

PSBMS Engineering Tool — User Manual

A field commissioning tool for BMS/controls engineers: BACnet and Modbus device testing, network discovery, live trending and bench calculators, in one app.

- Version 1.21 · Windows 10/11 (64-bit) · © PSBMS Ltd
- Support / info: <https://psbms.com>

1. Installing, editions & first run

Install. Run `PSBMS Engineering Tool Setup.exe` and follow the prompts (accept the Windows elevation prompt). It installs to Program Files and adds Start-Menu and Desktop shortcuts. There is also a standalone `PSBMS Engineering Tool.exe` that runs without installing — just double-click.

On a PC that has never run it before, Windows may show a blue "unknown publisher" screen. Click **More info** → **Run anyway**. (This goes away once the app is code-signed.)

Editions. The tool is free for everyday work and unlocks its professional features with a licence:

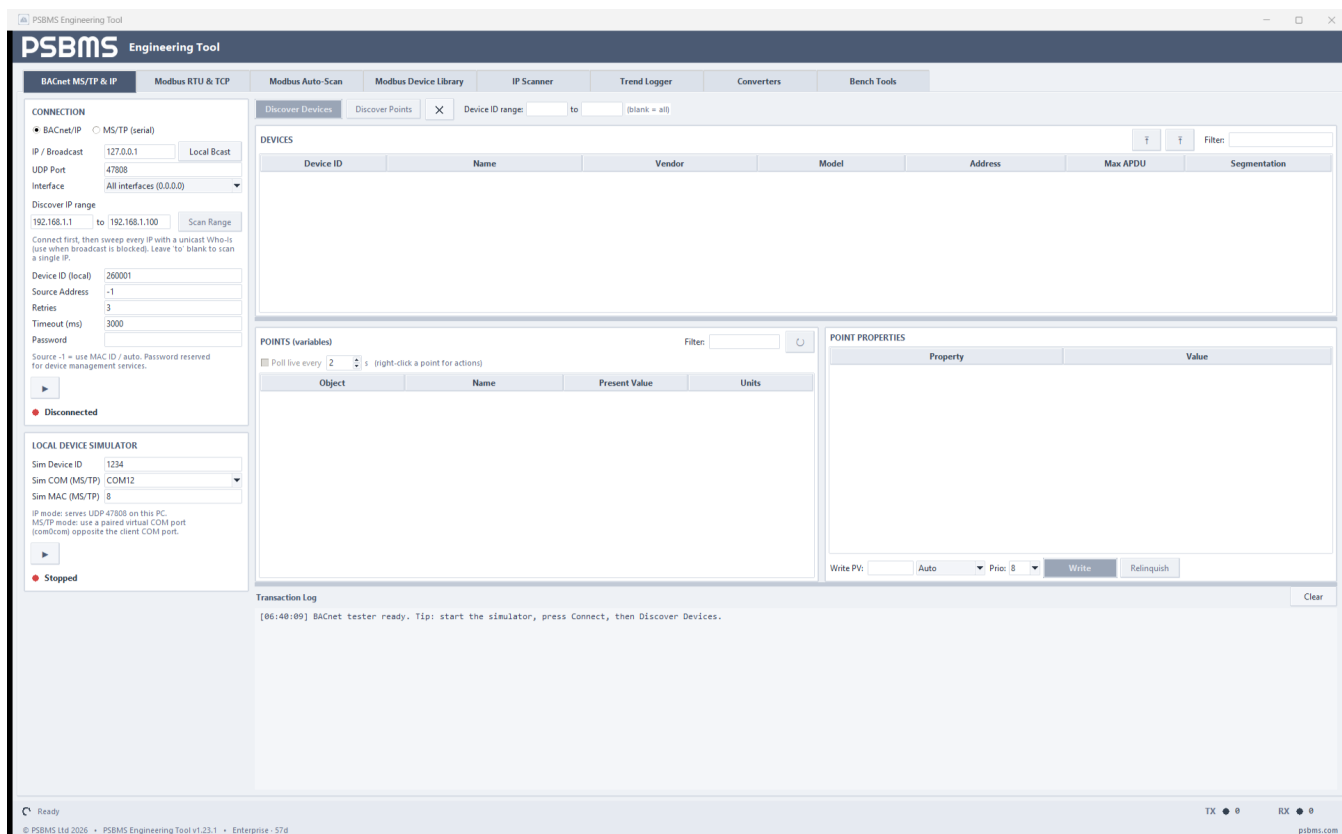
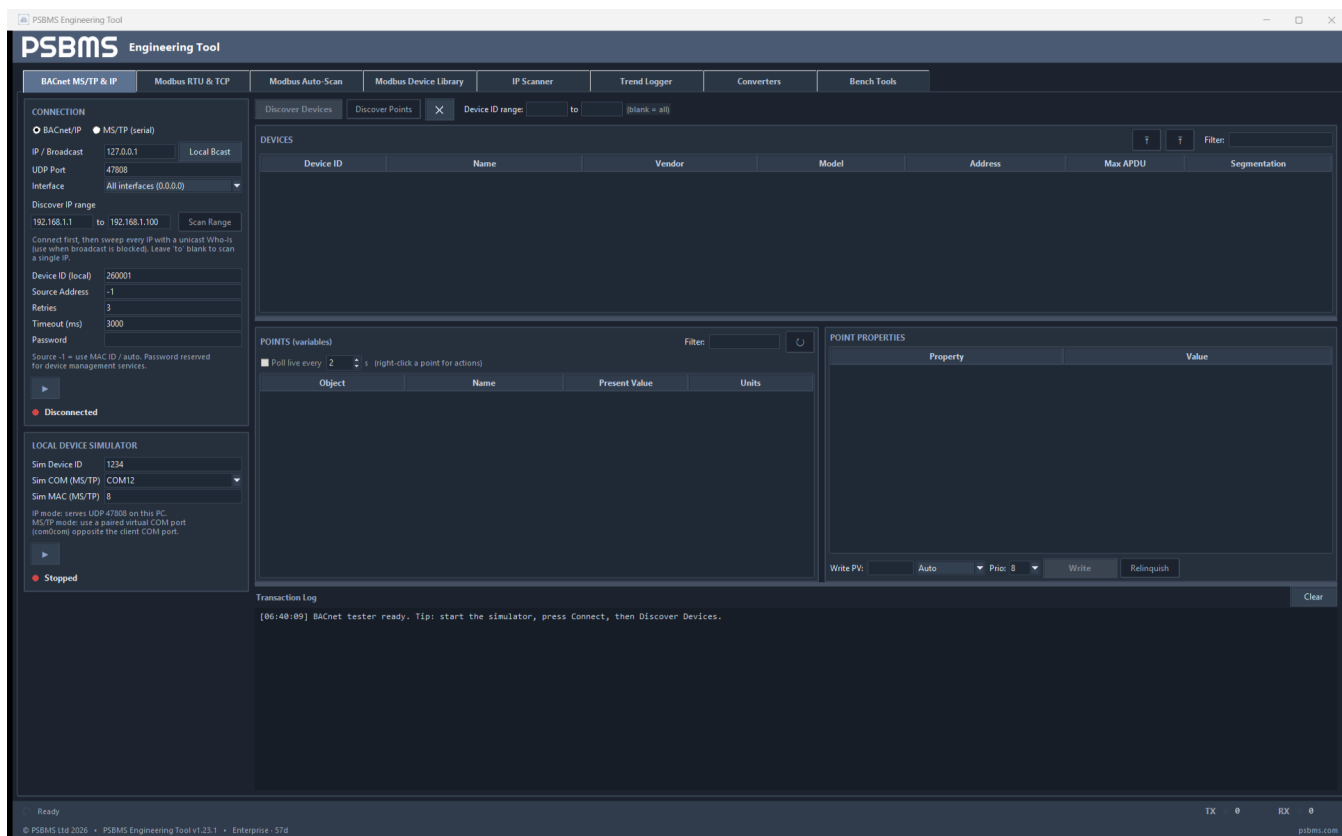
- **Community (free)** — the complete read-the-plant kit: BACnet, Modbus, IP Scanner, Bench Tools, the single-device simulators and a 2-channel Trend preview. No licence needed. **Importing and exporting data files — CSV import/export, the Excel/JSON round-trip and the branded PDF report — is a Pro/Trial feature;** clicking one shows a short upsell.
- **Pro** — unlocks all data **import/export** (clean, no watermark) plus the **Converters** module + branded report, unlimited Trend channels, saved connection profiles and batched operations.
- **14-day trial** — unlocks the full Pro feature set including import/export; Trial exports stay watermarked until you buy.

First run. The welcome screen offers three choices:

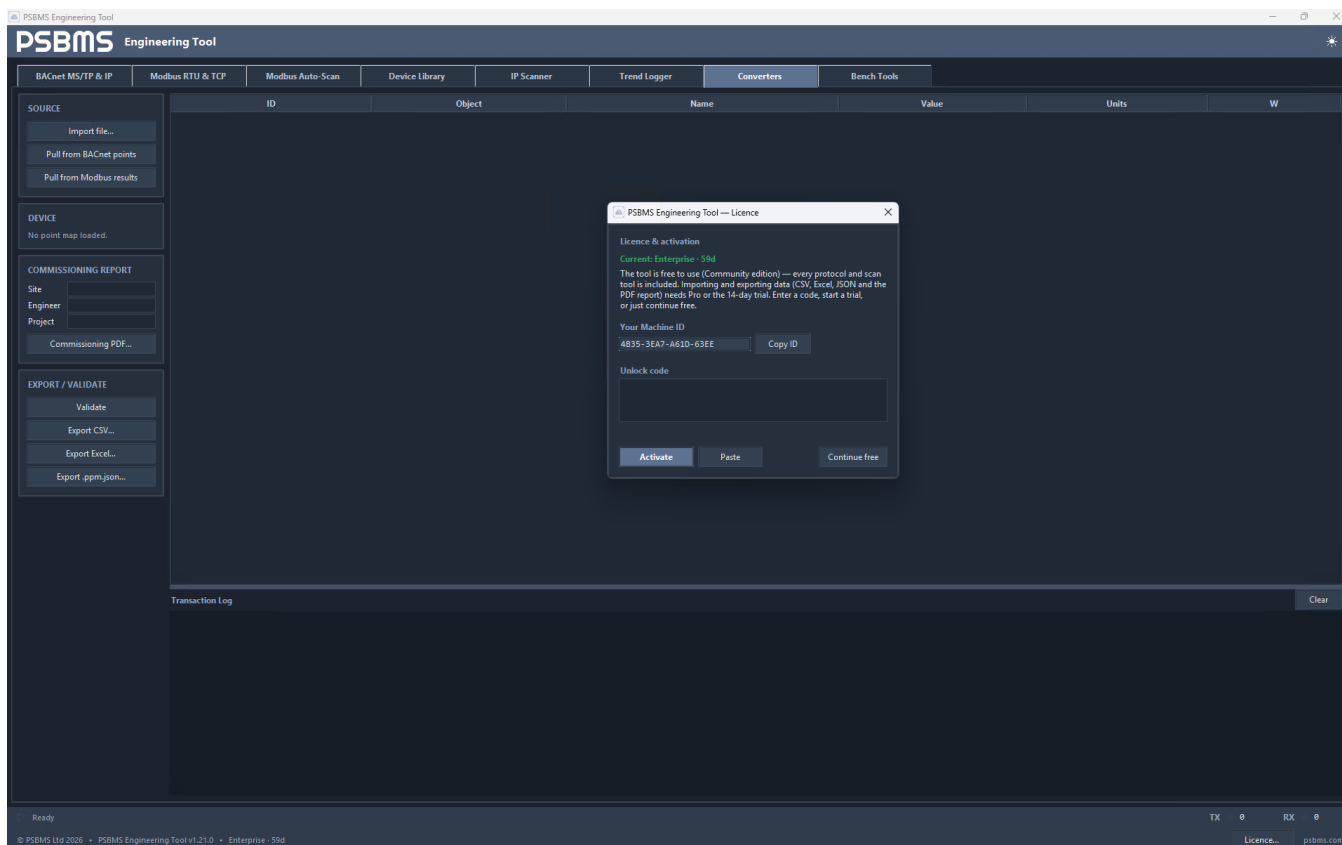
1. **Continue Free** — start using the Community tier straight away, no code.
2. **Start 14-day trial** — enter your email; the app unlocks the Pro feature set for 14 days.
3. **Activate** — paste a licence code. If you're offline, the screen shows **Your Machine ID** (e.g. `1973-4605-B4F0-FD21`) — send it to PSBMS, paste the code we return, and click **Activate**.

The footer always shows the current edition and, for a licence, when it expires. When a subscription lapses the app simply drops back to Community — it is never locked out, and your saved work is kept.

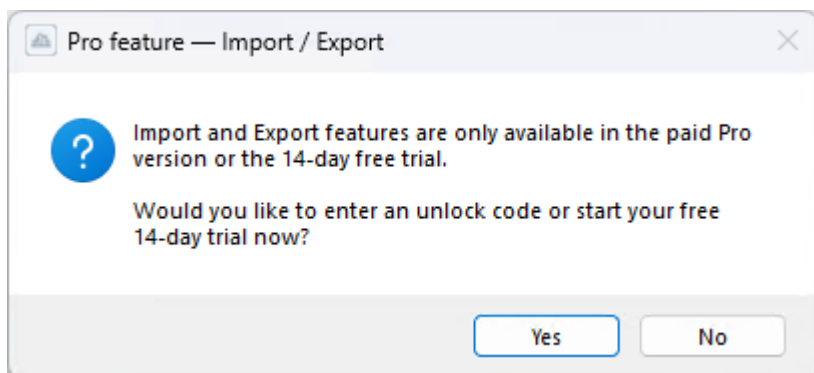
Appearance. The app ships in a **dark** theme; a sun/moon toggle on the brand bar (top-right) switches to a **light** theme, and your choice is remembered — logs and the trend chart follow the theme too. Many button rows are compact **icons with hover tooltips** — hover any button to see its label. The window remembers its **last position, size and maximized state** and restores them next launch. Each tab has a **drag handle above the Transaction Log** — drag it to make the log taller or shorter against the content above it; the Modbus tab adds a divider **between Read Results and Discovered Devices**, and the BACnet tab one **between the Devices list and the Points area**. A **progress bar** appears while a scan or discovery runs. Scrollbars **appear only when a table or log overflows**, so empty views stay clean.



The same tab in dark and light themes — toggle with the sun/moon on the brand bar.

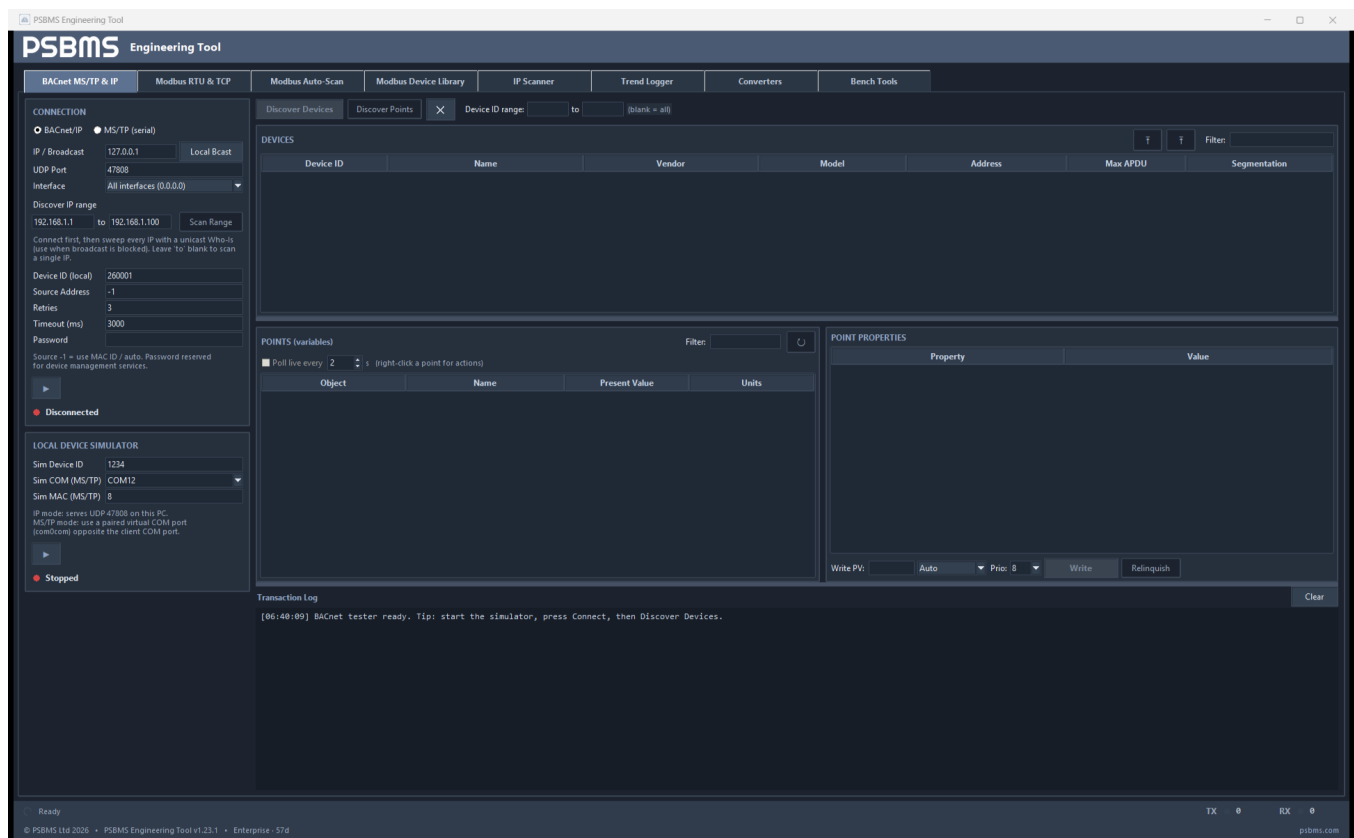


The Licence window: your Machine ID (to send to PSBMS) and the unlock-code box.



Community keeps every tool; importing/exporting a data file prompts to activate Pro or start the 14-day trial.

2. BACnet MS/TP & IP



Test BACnet devices, browse points, and write values. A **built-in simulator** lets you try everything with no hardware.

Connect

1. Choose **BACnet/IP** or **MS/TP (serial)**.
2. **IP**: set **IP / Broadcast** — the device's IP for a single device, or press **Local Bcast** to fill your subnet broadcast (e.g. 192.168.1.255) to find everything. 127.0.0.1 only searches your own PC. **MS/TP**: set COM port, baud and this tool's MAC ID.
3. **Interface** (IP only) — on a laptop with more than one network (e.g. Wi-Fi *and* a USB-Ethernet on the panel network), pick which adapter to send from. Leave it on *All interfaces* unless broadcasts are going out of the wrong NIC.
4. Set retries/timeout if needed and press the ► **Connect** button (a single toggle — it becomes ■ to disconnect once connected).

Not finding real devices? The tool must own UDP port 47808 to hear replies. If another BACnet program (e.g. IQVISION/Niagara) or this tool's own simulator is using it, the log shows an amber warning — close the other program, or point at the device's IP directly.

Try it with the simulator

In the **Local Device Simulator** card press **Start Simulator** (IP mode serves UDP 47808 on this PC), keep IP 127.0.0.1, **Connect**, then **Discover Devices**. A simulated AHU controller (device 1234) appears.

Discover & browse

- **Discover Devices** lists devices. Use **Device ID range** to limit the search to a band of device instance numbers (blank = all).
- **Discover an IP range** (BACnet/IP): in the Connection card set the **Discover IP range** from/to fields (e.g. `192.168.1.1` to `192.168.1.100`) and press **Scan IP Range**. This sends a unicast Who-Is to every address in the range and collects the replies — use it when broadcast is blocked or filtered on the network. (Connect first; max 1024 addresses.)
- Select a device and press **Discover Points** to list its objects with live **Present Value** and **Units**. Binary points read *Active/Inactive* and multi-state points show their *state text*.
- Click a point to see **all its properties** on the right.
- **Filter the devices** with the box beside the DEVICES heading — matches device ID / name / vendor / model / address.
- **Filter the points** with the box beside the POINTS heading — matches object / name / present value / units (the header shows "X of Y"). Handy on large controllers with hundreds of points. Both filters work for BACnet/IP and MS/TP.
- **Clear** empties the tables before a fresh search.

Live values

- ↻ **Refresh** re-reads the present values once.
- Tick **Poll live every N s** to keep them updating — ideal while you force a point and watch it move.

Right-click a point

- **Read all properties, Refresh present value**
- **View priority array...** (commandable points) — a 16-row window; the green highlighted level is the one in control.  = relinquished.
- **Copy object id / name**

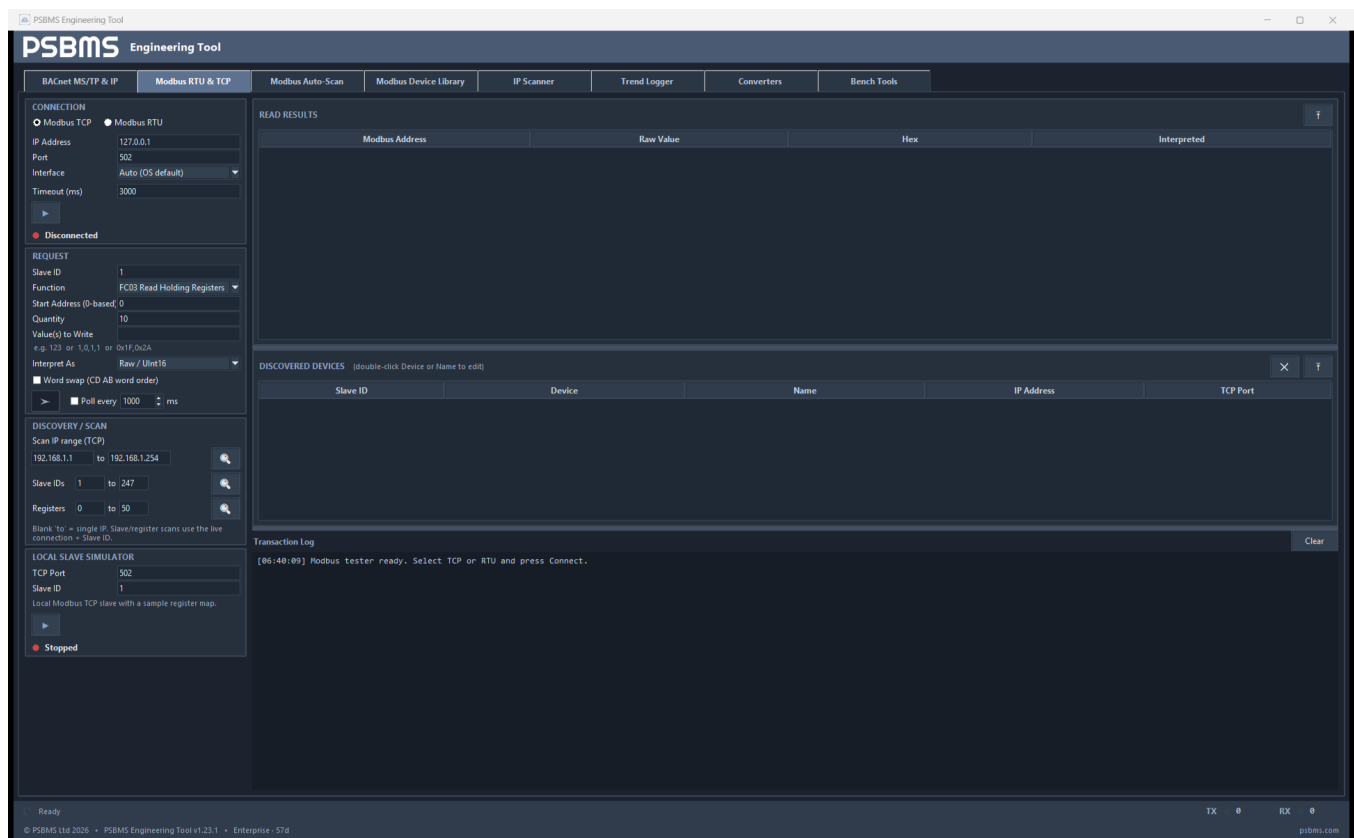
Write a value

Select a commandable point (setpoint, valve, mode...). The **Write PV** box is pre-filled with the current value. Choose a value type (Auto is fine) and a **Priority** (1–16), then **Write**. Choose **Null (relinquish)** to release a priority level.

Export

Export Devices CSV and **Export Points CSV** save the current tables.

3. Modbus RTU & TCP



A slave-ID scan of a gateway — each hit lands in the *Discovered Devices* table (*Device* + *Name* are editable), with the scan progress bar under the buttons.

A full Modbus master with a built-in slave simulator and discovery scans.

Connect

- **Modbus TCP:** IP + port (default `127.0.0.1:502`). **Interface** picks the source NIC on a multi-homed laptop (default *Auto* lets Windows route); it also applies to the IP-range scan.
- **Modbus RTU:** COM port, baud, data bits, parity, stop bits (↻ refreshes the port list).
- Press the ► **Connect** button. It's a single toggle — once connected it becomes ■ to disconnect (the same style as the LOCAL SLAVE SIMULATOR Start/Stop button).

Read / write

1. Set **Slave ID**.
 2. Choose a **Function** (FC01–06, 15, 16).
 3. Set **Start Address** (0-based) and **Quantity**.
 4. For writes, put value(s) in **Value(s) to Write** — a single value, or a comma list (`1,0,1,1`); hex like `0x1F` is accepted.
 5. **Send Request**. Results appear in **Read Results**.
- **Interpret As** re-reads registers as Int16/UInt16, 32-bit (Int32/UInt32/Float32) or 64-bit (Int64/UInt64/Float64); tick **Word swap** to reverse the word order (e.g. CD-AB).
 - Tick **Poll every ... ms** to read continuously.

Try it with the simulator

In **Local Slave Simulator** set a TCP port + unit and press **Start Slave**. Then connect TCP to `127.0.0.1` on that port and read holding registers.

Discovery / Scan

- **Scan IP range (TCP)** — sweep e.g. `192.168.1.1 ... 192.168.1.254` for Modbus devices at *different IP addresses* (no connection needed).
- **Scan Slave ID range** — find which slave addresses answer on the link. The log shows the value read from each unit.
- **Scan Register range** — read a block on the selected Slave ID to map which addresses hold data (great for undocumented devices). Results fill the table.

Discovered Devices table. The IP-range and Slave-ID scans list each device they find in the **Discovered Devices** table (Slave ID · Device · Name · IP Address · TCP Port), so you don't have to read them out of the log. **Double-click the Device or Name** cell to label a device (e.g. Device = *Schneider PM5000 Series*, Name = *PDU 1-7B*), use **Clear** to empty the list, and **Export** to save it to CSV (`Slave ID, Device, Name, IP Address, TCP Port`). Rows de-duplicate on (slave, IP, port), so re-running a scan won't create duplicates or lose the labels you typed. (Slave-ID scans on an RTU link leave IP/Port blank.) A **progress bar** under the scan buttons fills while a scan runs.

Scanning a Modbus/TCP gateway for its slaves

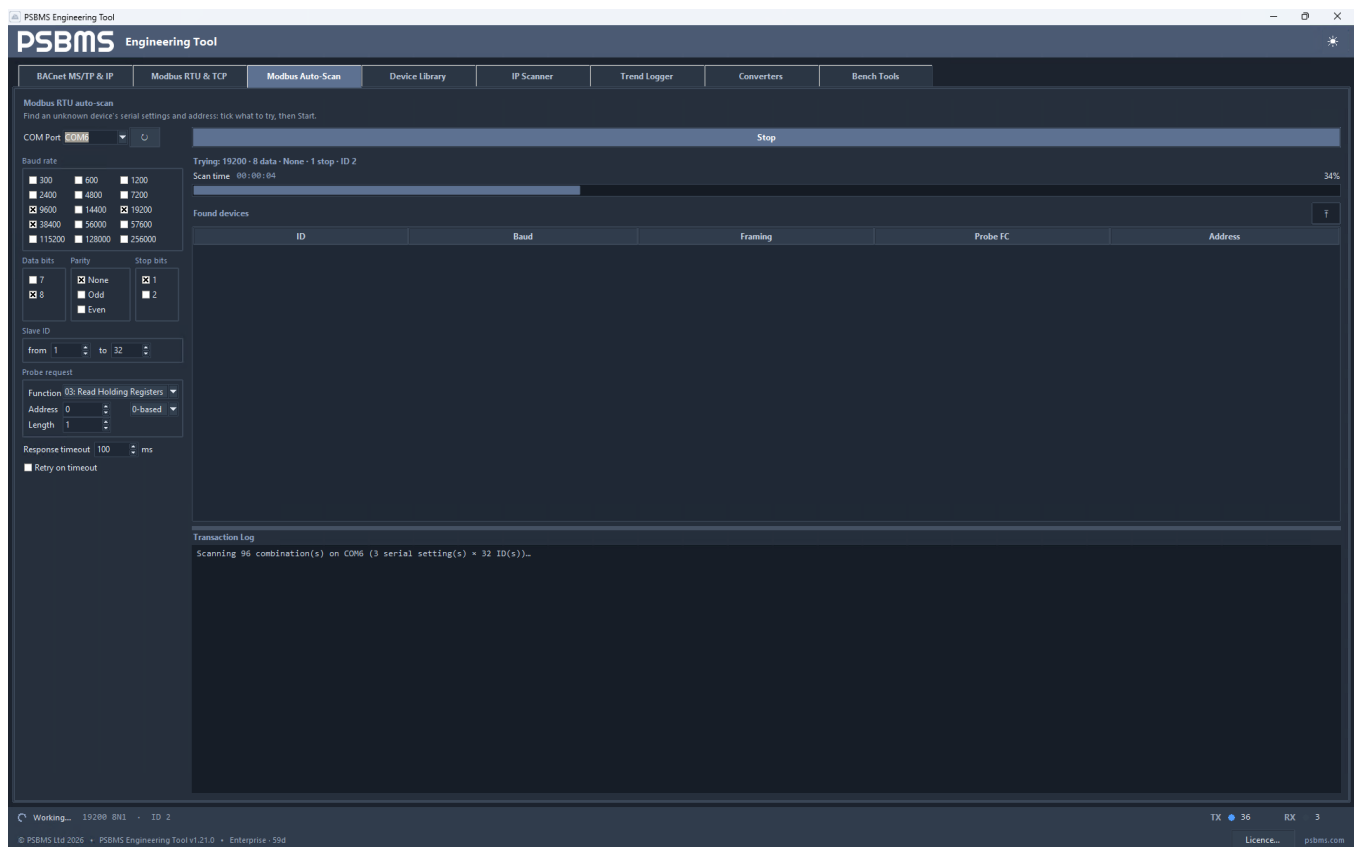
A gateway hosts many serial (RTU) slaves behind **one IP address**, addressed by **Slave ID** — not by IP. So:

1. Set **Modbus TCP**, enter the **gateway's IP** and port (usually 502), and press **Connect**. (Or right-click the gateway in the IP Scanner and choose "test in Modbus TCP tab" to fill this in for you.)
2. In **Discovery / Scan**, set the **Slave ID range** (e.g. 1–247) and press **Scan Units**.
3. Read the log: units returning **real data** are live slaves; units answering **0** (or a *gateway path unavailable / target failed* exception) are empty addresses. Tip: if a slave holds no data at address 0, set the Request card's **Start Address** to a register you know exists before scanning.

Auto-detect serial settings (Auto-Scan)

Handed an unknown RS-485 device and don't know its baud rate, parity or address? Open the dedicated **Modbus Auto-Scan** tab (§4), which brute-forces the serial settings and device ID until a device answers.

4. Modbus Auto-Scan



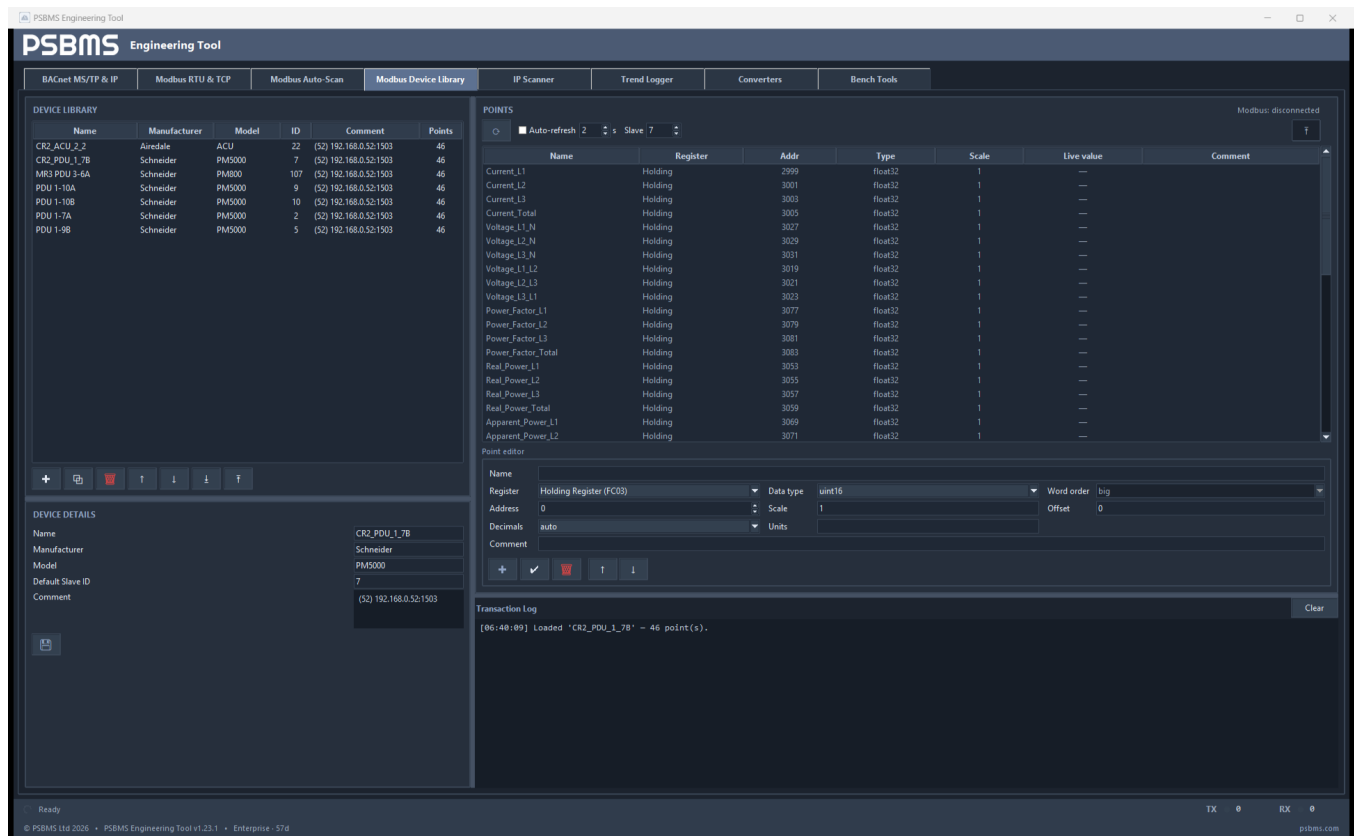
Brute-forcing an unknown RTU device's serial settings — the current combination, timer and progress bar update as it scans.

Find an unknown RTU device's serial settings and address by trying every combination until one answers.

1. Pick the **COM Port**.
2. **Tick what to try** — the **baud rates**, **data bits**, **parity** and **stop bits** to sweep (a quick, sensible set is pre-ticked; add more to widen the search), and the **Device ID** range (from / to).
3. Set the **Probe request** it knocks with (function — Read Holding Registers by default — address, 0/1-based, length), the **Response timeout** in milliseconds, and **Retry on timeout** if the link is marginal.
4. Press **Start scan**. It shows the settings it's **trying now**, the **scan time**, a **0–100 % progress bar**, and lists every device it **finds** in the **Found devices** table — any valid reply (real data or a Modbus exception) means a device is there at those settings, so you get its working baud/parity and address. **Stop** ends it early.
5. **Export CSV...** saves the found-devices table (ID, baud, framing, probe function, address) for your records.

The scan tries every combination, so it can take a while — a full sweep of all baud rates × parities × 247 IDs is thousands of probes. Start narrow (a few likely bauds, one parity) and widen only if nothing answers. The footer TX/RX LEDs flash on every frame; click them to watch the raw packets (see §10).

5. Modbus Device Library



The device list (Name, Manufacturer, Model, ID, Comment, Points) beside a stored register map, with the Point Editor below. Drag the divider between the two to widen either side.

A reusable database of Modbus register maps. Build a device's point table once — names, register kinds, addresses, data types, scaling and comments — store it, and reuse it on every visit. When you're connected to a slave, it reads every point and shows the value **beside its label**, turning a wall of raw registers into a named, commented point list.

Build a device

1. **New** creates a device; give it a **Name**, and optionally **Manufacturer**, **Model**, default **Slave ID** and a **Comment**. **Save** stores it in the library on the left (files live in `%APPDATA%\PSBMS Engineering Tool\modbus_devices\`). The library lists every device in a table — **Name**, **Manufacturer**, **Model**, **ID**, **Comment**, **Points** — and the **↑ ↓ Move device up / down** buttons reorder it; that order is remembered between sessions.
2. Add points with the **Point editor**: **Name**, **Register** (Holding / Input / Coil / Discrete), **Address** (0-based), **Data type** (bool, uint16, int16, uint32, int32, float32, **uint64/int64/float64** — 4 registers, for energy totalisers etc.), **Word order** (for 32/64-bit values, **big** = most-significant word first), **Scale** and **Offset** (engineering value = raw × scale + offset), **Decimals**, **Units** and a **Comment**. **Add** appends it; select a row and the **✓ update** / **🗑 remove** / **↑ ↓ move** buttons edit it. Remember to **💾 Save**.
3. **📄 Duplicate** clones the open device (handy for a family of similar units); **🗑 Delete** removes one from the library. (Hover any icon button for its label.)

Decimals controls how the live value is shown: **auto** prints whole numbers as integers and trims trailing zeros (and never uses scientific notation — a large run-hours count reads `262446`, not `2.62e+05`); set **0–6** to fix the number of decimal places.

Import / export

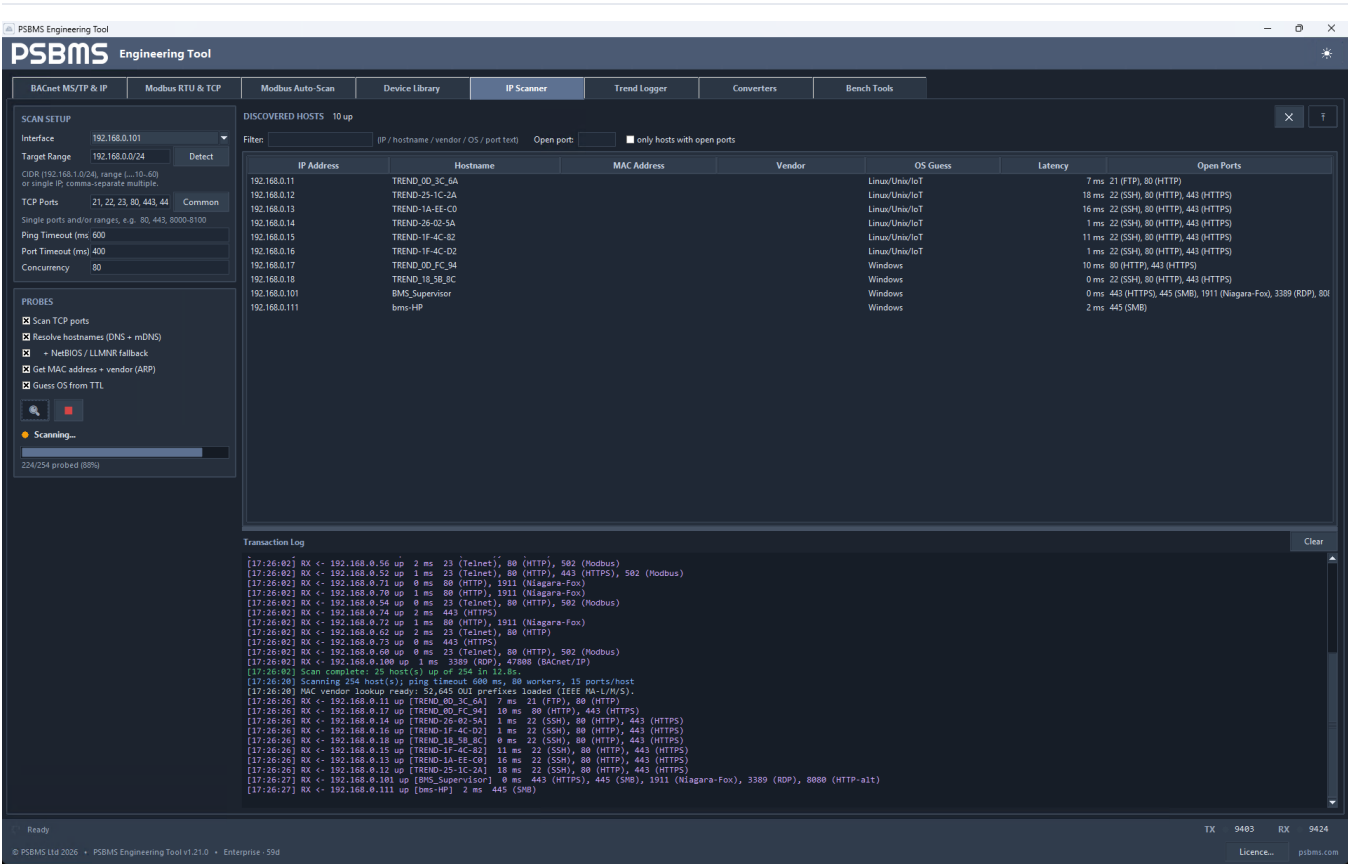
- **⌵ Import CSV** creates a device from a points CSV (columns: Name, Register, Address, DataType, WordOrder, Scale, Offset, Decimals, Units, Comment — register may be a name like *Holding Register* or a function-code number).
- **⌵ Export** opens a menu with three choices: a **Device list** summary CSV (one row per device — Name, Manufacturer, Model, Slave ID, Comment, Points), **This device's registers** (the open device's point map), or **Every device** → **folder** (one re-importable CSV per device). Share maps with colleagues or keep them under version control.

Read live values

1. Connect to the slave on the **Modbus RTU & TCP** tab (§3) — the Device Library shares that connection, so this works for both TCP and RTU.
2. Open the device, set the **Slave** to read (defaults to the device's), and press **⌵ Read live** — the **Live value** column fills with each point's scaled value **and its units** (e.g. `23.5 °C`). The points table no longer has a separate Units column — units are still set in the Point Editor, exported to CSV, and shown next to each value. Tick **Auto-refresh** to re-read every few seconds. While it reads, the footer **RX/TX LEDs** flash and the packet inspector captures the traffic, exactly as on the Modbus tab.
3. Points that couldn't be read (wrong address, not connected) show — and are greyed. Bits show **ON/OFF**; registers show the engineering value.
4. **⌵ Export live** writes the current readings (name, value, units, raw, comment, slave and timestamp) to CSV — a snapshot of what's on screen.
5. **Right-click** one or more points and choose **Add to Trend Logger** to chart them live — the channels carry their data type, word order and scale.

The live read uses the Modbus tab's own connection, so you never open a second COM port — essential for RS-485, where the port can only have one master.

6. IP Scanner



A subnet sweep — hostnames, MAC vendors, OS guess and open ports, with the progress bar and "x/y probed" count.

Discover and map devices on the local network.

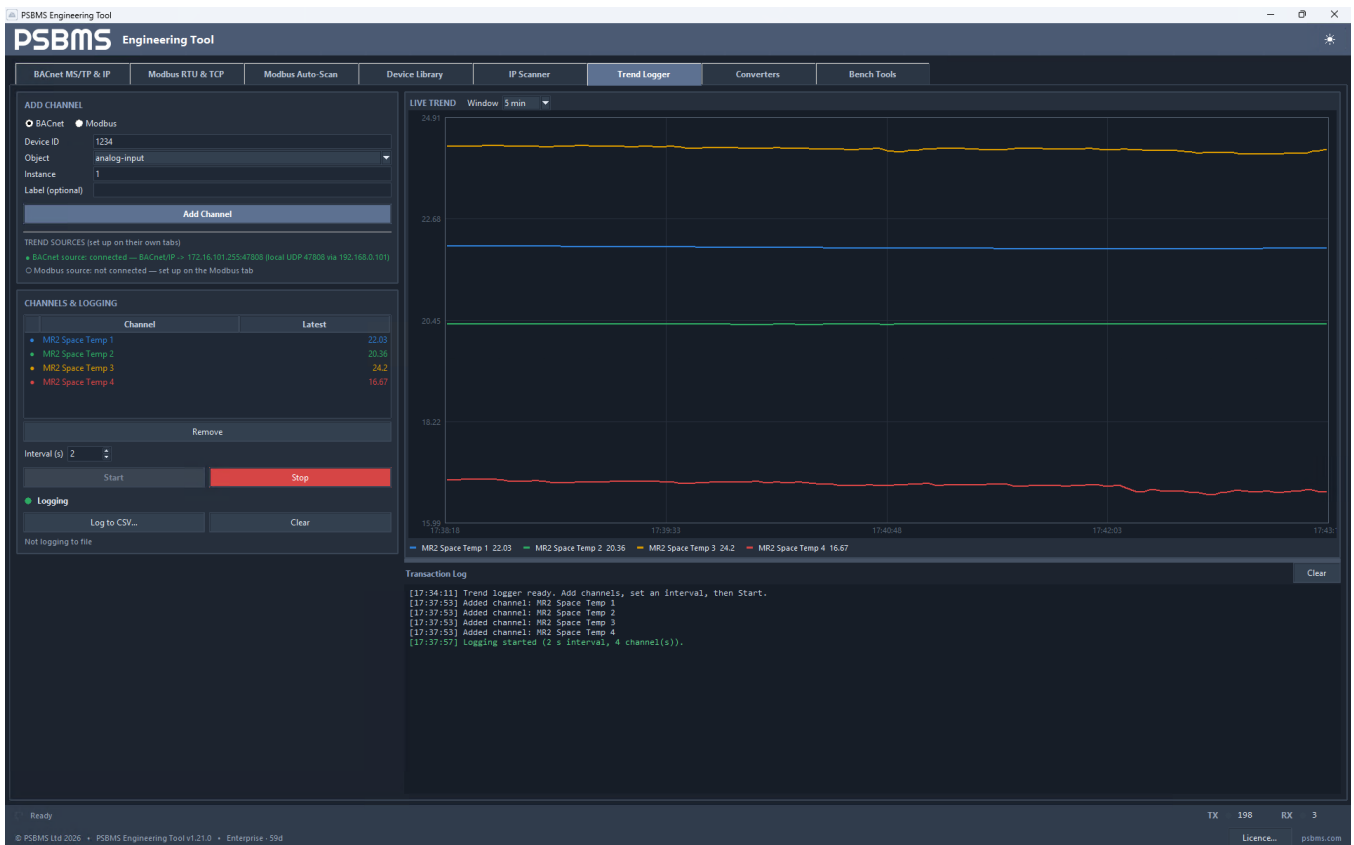
1. Press **Detect** to fill your local subnet, or type a range — CIDR (192.168.1.0/24), a range (.10-.60), or single IPs comma-separated.
2. Set **TCP Ports** — single ports and/or ranges (80 , 443 , 8000-8100); press **Common** for a sensible default set.
3. Tick the probes you want (ports / hostname via DNS + mDNS, with a NetBIOS + LLMNR fallback / MAC + vendor / OS guess) and press **Scan**.
4. Results stream in: IP, hostname, MAC, vendor, OS guess, latency, open ports. Click a column heading to sort. **Export CSV** to save.

The **Vendor** column is resolved from the device's MAC against a bundled IEEE manufacturer database (~52,600 prefixes) — it works fully **offline**, so manufacturers appear even on isolated plant networks. Randomised/private MACs (common on phones) have no manufacturer and are shown blank.

Filter the results with the bar above the table: type in **Filter** to match IP / hostname / vendor / OS / port text, enter an **Open port** to show only hosts with that port open, or tick **only hosts with open ports**. The count reads "X shown / Y up" while a filter is active.

Right-click a host for actions based on its open ports: open its web page (HTTP/HTTPS) in the browser, launch Remote Desktop (RDP), open a file share (SMB), copy an ssh/telnet/VNC command, or **test in the Modbus/BACnet tab** (loads the host's IP into that tab and switches to it). **Double-click** a host to open its web UI.

7. Trend Logger



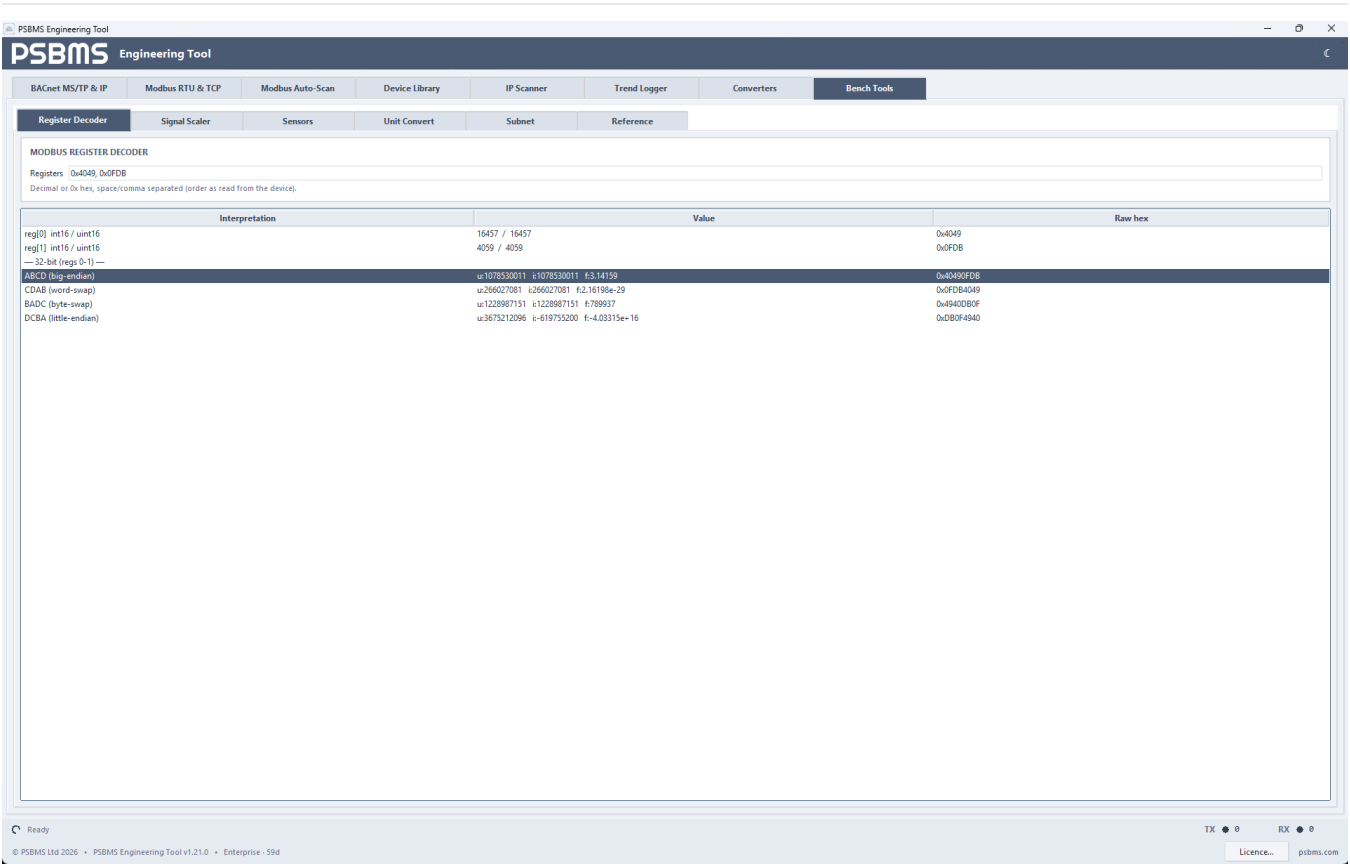
Sampling BACnet and Modbus points on an interval, plotted live and logged to CSV.

Watch and record values over time — the tool for proving a control loop.

1. First **connect** on the BACnet and/or Modbus tab. The **Trend Sources** line on this tab shows what each is connected as (or "not connected").
2. **Add a channel:** choose BACnet (device + object + instance) or Modbus (unit + register + type), optionally name it, and **Add Channel**.
3. Set the **Interval** and press **Start**. Values plot live on the chart and the latest reading shows next to each channel. Use the **Window** dropdown to change the visible time span.
4. **Log to CSV...** to also write every sample to a file. **Stop** to pause, **Clear** to reset the chart.

Community charts up to two channels (a preview); **Pro** removes the limit for unlimited channels.

8. Bench Tools



Offline calculators for daily work — register decoder, signal scaler, RTD/thermistor, unit and subnet tools.

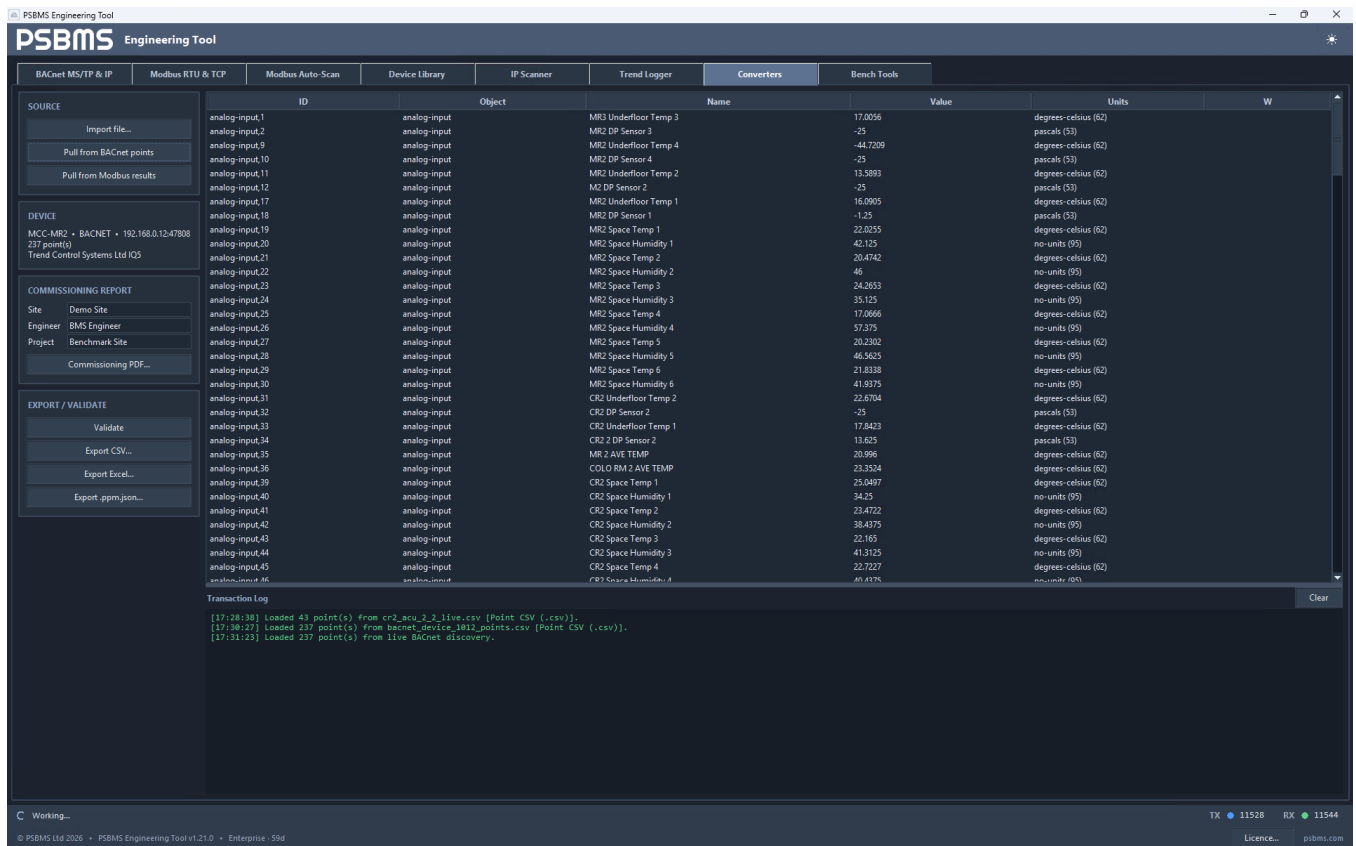
Offline calculators — no connection needed. Sub-tabs:

- **Register Decoder** — paste Modbus registers (decimal or 0x hex) and see Int16/UInt16, and 32-bit Int/UInt/Float across all four byte/word orders (plus Float64). Answers "which byte order is this meter?" instantly.
- **Signal Scaler** — convert 4–20 mA / 0–10 V (etc.) to engineering units and back. Set the signal and EU spans; type a value on either side.
- **Sensors** — NTC thermistor (10k Type II/III presets, editable Beta) and PT100/PT1000. Temperature ↔ resistance both ways, with a reference table.
- **Unit Convert** — temperature, pressure, air flow, power/heat, length.
- **Subnet** — IPv4 calculator: network, mask, broadcast, first/last host, usable count, from a CIDR or IP mask .
- **Reference** — Modbus exception & function codes, BACnet error codes & object types, common TCP ports.

9. Converters — point-list conversion & reports (Pro)



On Community the tab shows a Pro/Trial upsell bar...



...and once unlocked, imports point lists (here 237 live BACnet points) ready to validate, convert and turn into a branded PDF report.

Turn a controller's point list into a clean, portable point map and a branded commissioning report. A **Pro** feature — on Community the tab shows an unlock prompt; click **Activate Pro...** to enter a code.

1. **Load points.** **Import file...** reads a CSV, Excel or `.ppm.json` point list; or **Pull from BACnet points / Pull from Modbus results** grabs the points you've just discovered on those tabs.
2. **Validate.** Press **Validate** to flag missing names, duplicate addresses and out-of-range values before you hand anything on.
3. **Export.** Save a portable `.ppm.json` point map (**Export .ppm.json...**), a **CSV (Export CSV...)** or an **Excel** workbook (**Export Excel...**) — or generate a **Commissioning PDF...**: a branded, dated report of the point list for the project record. (Import/export needs Pro or the 14-day trial; Trial exports are watermarked, Pro is clean.)

10. The status bar, transaction log & packet inspector

The transaction log. Every tab has a colour-coded log at the bottom: blue = sent, purple = received, green = success, amber = warning, red = error, with timestamps. It's your first stop for diagnosing anything — and the best thing to copy when asking a device manufacturer's support for help. **Clear** empties it.

The status bar (bottom of the window) shows the tool is working and the comms traffic live:

- An **activity spinner** turns while a request is in flight, reading **Working... / Ready**.
- The active **serial settings** appear when connected or scanning (e.g. `9600 8N1 · ID 31`).
- **TX** and **RX** counters, each with an **LED that flashes on every frame** — real-time send/receive traffic at a glance.

The packet inspector. Click the **TX or RX counter** to open a live window listing every frame sent and received as **hex** — a running packet monitor, newest at the bottom. It keeps updating while you scan or poll; **Clear** empties the capture, **Close** dismisses it (the traffic keeps being captured).

11. Tips & troubleshooting

- **BACnet finds nothing:** you're probably pointed at `127.0.0.1`, on the wrong subnet, or another app owns UDP 47808 (watch the log warning). Use the device IP or **Local Bcast**, and close IQVISION/Niagara if it's running.
- **Modbus times out:** check unit ID, 0-based addressing, and (RTU) the serial settings match the device.
- **Values look wrong (Modbus):** try a different **Interpret As** / **Word swap**, or use the Bench Tools **Register Decoder** to see every option.
- **Windows Firewall** may prompt on first network use — allow it on private networks.
- **Licence lapsed:** the app drops back to the free **Community** tier — it is never locked out, and your saved work is kept. Activate a new code to restore Pro.