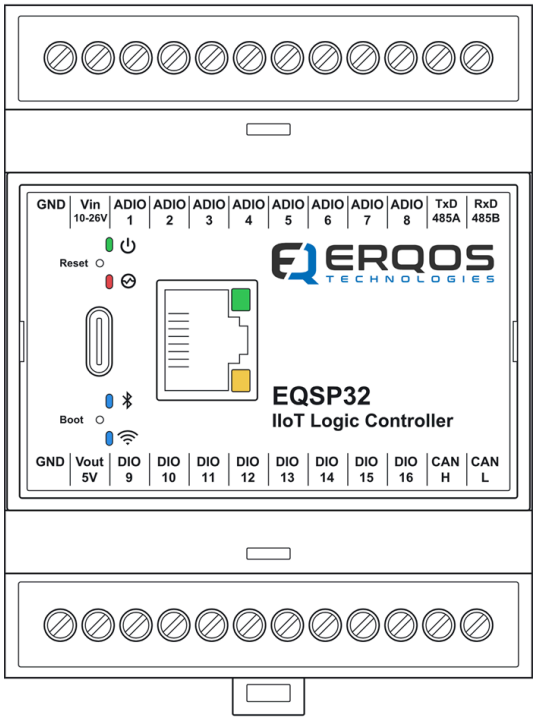
ERQOS

EQSP32

OpenPLC board package

USER MANUAL



User Manual v1.0.0 · Rev 102226

© 2026 Erqos Technologies · erqos.com

Contents

GETTING STARTED

1. What this package provides
2. Prerequisites
3. Installing the board package (customer kit)
4. Creating a project and selecting the device

CONFIGURATION

5. System Layout — how EQSP32 I/O is organized
6. Main-unit pin modes and PLC values
7. Aliases and IEC addresses (important)
8. EQX expansion modules in detail

PROGRAMMING

9. Working with scaled values in the PLC program
10. Modbus and networking
11. Compile, upload, and first run

REFERENCE

12. Limits and practical cabinet notes
13. Upgrading the package / missing channels
14. Troubleshooting
15. What stays with EQConnect / the EQSP32 library
16. Support scope for this package version (1.0.0)

DOCUMENT

Notices

Revision history

1. What this package provides

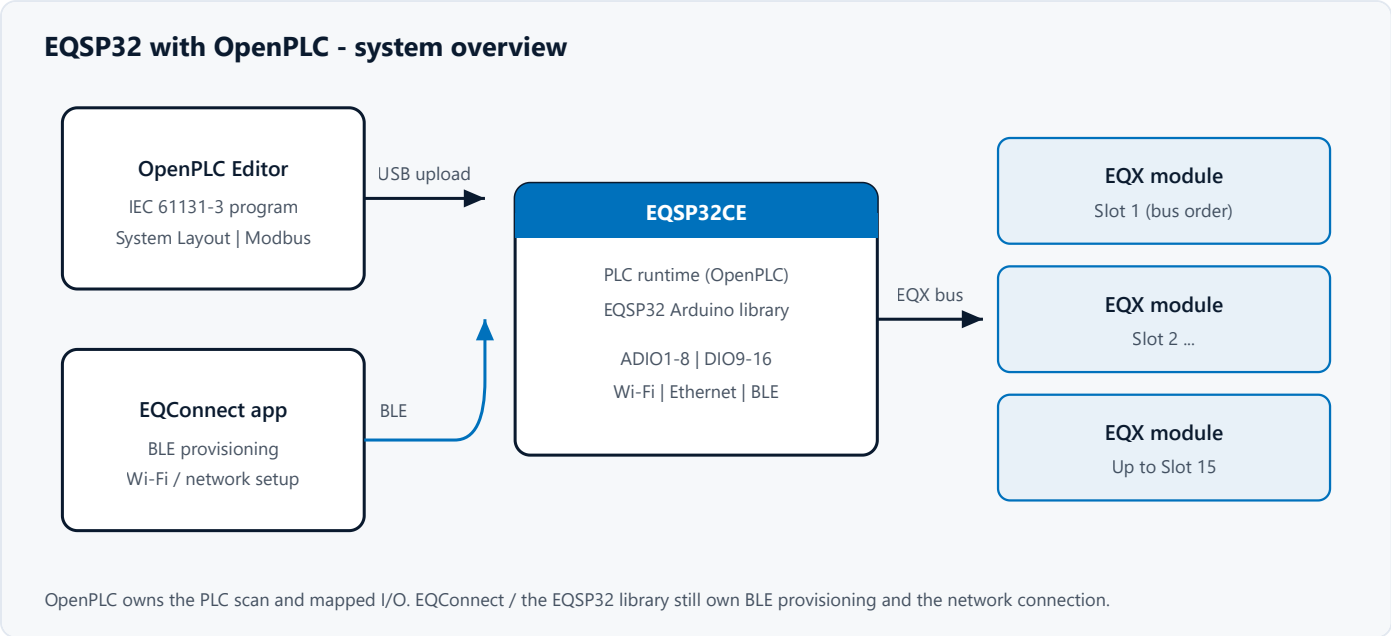


Figure 1: OpenPLC Editor and EQConnect work with the EQSP32CE and chained EQX modules.

Once installed, OpenPLC Editor can:

- Select **EQSP32CE** as the compile/upload target
- Configure the EQSP32 main-unit terminals and up to 15 EQX modules in one place (**System Layout**)
- Compile and upload PLC programs through the official **EQSP32 Arduino library**

The package **carries its own copy of the EQSP32 library** (precompiled). OpenPLC compiles every EQSP32CE project with that copy, even when another EQSP32 library version is installed on the PC, so a given package version always builds with the library it was tested with. A newer library reaches OpenPLC users with a new package version.

Because I/O goes through that library, these continue to work alongside the PLC runtime:

- **BLE provisioning** with the EQConnect mobile app
- Automatic **Wi-Fi / Ethernet** connection

There is **no separate Pin Mapping table** for this board. All I/O is configured under **Configuration → System Layout**.

2. Prerequisites

Item	Requirement
OpenPLC Editor	4.3.0 or newer (Windows; tested with 4.3.2). Start it once before the trust patch.
Node.js	Required once, to run the trust-patch script
USB	USB cable to the EQSP32CE for first upload (COM port)
Internet	For the trust patch, and for the first build (the editor downloads the ESP32 core 2.0.17 and RadioLib 7.7.1)

The editor installs the **ESP32 Arduino core 2.0.17**, which the EQSP32 needs, by itself on the first build.

Build folder: the trust patch also moves OpenPLC's Arduino folder to `C:\EQSP32-OpenPLC`, so builds work with any Windows user name. Files downloaded before the patch are downloaded again on the next build (a few minutes, once).

3. Installing the board package (customer kit)

The OpenPLC Editor only accepts **cryptographically signed** board packages. Until Erqos is an official signed vendor in the editor, the download kit includes a one-time editor patch that adds Erqos' public signing key.

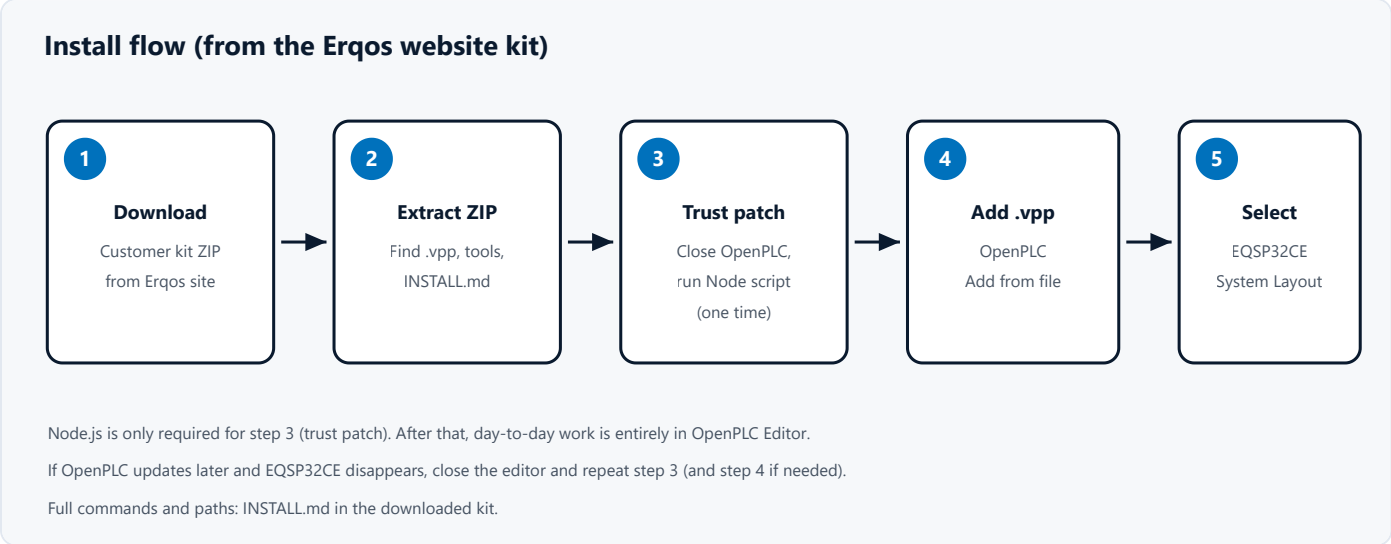


Figure 2: Download the kit, extract it, run the one-time trust patch, add the .vpp, then select EQSP32CE.

For step-by-step commands aimed at website downloads, follow **INSTALL.md** in the kit (extract the ZIP, close OpenPLC, run the Node trust patch, then add the `.vpp` in the editor's Package Manager: Device → Configuration → Device list → **+ Install additional boards...** → **Installed** → **+**).

3.1 Contents of the download kit

```
com.erqos.eqsp32-1.0.0.vpp     Signed board-support package (includes the EQSP32
library)
INSTALL.md                     Quick install for website customers
USER_MANUAL.html               This manual (open in a browser)
RELEASE_NOTES.md               What is new in this version
tools/patch-editor-trust.mjs   One-time editor trust, short build path and display
patch
tools/migrate-project-config.mjs  Fixes projects saved with an older package
(section 13)
keys/eqsp32-dev.pub.pem        Public key only (safe to share)
examples/                      One example project per language (section 11.1)
LICENSE.md                     License of the package and the EQSP32 library
```

3.2 After an OpenPLC Editor update

If the editor reinstalls or updates itself, the trust patch may be lost. Close the editor, extract the kit again if needed, and re-run the trust patch from **INSTALL.md**. Reinstall the `.vpp` if **EQSP32CE** is missing.

4. Creating a project and selecting the device

1. Create or open a project in OpenPLC Editor.
2. Open **Configuration** and select device **EQSP32CE**.
3. Open **Configuration → System Layout**.

You should see:

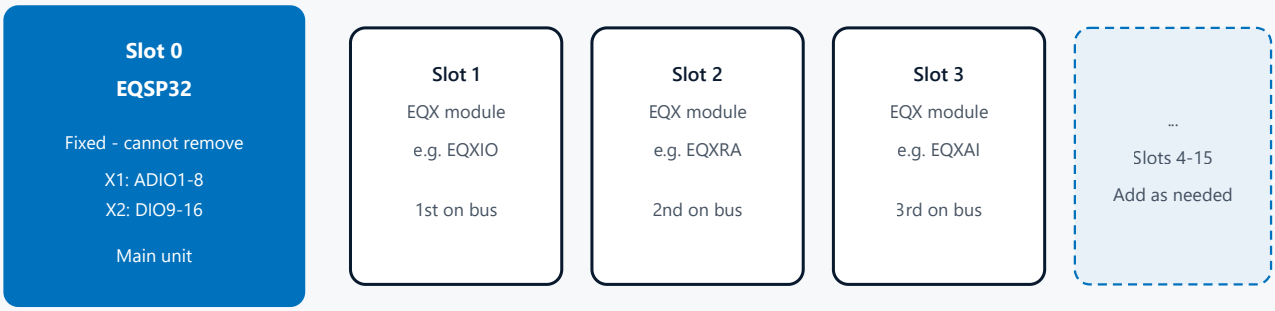
- **Slot 0 — EQSP32** (fixed; cannot be removed)
- Optional expansion slots (**Add module**) for EQX modules

If slot numbers show as 1, 2, 3… instead of 0, 1, 2… the trust/display patch was not applied. Behavior still works; only labels differ.

5. System Layout — how EQSP32 I/O is organized

System Layout - slots match the physical EQX chain

Typical DIN-rail order (left = EQSP32, then modules as chained)



1. Add EQX modules in OpenPLC in the same order they are physically chained.
2. Slot number is the module place in that chain - the EQSP32 finds each module by this order.
3. Without the editor patch, labels may show as Slot 1, 2, 3... (same slots; display only).

Figure 3: Slot 0 is the fixed EQSP32. Slots 1-15 are EQX modules in physical bus order.

I/O Mapping - Mode, address, and alias

Channel	Type	Mode	IEC Address	Alias
ADIO1	digitalInput	DIN (Digital Input)	%IX0.0	ADIO_1
ADIO5	analogInput	TIN (NTC Temperature)	%IW0	ADIO_5

Mode

Electrical function of the terminal (EQSP32 library). Can change the IEC class.

IEC Address

Auto-assigned by OpenPLC. May reflow when Modes change. Do not hard-code in logic.

Alias

Stable name for variables (Location dropdown). Always bind via aliases.

Figure 4: Bind PLC variables with aliases. IEC addresses are auto-assigned and may reflow when Modes change.

5.1 Slot 0 — EQSP32 main unit

Connector	Channels	Capabilities
X1	`ADIO1` – `ADIO8`	Digital and analog modes
X2	`DIO9` – `DIO16`	Digital, power PWM, relay, pulse counter

Each row in the **I/O Mapping** table has:

- **Channel** — faceplate / connector name
- **Type** — IEC class (digital/analog in/out)
- **Mode** — electrical function (EQSP32 library mode)
- **IEC Address** — auto-assigned (`%IX` , `%QX` , `%IW` , `%QW`)
- **Alias** — friendly name for PLC variables

5.2 Slots 1–15 — EQX expansion modules

Add modules **in the same physical order** they are chained on the EQSP32 expansion bus. The EQSP32 finds each module by its place in the chain, so the slot order must match.

Supported modules in 1.0.0:

Module	Role
EQXIO	10× digital I/O (DIN / SWT / POUT)
EQXAI	8× analog in (AIN / CIN / TIN)
EQXRA	8× relays + 2× 0–10 V analog outs
EQXTC	6× thermocouple
EQXPT	2× PT100 (2/4-wire or 3-wire per channel)
EQXPH	1× pH
EQXHI	10× isolated AC/DC voltage-presence inputs, frequency and quality per input

The number of modules is also limited by OpenPLC's I/O limits (see §12).

5.3 Configuration section (per slot)

Below each slot's I/O Mapping table is a **Configuration** panel. Only parameters relevant to selected modes appear (for example NTC coefficients when a pin is TIN). Module-level settings (PWM frequency,

EQXRA watchdogs) appear at the top of that panel.

6. Main-unit pin modes and PLC values

Mode labels always start with the EQSP32 library code.

Mode	Where	IEC	PLC value / notes
DIN (Digital Input)	X1/X2	`%IX`	0 / 1 — default ADIO1–4
SWT (Switch Input)	X1/X2	`%IX`	0 / 1 — debounce in Configuration
AIN (Analog Voltage Input)	X1 only	`%IW`	millivolts (0–5000) — default ADIO5–8
CIN (Current Loop 4-20mA)	X1 only	`%IW`	mA × 100 (e.g. 1200 = 12.00 mA)
TIN (NTC Temperature)	X1 only	`%IW`	°C × 10, **signed** — use `WORD_TO_INT`
RAIN (Relative Analog Input)	X1 only	`%IW`	0–1000 relative to 5 V VOut
PCC (Pulse Counter)	X2 only	`%IW`	pulses since the previous scan
PCC (Pulse Total)	X2 only	`%ID` (DINT)	running total, never cleared (§6.2)
QENC A (Quad encoder signal A)	DIO9, 11, 13, 15	`%ID` (DINT)	encoder position; the next terminal must be QENC B (§6.2)
QENC B (Quad encoder signal B)	DIO10, 12, 14, 16	`%IX`	TRUE while the encoder counts up
POUT (Digital Output)	X1/X2	`%QX`	off / on — default DIO9–12, 15–16
POUT (Power PWM Output)	X1/X2	`%QW`	0–65535 → 0–100% duty — default DIO13–14
RELAY (Relay Derated Output)	X1/X2	`%QX`	pull-in then hold; hold % and delay in Configuration

6.1 Main-unit Configuration parameters

Parameter	Applies when	Notes
PWM Frequency	Any POUT/RELAY on main unit	Shared for the whole POUT group (Hz)
Relay Sequencer	RELAY	Off (default): RELAY outputs switched on in the same scan pull in together. On: they pull in one after another, to spread the pick-up current

Parameter	Applies when	Notes
Switch Debounce	SWT	Milliseconds
NTC Conversion + coeffs	TIN	Beta, Steinhart-Hart, or Normalized S-H
Pulse Edge	PCC, PCC (Pulse Total)	Rising, Falling or Both edges
Encoder Signal Format	QENC A	A/B x4 (default), x2, x1; pulse + direction; up/down pulses
Encoder Direction	QENC A	Normal or Reversed
Relay Hold Power	RELAY	/1000 (1000 = 100%)
Relay Derate Delay	RELAY	ms at full power before hold

6.2 Counters and encoders

Pulse totals and encoder positions are 32-bit signed numbers (`DINT`) at `%ID` addresses. Declare the variable as `DINT` , or pick it by its alias.

- **The counters run free.** The package never clears a total or a position while the PLC runs (both start at 0 after a restart). To zero one, for example at a home switch, keep a reference value in the program and subtract it:

  ```
  IF Home_Switch THEN
    Home_Pos := Conveyor_Pos;          (* remember where "home" is *)
  END_IF;
  Position := Conveyor_Pos - Home_Pos;  (* 0 at home, counts from there *)
  ```

  In ladder: one rung with a `Home_Switch` contact driving `MOVE Conveyor_Pos → Home_Pos` , and one rung `SUB Conveyor_Pos - Home_Pos → Position` . A pulse total works the same way (`Batch := Flow_Total - Batch_Start;`).
- **Roll-over is safe.** After about 2.1 billion counts a counter rolls over as a normal 32-bit counter (from +2,147,483,647 to -2,147,483,648), and `Now - Start` stays right across it.
- **An encoder uses a terminal pair:** DIO9/10, 11/12, 13/14 or 15/16. Set the first terminal to `QENC A (Quad encoder signal A)` and the second to `QENC B (Quad encoder signal B)` , and wire the encoder's signals A and B to them. In the program the first terminal gives the position and the second reads TRUE while the position counts up. The Signal Format in Configuration sets what the B input does: A/B x4, x2 or x1 (counts per signal period), pulse + direction (pulses on the first terminal, direction

level on the second, high counts up), or up/down pulses (the first terminal counts up, the second counts down).

- **The build stops with a message** when the second terminal of a `QENC` pair is not `QENC B` (or `QENC B` has no `QENC A` before it), or when more than four terminals and pairs use the counters: `PCC (Pulse Counter)`, `PCC (Pulse Total)` and `QENC` pairs share four hardware counters.
- **Power the EQSP32 through VIn when an encoder is connected.** On USB power alone the first counts after a pause can be lost.
- **Modbus does not read `%ID`.** To give a position or a total to an HMI or a Modbus master, copy it to a `%MD` variable every scan (§10):

```
Position_MB AT %MD0 : DINT;   (* holding registers 52-53 *)
...
Position_MB := Position;
```

7. Aliases and IEC addresses (important)

IEC addresses are **auto-assigned** (lowest free address per class). Changing a Mode can change a channel's IEC class and **reflow** addresses.

Always bind PLC variables using aliases, not hard-coded addresses.

Default aliases (when the editor patch is applied):

Hardware	Example alias
Main unit ADIO1	`ADIO_1`
Main unit DIO9	`DIO_9`
First EQXIO, DIO3	`XIO_1_DIO3`
Second EQXRA, RL5	`XRA_2_RL5`
First EQXTC, CH1	`XTC_1_CH1`

You may rename aliases; user edits always win. Use letters, digits and `_` (the editor does not accept `.` or spaces in an alias).

Pin names in EQMonitor. A terminal or module channel you rename here gets that name on the EQSP32 when the program starts, so EQMonitor shows it next to the pin (for example `Start_Button` beside terminal 3). Default aliases are not sent: they only repeat the pin number. Names are set at every start and are not stored in the EQSP32.

8. EQX expansion modules in detail

Channel names include the connector prefix (`X1.` , `X2.` , `X3.`) to match the module housing.

A module that is missing or stops answering (unplugged, loose cable, module reset): its digital inputs read FALSE, its analog inputs read 0, its temperatures read −32768, and its outputs are not written. Write your program so that FALSE is the safe state of every input that permits motion or heat.

8.1 EQXIO

Channels	Modes	PLC
X1.DIO1–5, X2.DIO6–10	DIN / SWT / POUT	`%IX` or `%QX`

SWT debounce appears in Configuration when SWT is selected.

8.2 EQXAI

Channels	Modes	PLC
X1.CH1–4, X2.CH5–8	AIN / CIN / TIN	`%IW` — mV / mA×100 / °C×10

TIN conversion parameters appear in Configuration (same options as the main unit).

8.3 EQXRA

I/O Mapping (physical terminals only):

Channels	Mode	PLC
X1.RL1–2, X3.RL3–8	RELAY (Relay Output)	`%QX` on/off
X2.AO1–2	AOUT or SAOUT	`%QW` 0–65535 → 0–10000 mV

RL1–2 have independent COM; RL3–8 are paired COM (3–4, 5–6, 7–8).

The EQXRA's internal status values are not PLC channels in this release.

AOUT vs SAOUT

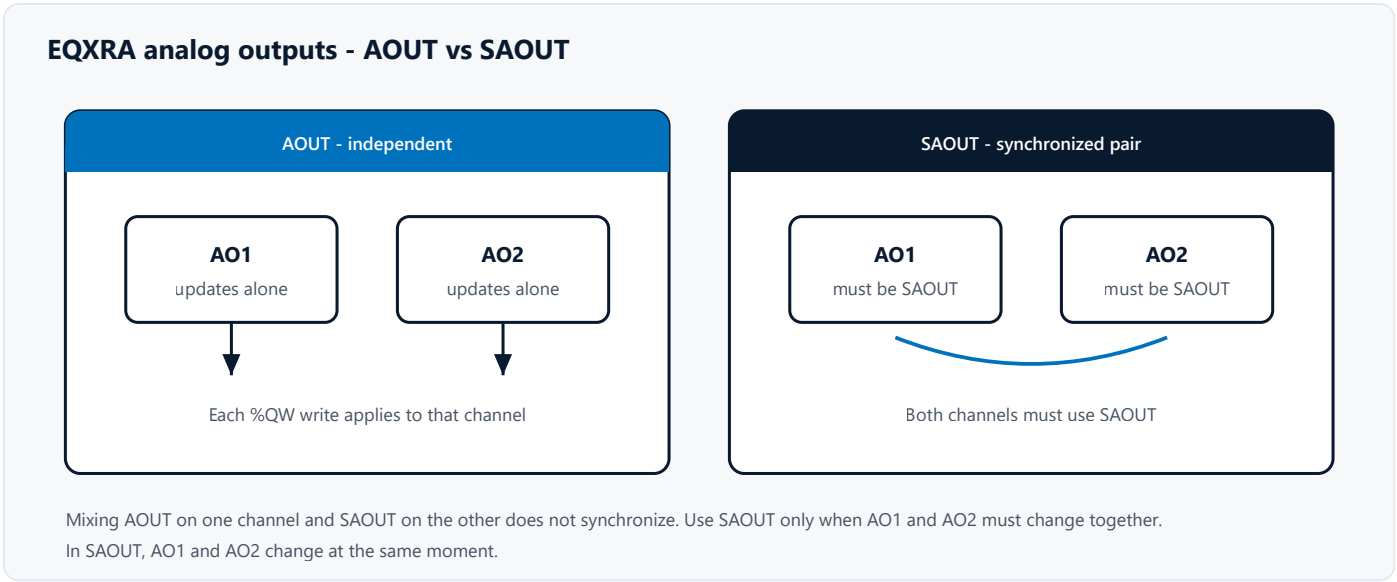


Figure 5: AOUT is independent per channel. SAOUT requires both AO1 and AO2 set to SAOUT.

Mode	Behavior
AOUT	Independent 0–10 V channel
SAOUT	Synchronized AO pair — **set both AO1 and AO2 to SAOUT**

Mixing AOUT on one channel and SAOUT on the other does not synchronize. In SAOUT, AO1 and AO2 change at the same moment.

EQXRA Configuration (protection)

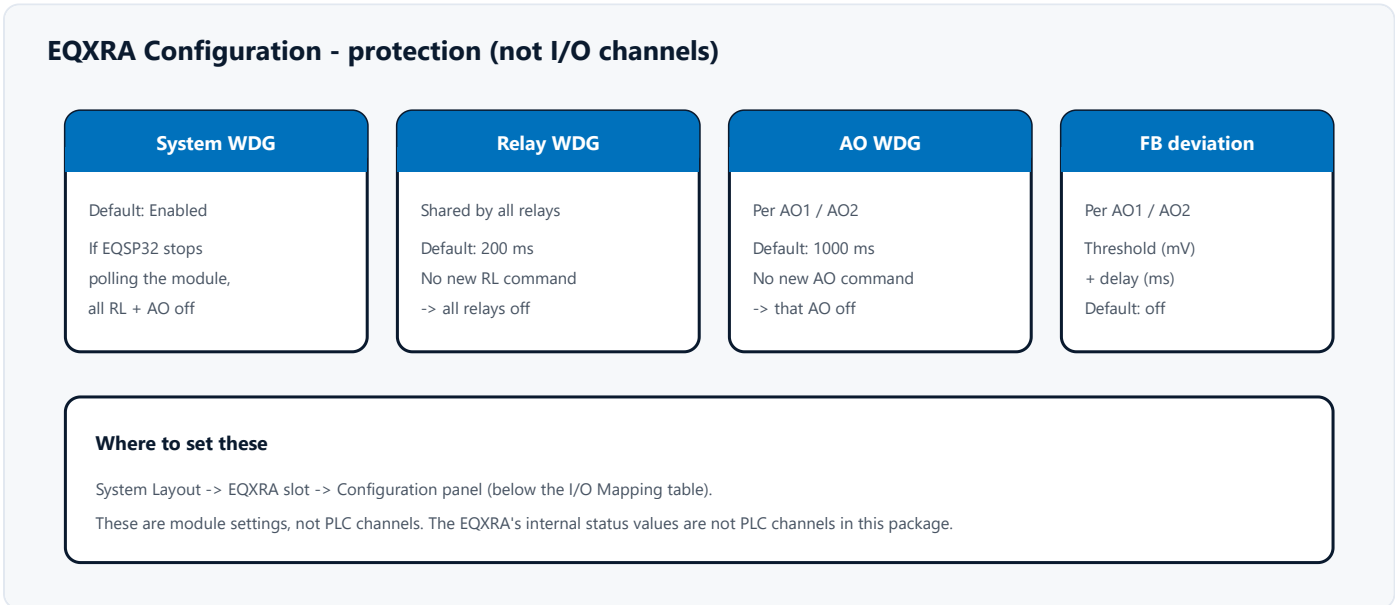


Figure 6: EQXRA watchdogs and feedback-deviation settings live in the Configuration panel, not as PLC channels.

Setting	Default	Meaning
System Watchdog	Enabled	If EQSP32 stops polling the module in time, all relays and AOs disable. Keep enabled.
Relay Watchdog Timeout	200 ms	Shared; no relay command within timeout → all relays off. `0` = disabled.
AO1 / AO2 Watchdog Timeout	1000 ms	Per AO; `0` = disabled.
AO1 / AO2 Deviation Threshold + Delay	Off (0 mV)	Fault when command vs feedback deviate too long.

8.4 EQXTC

Channels	Mode	PLC
X1.CH1–3, X2.CH4–6	TC	`%IW` °C × 10, signed → use `WORD_TO_INT`; sensor fault = −32768

8.5 EQXPT

Channels	Mode	PLC
X1.M1, X2.M2	PT100 (2/4-wire) or PT100 (3-wire)	`%IW` °C × 10, signed → use `WORD_TO_INT`; sensor fault = −32768

Pick wiring per channel with the Mode dropdown.

8.6 EQXPH

Channel	Mode	PLC
X1.PH1	PH	`%IW` pH × 100 (704 = 7.04)

8.7 EQXHI

Channel	PLC	Value
X1.IN1-5, X2.IN6-10	`%IX`	TRUE while voltage (AC or DC) is present on the input
FREQ1-10 (optional)	`%IW`	measured frequency in Hz × 10 (500 = 50.0 Hz); 0 for DC or no input

Channel	PLC	Value
QUAL1-10 (optional)	`%IW`	frequency quality: bits 0-2 the live code, bit 7 the delayed frequency error

PLC Channels in the Configuration section chooses which rows exist: *Inputs*, *Inputs + frequency*, or *Inputs + frequency + quality*. Choose only what the program needs; each frequency or quality channel uses one analog input address.

Quality codes (bits 0-2): 0 not available (no input), 1 acquiring, 2 bad (off target), 3 marginal, 4 good, 5 excellent, 6 not monitored. Bit 7 (value 128 added) is the delayed frequency error. Test the code with `(QUAL MOD 8) = 4` and the error with `QUAL ⩾ 128`.

Frequency monitoring is set per COM group (IN1-IN5 and IN6-IN10): Target Frequency (0 for DC, or 25 to 1000 Hz), Allowed Deviation (0 switches monitoring off) and Error Delay (0 to 60,000 ms).

9. Working with scaled values in the PLC program

Value scaling - what the PLC number means

Signal	IEC	Raw example	Physical meaning
AIN voltage	%IW	2500	2.500 V
CIN 4-20 mA	%IW	1200	12.00 mA
TIN / TC / PT temp	%IW	253	25.3 C (use WORD_TO_INT)
pH (EQXPH)	%IW	704	pH 7.04
POUT PWM duty	%QW	32767	~50% duty
EQXRA AO 0-10 V	%QW	32767	~5.000 V (0-65535 to 0-10 V)
Digital / relay	%IX / %QX	0 or 1	Off / On

Negative temperatures require WORD_TO_INT (or equivalent) because OpenPLC analog words are unsigned.

Figure 7: Common raw PLC values and their physical meaning.

Signal	Raw `%IW` / `%QW`	How to interpret
Voltage (AIN)	millivolts	e.g. 2500 = 2.500 V
Current (CIN)	mA × 100	e.g. 2000 = 20.00 mA
Temperature (TIN/TC/PT)	°C × 10, signed	Convert with `WORD_TO_INT` before using as INT
pH	× 100	e.g. 704 = 7.04
EQXHI frequency (FREQn)	Hz × 10	e.g. 500 = 50.0 Hz
PWM duty (`%QW`)	0–65535	Maps to 0–100%
EQXRA AO (`%QW`)	0–65535	Maps to 0–10000 mV (0–10 V)

Signed temperatures: OpenPLC analog words are unsigned. Negative °C values require `WORD_TO_INT` (or equivalent) in the program.

Temperature faults: TIN (open or short circuit), TC and PT sensor faults, and a temperature module that is missing, read **−32768** after `WORD_TO_INT` (−3276.8 °C, never a real reading). Test for it before you use the value.

10. Modbus and networking

Since OpenPLC Editor 4.3 the Modbus server belongs to the project, and the package supplies the EQSP32's ports. Three places:

1. **Device → Network** (package screen): switch on **Enable network**, and choose DHCP or a static address. The EQSP32 library runs the RJ45 Ethernet port and Wi-Fi and switches between them by itself: Ethernet works as soon as the cable is plugged in. For Wi-Fi, enter the network name and password, or leave them empty to use the Wi-Fi set up with the EQConnect app.
   - **A static address** set here is applied by a network reset: hold BOOT for 3 s (this also clears the Wi-Fi network). Otherwise set the address with the EQConnect app.
   - **Keep Ethernet and Wi-Fi both on**: no switching; both stay connected. A static address goes to Ethernet, and Wi-Fi uses DHCP. Modbus TCP and the debugger answer on both addresses. Use it, for example, for Modbus on a fixed Ethernet address with Wi-Fi for EQConnect.
   - **Disable Erqos IoT**: turns off BLE, the EQConnect app, EQMonitor (pin names, forcing) and Wi-Fi. The EQSP32 then works on Ethernet only, with an address from DHCP. The screen hides the settings that no longer apply.
   - **BLE advertising time**: how long the EQSP32 stays visible to EQConnect and EQMonitor after it starts (default 3 minutes, 0 = always). After that, press BOOT to make it visible again.
2. **Device → Serial** (package screen): the **Parity** of the RS232/RS485 port: None, Even or Odd (8 data bits, 1 stop bit). Many Modbus masters expect Even.
3. **Servers → Modbus server** (editor screen): choose **RTU**, **TCP** or both, the slave ID, and for RTU the **Serial Port** and baud rate:
   - `RS485_Port`: the EQSP32's RS485 terminals. Set the Modbus master to wait at least 10 ms between a reply and its next request (most masters wait longer by default).
   - `RS232_Port`: the RS232 terminals.

   The EQSP32 has **one** serial port, used as RS232 or as RS485: pick one. The **USB port belongs to the editor** (upload and debugger); a server set to `Serial` stops the build with a message.

Modbus TCP answers on port 502 (or the server's port) over Ethernet and Wi-Fi. Set the Modbus master's timeout to at least 2 s: a few requests per thousand take about 1 s on Ethernet and on Wi-Fi.

Register map (register numbers start at 0):

PLC address	Modbus	Registers
`%QX` outputs	Coils	`%QX0.0` = coil 0, `%QX0.1` = coil 1, … `%QX1.0` = coil 8
`%IX` inputs	Discrete inputs	`%IX0.0` = input 0, … (same numbering as the coils)
`%IW` inputs	Input registers	`%IW0` – `%IW31` = 0–31

PLC address	Modbus	Registers
`%QW` outputs	Holding registers	`%QW0` – `%QW31` = 0–31
`%MW` memory (`INT` , `UINT` , `WORD`)	Holding registers	`%MW0` – `%MW19` = 32–51
`%MD` memory (`DINT` , `UDINT` , `DWORD`)	Holding registers	`%MD0` – `%MD19` = 52–91, two registers each, high word first
`%ML` memory (`LINT` , `ULINT` , `LWORD`)	Holding registers	`%ML0` – `%ML19` = 92–171, four registers each, high word first

`%ID` (pulse totals and encoder positions, §6.2) is **not** in the map. Copy the value to a `%MD` variable in the program.

Debugger: always available over USB at 115200 baud; it needs no Modbus settings. With the network enabled the editor also offers the debugger over the network and asks once for the EQSP32's IP address (EQConnect shows it).

Projects made with package 0.7.x: editor 4.3 does not carry the old Modbus screen settings forward. Set up the Network screen and a Modbus server again.

BLE / EQConnect provisioning remains available whatever the network settings, unless **Disable Erqos IoT** is on.

11. Compile, upload, and first run

1. Finish System Layout and program logic.
2. Select the correct **COM port** for the EQSP32CE.
3. Build / upload from OpenPLC Editor.
4. First compile may take several minutes while core 2.0.17 and `RadioLib` 7.7.1 download. The EQSP32 library comes with the package and is not downloaded. On the first upload, Windows Firewall may ask about `mdns-discovery`. It only looks for network-connected boards; uploads over USB work if you deny it.
5. After upload, the PLC scan runs on the device; use EQConnect if you still need BLE setup.

If compile fails after an editor update, re-apply the trust patch (§3.2) and reinstall the `.vpp` if needed.

11.1 Example projects

The kit has one example per language in `examples/`. Copy a folder, open it with **Open** in OpenPLC Editor, pick the COM port, build and upload. Each `README.md` has the wiring and what to watch in the debugger and in EQMonitor.

Folder	Language	What it shows
`examples/ST/EQSP32_Conveyor`	Structured Text	Start/stop, a belt position from an encoder (QENC) and a home switch, a part counter (PCC), values for an HMI over Modbus
`examples/Ladder/EQSP32_Start_Stop`	Ladder Diagram	Start/stop with a set coil and a reset coil, a run lamp
`examples/FBD/EQSP32_Fan_Timer`	Function Block Diagram	A fan with an on-delay (TON) and an off-delay (TOF) timer
`examples/IL/EQSP32_Guard_Interlock`	Instruction List	A motor interlocked with a guard switch, a warning lamp

12. Limits and practical cabinet notes

OpenPLC's I/O limits (approximate):

- 56 digital inputs / 56 digital outputs
- 32 analog inputs / 32 analog outputs

The EQSP32 can address up to **15** EQX modules. Typical DIN-rail cabinets fit fewer modules beside the controller; stay within both the cabinet space and the I/O limits above.

Counters on the main unit: PCC terminals and QENC pairs share **4** hardware counters; the build stops with a message beyond four (§6.2).

13. Upgrading the package / missing channels

If channels disappear from the I/O Mapping table after a package upgrade (for example AO1/AO2 or ADIO1), the project usually still has an **old Mode string** that the new catalog no longer lists. The editor silently drops those rows.

Fix:

1. Close the project (or quit the editor).
2. Run:

   ```
   node tools/migrate-project-config.mjs "<path-to-your-project-folder>"
   ```

3. Reopen the project.

The tool writes `devices/configuration.json.bak` before changing anything.

14. Troubleshooting

Every build message of the package starts with `EQSP32:`. When a build stops, search the editor's console for `EQSP32:` to find the reason among the compiler output.

Symptom	Likely cause	What to do
Cannot install `.vpp` / signature error	Trust patch not applied	Close editor, run `patch-editor-trust.mjs`
EQSP32CE missing after editor update	The update removed the trust patch	Re-run the trust patch; reinstall the `.vpp` if needed
Channel missing from I/O Mapping	Stale Mode in project	Run migrate tool (§13)
Slot numbers start at 1	Display patch not applied	Cosmetic only; or re-run trust patch
Empty aliases	Display patch not applied	Enter aliases manually
Compile errors	Another ESP32 core version	The package needs 2.0.17: let the editor install it
Negative temperature looks huge	Treating `%IW` as unsigned	Use `WORD_TO_INT`
EQXRA AOs not syncing as expected	Only one AO set to SAOUT	Set both AO1 and AO2 to SAOUT (§8.3)
Compile stops with `bits/c++config.h: No such file or directory`	Arduino folder path too long	Run the trust patch again (it sets the short path `C:\EQSP32-OpenPLC`)
Build stops: `EQSP32: DIO9 is set to QENC A (Quad encoder signal A): set DIO10 to QENC B ...`	The second terminal of the encoder pair has another mode	Set it to `QENC B (Quad encoder signal B)` (§6.2)
Build stops: `EQSP32: more than 4 counters on DIO9-16 ...`	Too many PCC terminals and QENC pairs	Use at most four (§6.2)
Encoder position a few counts short after a pause	EQSP32 on USB power only	Power the EQSP32 through VIn
Windows Firewall asks about `mdns-discovery` on upload	Network board discovery	Deny is fine for USB uploads

Symptom	Likely cause	What to do
Build stops: `EQSP32: Modbus RTU cannot use the USB port ...`	The Modbus server's Serial Port is `Serial` (USB)	Set it to `RS485_Port` or `RS232_Port` (§10)
No Modbus register holds an encoder position or a pulse total	`%ID` is not in the Modbus map	Copy the value to a `%MD` variable (§6.2, §10)
Modbus TCP does not answer	Network off, or Modbus server without TCP	Device → Network: Enable network; Servers → Modbus: tick TCP (§10)
Modbus RTU answers garbage or nothing	Baud rate or parity differ from the master	Match the server's baud rate and the Serial screen's parity to the master
All inputs of a module read FALSE / 0 / −32768	Module missing or not answering	Check the module cable and the slot order (§5.2, §8)
Relays/AOs drop unexpectedly	Watchdog timeout	Increase timeout or keep commanding outputs each scan

15. What stays with EQConnect / the EQSP32 library

OpenPLC owns the PLC scan and mapped I/O. The EQSP32 library still owns:

- BLE / EQConnect provisioning
- Wi-Fi and Ethernet connection

Do not expect OpenPLC's System Layout to replace EQConnect for device provisioning.

16. Support scope for this package version (1.0.0)

Included:

- EQSP32CE main unit full Mode set via System Layout
- EQXIO, EQXAI, EQXRA, EQXTC, EQXPT, EQXPH, EQXHI
- EQXRA watchdog / feedback-deviation configuration
- Modbus TCP (Ethernet / Wi-Fi) and Modbus RTU (RS232 or RS485) through the editor 4.3 Modbus server
- EQSP32 library v1.6.0 bundled in the package
- Quadrature encoders (QENC, all formats) and the PCC running total (§6.2)
- Missing-module handling (inputs read FALSE / 0)

Not included / deferred:

- The EQXRA's internal status values as PLC channels
- Thermocouple type selection (library exposes a single TC mode)
- Listing in OpenPLC Editor's package catalog (this kit installs with the trust patch)
- CAN bus (no OpenPLC function blocks for it yet)

Notices

Erqos Technologies P.C. may change products, software, specifications and this document at any time without notice. Always use the latest revision, available at erqos.com.

These products and this document are intended for qualified professionals. The information in this document is provided as is. Erqos Technologies P.C. assumes no liability for its use. Customers are responsible for designing and testing their own applications, including the protection of connected equipment and persons.

For hardware details, see the EQSP32 and EQX module datasheets at erqos.com.

Revision history

Version	Revision	Date	Status	Changes
v1.0.0	Rev102226	22 Oct 2026	Released	First release, for the EQSP32 OpenPLC package 1.0.0 with the EQSP32 library v1.6.0.