks: default firmware caps tuning at 2.4 GHz. If you have F5OEO's Tezuka or similar extended-range firmware flashed, the bridge picks it up automatically. Pluto+ (plutoplus.github.io) has dual RX channels but only RX1 is wired here.

pluto-wsjtx-bridge — ADALM-Pluto with TX

Same hardware as pluto-rx-bridge but with the Pluto's TX path enabled. Use this if the Pluto is your only radio (e.g. QO-100 satellite work, indoor low-power tests). Otherwise stick with pluto-rx-bridge so a real rig handles TX.

Drivers / settings: same as pluto-rx-bridge.

Known quirks: at full TX power the AD9363 can heat the case; if you push high duty cycle Q65-60 add a fan. WSJT-X TX audio routes through the bridge's local audio device — confirm WSJT-X's TX audio source matches.

iq-rx-bridge — Sound-card IQ (FunCube Pro+ V2, FlexRadio DAX-IQ, Malachite)

Hardware: any USB sound card delivering stereo IQ as audio (L=I, R=Q). Common cases: FunCube Pro+ V2 (192 kHz dongle), FlexRadio DAX-IQ virtual card, Malachite SDR audio output. Wide-mode capped at the card's native rate (192 kHz on FunCube Pro+ V2; 256 kHz works on FlexRadio DAX).

Drivers: Windows audio class — no special drivers. Card must be visible in Sound Control Panel as a recording device. Set Windows audio device to 24-bit, card-native-sample-rate, stereo.

Settings: Pick the IQ card from the bridge's Device list. Sample rate auto-snaps to what the card reports. DC blocker default-ON (sound-card sources have no hardware DC mitigation; the LO leakage at centre is real and the IIR HP notches it cleanly). Capability-gated rate combo: the bridge only offers rates the card actually supports.

Known quirks: some FunCube units quote 192 kHz but actually deliver slightly drift-y rates (191.9xxx); the bridge's resampler tolerates ±0.1%. If you see decode rate drop over time, check Windows' "Exclusive mode" on the audio device — disable it for the bridge.

5. Troubleshooting

Symptom	Likely cause / fix
QMAP says "probe failed at 127.0.0.1:49812"	Bridge isn't running, or it crashed, or the TCP port in qmap.ini is stale. Re-launch bridg
Yellow rate-mismatch warning in WideGraph	CLI/INI override forces a rate the bridge isn't actually sending. Either change the overri
Waterfall is alive but no decodes	Sub-mode mismatch (e.g. file is Q65-60D but QMAP set to 60A). Try the other sub-mo
QMAP relaunches itself unexpectedly	Bridge changed its IQ rate during operation. The auto-relaunch is the intended behavio
Centre-frequency spike on the waterfall	DC blocker setting. Try toggling it in bridge Settings (default OFF for RF-direct, ON for
First QMAP launch decodes, second one in the same minute fails	UDP port 50004 conflict between two QMAP instances. Use --config to give each its ow
Bridge crashes on close	Known SDRplay quirk (api device-handle release timing). Wait 10 s before re-launchin

6. Technical reference

For testers who want to understand what's actually been modified inside QMAP and the bridges. Skip this if you just want to make decodes happen.

Wire format (Linrad UDP)

QMAP listens on UDP port 50004 for Linrad-style packets: 24-byte header + IQ payload. Header carries centre frequency, block number, sample format flag, etc. The classic 96 kHz Linrad packet is 1416 bytes total (24 header + 1392 payload = 348 IQ pairs as little-endian int16 I,Q,I,Q,…).

Wide mode keeps the same on-wire format but scales the pairs-per-packet: 696 pairs at 256 kHz (so the per-packet duration stays constant at ~3.6 ms, packet rate goes up). QMAP derives the pair count from the packet's payload-byte size and the `nrx` mode flag — no explicit length field. Both the bridge and QMAP scale their buffers via runtime config.

Whitening: the bridge applies `(-1)^n·conj()` to the IQ before packing. This compensates for QMAP's FFT direction (FFTW_BACKWARD = positive exponent), which would otherwise flip the spectrum. It's a fixed transform, involutive, and matches what stock Linrad sources have always done. The bridge also handles it on the .iq mapsim path (harness-only) by pre-cancelling so LinradServer's transform ends up as identity.

Multi-instance plumbing

`--config <path>` tells QMAP to read the named INI instead of the default `qmap.ini` alongside the executable. Each instance reads its own UDPport, tcp_port, sample_rate_mode, etc. The instances don't share any state.

WSJT-X also gets its own `--rig-name <label>` for parallel ops, but that's a stock K1JT feature — unchanged in this fork.

Sample-rate plumbing

QMAP-improved exposes a `qmap_runtime` C++ namespace + Fortran `qmap_params` module. Active rate (`activeRateHz()`) is set once at startup from this precedence:

1. `--samplerate <hz>` CLI flag
2. `qmap.ini [Linrad]/sample_rate_mode = <hz>`
3. `qmap.ini [Linrad]/sample_rate_mode = auto` (default) → query bridge via Linrad TCP probe
4. Baseline 96 kHz if all else fails

FFT size scales with rate: NFFT = 32768 at 96 kHz baseline, 131072 at 256 kHz wide-mode. COMMON-block buffers (`dd`, `ss`, `savg`) are sized at the wide-mode upper bound (256 kHz × 60 s × 2 channels × 4 bytes = ~123

MB) so any active rate fits without re-allocation.

Decoder timing instrumentation

QMAP gained `--decode-log <path>` (per-decode line tap) and `--decoder-timing-log <path>` (K1JT's existing per-stage timer instrumentation, now wired up). Each decoder cycle writes one summary block showing time spent in fftbig vs candidate finder vs q65b inner. Useful for diagnosing why a 256 kHz cycle is approaching the 6 s real-time bail in qmapa.f90:87.

These flags are primarily for the regression harness at `C:\dev\Q65\testfiles\` but harmless if passed manually for diagnostic runs.

Real-time decode deadline

Q65-60 has three decoder triggers per TR cycle: ihsym = 130 / 330 / 390 (≈ 19.5 / 49.5 / 58.5 s past PC-clock minute). For auto-sequencing in WSJT-X, cycle 2 (ihsym=330) needs to finish in ≤6 s — leaves ~4 s of headroom before the next TR boundary, which WSJT-X needs to set up TX. K1JT enforces this with a hard `exit` if `tsec > 6.0` on the candidate loop. At 256 kHz on a modern CPU, real measurements show 0.3 - 6 s decoder kernel time, comfortably within budget.

7. Reporting issues

The most useful bug report for a tester to file:

1. **What hardware** — SDR model, real-radio model, antenna, frequency.
2. **What software** — exact version of WSJT-X (`Help → About`), QMAP version (`Help → About`), bridge version (in About box; should be 1.1.4).
3. **What you expected vs what happened** — screenshot of the WideGraph + decoded text browser is gold.
4. **QMAP stderr / log files** if available. The bridge writes its own log next to the .exe (look for `<bridge>.log`).

Issue tracker: each bridge has its own GitHub issues page at `github.com/n6nu/<bridge>/issues`. QMAP-improved bugs go to `github.com/dg2ycb/wsjtxi/issues` (or directly to N6NU since DG2YCB's repo is private).