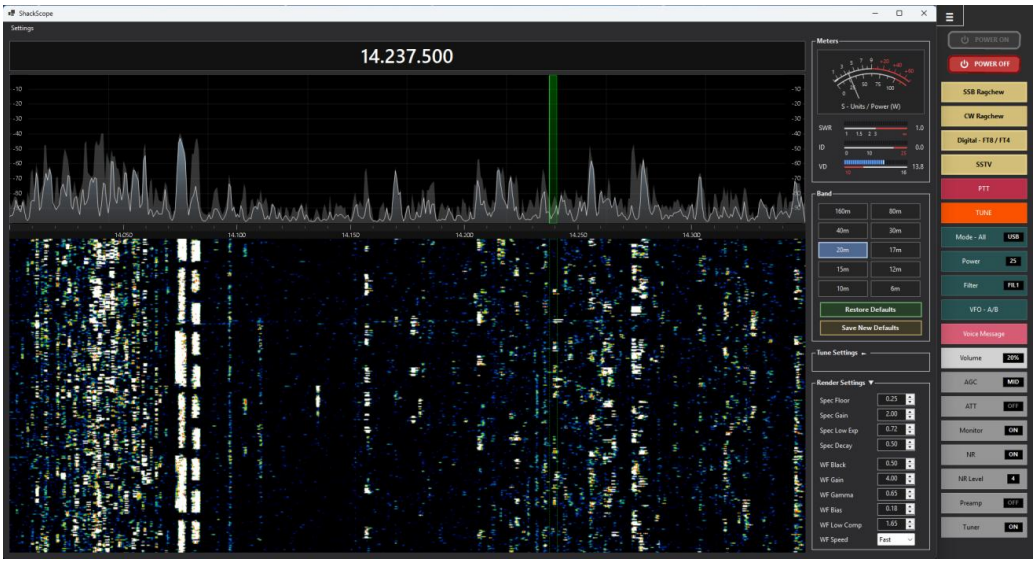
ShackMaestro

User Manual • Version 1.1.4

Windows desktop control software for amateur radio equipment and applications



ShackMaestro at a Glance

ShackMaestro is a Windows desktop application designed for controlling amateur radio equipment and companion software applications. It is fully configurable – no massive dashboard overloaded with every setting and option - just what you want, use and need to build your optimal operating environment with single click access.

Highlights:

- Direct CI-V control — no third-party CAT software required (supports the Icom IC-7300 and IC-705)
- Multi-operator support, so each person who uses the station keeps their own settings and macros
- A configurable macro system — Profile, Action, Toggle, Cycle, and Dropdown macros — for one-click operating
- ShackScope, a companion spectrum/waterfall display with analog and bar-style meters
- Optional integration with OmniRig, Log4OM, WSJT-X, GridTracker, SDR Console, and similar station software

ShackMaestro is intended to serve as the organizer and optimizer for your ham radio station.

Contents

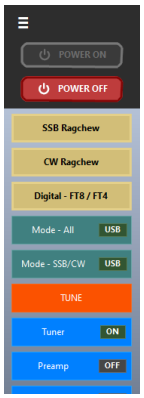
1. Introduction

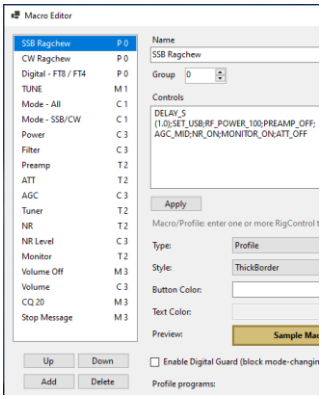
ShackMaestro was developed to provide a simple and efficient way to start, operate, and shut down a complete station operating environment with minimal effort — using a direct connection to your radio rather than depending on a stack of third-party CAT software just to get on the air.

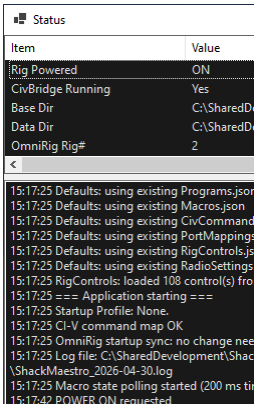
The application is designed for operators who want a larger, easier-to-use control surface than the radio's front panel alone can provide. It emphasizes speed, clarity, and repeatability through a macro-driven interface, and it supports multiple Operators on the same station — so if more than one person uses the shack, each person keeps their own settings, macros, and preferences without stepping on anyone else's.

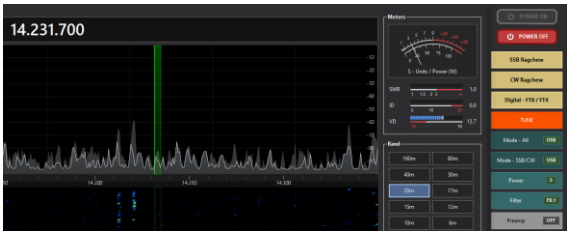
At its core, ShackMaestro provides:

- Direct CI-V control of your radio — no third-party CAT software required to get on the air
- A compact control surface for daily operation
- A configurable macro system for one-click actions
- Real-time visibility into radio state through the Status and Logging Panes
- A full spectrum/waterfall, tuning, meter and band switching app (ShackScope)
- Support for multiple Operators, so each person who uses the station keeps their own configuration









2. Design Overview

ShackMaestro's primary communication path to the radio is Direct CI-V. This is intentional: a new user should be able to install ShackMaestro and ShackScope, configure Radio Settings, and be on the air without installing or configuring any other CAT software first. OmniRig remains fully supported and is the recommended bridge when you want to add other station software, but it is optional rather than required.

In a typical setup, ShackMaestro acts as the control center, while ShackScope and other applications provide visualization and specialized functionality. This separation allows each component to focus on its strengths while maintaining a unified operating experience. When additional CAT-aware programs are added — Log4OM, WSJT-X, GridTracker, SDR Console, and similar tools — OmniRig is the recommended way to bridge them to the radio alongside ShackMaestro.

ShackMaestro also acts as a communication hub for companion applications by exposing virtual COM port pairs (created using virtual COM port software such as com0com) — one pair for ShackScope, and additional pairs for OmniRig or other bridged software as needed. This allows multiple applications to share a single physical radio connection without conflict. See Chapter 3, Installation, for how to set these pairs up.

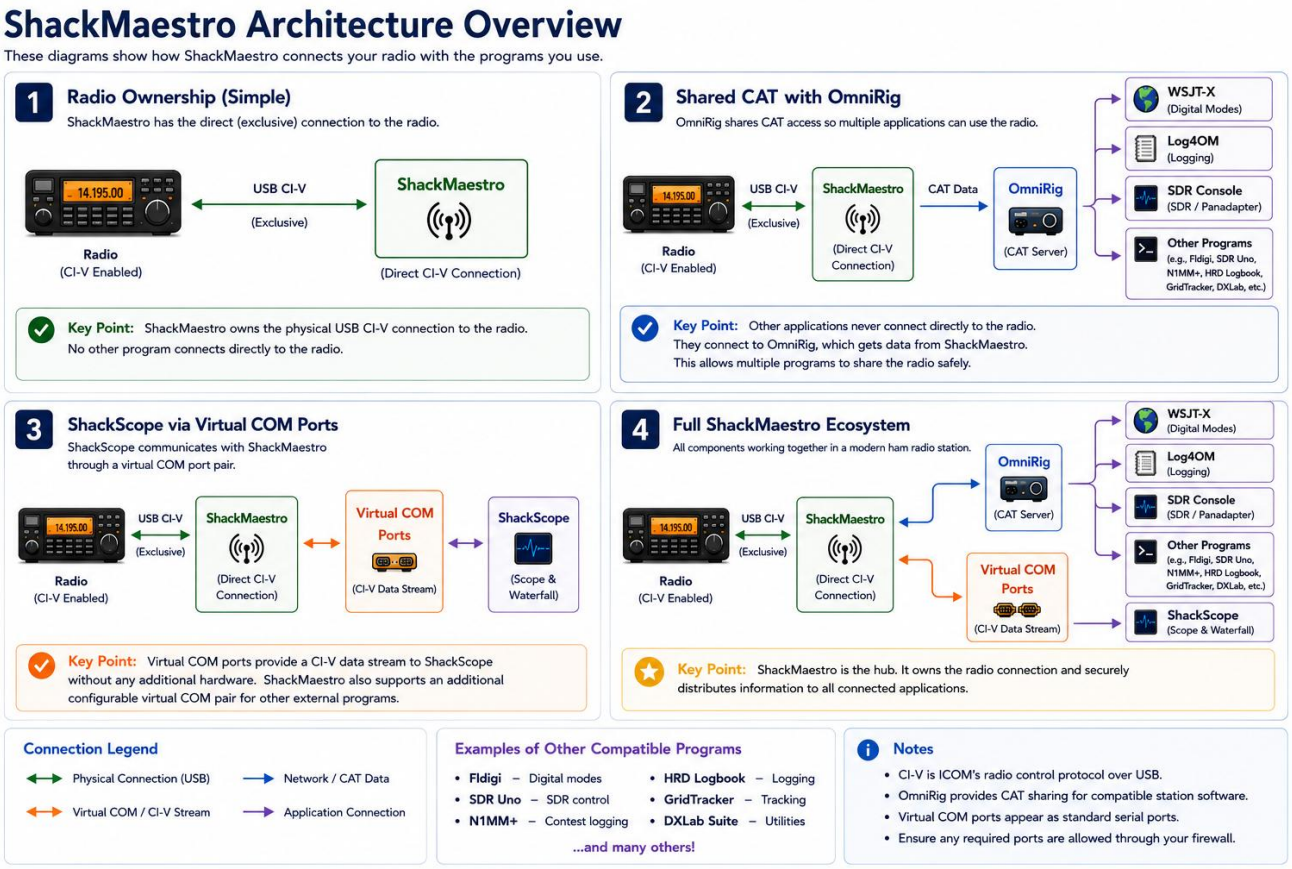


Figure 2-1. ShackMaestro architecture overview.

It implements a hybrid update model:

- Observed feedback from the radio when available
- Intelligent polling for controls that do not report state

This approach provides responsive and accurate UI updates while maintaining system stability.

The main interface consists of:

- Menu System
- Macro Control Area
- Status Pane (real-time radio state)
- Logging Pane (diagnostics and command flow)

The interface is designed for clarity, speed, and minimal operator distraction.

System Requirements

- Windows 10/11
- USB or serial connection to your radio
- .NET runtime (included when using installer)

3. Installation

ShackMaestro (and ShackScope) can be installed using the provided Windows installer, or run directly from a published folder for testing and development purposes. The recommended first-time setup is a seven-step sequence:

1. Install the Icom USB driver
2. Install virtual COM port software and create your COM pairs
3. Install ShackMaestro
4. Configure Radio Settings
5. Power ON
6. Verify ShackScope
7. Add optional software, if you use any

Only the first four are strictly required to get on the air — Direct CI-V control needs no other CAT software.

1. Install the Icom USB Driver

Install the Icom USB driver for your radio (available from Icom's website) so Windows can see it as a COM port.

2. Install Virtual COM Port Software

Install virtual COM port software — com0com is a free option many users choose — and create a virtual COM pair for ShackScope. We recommend using pair 15/14 for ShackScope, and pair 17/16 if you're also installing OmniRig. If you plan to bridge another program, create an additional pair for it. You'll select these pairs later in Radio Settings.

3. Install ShackMaestro

1. Run the ShackMaestro installer executable.
2. Follow the installation prompts.
3. Choose the installation location (default is recommended).
4. Complete installation.

ShackScope is installed alongside ShackMaestro in the same application directory and can be launched automatically by ShackMaestro depending on configuration. After installation:

- Start Menu shortcuts will be created.
- Optional Desktop shortcuts may be created.

The applications will be installed under: C:\Program Files\ShackMaestro\

4. Configure Radio Settings

Open Radio Settings and change if needed:

- COM port and baud rate for your radio
- CI-V address
- The virtual COM pairs you created for ShackScope (and OmniRig, if you use it)

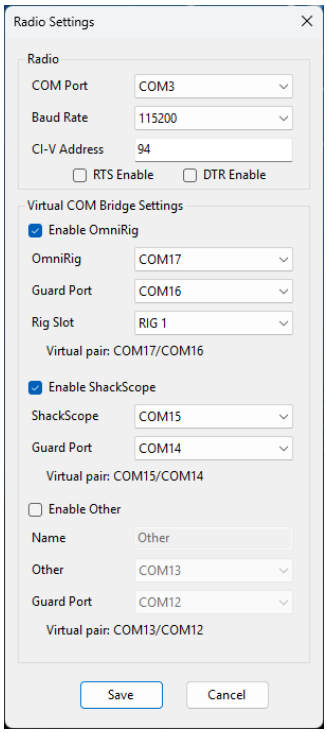


Figure 3-1. Radio Settings dialog.

5. Power ON

Click POWER ON. The radio should respond within a few seconds.

6. Verify ShackScope

Confirm ShackScope opens and shows a live spectrum/waterfall display. This confirms the virtual COM pair for ShackScope is working correctly. At this point you have a fully working station: Direct CI-V control through ShackMaestro, with ShackScope providing spectrum and waterfall visualization.

7. Add Optional Software

Once ShackMaestro and ShackScope are working, you can add other station software as needed. OmniRig is the recommended way to bridge these programs to your radio alongside ShackMaestro:

- OmniRig — recommended CAT bridge for the programs below
- Log4OM, N1MM, or other for logging
- WSJT-X / GridTracker — digital modes
- SDR Console, SDRSharp, Fldigi, MMSTV, and other CAT-aware applications

None of these are required for basic operation — add only what your operating style needs. See Chapter 8, External Program Integration, for setup details on each.

First Launch Behavior

On first launch:

- ShackMaestro and ShackScope create user configuration files in %AppData%\ShackMaestro\
- Default configuration files are copied automatically if needed.
- No manual setup of configuration files is required.

Portable / Development Use

ShackMaestro and ShackScope may also be run directly from a published folder:

1. Copy the published output folder (e.g., win-x64) to your system.
2. Run ShackMaestro.exe.

In this mode:

- Configuration files are stored in the same folder as the executable
- This is useful for testing and portable operation

Uninstalling

To remove ShackMaestro (and ShackScope):

1. Open Windows Settings → Apps.
2. Locate Shack Maestro.
3. Select Uninstall.
4. When prompted, choose whether to keep your configuration files. Keeping them (the default) is recommended if you plan to reinstall or upgrade later.

4. Quick Start

You've completed the one-time setup in Chapter 3. This is what using ShackMaestro day to day looks like:

1. **Start ShackMaestro.** *If your station has more than one Operator configured, select yours first.*
2. Click **POWER ON**. *The radio responds, and ShackScope (and any other configured companion applications) start automatically.*
3. Check ShackScope — you should see a live spectrum/waterfall display tracking your radio's current frequency.
4. Use macro buttons for routine actions such as:
    - Band changes
    - Mode selection
    - Tuning and control adjustments
5. Open the Query Pane and Logging Pane if you want more detail on what the radio is doing.

Figure 4-1. Before power-up.

Figure 4-2. After power-up.

Practical tips:

If the radio does not respond, verify COM port, baud rate, and CI-V settings in Radio Settings.

For regular operating, many users will spend most of their time in the main window with only the controls they need visible. The larger panes are best opened when you want more state feedback or diagnostic visibility.

5. Main Window and Layouts

ShackMaestro supports three primary layouts: **Normal, Tall Band, and Wide Band**. Each layout is optimized for a different operating style or monitor configuration.

Layout selection is available from the **Layout** menu or in General Settings.

Each layout preserves its own size, position, and pane configuration (“persona”), allowing you to return to a consistent working environment across sessions.

- **Normal Layout**
  Default all-purpose view, often used for multi-monitor operation
- **Tall Band Layout**
  Optimized for wide displays or side-by-side use with other applications
- **Wide Band Layout**
  Designed for vertical screen space or stacked window arrangements

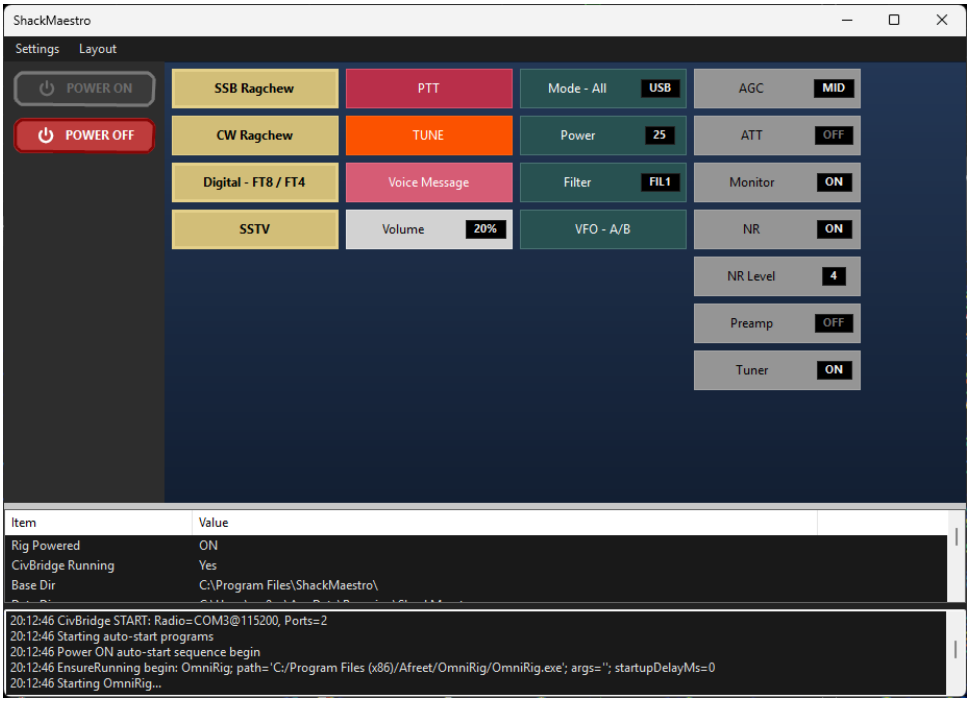


Figure 5-1. Normal layout with the Status Pane shown beneath the main control surface.

Normal layout is the default all-purpose view and works well with multiple monitors or window panels.

Figure 5-2. Wide Band layout.

Wide Band layout emphasizes vertical screen usage and is useful when screen height is available but width is limited.

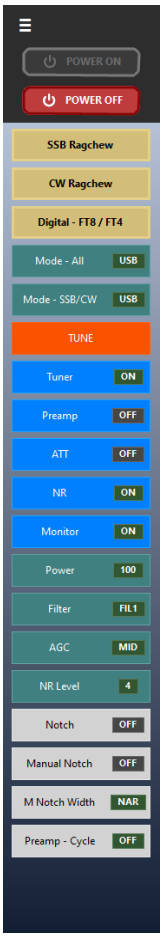


Figure 5-3. Tall Band layout.

Tall Band layout spreads applications across the screen and works well with wide monitors or side-by-side operating tools.

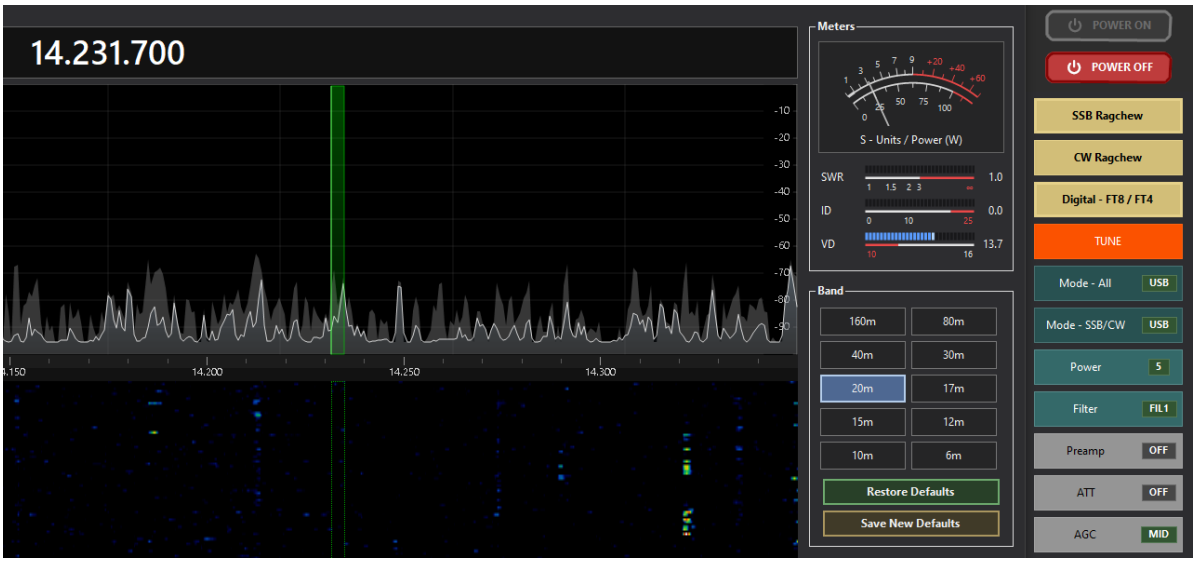


Figure 5-4. Clip of Tall Band layout next to ShackScope.

6. Settings

The Settings menu is intentionally compact so that frequently used commands remain easily accessible.

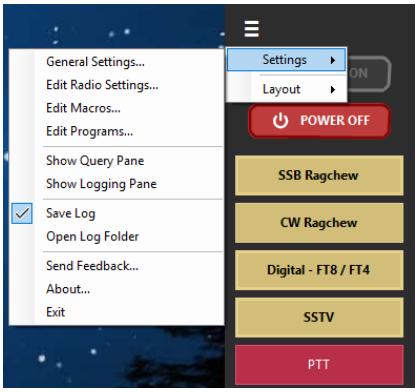


Figure 6-1. Settings menu commands.

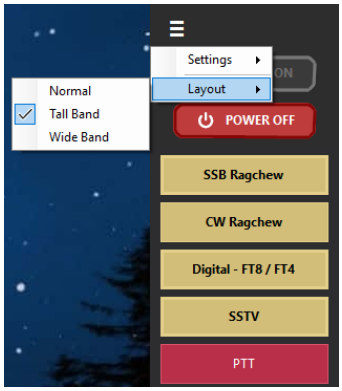


Figure 6-2. Layout menu commands.

- **General Settings**
  Controls layout behavior, pane visibility, UI preferences based on Layout Mode, and Operator management (see Operators below)
- **Radio Settings**
  Defines serial/CAT communication parameters used to connect to the radio and companion applications
- **Show Logging/Query Pane**
  Toggles visibility of diagnostic panes

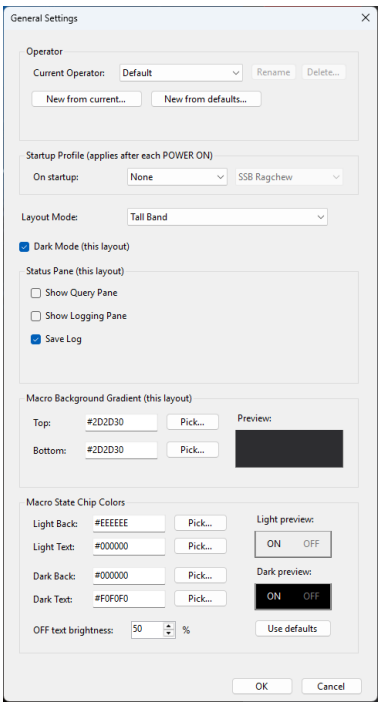


Figure 6-3. General Settings dialog.

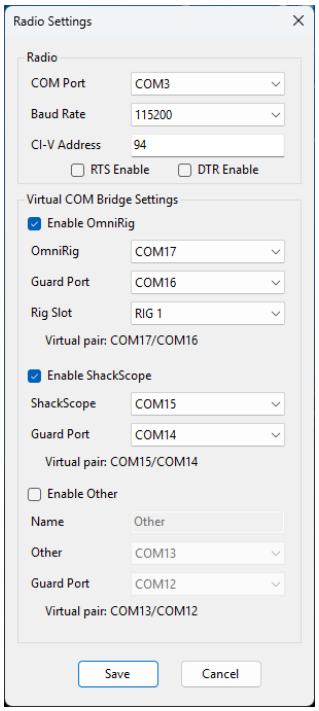


Figure 6-4. Radio Settings dialog.

General Settings Fields
Most of what's in this dialog is saved per Layout Mode, not globally — the Layout Mode dropdown near the top selects which layout (Normal, Tall Band, Wide Band) you're currently viewing and editing settings for. Everything below it applies to that layout specifically, until you switch it.

- **Startup Profile —** Choose a Profile macro to apply automatically every time you click POWER ON, so you don't have to select it by hand each session. Leave it None to start with whatever state the radio was already in.
- **Layout Mode —** Selects which layout's settings this dialog is showing and editing (see above).
- **Dark Mode (this layout) —** Toggles dark/light theme, saved independently for this layout.
- **Status Pane (this layout) —** Show Query Pane and Show Logging Pane control visibility, as covered in Chapter 9. Save Log controls whether this layout's session activity is written to the log file on disk (Chapter 12) — turning it off still shows activity live in the Logging Pane, it just won't be saved afterward.
- **Macro Background Gradient (this layout) —** Sets the background color gradient (top and bottom) behind the macro buttons, for this layout.
- **Macro State Chip Colors —** Customizes the colors used for macro state indicators (the ON/OFF-style chips on Toggle, Cycle, Dropdown, and Action macros — Chapter 7), with separate colors for light and dark mode, an OFF-text brightness adjustment, and a one-click reset to defaults.

Operators
If more than one person uses the station, General Settings includes an Operator group where each person can keep their own settings, macros, and rig controls, completely separate from anyone else's.

- **Current Operator —** Shows who's currently active.
- **New from current —** Creates a new Operator by copying everything from whoever is currently active — their General Settings, Radio Settings, Programs, Macros, and RigControls. Use this when a new operator wants to start from an existing, working setup and tweak it.
- **New from defaults —** Creates a new Operator starting from ShackMaestro's factory defaults instead, ignoring any customization the current Operator has made. Use this for a genuinely clean slate.
- **Rename —** Renames an Operator. Not available for Default, and not available for whichever Operator is currently active — switch to a different Operator and restart first.
- **Delete —** Removes an Operator and its settings entirely. Same restrictions as Rename: not Default, and not the one currently active.

Switching Current Operator takes effect the next time ShackMaestro starts — the dialog reminds you a restart is required.

Each Operator's data is fully isolated, including its own Logs folder. Default lives directly under:
%AppData%\ShackMaestro
Named Operators each get their own subfolder under:
%AppData%\ShackMaestro\Operators\[Operator Name]

Operators aren't only for multiple people. A single operator can use the same mechanism to keep separate configurations for different situations — a different radio, a portable setup, or a macro set built for a specific contest or operating style — and switch between them the same way, via Current Operator and a restart.

7. Macro System

The macro system is the heart of ShackMaestro's one-click workflow. A macro is a named button that runs one or more radio commands — anything from flipping a single toggle to reconfiguring the whole radio for a mode change. Everything on the main window's button grid, other than POWER ON/OFF, is a macro.

This chapter explains the six macro types, walks through building one of each, and covers the Commands syntax and a few practices that make a macro set easier to live with long-term.

The Macro Editor

Open it from Settings → Edit Macros. The editor has three parts:

- A list of existing macros on the left, with Add, Delete, Up, and Down to reorder buttons within their column.
- Macro properties in the middle: Name, Type, Group, Style, Text Color, and a live Preview of the button.
- A Controls list on the right, filtered to whatever is valid for the selected Type, plus a Commands box where you build the actual sequence.

Two fields only appear for Profile macros: Enable Digital Guard and Profile programs — covered under Profile below.

Macro Types

ShackMaestro has six macro types. The right one depends on what you're controlling — each type exists because it matches a different kind of radio state.

- **Direct** — runs a single control command with no state to track. Use it for one-shot actions: tuning the antenna, swapping VFOs, sending a canned sequence. *Example:* TUNE *macro executes the* TUNER_TUNE *control.*

- **Profile** — applies a complete operating configuration in one click: e.g. including mode, power, filter, AGC, noise reduction, and which companion programs should be running. Use it for the operating modes you switch between all the time. *Example:* SSB Ragchew *sets USB, 100 W, preamp off, AGC mid, NR on, monitor on, and ATT off.*

- **Toggle** — flips a control between exactly two states (on/off). ShackMaestro queries the radio to confirm which state it's actually in and shows that on the button's chip — it doesn't just assume. Use it for anything genuinely binary. *Example:* ATT *macro executes the* ATT_TOGGLE *control to toggle the ATT function ON/OFF.*

- **Cycle** — steps through two or more states in a fixed order each time you click. Use it when there's a short, ordered list of values and clicking through them is the natural interaction. *Example:* Filter *cycles through the radio's three filter widths.*

- **Dropdown** — same underlying idea as Cycle (a control with several states), but presented as a dropdown you pick from directly instead of clicking through in order. Use it when there are enough options that clicking through one at a time is tedious, or the current selection matters more than the sequence. *Example:* Mode - All *presents every mode in a dropdown (with current selected state indicated) instead of cycling.*

- **Action** — looks similar to Toggle but is for controls where you want an activity indicator instead of a persistent ON/OFF state — a blinking lamp on the button while active, rather than a chip that just says ON. Use it for things like PTT, where you want indication that it is in an active state. *Example:* PTT *macro executes* PTT_TOGGLE *control to toggle the PTT state and blinks while PTT is active*.

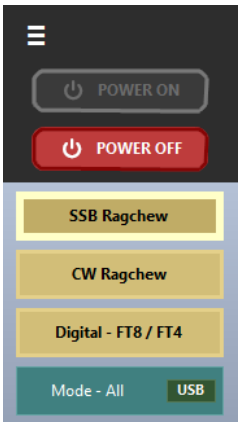


Figure 7-1. Example of a Profile macro when active (SSB Ragchew).

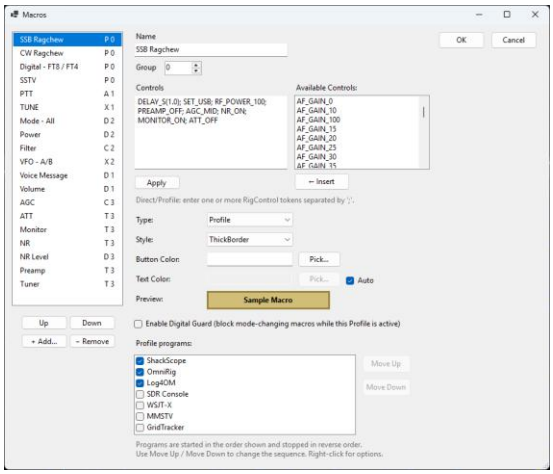


Figure 7-2. Macro Editor with Profile macro options.

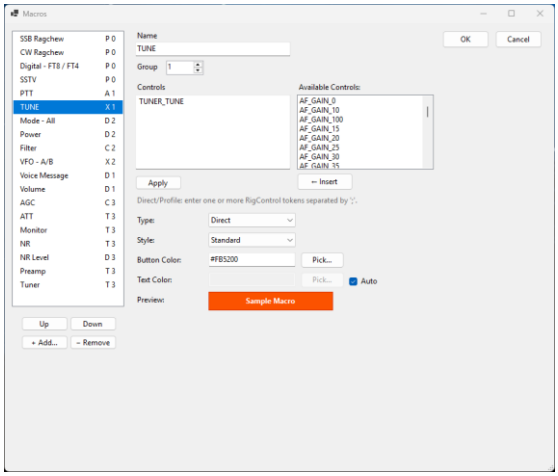


Figure 7-3. Direct macro options.

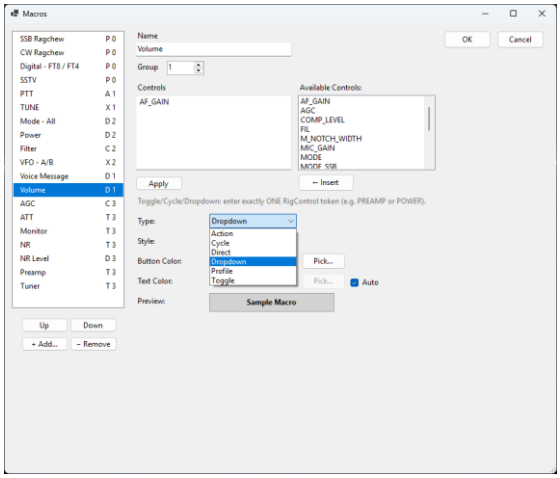


Figure 7-4. Macro types.

Creating a Direct Macro

1. In the Macro Editor, select Add and set Type to Direct.
2. Select a control from the list and click ← Insert to add it to Commands (a semicolon separator is added automatically). Repeat for each control you want, in order.
3. Give it a Name, choose a Group/column, and optionally set Style and Text Color.
4. Select Apply, then test it from the main window.

Creating a Profile Macro

1. In the Macro Editor, select Add and set Type to Profile.
2. Build the Controls sequence using the same method to add controls as with a Direct macro. Example: mode, power, filter, AGC, and any other settings this operating style needs. Add DELAY_S(seconds) between steps if the radio needs time to settle.

3. Under Profile programs, select the companion applications (ShackScope, WSJT-X, Log4OM, etc.) that should be running when this Profile is active. Order matters: programs start in the order listed (use Move Up/Move Down to change it) and stop in reverse order.
4. Right-click a selected program for one more option: **Restart when this profile starts**. This stops and freshly restarts that program at its position in the sequence, rather than leaving an already-running copy in place — useful when a program needs to bind to another one that just started. For example, some Log4OM/WSJT-X setups need Log4OM listed *after* WSJT-X with restart enabled, so Log4OM re-establishes its UDP connection to WSJT-X cleanly instead of hitting the timing issue that happens when both start together.
5. If this Profile represents a digital mode, consider checking Enable Digital Guard. This blocks other mode-changing macros from being triggered while the Profile is active — useful for preventing an accidental click from breaking a digital mode setup mid-session.
6. Name it, set Style/Color, Apply, and test.

Creating a Toggle Macro

1. Select Add and set Type to Toggle.
2. Choose a control from the list — only toggle-capable controls (like ATT_TOGGLE) appear here.
3. Toggle macros take exactly one control and no DELAY — the Commands field is populated for you when you pick the control.
4. Name it, set Style/Color, Apply, and test. Confirm the button's chip correctly reflects ON/OFF after a click.

Creating a Cycle Macro

1. Select Add and set Type to Cycle.
2. Choose a control from the cycle-capable list (like FIL for filter width).
3. As with Toggle, Cycle takes exactly one control — the sequence of steps is defined by the control itself, not typed in by hand.
4. Name it, set Style/Color, Apply, and click through it a few times to confirm the order makes sense for how you'll use it.

Creating a Dropdown Macro

1. Select Add and set Type to Dropdown.
2. Choose a control from the same cycle-capable list used for Cycle macros (like MODE) — Dropdown and Cycle share the same underlying controls, just presented differently.
3. Name it, set Style/Color, Apply, and confirm the dropdown shows the current state and lets you jump directly to any option.

Creating an Action Macro

1. Select Add and set Type to Action.
2. Choose a control from the toggle-capable list, same as you would for a Toggle macro (like PTT_TOGGLE).
3. The difference from Toggle is entirely visual: Action shows a blinking activity lamp while the state is active, instead of a static ON/OFF chip. Use it when "currently happening" is the more useful signal than "left on."
4. Name it, set Style/Color, Apply, and test by triggering the underlying condition (e.g., keying PTT) to confirm the lamp behaves as expected.

Understanding Macro Commands

A macro's Commands field is a semicolon-separated list of tokens, run in order:

DELAY_S(1.0);SET_USB;RF_POWER_100;PREAMP_OFF;AGC_MID;NR_ON;MONITOR_ON;ATT_OFF

This is the actual Commands string behind the SSB Ragchew Profile macro: wait one second, then set USB mode, 100 W power, preamp off, mid AGC, noise reduction on, monitor on, and attenuator off — in that order.

Delays use one of three equivalent forms:

- DELAY_S(1.0) — seconds, as a decimal
- DELAY_MS(1000) — milliseconds, as an integer
- DELAY(1000) or DELAY(1.0) — milliseconds by default, but treated as seconds if you give it a decimal

DELAY_S is the clearest to read and the one you'll see used by default. Use a delay whenever a command needs the radio a moment to settle before the next one is sent — the Digital - FT8/FT4 Profile, for example, waits 5 seconds before changing power, since some radios need a moment after a mode change before other settings take effect reliably.

Order matters: commands run top to bottom exactly as listed, so put anything time-sensitive (like a settling delay) where it's actually needed rather than at the start out of habit.

How ShackMaestro Confirms Macro State

Toggle, Cycle, Dropdown, and Action macros aren't just fire-and-forget — each one is backed by a control definition that also knows how to query the radio and read back its actual state. After a macro runs, ShackMaestro can confirm the change actually took effect rather than just assuming it worked, and updates the button's indicator accordingly.

ShackMaestro also polls certain controls in the background, so if the radio's state changes some other way — the front panel, another program sharing the connection — the macro button's indicator catches up rather than showing stale information.

This is why Toggle and Cycle macros are restricted to controls that support it, and why they take exactly one control each: the verification behavior depends on knowing precisely which state to query.

Good Macro Design

- Keep Profiles focused on one operating style each. A Profile that tries to cover every possibility stops being a fast one-click action.
- Use Enable Digital Guard on digital-mode Profiles if accidental mode changes mid-session are a real risk for how you operate.
- Order commands the way the radio actually needs them — mode before mode-dependent settings, with a DELAY_S wherever the radio needs to catch up.
- Use Toggle for anything genuinely binary, Cycle for a short ordered list you'll click through, and Dropdown once that list gets long enough that clicking through it repeatedly is annoying.
- Reserve Action for things you want to see happening (PTT, a blinking process) rather than things you want to remember are on.
- Use Style and Text Color consistently — for example, color-coding by function (mode, power, digital) — so a busy button grid stays scannable at a glance.

Editing an Existing Macro

1. Open Edit Macros from the Settings menu.
2. Select the macro to modify.
3. Confirm the Type is still correct — changing Type changes which controls are valid.
4. Update the Commands and settings as needed.
5. Select Apply, then test directly from the main window before relying on it during operation.

Best practice: always test a macro after editing it, before you're depending on it mid-contact.

Removing and Restoring Macros
Deleting a macro from the editor gives you a choice, rather than deleting outright:

- Remove from panel and save for later — the macro disappears from your button grid but stays defined. It doesn't clutter the interface, but nothing about it is lost.
- Delete permanently — the macro definition is gone for good.

To bring a saved macro back, select Add and choose Restore Saved instead of New Macro. This is a good way to keep macros you use occasionally — or ones that need more setup than you want visible by default — without deleting them outright.

8. External Program Integration

ShackMaestro is designed to operate alongside other radio applications. Typical integrations include:

- ShackScope
- SDR Console
- WSJT-X
- Fldigi
- Log4OM
- OmniRig

This is where the magic happens – you turn your station startup/shutdown and operating profile changes into a single mouse click.

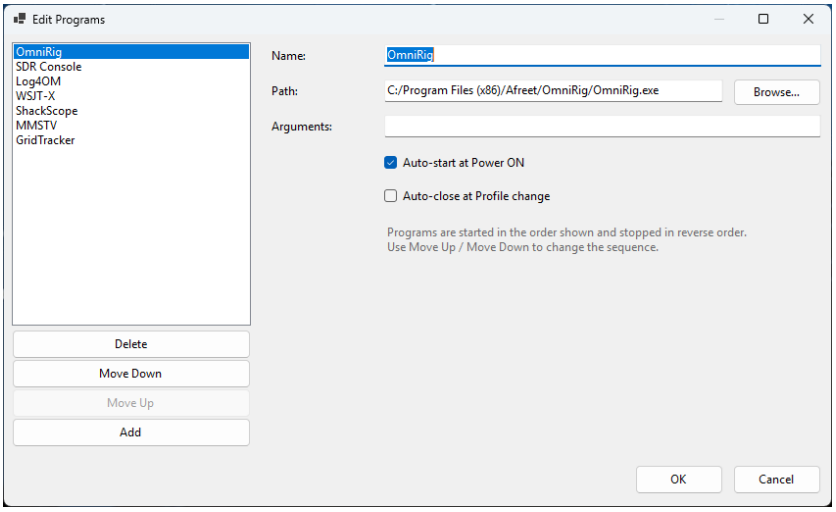


Figure 8-1. Edit Programs dialog.

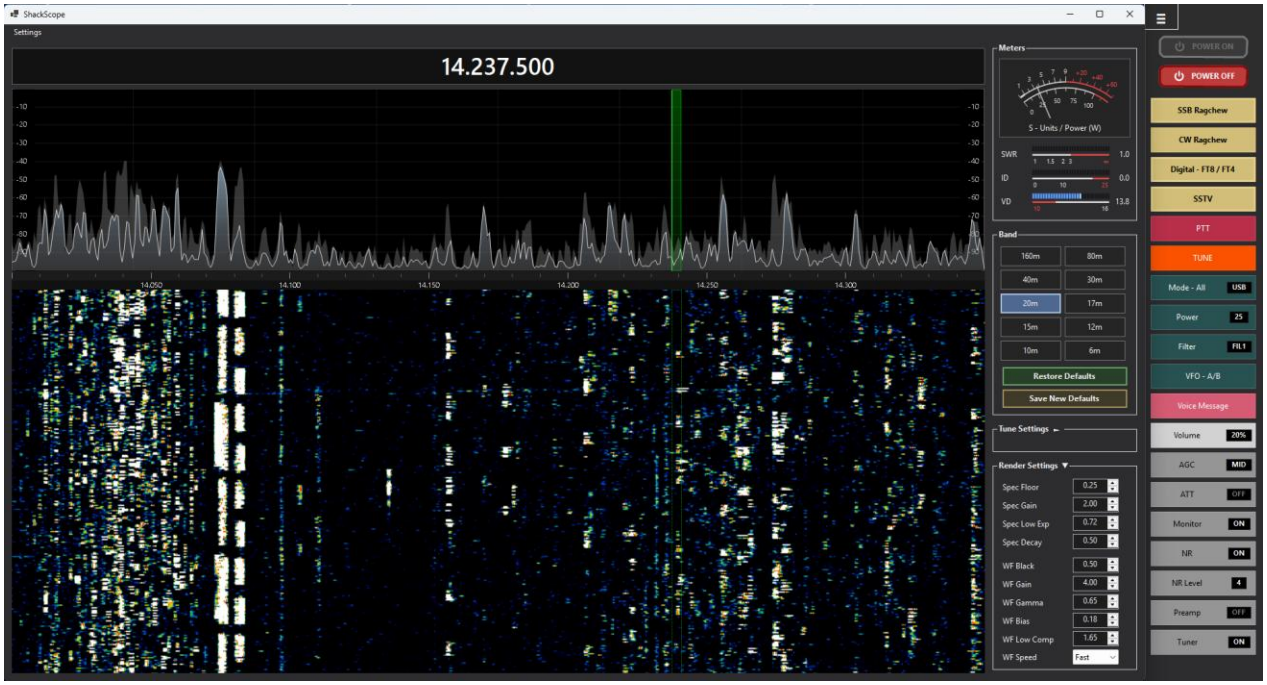


Figure 8-2. Tall Band alongside ShackScope.

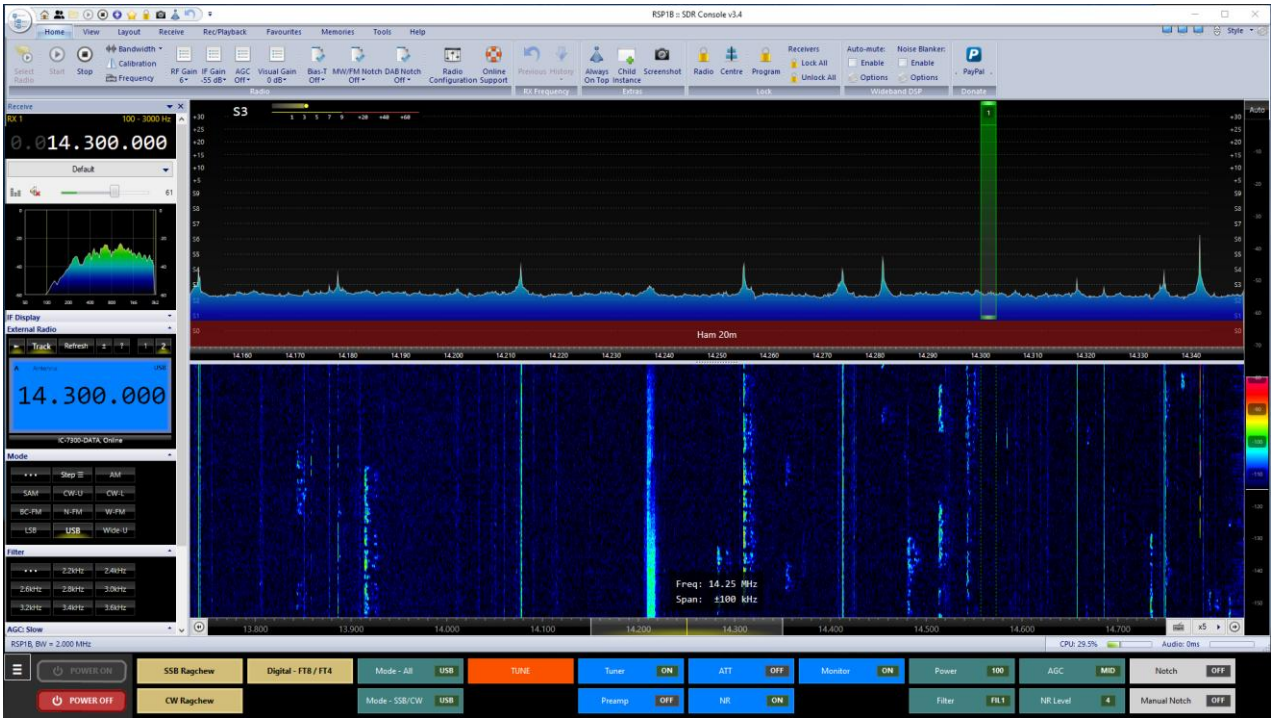


Figure 8-3. Wide Band layout with SDR Console displayed alongside ShackMaestro.

Two Ways Programs Get Started

ShackMaestro can start companion programs in two independent ways, and it helps to keep them straight:

- **Auto-start at Power ON —** Global, one-time launch when ShackMaestro powers on — configured per program in Edit Programs.
- **Profile programs —** Started or stopped as part of a specific Profile macro, so different operating styles bring up different programs. This is covered in Chapter 7, Macro System.

These work together, not against each other: a program can be started at Power ON and also referenced in Profile programs (which then just keeps it running, or restarts it, depending on how the Profile is set up).

Edit Programs

Open it from Settings → Edit Programs. For each program you define:

- Name and Path (Browse... to locate the executable), plus optional Arguments.
- Auto-start at Power ON — launches this program automatically every time ShackMaestro powers on.
- Auto-close at Profile change — closes this program automatically when a Profile switch no longer includes it.

As with Profile programs, order matters here too: programs start in the order shown and stop in reverse order. Use Move Up/Move Down to change the sequence.

OmniRig Is a Special Case

Checking Auto-start at Power ON for OmniRig isn't enough by itself. ShackMaestro only auto-starts OmniRig if Enable OmniRig Bridge is also turned on in Radio Settings — consistent with Direct CI-V being the primary connection and OmniRig being strictly optional. If OmniRig doesn't come up automatically, check Radio Settings first, not just the Auto-start checkbox.

OmniRig is typically used as the CAT interface for third-party applications, while ShackMaestro maintains direct control of the radio. For simplest setup, configure external programs to use OmniRig, rather than direct COM port selection, for CAT control.

A common operating setup is to place ShackMaestro alongside ShackScope or another SDR program, combining control with visual spectrum/waterfall monitoring.

After changing COM ports or radio configuration, verify OmniRig and companion software reconnect — they read radio settings independently and don't always pick up a change automatically.

9. Query Pane and Logging Pane

ShackMaestro includes two optional panes that show what's happening under the hood: the Query Pane and the Logging Pane. Neither is required for normal operation, but both are the fastest way to see what the software is actually doing, especially during setup or when something isn't behaving as expected.

Query Pane

Shows a snapshot of current state, refreshed as things change:

- **Live radio state —** Rig Powered, CivBridge Running, Rig Mode
- **Paths —** Base Dir and Data Dir (where ShackMaestro is installed and where this Operator's config lives — Chapter 12)
- **OmniRig —** Whether OmniRig is bridged and which rig number it's using
- **Configuration file health —** Programs.json, Macros.json, RigControls.json, and related files, each shown as OK with a last-modified time, or MISSING.

That last item is worth knowing about specifically: if something references a config file that shows MISSING here, that's usually the actual problem — see Chapter 13, Troubleshooting.

Logging Pane

A running, timestamped feed of operational messages — commands sent, macros triggered, programs starting and stopping, polling activity. This is the same information written to the log files described in Chapter 12, shown live as it happens.

Visibility and Docking Are Per-Layout

Whether each pane is shown, and whether it's docked in the main window or detached into its own floating window, is remembered separately for each Layout Mode (Normal, Tall Band, Wide Band — Chapter 5). Toggle visibility from Settings → Show Query Pane / Show Logging Pane (Chapter 6); a detached pane remembers its own size and position per layout.

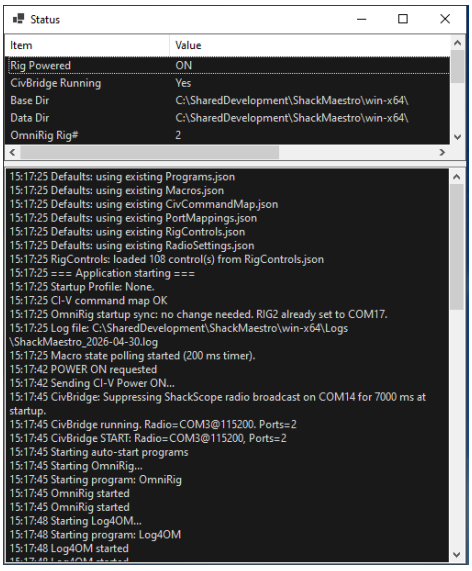


Figure 9-1. Detached Status Pane with Query and Logging.

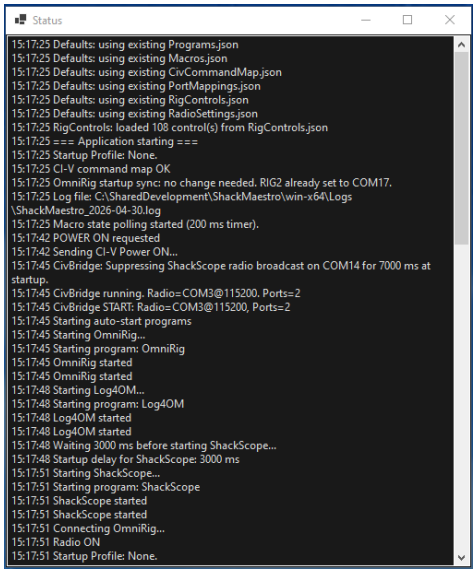


Figure 9-2. Detached Status Pane with only Logging.

When troubleshooting, the Query Pane is often the fastest way to confirm that commands, polling activity, and external integrations are behaving as expected.

10. ShackScope

ShackScope is a companion application designed to provide real-time spectrum, waterfall, and meter visualization for supported radios when your station doesn't include a separate SDR and scope software. ShackScope extends ShackMaestro by providing a visual operating surface that complements the macro-driven control interface.

While ShackScope can operate as a standalone application, it is most commonly used alongside ShackMaestro as part of an integrated operating environment.

What ShackScope provides

- Real-time spectrum/waterfall display
- Analog and bar-style meters
- Band selection with full settings memory
- Tune step and edge adjustments
- Visual feedback aligned with radio operation

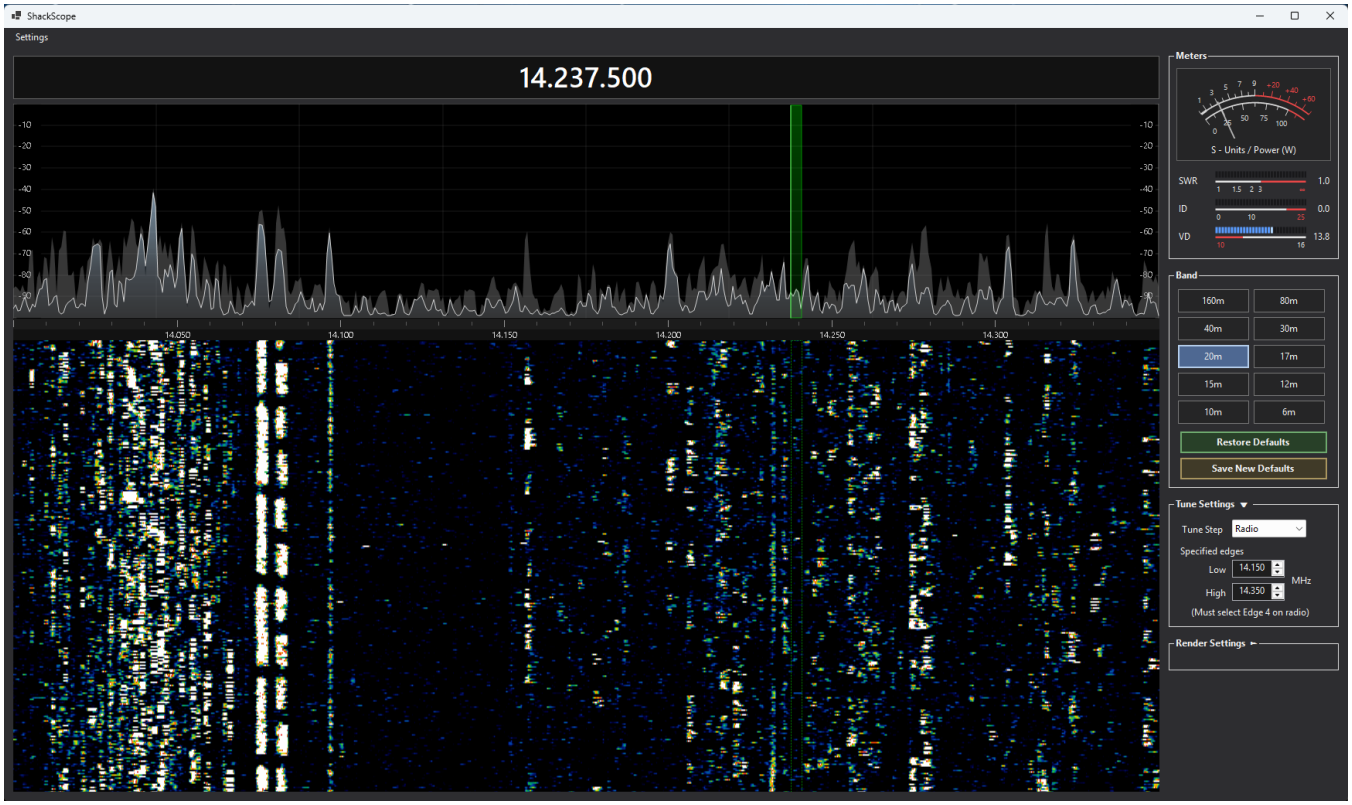


Figure 10-1. Full screen view of ShackScope.

How ShackScope connects

ShackScope does not communicate directly with the radio. Instead, it connects through ShackMaestro using a virtual COM port bridge.

Radio ⇄ ShackMaestro ⇄ Virtual COM Pair ⇄ ShackScope

This design ensures:

- A single, controlled CI-V communication path
- Reduced conflicts between applications
- Consistent and reliable operation

Virtual COM configuration

A virtual COM port pair is used to connect ShackScope to ShackMaestro.

Typical configuration:

- ShackMaestro (CivBridge side): COM14
- ShackScope: COM15

(These values may be adjusted in Radio Settings.)

Startup behavior

When properly configured:

1. ShackMaestro starts
2. POWER ON is pressed
3. ShackScope is launched automatically (if configured)
4. ShackScope connects through the virtual COM port
5. Spectrum/waterfall and meters begin updating

Important notes

- When running with ShackMaestro, ShackScope depends on ShackMaestro for CI-V access
- ShackScope may take a few seconds to begin streaming after startup
- High CI-V traffic (e.g., rapid macro use) may temporarily affect update timing - this is expected and does not indicate a failure
- ShackScope can operate as a stand-alone scope utility with your radio – you just need to configure ShackScopeSettings to connect directly with the radio's COM port

Settings Menu

ShackScope's Settings menu has two entries:

- **General Settings... —** Opens the same style of preferences dialog as ShackMaestro's General Settings.
- **Edit Meter Settings... —** Opens the Meter Settings dialog, covered below.

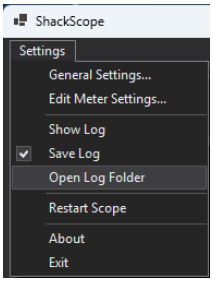


Figure 10-2. ShackScope Settings menu.

General Settings Dialog

Covers ShackScope's own layout and display preferences, including the Render Settings explained in Appendix A. As noted in Chapter 12, these settings live entirely separately from ShackMaestro's, in ShackScope's own configuration files — seeded from your Radio Settings the first time ShackScope runs, then owned by ShackScope from that point on.

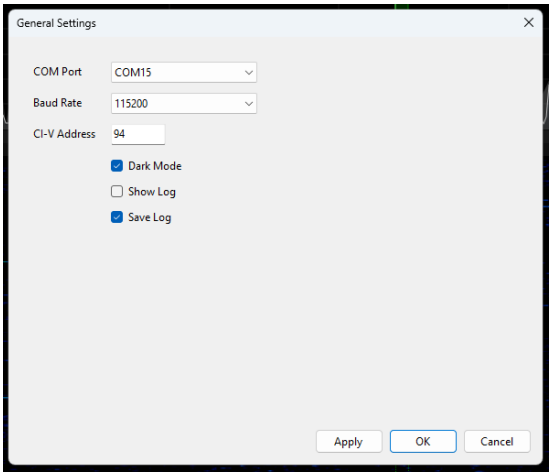


Figure 10-3. ShackScope General Settings dialog.

Band and Tune Settings

ShackScope remembers a separate preset for each ham band — frequency edges, tune step, mode, and filter — so switching bands restores how you last had that band configured. A band can also carry its own Render Settings override, if a particular band benefits from different Spectrum Floor or Waterfall settings than your general defaults (Appendix A).

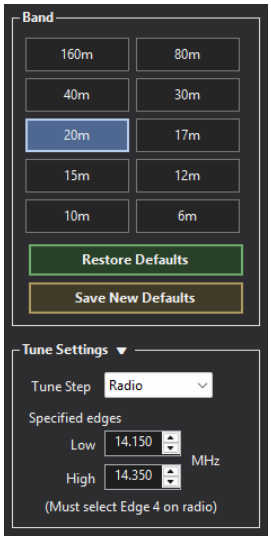


Figure 10-4. ShackScope Band and Tune settings.

Meter Settings

Configures the analog and bar-style meters shown alongside the spectrum/waterfall display. A few limits worth knowing:

- Up to three bar-type meters can be shown at once.
- Each meter's numeric value display is optional — you can show just the bar/needle, or the bar/needle plus its number.
- Only one transmit meter can be the analog (needle) style — any additional transmit meters are bar-type.

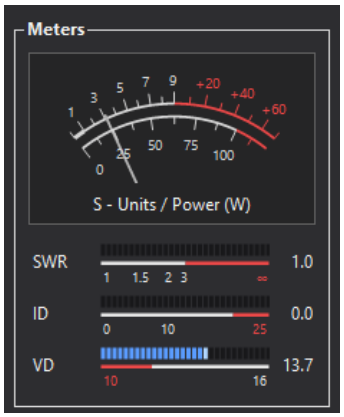


Figure 10-5. Meter panel in the main ShackScope window.

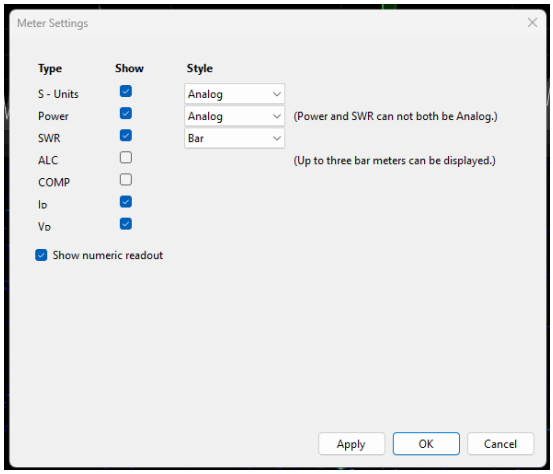


Figure 10-6. ShackScope Meter Settings dialog.

Render Settings

Controls the visual appearance of the spectrum and waterfall — explained in full in Appendix A, ShackScope Render Settings Explained.

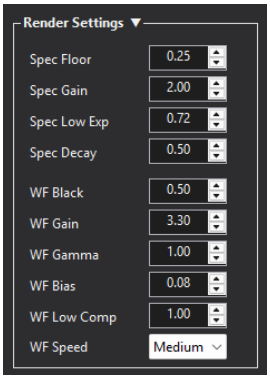


Figure 10-7. Render Settings panel (Appendix A).

How Waterfall Speed Actually Works

There are five levels — Min, Slow, Medium, Fast, Max — and the mechanism is worth understanding since it isn't quite a simple slider:

- **Slow is the true baseline —** Genuinely 1:1 with incoming data. Every spectrum update from the radio produces exactly one waterfall row — no duplication, no skipping.
- **Min is slower than real-time, on purpose —** Deliberately slower than 1:1. Occasionally an incoming update doesn't get its own row, so the waterfall falls slightly behind in exchange for compressing more time into the same number of visible rows — the most history-dense setting.
- **Medium, Fast, and Max all exceed 1:1 —** Each moves faster by repeating — duplicating a real update across more than one waterfall row (roughly 1.4, 2, and 2.6 rows per update respectively) — rather than inventing new data in between. It's repetition, not interpolation.

This setting only changes how fast the waterfall history scrolls. The live spectrum trace above it redraws on its own fixed schedule regardless of Waterfall Speed, so a faster setting doesn't make the spectrum itself more responsive — only the waterfall.

11. Recommended Daily Operating Workflow

Once setup (Chapter 3) is done, a typical session looks like this:

1. Launch ShackMaestro — or leave it always open. If you use multiple Operators (Chapter 6), the one selected in General Settings is who's active until you switch and restart.
2. Click POWER ON. Confirm the radio responds and that any programs set to Auto-start (Chapter 8) come up.
3. Pick a Profile macro for how you're operating today — SSB, CW, a digital mode, whatever matches (Chapter 7). This sets mode, power, filter, and starts the right companion programs in one click.
4. Operate using the macro buttons — Toggle, Cycle, Dropdown, and Action macros for the moment-to-moment adjustments (attenuator, filter, mode, PTT).
5. Tune frequency and change bands using ShackScope or your radio's controls.
6. Switching operating styles mid-session? Click a different Profile macro rather than powering down — that's exactly what Profiles are for.
7. If something looks off — a program didn't start, a control isn't responding — check the Query Pane (Chapter 9) before digging further. Config file health and current radio state are both right there.
8. Use Settings or Edit Macros only when actually changing configuration, not as part of routine operating.
9. Click POWER OFF to turn off the radio and close managed programs (per each program's Auto-close setting, Chapter 8).

Best Practice
Maintain a backup of your configuration files (in *%AppData%\ShackMaestro* — or the relevant Operator subfolder, Chapter 12) before making significant changes.

12. Configuration Files and Advanced Behavior

ShackMaestro stores settings, macros, and program definitions as JSON files. Nothing here is required reading for day-to-day use — the UI covers everything most operators need — but it's useful if you want to understand what a setting actually changes, back things up, or troubleshoot.

Configuration File Locations

When installed, user-editable configuration files live in:

%AppData%\ShackMaestro

If you've created additional Operators (Chapter 6), each one has its own isolated copy under:

%AppData%\ShackMaestro\Operators\[Operator Name]

When running in portable mode, configuration files are stored alongside the executable instead.

ShackScope keeps its own settings entirely separate from ShackMaestro's, in its own folder:

%AppData%\ShackScope

ShackMaestro Configuration Files

Each Operator's folder contains the same set of files:

- **GeneralSettings.json —** Layout Mode, pane visibility, and other UI preferences set in General Settings.
- **RadioSettings.json —** COM port, baud rate, CI-V address, and virtual COM pairs set in Radio Settings. Unlike the others below, this file has no shipped template — it's created with built-in defaults the first time ShackMaestro runs.
- **Programs.json —** The external programs defined in Edit Programs (Chapter 8): paths, arguments, Auto-start, Auto-close.
- **Macros.json —** Every macro and Profile button (Chapter 7): type, Commands, Profile programs, styling.
- **RigControls.json —** The control layer macros are actually built from — what each control key (like ATT_TOGGLE or MODE) does, how to query it, and how to verify it. This is the layer described throughout Chapter 7; you work with these control names, not raw CI-V.
- **CivCommandMap.json —** The raw CI-V command mapping underneath RigControls.json. This is an internal layer — you shouldn't need to touch it, and incorrect edits here can affect CAT communication broadly rather than one control at a time.

GeneralSettings, Programs, Macros, and RigControls each ship with a default template that's copied in automatically the first time an Operator needs one (including when you create an Operator with New from defaults — see Chapter 6).

ShackScope Configuration Files

ShackScope has its own configuration files (see location above):

- **ShackScopeSettings.json —** Everything in the Render Settings and Meters panels, plus window layout and scope mode. Seeded from your Radio Settings the first time ShackScope runs; after that, ShackScope owns it. ShackMaestro keeps a saved copy of this file for each Operator and swaps it into place right before launching ShackScope, then copies it back out when ShackScope closes or the Operator is about to switch — so each Operator\u2019s ShackScope configuration follows them, without ShackScope itself needing to know Operators exist.
- **BandSettings.json —** Per-band tuning presets — frequency range, tune step, mode, filter, and even per-band Render Settings overrides, saved separately for each ham band. Tracked per Operator the same way as ShackScopeSettings.json, above.

Advanced Editing (Optional)

Advanced users may edit these files directly. A few things worth knowing that the UI doesn't fully expose:

- **DELAY_S / DELAY_MS —** In a macro's Commands (Macros.json), used to give the radio time to settle between steps — see Chapter 7.

- **StartupDelayMs —** On a program entry (Programs.json), delays that program's launch — useful for integrations like ShackScope that need the radio link ready first. Not exposed in the Edit Programs dialog — see Chapter 8.
- **RestartPrograms —** On a Profile's program list (Macros.json), forces a program to stop and restart fresh at its position in the sequence — the mechanism behind the right-click option covered in Chapter 7.
- **pollPriority —** On a control definition (RigControls.json), how often that control is checked during idle background polling — higher means checked more often.
- **verifyDelayMs —** On a control definition (RigControls.json), overrides how long ShackMaestro waits before re-querying the radio to confirm a macro's change took effect (see "How ShackMaestro Confirms Macro State" in Chapter 7).

Caution

- Always back up a configuration file before editing it directly.
- Invalid JSON, or an invalid value, can affect application behavior in ways that aren't obvious — test after any manual edit.

Logging and Diagnostics

ShackMaestro maintains log files to help diagnose issues. Each Operator has its own log folder:
%AppData%\ShackMaestro\Logs
(or the corresponding Operators\[Operator Name]\Logs\ folder for a named Operator.)

The Logging Pane (Chapter 9) shows this same activity in real time inside the app; the log files are the persistent record, useful for reviewing what happened after the fact or sharing with support.

ShackScope keeps its own separate log, optionally enabled from its Settings menu:
%AppData%\ShackScope\Logs

Unlike its settings, ShackScope's logs are deliberately left alone by the Operator-tracking mechanism above — there's one shared log location regardless of which Operator is active. Logs are runtime history, not configuration, so there's less reason to isolate them per Operator.

13. Troubleshooting

If the system does not behave as expected, use the following checks:

Symptom	What to check
No radio response	Verify COM port, baud rate, and CI-V address in Radio Settings, and confirm CivBridge Running shows Yes in the Query Pane (Chapter 9).
OmniRig doesn't start automatically	Auto-start alone isn't enough — check Enable OmniRig Bridge in Radio Settings too (Chapter 8).
A companion program isn't running when expected	Confirm it's in the active Profile's programs (Chapter 7) or has Auto-start enabled (Chapter 8), and check Configuration file health in the Query Pane (Chapter 9).
A program isn't syncing correctly with another (e.g., Log4OM with WSJT-X)	Check program order and Restart when this profile starts in Profile programs (Chapter 7).
A Toggle, Cycle, Dropdown, or Action button shows the wrong state	Give it a moment — state is confirmed by querying the radio, not assumed (Chapter 7). If it's consistently slow, verifyDelayMs can be tuned per control (Chapter 12).
Macros not working	Check the macro's Type and Commands in the Macro Editor, and watch the Logging Pane while triggering it (Chapters 7 and 9).
Layout or pane settings not persisting	Confirm you're changing the Layout Mode you expect to persist — Normal, Tall Band, and Wide Band each save their own configuration independently (Chapter 5).
A config file seems missing or corrupted	Check Configuration file health in the Query Pane (Chapter 9); restore from a backup, or delete the file to let ShackMaestro recreate it from defaults (Chapter 12).
Need more detail on what's happening	Enable the Query Pane and Logging Pane (Chapter 9).

CI-V Behavior Notes

ShackMaestro operates over a shared CI-V communication channel. Due to the nature of serial communication:

Radio
↓
ShackMaestro (CI-V Controller)
↓
CivBridge (Virtual COM)
↓
ShackScope / OmniRig / Other Apps

- Some commands may be delayed under heavy activity
- Certain settings may take up to a few seconds to reflect on the UI
- Occasional retries may occur to ensure the radio reaches the intended state

These behaviors are normal and are part of ensuring reliable operation across multiple integrated applications.

For additional assistance or product requests, please contact us at **support@ShackMaestro.com**.

Appendix A: ShackScope Render Settings Explained

The render controls do two different jobs. Spectrum controls affect the live spectrum trace; Waterfall controls affect the waterfall's image history. Either way, the radio data itself doesn't change — these controls only change how ShackScope displays it.

Spectrum Floor
Controls how much of the weaker signal data is visible in the spectrum display — think of it as how much of the bottom of the signal range gets thrown away.

- **Lower (e.g. 0.10):** More weak signals and noise visible; spectrum looks busier; band floor rises.
- **Higher (e.g. 0.50):** Weak signals and noise suppressed; strong signals stand out; spectrum looks cleaner.

Practical recommendation: 0.20–0.35 for general HF; 0.10–0.20 for very weak-signal work; 0.30–0.50 for contest/CW.

Waterfall Black Level
Determines what signal level becomes pure black in the waterfall.

- **Lower:** More pixels become colored — more activity and noise visible.
- **Higher:** More pixels become black — cleaner waterfall, weak signals disappear sooner.

Practical effect: usually the first control people adjust. If the waterfall looks washed out, increase Black; if it looks empty, decrease Black.

Waterfall Gain
Amplifies signal intensity before coloring — think of it as a brightness control for signals.

- **Lower:** Signals look weaker.
- **Higher:** Signals become brighter and weak signals more visible; too much and everything glows, losing contrast.

Practical effect: Black and Gain work together — Black controls where color begins, Gain controls how fast color grows.

Waterfall Bias
Shifts the entire intensity curve up or down — think of it as a waterfall exposure adjustment.

- **Positive bias:** Everything brighter.
- **Negative bias:** Everything darker.

Practical effect: small adjustments can have surprisingly large effects. Useful when Black and Gain both look right but overall brightness still feels wrong.

Waterfall Compression
Controls how much difference exists between weak, medium, and strong signals.

- **Low compression:** Large differences remain; strong signals dominate; weak signals disappear.

- **High compression:** Differences are squeezed together; weak signals become easier to see; strong signals become less overwhelming.

Practical effect: think of it like audio compression — without it, weak stays very weak and strong stays very strong; with it, weak becomes easier to see and strong becomes less overpowering.

Waterfall Gamma
Changes how brightness is distributed — often the most misunderstood control.

- **Lower gamma:** Emphasizes weaker signals; waterfall appears brighter.
- **Higher gamma:** Emphasizes stronger signals; waterfall appears darker.

Practical effect: Gamma changes where contrast exists, rather than how much contrast exists.

Waterfall Speed
Controls how quickly rows scroll downward. Five levels are available \u2014 Min, Slow, Medium, Fast, and Max \u2014 covered from a mechanism standpoint in Chapter 10.

- **Slow:** More history visible — good for weak-signal work, WSPR, FT8, and monitoring.
- **Fast:** More real-time feel — good for SSB, CW, and band scanning.

How They Interact
The controls aren't independent. The most important relationships:
- **Black + Gain —** The primary controls — Black is the threshold, Gain is the brightness.
- **Compression + Gamma —** The advanced controls — Compression is dynamic range, Gamma is contrast placement.
- **Bias —** Fine tuning, usually touched last.

Recommended Presets

General HF SSB:
Spectrum Floor 0.25, WF Black 0.25, WF Gain 1.00, WF Bias 0.00, WF Compression 1.00, WF Gamma 1.00, WF Speed Medium.

Weak Signal Digital (goal: see everything):
Spectrum Floor 0.15, WF Black 0.15, WF Gain 1.20, WF Compression 1.30, WF Gamma 0.85.

Contest / CW (goal: reduce clutter, highlight strong signals):
Spectrum Floor 0.35, WF Black 0.35, WF Gain 0.90, WF Compression 0.90, WF Gamma 1.10.

Appendix B: Third-Party Program Setup

These focused guides cover connecting WSJT-X, Log4OM, and SDR Console through ShackMaestro — common examples, not the only possibilities. The same OmniRig-based approach applies to many other CAT-aware programs (Fldigi, N1MM+, HRD Logbook, DXLab Suite, SDR Uno, GridTracker, and others).

In every case, the same rule applies: ShackMaestro owns the physical radio connection. The other program should connect through OmniRig, not by opening the same physical COM port — see Chapter 2, Design Overview, and Figure 2-1 for how this fits together.

WSJT-X
Connects for digital-mode CAT control. Avoid having WSJT-X open the same physical radio COM port ShackMaestro uses.

1. In WSJT-X, go to File → Settings, then select the Radio tab.
2. If you use OmniRig, set Rig to OmniRig Rig 1 or OmniRig Rig 2, matching the slot configured in ShackMaestro Radio Settings.
3. Set PTT Method to CAT, Mode to None, and Split Operation to Fake It.
4. Click OK, then confirm WSJT-X tracks the radio's frequency.

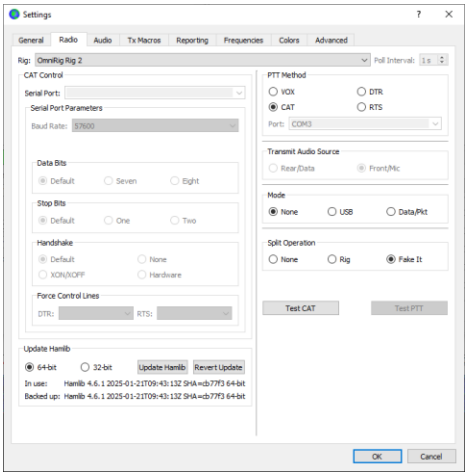


Figure B-1. WSJT-X Radio settings configured for OmniRig.

Success check: WSJT-X tracks frequency changes and CAT is active, without opening ShackMaestro's physical radio COM port.

Log4OM
Receives frequency and mode for logging. Avoid having Log4OM open the same physical radio COM port ShackMaestro uses.

1. In Log4OM, go to Settings → Program Configuration.
2. Select Hardware Configuration → CAT Interface.
3. If you use OmniRig, set CAT Engine to OmniRig.
4. Click Save and Apply.

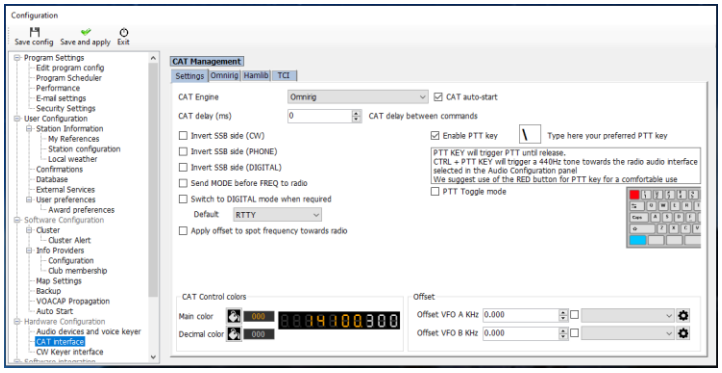


Figure B-2. Log4OM CAT Interface settings configured for OmniRig.

Success check: Log4OM shows the correct frequency and mode, without opening ShackMaestro's physical radio COM port.

SDR Console

Tracks radio frequency and mode for SDR/panadapter use. Avoid having SDR Console open the same physical radio COM port ShackMaestro uses.

1. In SDR Console, open the External Radio window.
2. Click the ? in the External Radio window's menu bar to open External Radio Options.
3. If you use OmniRig, choose the OmniRig radio slot matching your ShackMaestro/OmniRig configuration.
4. Enable frequency and mode tracking in the directions you want (e.g., SDR → Radio and Radio → SDR).
5. Click OK, then confirm SDR Console tracks the radio.

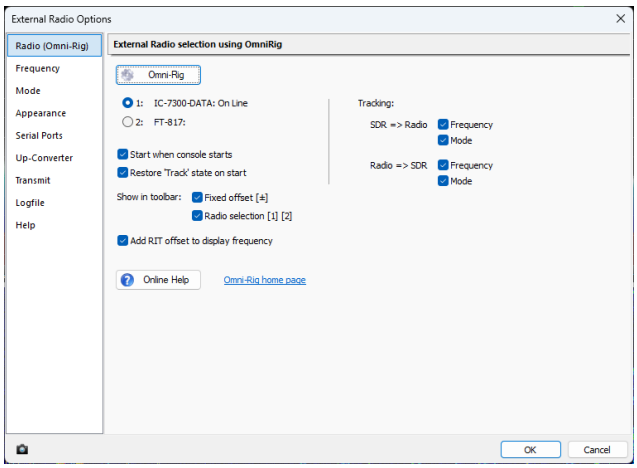


Figure B-3. SDR Console External Radio Options configured for OmniRig.

Success check: SDR Console follows the radio's frequency and mode changes, without opening ShackMaestro's physical radio COM port.

Other Compatible Programs

- **Digital modes —** Fldigi and other digital-mode programs that support shared CAT control.
- **Logging and contesting —** N1MM+, HRD Logbook, DXLab Suite, and similar tools that support OmniRig or CAT.

- **SDR and companion tools —** SDR Uno, GridTracker, and other tools that can follow radio frequency or mode.

Best practice: add one integration at a time, and confirm each program's success check before moving to the next — this makes setup problems much easier to isolate.