Modbus-Wärmepumpe: Verhalten bei WAN-Ausfall

> Quelle: `Stations/Dokumentation/Anleitung_Modbus_WP_WAN_Ausfall.md` im onekey-Repo (Stand der Übernahme: 2026-07-26). Bei Codeänderungen dort zuerst aktualisieren.

Anleitung: Modbus-Wärmepumpe stockt nach WAN-/Internet-Ausfall

Technischer Leitfaden für Service/Entwicklung. Beschreibt Symptom, Ursache und die im Projekt umgesetzten Gegenmaßnahmen sowie deren Verifikation.

Symptom

- Energiemanager / Cloud-Funktion läuft, dann fällt das Internet aus (z. B. ISP-Störung).
- Danach **stockt die lokale Modbus-TCP-Kommunikation zur Wärmepumpe (ACond)**.
- Die Verbindung wirkt teils noch „OK", es fließen aber keine Daten mehr.
- **Nur ein vollständiger SPS-Neustart** stellt die Kommunikation wieder her.
- Modbus im Debug zu deaktivieren/aktivieren hilft **nicht**.

Ursache (zwei Ebenen)

1) Auslöser (WAN): Alle TCP-Verbindungen (lokaler Modbus **und** Cloud-MQTT/HTTP) teilen sich denselben OS-Socket-/Filedescriptor-Pool (unterliegendes Linux, kein Zugriff). Bei WAN-Ausfall versucht der MQTT-Client dauerhaft, den Cloud-Broker `mqtt.onekey.1sinq.de:8883` (TLS, per Hostname → DNS) zu erreichen. Über einen langen Ausfall summieren sich Reconnect-/DNS-/TLS-Versuche und zehren am gemeinsamen Socket-Pool.

2) Warum nur Neustart hilft (der eigentliche Defekt): In der Library-Klasse `_TCPIP_CLIENT` (`CyclicCall`) geht eine Verbindung bei fehlgeschlagener Socket-Allokierung (`OS_TCP_USER_SOCKET() < 0`, typisch bei erschöpftem Pool) nach `_STATE_ERROR`. Dieser Zustand war ein **toter Endzustand**: kein Retry, und das `DelConn`-Flag wurde nicht geprüft.

Konsequenz: Die Verbindung bleibt dauerhaft hängen. `DelConnection()` (= das, was ein Disable im Debug auslöst) wird ignoriert; bei `cMaxConnections = 1` schlägt ein erneutes `AddConnection()` mit `TCP_CLT_ERR_MAX_CONN` fehl. Der Zustand ist Member-Variable und überlebt jedes Klassen-Disable/Enable — nur ein voller SPS-Neustart (Reinit von Objekt + OS-Stack) räumt auf.

Umgesetzte Gegenmaßnahmen

Fix A — `_TCPIP_CLIENT` selbstheilend (Hauptfix)

Datei: `Stations/Terminal/LSL/Class/_TCPIP_CLIENT/_TCPIP_CLIENT.st` (markiert mit `// OneKey-Patch`)

- In `_STATE_MAIN_SOCK` wird beim Allokierungsfehler ein Zeitstempel gesetzt.
- `_STATE_ERROR` ist aus der fatalen Init-Fehlergruppe herausgelöst und **recoverbar**: berücksichtigt `DelConn` **und** versucht nach 5 s Backoff erneut (`→ _STATE_IDLE → _STATE_MAIN_SOCK`, fordert einen neuen Socket an).
- Echte Init-Fehler (`_STATE_ERROR_ALLOCATING_MEMORY` / `_CREATING_MUTEX` / `_CREATING_TASK`) bleiben absichtlich terminal.

Wirkung: Jede `_TCPIP_CLIENT`-Verbindung (inkl. lokaler Modbus-WP) erholt sich automatisch, sobald wieder Sockets frei werden — ohne SPS-Neustart. Disable/Enable wirkt wieder.

> Hinweis: `_TCPIP_CLIENT` ist Sigmatek-Library-Code und wird von **allen** TCP-Verbindungen genutzt. Der Patch ist minimal und markiert; bei einem Library-Update kann er überschrieben werden → Bug ist an Sigmatek gemeldet (siehe unten).

Fix B — Reconnect-Churn senken (Trigger entschärfen)

Datei: `MQTT/mqtt_conf/mqtt_client.xml`

- `Reconnection/max_interval` 30 → 120 s. Reduziert bei langem WAN-Ausfall die Anzahl der Reconnect-/DNS-/TLS-Versuche und damit den Druck auf den gemeinsamen Socket-Pool.
- Trade-off: Nach WAN-Rückkehr dauert der nächste MQTT-Versuch bis zu 120 s (lokale Regelung ist davon nicht betroffen).

- Der Broker bleibt per Hostname adressiert — die TLS-Zertifikatsprüfung benötigt den Hostnamen, eine feste IP ist daher keine Option.

> Fix B ist eine Entschärfung, keine Heilung: Der eigentliche Socket-/fd-Leak liegt in der Sigmatek-MQTT/TCP/SSL-Library + Linux-OS und ist von uns nicht behebbar.

Fix C — Connect-Timeout (Socket-Haltezeit begrenzen)

Datei: `Stations/Terminal/LSL/Class/_TCPIP_CLIENT/_TCPIP_CLIENT.st` (markiert mit `// OneKey-Patch`)

- `_STATE_CONNECT` hatte keinen oberen Timeout: ein unbeantwortetes SYN (typisch bei WAN-Ausfall) hielt den Socket bis zum OS-Default (~1–2 min). Jetzt wird die Connect-Startzeit gemerkt und ein hängender Connect nach `cConnectTimeout` (Default **20 s**) aktiv abgebrochen → der Socket wird sofort wieder frei (`_STATE_CLOSE_MAIN_SOCK` schließt ihn real).
- Wirkung: Bei WAN-Ausfall belegen Cloud-Reconnects den gemeinsamen Pool deutlich kürzer → geringerer Erschöpfungs-Druck. Greift den eigentlichen Socket-Fresser an der Wurzel an.

Fix D — Reboot-Eskalation als letzter Schutz

Datei: `Stations/Terminal/LSL/Class/AcondModbusWatchdog/AcondModbusWatchdog.st`

Da der fd-Leak in der Library selbst nicht behebbar ist, fängt der Watchdog den Endzustand ab — als feste Eskalationsleiter, ohne zu unterscheiden, ob ACond oder SPS schuld ist (von außen nicht erkennbar). Die Position auf der Leiter überlebt den SPS-Reboot über den retentiven `RebootCounter`:

1. **Stufe 1:** Kommunikation tot > `ComTimeout` (Default 5 min) → HTTP-Reset der ACond (wie bisher).
2. **Stufe 2:** Bleibt die Kommunikation trotz ACond-Reset > `RebootTimeout` (Default 2 min) tot → **sauberer SPS-Reboot** über `255::DiagnoseClass1.Reboot` (OS-Reboot, **nicht** der CMD_RESET-Pfad, der die ET-0710 crasht). Ein Reboot re-initialisiert den OS-Socket-Stack und behebt damit den Sigmatek-seitigen Fall.
3. **Stufe 3:** Ist die Kommunikation nach dem SPS-Reboot weiterhin > `ComTimeout` tot → nochmals ACond-Reset und **sofort danach** ein zweiter SPS-Reboot (beide Steuerungen starten gemeinsam frisch) — ohne Verifikationswartezeit und ohne Karenz.
4. **Schluss:** Bleibt die Kommunikation auch danach tot → keine weiteren Aktionen, nur noch Dauer-Alarm (`Status` = 4).

Schleifenschutz gegen Dauerfehler: `RebootCounter` (retentiv) zählt die Reboots; ab `MaxReboots` (Default 2) wird **nicht** mehr rebootet (= Schluss-Stufe). `MinRebootInterval` (Default 1 h) wirkt als Mindest-Betriebszeit vor dem **ersten** Reboot: Die SPS rebootet nicht, solange sie selbst erst kürzer als diese Zeit läuft (Flatter-Schutz; der ACond-Reset der Stufe 1 kommt trotzdem sofort). Sobald die Kommunikation wieder gesund ist, werden alle Zähler zurückgesetzt — die Leiter läuft also nur bei durchgehendem Fehler bis zum Ende durch.

> Konfigurierbar (Server-Channels): `RebootTimeout`, `MinRebootInterval`, `MaxReboots`. Beobachtbar: `RebootCounter` (extern/Visu lesbar) sowie `Status` im Online-Debug (0=OK, 1=ACond-Reset, 2=Reboot-Prüfung, 3=Reboot ausgelöst, 4=Alarm). Hinweis: `Status` ist `WriteProtected` und erhält im aktuellen Codegen keinen externen Lese-Pointer — für eine Visu-Anzeige müsste der Channel entsprechend angepasst werden (vorbestehender Effekt, nicht reboot-relevant).

Fix E — Enable-Sackgasse in der `ACond_Heatpump`-Statemachine

Datei: `Stations/Terminal/LSL/Class/ACond_Heatpump/ACond_Heatpump.st`

Am Gerät beobachteter Klemm-Zustand, den auch die Watchdog-Eskalation (Stufe 1, ACond-Reset) nicht heilen konnte: `EnableReadRegister`/`EnableWriteRegister` = 0 bei `ModbusOnline` = 1, `HPOnline` = 0, **kein** Kommunikationsalarm, eingefrorene Istwerte. Die Modbus-Function-Objekte (`ACond_ReadInputRegisters`/`ACond_WriteHoldingRegister`) waren dauerhaft disabled — nur ein SPS-Reboot half (die Enables starten mit DefValue = 1 neu).

Ursache (vier zusammenwirkende Lücken):

1. Der `WPActive = 0`-Skip in `CyWork` nullte die Enables, ließ aber die Statemachine in ihrem letzten Zustand eingefroren stehen. Da die Enables **nur** in der Init-Leiter (`Heatpump_Init`) wieder gesetzt werden, blieben sie nach einer Re-Aktivierung (`WPActive` 0 → 1, z. B. Boot-Race, Config-Übernahme, kurzes Modul-Deaktivieren) dauerhaft 0, wenn die FSM bereits hinter Init stand.
2. `_HeatpumpInit_SetInitData` hatte **keinen Timeout**: Traf Lücke 1 mitten in der Init-Leiter, wartete der Zustand ewig auf `SetHPMode(Off)` — das ohne Write-Enable nie fertig wird.
3. In `_HeatpumpMain_Init` unterdrückt `GetHPErrors` Alarm **und** Recovery komplett — die Sackgasse war dadurch unsichtbar (kein `Alarm_HPCommunication`) und unerreichbar für `Reset_ComError`.
4. Die Error-Recovery verlangte `HPOnline = FALSE`; mit stale `HPOnline = TRUE` wurde `Reset_ComError` jeden Zyklus gelöscht, bevor er konsumiert wurde (zweite, latente Sackgasse).

Umgesetzte Korrekturen:

- `WPActive = 0`-Skip pinnt die FSM jetzt auf `_HeatpumpMain_Init`/`_w4ModbusOnline` (inkl. frischer Timeout-Basis, `HPOnline := FALSE`) → nach jeder Re-Aktivierung läuft garantiert die volle Init-Leiter inkl. Enable-Setzen.
- `_HeatpumpInit_SetInitData` bekam einen 120-s-Timeout (→ Error → reguläre Recovery) und assertet die Enables jeden Zyklus.
- `_HeatpumpMain_Active` assertet die Enables jeden Zyklus (Selbstheilung; „Active ⇒ Enables an" gilt jetzt per Konstruktion).
- Error-Recovery in `GetHPErrors` ist von `HPOnline` entkoppelt und stampt die Timeout-Basis neu.

Wirkung: Der Zustand „Komm tot, alles sieht online aus, nur Reboot hilft" kann strukturell nicht mehr entstehen; die Watchdog-Stufe 1 (ACond-Reset) ist wieder in allen Fällen wirksam, weil jeder TCP-Abriss zuverlässig in die Recovery-Schleife führt.

Bug-Report an Sigmatek

Der Library-Defekt ist dokumentiert in `Stations/Dokumentation/Bugreport_TCPIP_CLIENT_STATE_ERROR.md` (inkl. Code-Stellen, Reproduktion und Korrekturvorschlag). Dieser sollte an Sigmatek übermittelt werden, damit der Fix dauerhaft in die Library einfließt.

Verifikation am Gerät (ohne OS-Zugriff, via LASAL-Debug)

1. WAN-Ausfall simulieren (Internet trennen), Wärmepumpe lokal weiterlaufen lassen.
2. Im Debug am Modbus-`_TCPIP_CLIENT` beobachten:
   - `pActConn^.FSM_TCP` — tritt `_STATE_ERROR` auf?
   - `LastError` — Hinweis auf Socket-/`MAX_CONN`-Fehler?
3. WAN wiederherstellen. **Erwartung mit Fix A:** Die Verbindung löst sich nach ≤ 5 s Backoff selbst und nimmt die Modbus-Kommunikation wieder auf — **ohne SPS-Neustart**.
4. `ComDiagnosis_MQTT` zur Beobachtung der MQTT-Reconnect-Aktivität.

Hinweis zum Stale-`IsConnected`-Fall

Der `AcondModbusWatchdog` erkennt den Komm-Verlust nicht über `IsOnline`/`IsConnected` allein, sondern über den **Datenfluss** (Modbus-Callback-Zähler bewegt sich + `cIsOnline`). Damit ist auch der Stale-Fall abgedeckt, in dem `IsConnected` fälschlich TRUE bleibt (Socket halb-offen), aber keine Antworten mehr kommen: Bleibt der Callback-Zähler stehen, greift die Eskalation (Tier 1 → Tier 2). Am Gerät verifizieren, dass der Callback-Zähler im realen Fehlerfall tatsächlich stehen bleibt.

Revision #1
Created 2026-07-26 18:33:46 UTC by Christian
Updated 2026-07-26 18:33:46 UTC by Christian