ZT-100-C Water Cut Meter — Technical Manual

Technical Manual - Zelentech

Manufacturer: Zelentech
Product: ZT-100-C Digital Water Cut Meter
Audience: service engineers, commissioning engineers, factory test and production
Firmware covered: CAP card `0.16.37-cap-b38` · SIGNAL card `0.17.35-sig-c59`
Document: MN-ZT100C-200, Issue 1.0, August 2026

Contents

About This Manual

This is the service and factory reference for the Zelentech ZT-100-C. It assumes the ZT-100-C User Manual and goes beyond it: per-card architecture, the complete CLI surface of both cards, every configuration parameter, every calibration flow, the HART command set as actually implemented, the inter-card link, the record store, forensics, and factory procedures.

Scope and firmware identity

This manual describes exactly two firmware builds:

Card	Product string	Firmware version	MCU
CAP	`zt-100c-cap`	`0.16.37-cap-b38`	SAMD51
SIGNAL	`zt-100c-signal`	`0.17.35-sig-c59`	SAMD51

Version strings are bumped **per card, independently**. Always record both when reporting a problem:

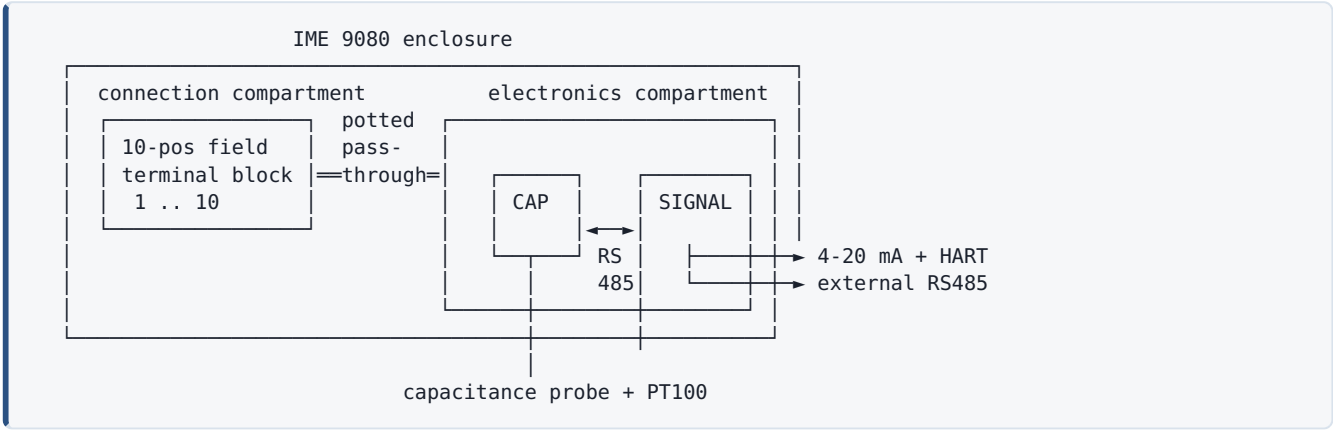
```
sys version            # SIGNAL
cap sys version        # CAP, through the proxy
```

Ground rules for this document

1. **Nothing here is estimated.** Every behavioural statement is taken from the verified fact base for these two builds. Where a figure is conventionally expected but not established, it is written **[TBC]**.
2. **Modbus is implemented on SIGNAL** — an RTU slave at address 1 on the external RS485 port, behind the same portdetect gate as the console. It is documented here as built, **including the registers that are absent on this card**, which are the ones holding CAP-owned data. The authoritative register map is printed by the card itself with `mb map`.
3. **The DISPLAY card does not exist.** See DISPLAY Card — Not Released.
4. Terminal numbering is the enclosure's own **0–9** scheme, printed on the wiring-diagram label in the connection compartment. The map is in Enclosure, Terminals and Wiring. The backplane strip at the other end of the harness is identified by **wire colour** and carries no position numbers.
5. Figures are placeholders. Where a usable photograph exists in the previous document set it is named by path.

System Architecture

Two cards, one enclosure, one link



	CAP	SIGNAL
Role	Measurement	Output
Composed units	`con_usb`, `meas`, `nvm`, `monitor`, `link485`	`con_usb`, `con_485`, `link485`, `linkmirror`, `aout`, `hart`, `nvm`, `monitor`
Sensors	Capacitance bridge, PT100, board temp, MCU die temp	MCU die temp only
Analog output	**None, by design**	4-20 mA
HART	None	HART slave, declares revision **6** (7 at the factory)
Consoles	USB only	USB + external RS485
Registered commands	36 (+7 built-ins)	23 (+7 built-ins)

Data flow

1. **CAP** runs a continuous measurement cycle: probe channel, reference channel, compute, publish. Every published value carries a quality.
2. **CAP** broadcasts a 38-byte enclosure heartbeat at **1 Hz** on the internal RS485 link, carrying water cut, capacitance, dielectric constant, process temperature, board temperature, all five qualities, a configuration generation counter and uptime.
3. **SIGNAL** mirrors the payload into its own store, **preserving the source quality byte** — SIGNAL never invents a quality.
4. **SIGNAL**'s `aout` unit maps the mirrored water cut onto 4-20 mA at the configured span; its `hart` unit serves the same values to a HART master.
5. If no fresh heartbeat arrives within the mirror TTL (**2500 ms**), every mirrored variable goes `Q_STALE` and the normal quality machinery takes over: the loop goes to the 3.6 mA alarm and HART raises PV-out-of-limits. Recovery is automatic.

Design principles that show up everywhere

Principle	Consequence you will observe
Honest data	Every value carries a quality. Absence and fault are surfaced. A stale number is never presented as live

Principle	Consequence you will observe
No blocking	Every driver is a start/poll/result state machine; every poll returns in microseconds. A blocking call would show up as a scheduler stall
No dynamic allocation	All memory is static and composition-owned
Atomic-or-fail persistence	A calibration commit either lands completely or changes nothing; the previous record survives
Signal, don't freeze	A dead source drives the alarm current, not the last good value

Firmware architecture (orientation)

The firmware is composed from a portable C11 core:

- **libs** — pure algorithms: the water-cut maths (`wcut`), the command language, the link protocol, the HART protocol, the port classifier.
- **drivers** — one non-blocking state machine per IC (ADS1115, MAX31865, MCP9808, AD5420, AD5700).
- **units** — composable application blocks: `meas`, `aout`, `hart`, `monitor`, `con_485`, `link485`, `linkmirror`, `nvm`, `con_usb`.
- **services** — the link engine, HART engine, scheduler, command dispatcher, configuration, record store and forensics.
- **platform** — the SAMD51 hardware abstraction.

`unit list` on either card prints the composed units and their state.

Enclosure, Terminals and Wiring

Mechanical arrangement

The ZT-100-C is housed in an **IME 9080 enclosure with a separate connection compartment**. Field conductors land on the terminal block in the connection compartment and are **potted through into the electronics compartment**, where the internal harness reaches the CAP and SIGNAL cards.

The pass-through is **potted, not a connector**. Field wiring work is confined to the connection compartment. Rework on the electronics side of the potting is a workshop operation.

Field terminal map — the enclosure block

The numbering is the enclosure's own **0–9** scheme, moulded into the wiring-diagram label in the connection compartment, so the manual and the thing the installer is looking at agree.

Terminal	Function	Wire colour	Field use
0	4-20 mA IN (+) — feeds the display	Red	Not used in this revision
1	4-20 mA IN (−) — feeds the display	Black	Not used in this revision
2	Internal RS485 **A**	Grey	Factory wired — do not connect
3	Internal RS485 **B**	Purple	Factory wired — do not connect
4	4-20 mA (−)	Blue	Process output
5	4-20 mA (+)	Green	Process output
6	External RS485 **B**	Orange	Operator console
7	External RS485 **A**	Yellow	Operator console
8	24 VDC (−)	White	Supply
9	24 VDC (+)	Brown	Supply

The 10-position field terminal block, looking into the connection compartment. The wiring-diagram label in the centre carries the 0–9 position numbering used throughout this manual.

Backplane terminal strip — the internal block

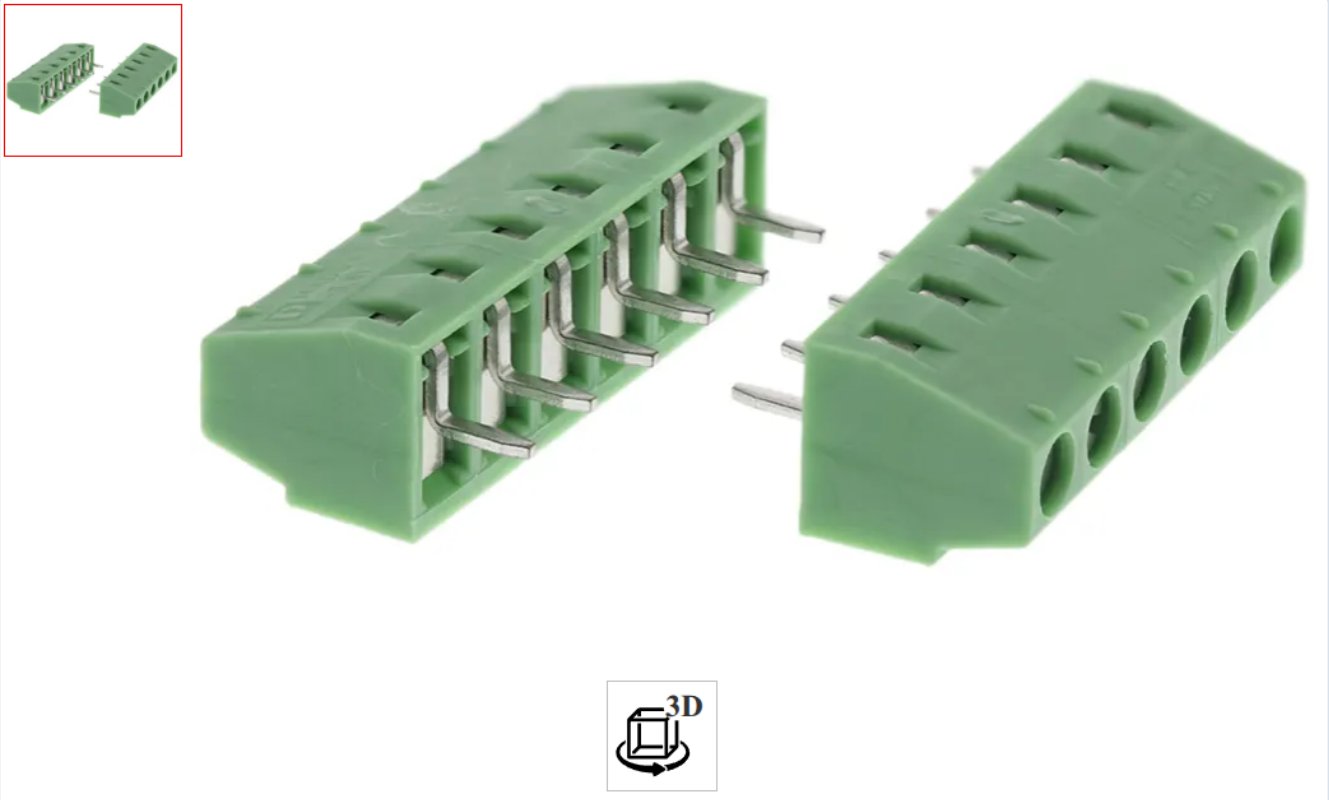
The other end of the potted harness lands on the backplane's own strip. **It is identified by wire colour, not by position number** — the pads carry function and colour beside them and no numbering, which is deliberate.

Wire colour	Function
Red	4-20 mA IN (+)
Black	4-20 mA IN (−)
Grey	Internal RS485 **A**
Purple	Internal RS485 **B**

Wire colour	Function
Blue	4-20 mA (−)
Green	4-20 mA (+)
Orange	External RS485 **B**
Yellow	External RS485 **A**
Brown	24 VDC **(+)**
White	24 VDC **(−)**

Land by colour. This is a factory operation — the strip is not a field wiring point and carries no installer guidance.

> **First-batch boards carry a correction sticker.** Two adjacent pairs at the top of the strip — the 24 VDC pair and the external RS485 pair — are printed the wrong way round on that batch. **The sticker is the authority on those boards; land the harness as the sticker shows.** Boards from the following revision are printed correctly and carry no sticker.



Terminal block, mechanical detail.

⚠ RS485 A/B polarity

A/B polarity is the most common wiring failure on this product.

The A/B lettering convention is **not standardised between manufacturers**. The same physical conductor may be labelled A by one vendor and B by another. There is no reliable way to resolve this from the labels.

- **If RS485 is silent or garbled, swap A and B.** It is safe, it is the only polarity choice on a differential pair, and it is the fix in the overwhelming majority of cases.

- **Do not instruct anyone to match by cable colour.** The colours in the table describe *this enclosure's* harness only. They have no relationship to any adapter cable, field cable or third-party device.
- Land by **terminal number**, verify by **communicating**, then swap if it does not work.

Internal interconnect

CAP and SIGNAL are joined by a half-duplex RS485 bus wired **A to A, B to B** between the cards, brought out to field terminals 2 (A) and 3 (B). The bus runs at a fixed **115200 baud** and carries both the 1 Hz measurement heartbeat and the remote-execution proxy traffic.

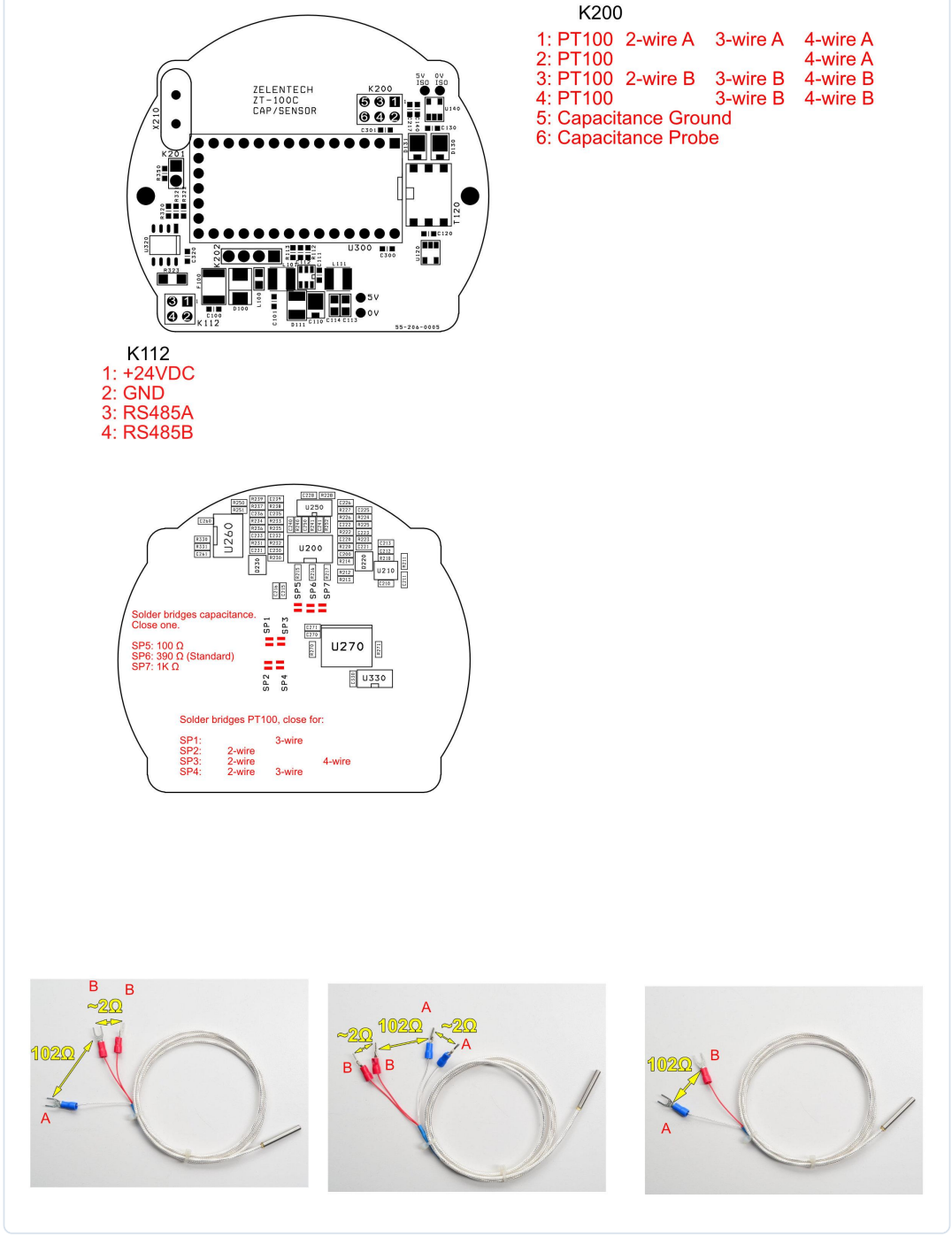
The internal interconnect is a stacked bus, not a discrete harness. All field wiring lands on a **backplane card** (`K245-0230-01`); CAP and SIGNAL piggyback onto it as a modular assembly. The backplane is pure interconnect — no active components, no regulators, no passives at all.

Path	Connector	Carries
Backplane → stack	**K112**, 2×2, 2.54 mm	24 V, GND, internal RS485 A/B
Backplane → CAP	**K200**, 2×3, 2.54 mm	PT100 × 4, probe screen, probe centre conductor
Backplane → cable	**K3**, 4-way JST PH	4-20 mA output pair, external RS485 pair
Backplane → cable	**K4**, 2-way JST PH	4-20 mA input pair

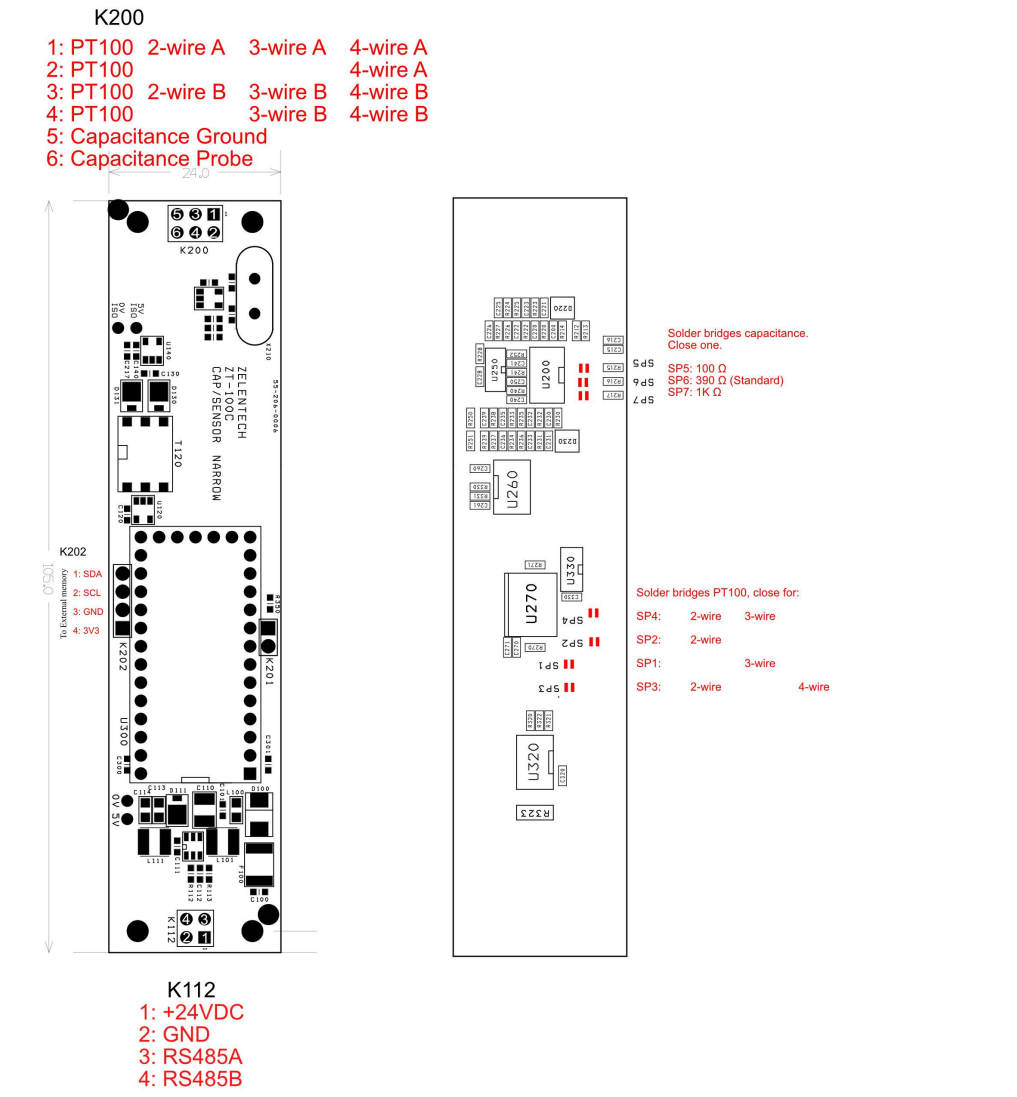
24 V enters at field terminals 9 (+) and 8 (−) and leaves the backplane at exactly one place — K112 — with no branch. **Both cards are therefore fed from a single rail**, each regulating its own internal voltages from it.

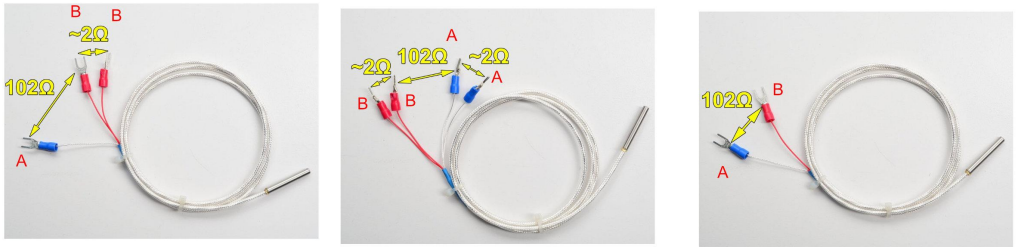
Neither CAP nor SIGNAL carries a field-wireable terminal. Everything they need arrives through the plug-in stack connectors, which is what makes card replacement a wiring-free operation: unplug, plug back in, nothing to strip or re-terminate.

> **The stack connectors are not polarity-keyed.** They are plain unshrouded socket strips; the polarising option was not fitted. A 2×2 or 2×3 connector can be mated rotated 180°, which would reverse the supply or transpose probe pairs. Protection rests on position count (4-way cannot be swapped with 6-way) and the **square pin-1 pad**. **Check pin 1 against the silkscreen before pressing a card home.**

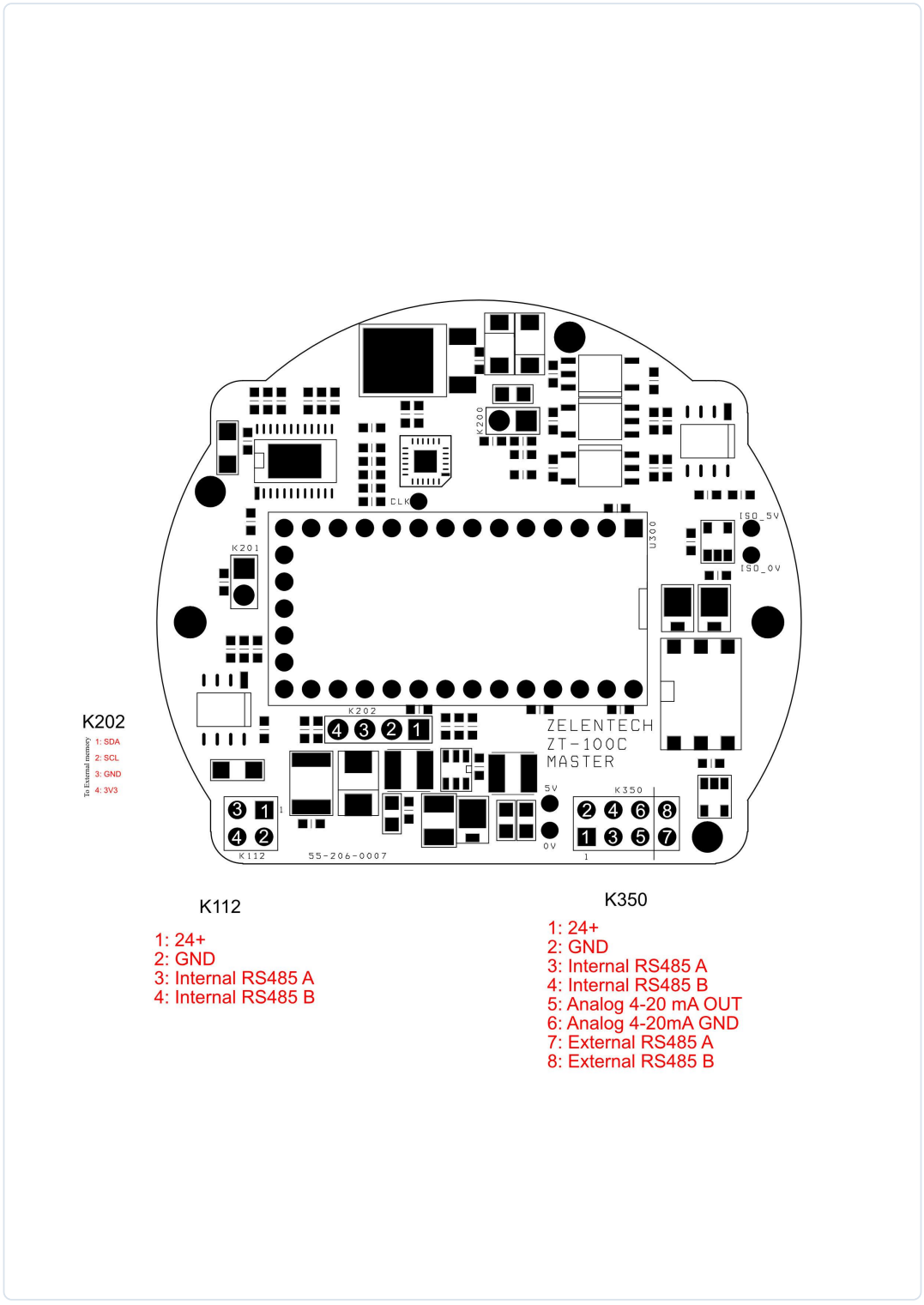


CAP card connection drawing, round card (ZT-100C CAP/SENSOR). Connector functions are as annotated on the drawing itself; the physical form of the interconnect on the production units remains TBC as noted above.





CAP card connection drawing, narrow card (ZT-100C CAP/SENSOR NARROW).



SIGNAL card connection drawing, round card (silkscreen: ZT-100C MASTER).

Probe and PT100 landing

The capacitance probe and the PT100 land on the **backplane card**, on a 6-position screw terminal (**Phoenix MKDS 1/6-3,81**), and reach CAP through K200 with no conditioning of any kind — no dividers, no filters, no guard drive, no mode selection. Straight copper, 1:1.

Backplane position	Silkscreen	CAP K200 pin
1	`PT100A1`	1
2	`PT100A2`	2

Backplane position	Silkscreen	CAP K200 pin
3	`PT100B1`	3
4	`PT100B2`	4
5	`CAP GND` — probe screen	5
6	`CAP PROBE` — probe centre conductor	6

PT100 wire count uses positions **A1 + B1** for 2-wire, **A1 + B1 + B2** for 3-wire, and all four for 4-wire. The selection itself is made on the CAP card by solder bridges **SP1 – SP4** — it is a build-time decision, not a switch. **CAP has no DIP switch.**

`CAP GND` is the probe screen return and is a probe-side net only: it does not appear on the field terminal block and is not tied to the supply negative anywhere on the backplane.

The probe and its screened cable are supplied as a **matched assembly** and are not cut or extended in the field, so no maximum cable length is specified. Cable capacitance sits in parallel with the probe and is absorbed by the card's calibration for the assembly it was supplied with — which is why the cable must not be altered.

Wiring specification gaps

Two different terminal blocks exist in this product, and only one of them is ever touched in the field.

	Where	Who wires it	Conductor limit
Field terminal block, 10 positions	Enclosure connection compartment	**Installer**	**0.82 mm² (AWG 18)** max, per the enclosure specification
Backplane terminal blocks	Electronics compartment	**Factory only**	See below

The installer's limit comes from the **enclosure**, not from the backplane. The backplane's own blocks are factory wiring and their capability is not an installation figure — do not publish it as one, and do not let it reach an installer as guidance.

Item	Established
Backplane field block	**Phoenix MKDS 1/10-3,81**, screw clamp, 3.81 mm pitch, 13.5 A / 200 V nominal (UL 10 A / 300 V)
Backplane conductor capability	0.14 – 1.5 mm² (AWG 26 – 16); 0.25 – 0.5 mm² with a ferrule. **Factory reference only**
Stripping length	5 mm
Backplane probe block	**Phoenix MKDS 1/6-3,81**, same connection data
RS485 termination	**120 Ω fitted on both ports** on the SIGNAL card (soldered, not selectable). No termination on the backplane
RS485 biasing	**None**, either port
Earth / shield landing point	**None exists.** The backplane has no earth stud, lug or pad, and its two mounting holes are non-plated with no annular ring — a screw through them touches no copper
RS485 signal common	**None is brought out.** The 10-position block has no ground position, and SIGNAL's isolated RS485 common reaches only test pad K141
Cable gland sizes	From the enclosure data sheet — not published here pending Ex d certification

Item	Established
Maximum RS485 bus length	Not specified. It is set by baud rate, cable type and cross-section, and by the installation, not by the meter
Terminal torque	Not published — see below

Torque is deliberately not published. The block is an M2 screw and the figure is available from the manufacturer if a site procedure demands it; publishing it invites over-tightening against a specification this document does not control.

Consequences of the two "none" rows above. A cable shield cannot be landed inside the ZT-100-C — it must be earthed at the control-system end. The external RS485 console is a **two-wire connection**, with no shared reference between the ends; it works, and it is how the product is used, but a long run between separated grounds is the case that will fail first.

CAP Card

Product string `zt-100c-cap` · **Firmware** `0.16.37-cap-b38` · **MCU** SAMD51

CAP is the measurement half of the enclosure. It owns the probe, the temperature sensors, the water-cut computation and **all calibration data**.

Composition

Units: `con_usb`, `meas`, `nvm`, `monitor`, `link485` — five units.

What CAP has

Function	Detail
Capacitance front end	ADS1115 16-bit I²C ADC at address `0x48`, single-shot, 128 SPS, gain setting `ADS_GAIN_4`. Channel 0 = probe, channel 1 = reference, both single-ended. 4 raw conversions averaged per channel per cycle
Process temperature	MAX31865 with a **3-wire PT100**, `rtd_nominal` 100.0 Ω, **reference resistor 430 Ω**, chip select on Arduino pin 2. Hardware SPI at 1 MHz, SPI mode 1, 60 Hz filter
Board temperature	MCP9808 at I²C `0x18`, read every 1000 ms
MCU die temperature	Published by `monitor` every 10 s
Internal RS485 link	Serial1 / SERCOM3, port 0, DE on Arduino pin 18 (PA04)
Console	**USB only**
Record store	2 MB GD25Q16 QSPI flash, 12 slots × 8 KB

What CAP does not have — by design

- **No analog output of any kind.** No 4-20 mA, no voltage output. The span lives on SIGNAL.
- **No HART.** No modem.
- **No console on RS485.** Its RS485 port is the link, not a terminal. CAP's only local console is USB.
- **No DIP switches and no jumpers.** The card is configured entirely by `cfg` keys and calibration records.
- No buttons.

> **The RTD reference resistor is 430 Ω on CAP**, and 430 Ω is also the correct value on the single-card ET-25001. Some shared driver documentation still presents **1460 Ω** as universal — it is not, and it is not either product's value. It is the earlier INV-era card's reference. Never carry it onto a CAP card.

CAP pin reference

CAP's pins are declared inline in its composition rather than in a board header. The Arduino number is authoritative — it is what the firmware writes.

Function	Arduino pin	Port
MAX31865 chip select	2	**PA07** — read from the CAP production files (`hardware_reports/hw_cap.md`), not assumed from the ET-25001. The two boards do agree, but that is now established rather than inferred
Internal RS485 DE	18	PA04
Internal RS485 TX / RX	D1 / D0 (Serial1, SERCOM3)	—
RGB status LED data / clock	D8 (PB03) / D6 (PB02)	PB03 / PB02
Heartbeat LED	D13	`LED_BUILTIN`
ADS1115 address strap	ADDR to GND → `0x48`	—
MCP9808 address strap	A2..A0 to GND → `0x18`	—

The SAM port designator for the MAX31865 chip select is not stated anywhere in the firmware. It was read from the CAP production files and is **PA07** — see the pin table above.

Measurement cycle

One small step per scheduler poll, never blocking:

```
PROBE_START → PROBE_WAIT ×4 → REF_START → REF_WAIT ×4
            → compute water cut → publish → repeat
```

The RTD one-shot runs on a **500 ms** cadence in the background of the ADC phase machine; the board temperature on **1000 ms**. The `meas` unit declares a **250 ms** progress deadline — the tightest on the card.

Heartbeat payload (enclosure v2)

CAP broadcasts this 34-byte payload at 1 Hz. Little-endian.

Bytes	Field	Type
0	version = `0x02`	u8
1..4	water cut, %	f32
5..8	capacitance, pF	f32
9..12	dielectric constant	f32
13..16	process temperature, °C	f32
17..20	board temperature, °C (CAP)	f32
21..25	qualities of the five values above, same order	u8 × 5
26..29	configuration generation	u32
30..33	uptime, s	u32

The layout is versioned: fields are never changed in place — the version byte is bumped and new fields are appended. **v2 is the only version this firmware knows**, in both directions.

The two cards are not reflashed simultaneously in the field, so a version bump that bricked a half-upgraded pair would be a worse defect than the feature is worth.

An **unknown** version byte, or a payload shorter than its own version's size, still causes the receiver to treat CAP's data as **stale**.

Remote execution

CAP answers command lines forwarded from SIGNAL over the link. **The forwarded session starts at operator tier and must authenticate for itself** (firmware 0.16.37-cap-b38):

```
cap auth commission 1234
cap fcal zero
```

The session is long-lived, so one `cap auth` covers the commands that follow it.

> **Why this changed.** CAP used to raise forwarded sessions to commission automatically, on the argument that physical access to the internal bus is the authorization. That argument was retired for the ET-25001's front panel on the same day, and it was weaker here: `cap` is an **operator** command on SIGNAL, so anyone with the SIGNAL console — no code, no enclosure opened — could run commission-tier work on CAP. Proximity to a bus is not the same as authority over what is on it.

See Reaching CAP through SIGNAL.

SIGNAL Card

Product string `zt-100c-signal` · **Firmware** `0.17.35-sig-c59` · **MCU** SAMD51

SIGNAL is the output half. It measures nothing.

Composition

Units: `con_usb`, `con_485`, `link485`, `linkmirror`, `aout`, `hart`, `nvm`, `monitor` — eight units.

What SIGNAL has

Function	Detail
4-20 mA output	AD5420, SPI mode 0 at 3 MHz, MSB first, **no chip select**; 24-bit frames committed by a microsecond LATCH pulse. REXT not fitted → internal RSET
HART slave	AD5700 modem on SERCOM4, 1200 baud 8-O-1
Internal RS485 link	Serial1 / SERCOM3, port 0, DE on D4 (PA14), fixed 115200
External RS485 console	SERCOM0, port 1, TX D18 (PA04), RX D15 (PA05), DE D3 (PB22), `rs485.baud` default 115200
Link mirror	Consumes CAP's heartbeats into its own store with a 2500 ms staleness TTL
Consoles	USB (factory setup) + external RS485 (field)
`cap` proxy	Forwards command lines to CAP over the link

What SIGNAL does not have — by design

- **No measurement front end.** No probe, no PT100, no board temperature sensor. There is **no `meas read` command on SIGNAL** — every measured value it holds is mirrored from CAP.
- **No voltage output.**
- **No DIP switches and no jumpers.** The 4-20 mA span is set by the `pv.range` configuration key, not by a range DIP. Any earlier document describing a range DIP on this card is wrong.
- **No probe, capcal or medium data of its own** — the Modbus registers and status flags that would carry them are absent rather than mirrored, and the action opcodes that would write them are refused. See Modbus.
- No buttons.
- **No external temperature sensors** — process temperature arrives from CAP over the link.

SIGNAL pin map

SIGNAL's board header is the compliant pin-truth reference for the product family: it is derived from the production build data and an oscilloscope-verified pin document.

Function	Arduino pin	Port	Notes
AD5420 LATCH	D2	PA07	
AD5420 CLEAR	D5	PA15	
AD5420 FAULT	D7	PA18	Input; loop-state sense
AD5420 REXT	—	—	**Not fitted** → internal RSET
Internal RS485 TX / RX	D1 / D0	—	Serial1 / SERCOM3, link port 0

Function	Arduino pin	Port	Notes
Internal RS485 DE	D4	PA14	
External RS485 TX	D18	PA04	SERCOM0 / PAD[0], port 1
External RS485 RX	D15	PA05	SERCOM0 / PAD[1]
External RS485 DE	D3	PB22	
HART TX	D16	PB08	SERCOM4 / PAD[0], scope-verified
HART RX	D17	PB09	SERCOM4 / PAD[1]
HART RESET	D9	—	Active low; released HIGH
HART nRTS	D10	—	Active low — carrier keyed when LOW
HART CD	D11	—	Active-high carrier detect, input
HART DUPLEX	D12	—	Driven LOW at init
RGB status LED data / clock	D8 / D6	PB03 / PB02	
Heartbeat LED	D13	—	`LED_BUILTIN`

External port traffic classifier

SIGNAL's external RS485 port (port 1) is the only port with a traffic classifier. It exists so that a foreign master on a shared bus cannot corrupt a console session.

Concept	Behaviour
Burst	Bytes separated by ≥ **4 ms** of line silence
`TEXT`	≥ 90 % printable ASCII/CR/LF/BS, or a burst ending in CR/LF that parses as a text line
`MODBUS`	4–256 bytes whose last two bytes are a valid Modbus CRC16 (poly `0xA001`) over the preceding bytes **and** whose function code is in the legal set
`UNKNOWN`	Neither — counted and dropped
Latch	A verdict latches the port personality for **10 s**, refreshed by every same-class burst, so a CLI session does not flap on a binary-looking paste

Only `TEXT` bytes reach the command session. `UNKNOWN` bursts are counted and dropped. **`MODBUS` bursts are routed to the Modbus slave** rather than discarded, which is what lets one pair of wires carry a console and a Modbus master without a mode switch.

The verdict names above are the ones the card itself prints. Counters are visible with `sys rs485`, and reading them is the fastest way to identify another device on a shared pair.

RS485 electrical and timing behaviour

- Half duplex; the platform bridge owns the DE pin per port. DE asserts before the first byte and drops after a computed drain window: 10 bits per 8N1 byte (`10000/baud` ms per byte, plus 1 ms slack). **No blocking and no `flush()`.**
- **Echo suppression:** while DE is high the transceiver loops TX back onto RX. Those bytes are discarded and counted as `echo_drops`.
- **No local echo of typed characters** on either console. On a half-duplex pair the operator sees their typing only with terminal-side local echo.
- Reply back-pressure: bytes the TX ring refuses are held in a 512-byte pending tail and pushed from poll. A reply exceeding ring plus tail drops its remainder, counted by `tx_dropped`.

DISPLAY Card — Not Released

> **The ZT-100-C DISPLAY card is planned but does not exist. There is no display firmware.**

What exists in the source tree is a stub: an empty composition, an empty board pin header, a build file and a README describing an *intended* composition. No display code, no pin map, no firmware.

Consequences:

- The ZT-100-C in this revision is a **two-card** product: CAP and SIGNAL.
- Field terminals **0 and 1** (4-20 mA IN, Red / Black) are provisioned to feed a display card and are **unused**.
- The heartbeat LED flash count is `1 + peer count`. With two cards the normal count is **2**. A third card would make it 3 — there is no third card in this revision.
- No display behaviour is described anywhere in this manual because none is implemented.

Do not commission, test or document display functionality against this revision.

Consoles, Sessions and Access Tiers

Console inventory

Console	Card	Physical	Baud	Role
USB	CAP	Native USB CDC	n/a	CAP's **only** local console
Internal RS485	CAP	Serial1 / SERCOM3, DE 18 (PA04)	`rs485.baud`, default 115200	**Link, not a terminal**
USB	SIGNAL	Native USB CDC	n/a	Factory setup
Internal RS485	SIGNAL	Serial1 / SERCOM3, DE 4 (PA14)	**Fixed 115200**, not configurable	Link
External RS485	SIGNAL	SERCOM0, TX D18 / RX D15, DE 3 (PB22)	`rs485.baud`, default **115200**	**Field operator console**
HART	SIGNAL	AD5700 on SERCOM4	1200 baud, 8-O-1	HART slave, declares revision **6** (7 at the factory)

USB writes are **accept-and-drop** when no host is attached — the card never wedges waiting for a terminal. USB and RS485 are **independent sessions on the same command registry**: a stream started on one does not disturb the other.

Access tiers

Tier	Entry	Scope
Operator	Default for every session	Reads; `cfg get`, `cfg list`
Commission	`auth commission 1234`	Configuration writes, calibration, output tests
Factory	`auth factory 9999`	Fault and stall injection

- The codes above are **fixed placeholders in this firmware revision**, identical on every card. Real per-device authentication arrives with the provisioning milestone.
- `auth` never **lowers** a session's tier.
- Command dispatch is longest-prefix: a strictly longer match wins; ties resolve to the product registry.
- CAP raises its **remote-exec** session (commands arriving over the link) to **commission tier** — physical bus access is the authorization model. SIGNAL's own link session stays at operator tier.

Output modes

```
mode human      # default
mode json       # newline-delimited JSON
```

In JSON mode every command answers with a `cmd` path, an `ok` flag and typed key/values; every measured value gets a companion `<name>_q` key carrying its quality.

Universal stop

Streaming commands — `meas stream start`, `cal start`, `cal enter`, `cal export`, `capcal add` — are stopped with **Ctrl-C** or **ESC**.

Two behaviours differ and matter:

- Stopping a **calibration progress stream** kills the *display* only. The capture keeps sampling and remains visible via `cal status`.
- Stopping a `capcal add` stream on CAP **aborts the whole capture**, deliberately. The working table is left unchanged.

CLI Command Reference

Both cards share one command language. Everything below is grouped by where it is registered.

Built-in commands — present on both cards

Registered by the command service itself.

Command	Arguments	Tier	Notes
`help`	—	operator	Works at any prefix, e.g. `meas help`
`mode`	`m:enum(human,json)`	operator	Output mode for this session
`auth commission`	`code:str`	operator	Raise to commission tier
`auth factory`	`code:str`	operator	Raise to factory tier
`cfg list`	`prefix?:str`	operator	List parameters
`cfg get`	`path:str`	operator	Read one parameter
`cfg set`	`path:str value:str`	operator	Write one; validated and tier-gated

Command matrix — CAP vs SIGNAL

● present · — absent. Tiers are the *registered* tier.

Command	Arguments	CAP	SIGNAL	Tier
`sys version`	—	●	●	operator
`sys nvm`	—	●	●	operator
`sys uptime`	—	●	●	operator
`sys boot`	—	●	●	operator
`sys reboot`	—	●	●	commission
`sys rs485`	—	● *(internal port only)*	● *(both ports + classifier)*	operator
`sys faultinject`	—	●	●	**factory**
`sys stallinject`	—	●	●	**factory**
`unit list`	—	●	●	operator
`meas read`	—	●	—	operator
`meas stream start`	`rate:f32[0.1..5]`	●	—	operator
`cal list`	—	●	●	operator
`cal start`	`flow:str`	●	●	operator
`cal status`	—	●	●	operator
`cal enter`	`val?:f32`	●	●	operator
`cal commit`	—	●	●	operator
`cal abort`	—	●	●	operator
`cal nudge`	`d:f32[-1..1]`	●	●	operator
`cal export`	`what:enum(capcal,probeoffset,fieldcal)`	●	—	operator
`capcal show`	—	●	—	operator

Command	Arguments	CAP	SIGNAL	Tier
`capcal add`	`pf:f32[0..3000]`	●	—	commission
`capcal del`	`idx:u32[0..15]`	●	—	commission
`capcal clear`	—	●	—	commission
`capcal save`	—	●	—	commission
`capcal load`	—	●	—	commission
`fcal show`	—	●	—	operator
`fcal save`	—	●	—	commission
`fcal adj`	`d:f32[-5..5]`	●	—	commission
`fcal set`	`wc:f32[0..98]`	●	—	commission
`fcal zero`	—	●	—	commission
`fcal clear`	—	●	—	commission
`probe show`	—	●	—	operator
`probe set`	`od:f32[10..200] id:f32[5..190] len:f32[50..2000]`	●	—	commission
`probe clear`	—	●	—	commission
`medium show`	—	●	—	operator
`medium set`	`dc:f32[1.2..10]`	●	—	commission
`medium table`	`e0 e25 e50 e75 e100 :f32[1.2..10]`	●	—	commission
`out status`	—	—	●	operator
`out test`	`ma:f32[3.5..21]`	—	●	commission
`out auto`	—	—	●	commission
`out trim`	`action?:str` (`clear`)	—	●	operator (`clear` needs commission)
`link status`	—	●	●	operator
`link txtest`	`secs?:u32[1..60]`	—	●	**factory**
`hart status`	—	—	●	operator
`cap`	`w0?..w7?:str`	—	●	operator
`sys id`	—	●	●	operator
`id show`	—	●	●	operator
`mb map`	—	—	●	operator
`mb stats`	—	—	●	operator

Totals: CAP 40 registered commands, SIGNAL 28 — plus the 7 built-ins on each.

`medium table` writes the whole five-point oil curve at 0/25/50/75/100 °C in one command. It exists because `medium set` installs a **flat** one-point medium, so using it to enter a customer's dielectric silently discarded the temperature slope; `medium table` is the way to keep it.

`link txtest` is a **factory-tier bench instrument** that floods the CAP link with requests and reports the retry rate. It saturates the link while it runs — never use it on a meter in service.

`cap` accepts up to **eight** words, widened from five so that `cap medium table e0 e25 e50 e75 e100` fits through the proxy.

`id show` — the shipping record's read

`sys id` identifies the **board**. `id show` identifies the **unit**, and it is the one to take at commissioning. **It is on all three cards** — CAP, SIGNAL and ET-25001 — so a record-collecting script asks one question per card rather than one per product.

```
product:     zt-100c-signal 0.17.35-sig-c59
mcu uid:     4DF031945357334E59202020FF0F3D07
label:       55KFXEXN   (PCB sticker)
unit serial: 10432   (HART final assembly no.)
probe serial:55021   (HART PV transducer serial)
id date:     20260810   (YYYYMMDD; HART cmd 18)
device tag:  SEP-A INLET
hart tag:    'ZTWCM'  long 'SEPARATOR-A INLET LINE 4'
descriptor:  'INLET SKID'
message:     'COMMISSIONED 2026-08'
hart:        devid 0x3417B2 poll 0 preambles 5 rev 6
hart id:     devtype 0x26E1 mfr 0 distributor 0 profile 0
config changes: 12   (HART cmd 0 / 38 counter)
(CAP's identity: `cap id show`)
```

SIGNAL adds the HART block — device id, poll address, declared revision, the factory identity and the configuration change counter. CAP prints the same reply without it, since it has no modem.

An unset serial prints `NOT SET`, not `0`, because 0 is a legal serial number and cannot mean both things.

`mode json` returns the same reply as one object, which is what a record-collecting script should read.

> **Take `cap id show` as well.** The two cards each carry their own copy of the identity block and their own MCU uid and PCB label. A record with only SIGNAL's half cannot say which CAP board is in the enclosure.

`sys id` — identifying a card

`sys id` is the production and service record for a board, at operator tier:

```
product:  zt-100c-signal 0.17.35-sig-c59
mcu uid:  4DF031945357334E59202020FF0F3D07
label:    55KFXEXN   (PCB sticker)
tag:      (unset)
```

Field	What it is
`product` / `version`	Which card this is and what it is running
`mcu uid`	The MCU's 32-hex unique id. The machine-readable key — calibration records and migration tooling identify a board by this
`label`	The **8-character identity printed on the PCB sticker**. This is the one to quote
`tag`	`device.tag`, the customer's own free-text name. Unrelated to the two above

Quote the label, not the uid. It is eight characters instead of thirty-two, it is printed on the board, and its alphabet deliberately excludes **I, L, O and U** so it cannot be misread as 1/1/0/V or misheard over a telephone.

The label cannot be changed or reassigned. It is derived from the MCU serial by a hash, so it is fixed for the life of the board — two cards cannot share one, and re-flashing does not alter it. `device.tag` remains the separate field for a site's own naming, and that one is freely editable.

> **Firmware older than CAP `0.16.21-cap-b22` / SIGNAL `0.17.3-sig-c27` / ET-25001 `0.16.7-m20` does not report `label`** — the field is simply absent from `sys id`. The **first units shipped predate it.** On those, read the sticker off the board, or identify the card by `mcu uid`.

Where the two cards differ most

	CAP	SIGNAL
Measurement (`meas …`)	●	**absent**
Field calibration (`fcal …`)	●	**absent**
Capacitance table (`capcal …`)	●	**absent**
Probe / medium (`probe …` , `medium …`)	●	**absent**
`cal export`	●	**absent**
Output (`out …`)	**absent**	●
`hart status`	**absent**	●
`cap` proxy	**absent**	●
Calibration flows	`capzero` , `capcal2`	`trim4` , `trim20`

There is **no `sys dip`** on either card — neither has DIP switches.

CAP command notes

Command	Behaviour
`meas read`	Prints every measured variable with its quality: water cut, capacitance pF, dielectric constant, process temperature, board temperature, MCU temperature, and the raw bridge averages `ad0` / `ad1`
`meas stream start <rate>`	Streams the same set. Rate 0.1 – 5. **Does not stream through the `cap` proxy** — use CAP's USB console
`sys rs485`	Internal link port counters only
`link status`	Link peers, addresses, heartbeat counters, frame and scanner counters, request stats, and the `reply:` line — `dropped=<bytes> overruns=<exchanges> hi=<high-water>/1024` for the peer-reply buffer
`cal export <what>`	Serialises the **current live** calibration — not the stored record — as interchange JSON, streamed in 192-byte chunks every 25 ms. Synthetic span anchors are never exported

SIGNAL command notes

Command	Behaviour
`out status`	Output state (`normal` / `sat-hi` / `sat-lo` / `alarm`), commanded and corrected mA, loop closed/open with quality, AD5420 status bits `iout` / `slew` / `overheat` , and the running **control-writes** counter
`out test <mA>`	Drives a fixed current, 3.5 – 21 mA, for **30 s**, then auto-reverts to AUTO. In FIXED mode the saturation rewrite is suppressed, so a bench 3.5 or 21 mA is driven as commanded

Command	Behaviour
`out auto`	Return to source tracking immediately
`out trim`	Show both trim offsets. `out trim clear` zeroes both and persists immediately (commission)
`hart status`	Modem and engine state, loop gate, frame counters, plus `devcmd:` (the outcome of the last HART 143/144 — OK or FAILED with CAP's verbatim error text, age, and a failure count) and `cache:` (command 147's cache — valid or EMPTY, age in seconds, and the reply-accumulator overflow count)
`link status`	Link peers and heartbeat counters, frame and scanner counters, request stats, and the `reply:` line — `dropped=<bytes> overruns=<exchanges> hi=<high-water>/1024` for the peer-reply buffer
`sys rs485`	**Both** ports plus the traffic classifier counters (text / modbus / unknown bursts, latch state, `echo_drops`, `tx_dropped`)
`cap`	See below

Reaching CAP through SIGNAL — the `cap` proxy

CAP has no console on the field terminals. The field path to CAP is SIGNAL's proxy.

```
cap                        # no arguments: link health + mirrored values
cap <command line>         # forward the line to CAP, print CAP's reply
```

Examples:

```
cap sys version
cap meas read
cap fcal show
cap probe set 52.5 42.16 328
cap cfg set meas.damping 5
cap capcal show
```

Rules and limits:

Rule	Detail
Token limit	Up to **5** positional tokens are rejoined into the forwarded command line
Concurrency	One outstanding link request at a time. If the shared request slot is busy, `cap` defers **one** retry after about 300 ms rather than failing immediately; a **second concurrent** `cap` is refused at once
Reply size	Remote replies are capped at roughly **400 bytes**
Streaming	Streams **do not** stream remotely. `cap meas stream start 1` will not work as a stream
Tier	CAP raises the remote-exec session to **commission tier** — commission-tier CAP commands work through the proxy without a separate `auth` on CAP

`cap` with **no arguments** is the single most useful diagnostic on the instrument. It reports:

- Link peer count and CAP's address
- Whether CAP's data is **fresh or STALE**
- Heartbeats seen and bad payloads counted
- **Every mirrored value with its quality** — and those qualities are CAP's own, not invented by SIGNAL

Configuration Parameters

All keys are persistent (record `REC_CFG`, id 4). **Tier is per key** — the Tier column below is authoritative. Until firmware 0.16.37-cap-b38 / 0.17.35-sig-c59 nearly every key on both cards sat at commission; see Consoles, Sessions and Access Tiers for the rule that spread them.

CAP

Key	Type	Range	Default	Tier	Effect
`device.tag`	str, 24 B	free text	empty	commission	Label only, no behaviour
`id.*`, `unit.serial`, `probe.serial`	as SIGNAL below			commission	Identity only
`rs485.baud`	u32	9600 – 460800	**115200**	commission	**Next boot**
`link.idle_ms`	u32	5 – 100 ms	**5**	commission	Live; bench knob, **not persisted**
`meas.damping`	f32	0.0 – 60.0 s	**0.0**	commission	**Live** — user damping, see below
`rtd.rref`	f32	100.0 – 5000.0 Ω	**430.0**	commission	**Live** — next reading

CAP has **no** `pv.range` — it drives no analog output.

> `meas.damping` **is a CAP key on this firmware**, at commission tier, and it is the only damping in the signal path. SIGNAL has no damping key of its own. A later release moves it to SIGNAL and renames what remains on CAP; a card running that firmware is documented by a different edition of this manual, because the same key on two cards at two tiers would be a support call waiting to happen.
>
> What this one does has not changed: it is an **anti-alias filter** sized against the 200 ms link poll, so that a single outlier sample is not the one SIGNAL happens to catch. We choose it; there is no field reason to move it.
>
> **A card upgraded from earlier firmware falls back to the 1 s default** rather than inheriting its old `meas.damping` value. That is deliberate: the old value was an operator's process damping, and promoting it to a factory constant would be a wrong number arriving with factory authority.

The internal bus runs at 115200 in practice; the key remains configurable on CAP but changing it will break the link unless SIGNAL's fixed 115200 internal port is changed too — **and SIGNAL's internal link baud is not configurable**. Do not change `rs485.baud` on CAP.

SIGNAL

Key	Type	Range	Default	Tier	Effect
`device.tag`	str, 24 B	free text	empty	commission	Label only

Key	Type	Range	Default	Tier	Effect
`meas.damping`	f32	0.0 – 60.0 s	**0.0**	operator	**Live** — the USER damping. See below
`rs485.baud`	u32	9600 – 460800	**115200**	commission	**Next boot**; **external console only** — the internal link is fixed 115200
`pv.range`	f32	0.1 – 100.0 %	**25.0**	commission	**Live** — 0..range % maps to 4-20 mA, re-applied to the output span on every store poll
`link.baud`	u32	9600 – 460800	**115200**	commission	**Next boot**; both cards must match
`link.idle_ms`	u32	5 – 100 ms	**5**	commission	Live; bench knob, **not persisted**
`link.poll_ms`	u32	50 – 5000 ms	**200**	commission	**Live**. See below
`id.tag`	str, 9 B (8 packed-ASCII + NUL)	≤ 8 chars	`ZTWCM`	commission	HART identity; also written by HART cmd 18/22
`hart.devid`	u32	0 – 16777215 (24-bit)	**1**	commission	HART identity
`hart.polladdr`	u32	0 – 63	**0**	commission	**Live**; also written by HART cmd 6
`hart.preambles`	u32	**5 – 20**	**5**	commission	**Live**; also written by HART cmd 59. The floor is the specification's, not a preference
`id.descriptor`	str, 17 B	≤ 16 chars	*(empty)*	commission	HART identity; also written by HART cmd 18
`id.message`	str, 33 B	≤ 32 chars	*(empty)*	commission	HART identity; also written by HART cmd 17
`id.longtag`	str, 33 B	≤ 32 chars	*(empty)*	commission	Identity; also written by HART cmd 22, and read/write over Modbus
`id.date`	u32	0 – 21551231	**0**	commission	Identity date as YYYYMMDD; also written by HART cmd 18. The loop carries year−1900 in one byte, so 1900–2155 is the whole representable span
`unit.serial`	u32	0 – 16777215	**0**	commission	The **assembly's** serial = HART final assembly number (cmd 16/19)
`probe.serial`	u32	0 – 16777215	**0**	commission	The **probe's** serial = HART PV transducer serial (cmd 49)

> `meas.damping` **is new on SIGNAL and is the operator-facing damping.** It used to be CAP's parameter, reached over the link, and that is what HART command 34 wrote. Around it sat a mirror of CAP's last *confirmed* value, a background poll refreshing that mirror every 30 s, and three device-specific status bits reporting never-confirmed / stale / unsettled. All of it is gone: command 34 is a local write and command 15 reads the same field back.
>
> `link.poll_ms` **is factory tier**, and it is worth knowing why a freshness knob became one. With the user damping on this card, the poll period **is the sample rate of that filter**. Setting it to 5000 ms would not merely slow the readings down — it would quietly wreck the damping. It belongs with the oversampling and the sample rate: things we choose.

The identity fields and both serials are **operator**. HART writes them as standard commands, and refusing a standard write is not ours to do — gating the same field locally would buy nothing a master respects. The card's real identity is the MCU unique id and the Label ID derived from it, neither of which can be edited at any tier.

Factory-tier HART identity

These describe what the meter *claims to be* to every master on the loop, which is why a commissioning engineer cannot move them. On this firmware they are the only factory-tier keys: `rtd.rref`, `link.poll_ms` and the capacitance table are all reachable at commission.

Key	Type	Range	Default	Notes
`hart.rev`	u32	6 – 7	**6**	The declared HART revision. Changes the shape of command 0
`hart.mfr_id`	u32	0 – 65535	**0**	Manufacturer identification code
`hart.distributor`	u32	0 – 65535	**0**	Private label distributor code
`hart.devtype`	u32	0 – 65535	**0x26E1**	Expanded device type
`hart.profile`	u32	0 – 255	**0**	Device profile code
`hart.cfg_changes`	u32	0 – 65535	**0**	The configuration change **counter**, not a setting. Writable only so a meter can leave manufacture at a known value

> **Two serial numbers, and they are different things.** `unit.serial` is the assembled unit; `probe.serial` is the sensor. They part company at an RMA, and HART carries both separately for that reason. Both are 24-bit because HART's own fields are, and `cfg set` refuses anything larger rather than letting the loop truncate it silently. `id show` returns the whole identity block in one read - see `id show`.

Live versus next boot — summary

Key	Card	When it takes effect
`rs485.baud`	CAP	Next boot
`rs485.baud`	SIGNAL (external console)	Next boot
`meas.damping`	**SIGNAL**	**Live**
`meas.damping`	CAP	**Live**

Key	Card	When it takes effect
`link.poll_ms`	SIGNAL	**Live**
`pv.range`	SIGNAL	**Live**
`hart.polladdr`	SIGNAL	**Live** (matches a HART cmd-6 write)
`hart.preambles`	SIGNAL	**Live** (matches a HART cmd-59 write)
`device.tag`, `id.tag`, `hart.devid`	either	Stored on set; no live side effect wired
Internal link baud	SIGNAL	**Not configurable** — hard-coded 115200

There is no live re-baud. A `rs485.baud` change requires a reboot, and the session that made the change finishes at the old rate.

HART writes fold back into configuration

HART commands 6, 17, 18, 22 and 59 mirror tag, device id, poll address and preamble count into the configuration shadow and best-effort persist `REC_CFG`. A HART master's identity writes therefore **survive a power cycle** and show up in `cfg get`.

Measurement Chain

Signal path

```
ADS1115 ch0 (probe) ─┐
                     ├─► normalize AD ─► interpolate capScaling(AD0)
ADS1115 ch1 (ref)  ──┘          │
                                ├─► factory ADC correction
                                ├─► bridge formula
                                ├─► board-temperature compensation
                                ├─► probe + user offsets
                                ├─► physics-based water content
                                └─► field calibration ─► water cut %
```

Front-end constants

Item	Value
ADC	ADS1115, 16-bit, I²C `0x48`
Sample rate	128 SPS, single-shot
PGA setting	`ADS_GAIN_4`
Conversion budget	10 ms, with a 3× grace before the driver gives up
Channels	0 = probe (AD0), 1 = reference (AD1), single-ended
Oversampling	**4** raw conversions averaged per channel per cycle
RTD cadence	500 ms one-shot
Board temperature cadence	1000 ms
MCU die temperature cadence	10 s

MAX31865 / PT100 details

- Hardware SPI at 1 MHz, SPI mode 1, 60 Hz filter.
- Steady-state CONFIG `0x10` — **bias OFF between conversions** to limit self-heating.
- One-shot sequence: clear faults and bias ON → ≥10 ms settle → trigger → ≥65 ms conversion → read and bias OFF.
- Temperature conversion is Callendar–Van Dusen (`A` = 3.9083e-3, `B` = −5.775e-7) with the quadratic closed form for T ≥ 0 °C and the AN709 5th-order polynomial below 0 °C.
- **CAP's reference resistor is 430 Ω**, `rtd_wires` 3, chip select on pin 2.

Implausible RTD readings

A conversion below **−100 °C** is **not published as a temperature**: it is marked faulted, the neutral 25 °C substitute is used, and the reading is counted. A wrong reference resistor converts cleanly and lies, so a believable-looking number is not evidence of a working sensor.

CAP's reference is **430 Ω**. The mis-stuffed batch that affects the ET-25001 does **not** apply to this card, and the automatic reference-correction carried by that product is **disabled on CAP**.

CAP therefore refuses an implausible reading but never "fixes" one: a conversion below −100 °C is marked faulted and counted, and the reference in force is left alone. That is deliberate. On a card where the defect does not exist, an automatic corrector could only ever write a wrong value onto a correctly built board.

Use `cfg get rtd.rref` to confirm the value in force, and `cfg set rtd.rref <ohms>` (commission tier, live, persisted) if a card is ever built with a different reference.

Water-cut defaults at first boot

Item	Default
Bridge resistor selection	`WCUT_R390`; fallback capScaling **102.02 pF**
Board temperature compensation table	0 – 80 °C in 5 °C steps (17 entries), normalised so the multiplier at 25 °C is 1
Probe geometry	ID **42.160 mm**, OD **52.500 mm**, length **328.000 mm** (2-inch LR probe)
Probe / user offsets	0.0 pF
Oil dielectric table	Default medium-oil table, also copied to a pristine baseline
Water dielectric table	**Empty** → the compute falls back to `88 − 0.4·T`
Field calibration	Disabled; scale 1.0, offset 0.0, saved temperature 25.0 °C
Maximum calibration points	**16**

Quality assignment — authoritative table

Variable	Condition	Quality
`wcut`	Compute OK, not undercut	`good`
`wcut`	Undercut (final water content < 0)	`underrange`
`wcut`	Unclamped water content > 100 (clamped high)	`overrange`
`wcut`	Bridge-domain error (AD1 ≤ AD0, probe below bridge zero)	Published with a 0 pF clamp, `underrange`
`wcut`	Any other compute error	`fault`, last value kept
`cap_pf`	Compute OK	`good` (`underrange` on bridge-domain)
`cap_pf`	Otherwise	`fault`, last value kept
`ad0` / `ad1`	Conversion pair completed	`good` **always** — the raw averages are valid observations whatever the maths says
`ad0` / `ad1`	Driver or bus error	`fault`; the cycle restarts, recovery is the next cycle
`temp_pcb`	Successful read	`good`
`temp_pcb`	Read failure	`fault`, last value kept — **and the last value is still used by the compute**
`temp_proc`	RTD one-shot completed	`good`
`temp_proc`	RTD fault, or the one-shot could not be started	`fault`, and both the published and the computed value become a neutral **25.0 °C**
all owned variables	`meas` unit stopped	`stale`

> **The RTD fault substitute is a fixed 25 °C — NOT the board temperature.** Some older in-source comments claim a board-sensor fallback; the code does not do this. The rationale is explicit: enclosures run hot, and the process medium is closer to ambient than to the electronics. **Document and diagnose against 25 °C.**

A faulted or absent PT100 is retried every 500 ms. Recovery needs no operator action.

Diagnostic value of the raw AD readings

`ad0` and `ad1` are published with quality `good` whenever a conversion pair completes, **even when the water-cut maths fails**. This is deliberate: it lets you distinguish "the ADC is fine but the calibration or the probe is wrong" from "the front end is dead". `capcal show` deliberately renders the raw AD columns for the same reason — a bad probe or rig connection shows up as implausible AD before it shows up anywhere else.

Analog Output and Trim

Mapping law (`aout`)

AUTO mode, source quality `good` / `underrange` / `overrange`:

```
ma = 4 + 16 × (v − src_lo) / (src_hi − src_lo)
```

Condition	Result
`ma > 20`	**Saturate high: 20.4 mA**
`ma < 4`	**Saturate low: 3.6 mA**
otherwise	Normal, range 4-20 mA

AUTO mode, source quality `fault` or `stale`: the **alarm current**, **3.6 mA**.

> The design note is explicit: "*the PoC held the last current on a dead source; we signal instead of pretending.*"

On SIGNAL, `src_lo` is 0.0 and `src_hi` is the `pv.range` configuration key (default **25.0 %**), re-applied to the unit on every store poll — so a range change is live.

Output states

`out status` reports an internal state code:

Code	State	Meaning
0	`normal`	Tracking within span
1	`sat-hi`	Above span → 20.4 mA
2	`sat-lo`	Below span → 3.6 mA
3	`alarm`	Source faulted or stale → 3.6 mA
other	init	Not yet driving

3.6 mA is ambiguous between `sat-lo` and `alarm`. `out status` is the only way to tell them apart. This distinction is worth putting into commissioning checklists.

Update cadence and register churn

- Default update cadence **50 ms**.
- **Range and control-register writes happen only on state transitions** (normal ↔ saturated/alarm), never every cycle. `out status` exposes a running **control writes** counter so churn is visible. A climbing control-writes counter in steady conditions means the output is flapping across a state boundary.

Loop-state detection

The AD5420 FAULT pin: **HIGH = loop closed (current flowing), LOW = open circuit or compliance failure**. It is only meaningful while the output is enabled and driving.

The `aout` unit samples it every poll and requires **3 consecutive identical samples** (`loop_debounce`, default 3) before flipping the `loop` store variable to 1.0 (closed) or 0.0 (open), quality `good`.

The `loop` variable is the gate that **HART depends on**. See HART Interface.

AD5420 chip facts

Item	Value
Bus	SPI mode 0 @ 3 MHz, MSB first, **no chip select**
Frame	24-bit (address byte + 16-bit word), committed by a **microsecond LATCH pulse**
Registers	Control `0x55`, data `0x01`, status readback `0x00`
Control bits used	OUTEN `0x10`, REXT `0x20`
Status bits	IOUT_FAULT `0x04`, SR_ACTIVE `0x02`, OVERHEAT `0x01` — surfaced by `out status` as `iout` / `slew` / `overheat`

The DAC's internal range selection and code scaling are firmware internals and are deliberately not published here: they are a property of how the firmware drives the part, not of the product. **The terminals carry 4-20 mA in every mode.** | REXT resistor | **Not fitted** on SIGNAL → internal RSET |

Trim maths and persistence

```
corrected = ma + zero_offset + (ma − 4)/16 × (span_offset − zero_offset)
```

Linear interpolation between the two offsets — `trim4` is the zero offset, `trim20` the span offset.

The two ends do not interact. Trimming zero does not move span, so there is no working back and forth between them as there was with potentiometer instruments. Set each once.

Only these two points can be trimmed; there is no way to trim at an arbitrary current. **Each offset is limited to ±0.25 mA** and a correction beyond that is refused rather than applied — a card needing more than that has a hardware fault (range resistor, output stage, loop compliance) and wants repair, not calibration.

Over HART, **command 45 trims zero and command 46 trims span**. A correction outside the limit answers **rc 3** (the card reads high) or **rc 4** (reads low) rather than a bare error, so a master can tell an operator which way the card is out.

Persisted as JSON under record id **6** (`REC_AOUT_TRIM`), using the previous generation's key names for interchange:

```
{"trim4mAOffset": …, "trim20mAOffset": …, "calibrationApplied": true}
```

Loaded at boot and overlaid onto the unit context, which defaults to 0 / 0.

```
out trim              # show both offsets (operator)
out trim clear        # zero both and persist immediately (commission)
```

Calibration Flows

Flow availability

Flow	CAP	SIGNAL	Tier
`capzero` — probe zero	●	—	commission
`capcal2` — two-point capacitance	●	—	**factory**
`trim4` — **zero** trim, 4 mA	—	●	**operator**
`trim20` — **span** trim, 20 mA	—	●	**operator**
`capcal` interactive table editor *(not a cal-engine flow)*	●	—	**factory** (`show` is operator)

Re-tiered in firmware 0.16.37-cap-b38 / 0.17.35-sig-c59 on one distinction: does the act need a **reference standard** at the card — something that *defines* what the card believes, rather than *verifies* what it does? A meter in the loop verifies, so a trim is operator work. A known capacitor on the probe defines what a picofarad means, so `capcal2` and the table editor are factory work, console only.

`capzero` sits between the two and stays **commission**: it needs a *condition* — the probe empty and dry — not a reference standard, and a field card swap is a documented procedure that simply needs the password.

The flow tier is separate from the `cal start` command tier. `cal start` is an operator command; the gate is the flow's own tier, listed above.

The old 8-point `capcal8` flow **no longer exists**. On CAP it was replaced by the interactive `capcal` editor.

Common flow mechanics

- One active run per card. A second session gets a **busy** error naming the owner.
- Verbs: `cal list`, `cal start <flow>`, `cal status`, `cal enter [val]`, `cal nudge <d>`, `cal commit`, `cal abort`.
- Step input kinds: no input (press enter), float value (`cal enter <value>`), or **capture** (the engine samples the store; no user value).
- **Inactivity auto-abort: 60 s** on both cards. CAP notifies the owning session and blips its RGB LED purple. The message names the reason: "*auto-aborted (60 s idle) — held output released, changes discarded*".

> ⚠ Some on-screen prompts still say idle flows auto-abort after **10 minutes**. **The behaviour is 60 seconds** — the prompt text is wrong and is a known documentation defect in the firmware. SIGNAL wires no auto-abort handler, so SIGNAL shows no purple blip.

- Capture steps auto-start a **500 ms progress stream** on the issuing session showing `[n/total]`. The universal stop kills the display without aborting the run.
- **Capture accepts only `good` samples.** A run fails if the store stays non-good.

`capzero` — probe zero (CAP)

Step	Prompt	Input
1/2	"Probe empty and dry? enter to continue"	`cal enter` (no value)
2/2	"capturing zero..."	Capture of `cap_pf`: settle **2000 ms**, **16 samples**, **250 ms** gap (≈6 s)

Commit semantics: `probe_offset_pf += captured average`. Adding the already offset-corrected reading makes the current reading the zero. Persists `REC_PROBE_OFFSET`. **Atomic-or-fail** — a failed serialise or queue restores the old offset.

On CAP's own console:

```
auth commission 1234
cal start capzero
cal enter
# ... capture runs ~6 s ...
cal commit
sys nvm            # wait for the write to drain
```

Proxied from SIGNAL — note that the forwarded session authenticates for itself, and stays authenticated for the commands that follow:

```
cap auth commission 1234
cap cal start capzero
cap cal enter
# ... capture runs ~6 s ...
cap cal commit
cap sys nvm
```

`capcal2` — two-point factory capacitance calibration (CAP)

Step	Prompt	Input
1/4	"known capacitance #1 (pF)?"	`cal enter <pF>`, range **0 – 3000**
2/4	"capturing point #1..."	Capture of `ad0` + `ad1`: settle 2000 ms, 16 samples, 250 ms gap
3/4	"known capacitance #2 (pF)?"	`cal enter <pF>`, **must differ from point #1**
4/4	"capturing point #2..."	Capture of `ad0` + `ad1`, same timing

Commit writes 2 calibration points (known pF, averaged AD0, averaged AD1, current board temperature), sets `factory_calibrated`, and persists `REC_CAPCAL`. Atomic-or-fail restores the previous table.

For production use the interactive editor below instead — `capcal2` is a minimal two-point path.

`capcal` interactive table editor — CAP only

This is the production capacitance calibration. It is *not* a cal-engine flow: it edits a composition-owned working table in RAM, point by point, and commits only on `capcal save`.

Command	Effect
`capcal show`	Render the working table — `Index | Known (pF) | Measured VRatio | AD0 Probe | AD1 Ref | Board Temp (C)` — then a status line: point count, saved/unsaved, whether the live calibration is `factory_calibrated`, and whether a capture is in progress. **Raw AD bits are deliberately visible** so a bad probe or rig connection shows as implausible AD
`capcal add <pF>`	Capture a point asynchronously: settle **2000 ms**, **16 samples**, **250 ms** gap (≈6 s), streaming `capcal: capturing <pF> pF [n/16]`. A pF within **0.5 pF** of an existing row **overwrites** it; otherwise it appends. **Ctrl-C / ESC aborts the capture** and leaves the working table unchanged
`capcal del <idx>`	Remove a point; the array is compacted
`capcal clear`	Empty the working table. **The live calibration is untouched**
`capcal save`	Commit working → live factory calibration. **Requires ≥ 2 points.** Re-runs the span-anchor check and persists `REC_CAPCAL`
`capcal load`	Re-seed the working table from the live calibration, discarding edits

Constants:

Constant	Value
Settle	2000 ms
Samples	16
Sample gap	250 ms
Consecutive non-good samples before failure	**50**
Overwrite match window	**0.5 pF**
Table capacity	**16** points

Only **one** capture may run at a time; a second gets a busy error. The capture accumulates only on `good` `ad0` / `ad1`. Fifty consecutive skips fail the capture with "*capture failed: AD never good (check probe/wiring)*".

At boot the working table is seeded from the live calibration, excluding synthetic span anchors.

The monotonic invariant — an impossible table is refused

More capacitance must always read a LOWER AD0 and a HIGHER VRatio. That is the physical behaviour of the bridge, and it is the operator's sanity check on every point captured.

A point that breaks it is not a slightly wrong number — it is an invisible one. Interpolation sorts by descending AD0 and brackets the sample between neighbours, so an out-of-order row yields smooth, plausible capacitance forever afterwards, with nothing in the reading to say it is wrong.

From CAP `0.16.20-cap-b21` the firmware objects in two places:

Where	What happens
`capcal add`	Prints a **WARNING** naming the offending pair, then tells the operator to re-seat the standard and add the point again. **The point is kept** — you may be mid-edit, and the table is allowed to be temporarily wrong
`capcal save`	**REFUSES**, with `impossible cal table:` and the same detail. Nothing is written and the live calibration is untouched

The warning names both rows, for example:

```
#1 47.00 pF (VRatio 0.8821) does not read ABOVE #3 220.00 pF
(VRatio 0.8454) - impossible; check the standard is connected
```

What to do about it. Re-seat the standard — a clipped or poorly connected one reads far too low — and run `capcal add <pF>` again with the same value. A capture within **0.5 pF** of an existing row overwrites that row rather than appending, so the bad point is replaced, not duplicated.

Two details worth knowing:

- **Equal AD readings at different capacitances are refused too.** The reading is ambiguous and the interpolation slope would be infinite.
- **Points may be captured in any order.** Only the AD ordering matters, so adding 470 pF before 220 pF is perfectly fine.

The refusal is deliberately at `save` rather than at `add`, because save is the point of no return: it overwrites the live factory calibration and marks the card `factory_calibrated`.

Span fool-proofing and the "incomplete calibration" warning

If a factory table exists (≥ 2 points, `factory_calibrated`) but has **no point at or below 10 pF**, or **none at or above 1000 pF**, default anchors measured on a shipped reference card are superimposed:

- **0 pF @ AD 18114 / 13385**
- **1500 pF @ AD 621 / 13373**

These are marked **synthetic**. They are **never persisted and never exported**. Their presence raises a warning at commit:

> "note: incomplete calibration - default anchor(s) superimposed (0 pF + 1500 pF); add real cal points at the range ends to remove"

The rationale: a table spanning only 470–1500 pF pins an empty probe at 470 pF, giving a field meter that can never read below about 8.8 mA. A ballpark at the extremes *with* an honest "incomplete" indication beats a meter stuck mid-range.

Treat this warning as a production defect — add real points at the range ends.

Zero and span trim — the 4-20 mA output (SIGNAL)

`trim4` is the **zero** trim (4.000 mA) and `trim20` is the **span** trim (20.000 mA). Nothing between them can be trimmed: two points fully determine the linear correction, and **the two ends do not interact** — setting zero does not move span.

- **On start** the output is driven at a fixed **4.000 mA** (zero) or **20.000 mA** (span) with **no timeout**. The flow's commit or abort — or the 60 s inactivity auto-abort — releases it. There is no confirm step: **the meter is live the moment the flow begins.**
- Single step, prompt "*fixed 4.000 mA driving - enter measured mA*".
- **Two separate checks, easily confused.** The **entry typo filter** — zero **3.0 – 5.0 mA**, span **18.5 – 21.5 mA** — only asks whether the number could plausibly be a reading of that endpoint. The one that matters is the **±0.25 mA limit on the resulting offset**: a correction beyond it is **refused, not clamped**. That is a hardware-fault threshold — a card needing more has a defect (range resistor, output stage, loop compliance) and wants repair, not calibration.
- Correction is additive: `new_trim = old_trim + (target − measured)`, so a re-run converges to zero.

- The entry is applied **live as a preview** — the meter visibly lands on the endpoint — but is **not persistent**. `cal commit` persists; `cal abort`, including the inactivity auto-abort, restores the entry value.
- `cal nudge <d>` (|d| ≤ 1) accumulates onto the previewed trim the same way. Flows are commit-any: a nudged-only value is committable.
- Commit persists `REC_AOUT_TRIM` and drops the unit back to AUTO **only on success**. On persist failure the unit stays FIXED with the entry value restored.

> **SIGNAL has no loop-open guard on the trim flows.** The single-card ET-25001 product refuses `cal start trim4/trim20` on an open loop; SIGNAL does not. Verify the loop is closed with `out status` before trimming, or you will be trimming against nothing.

Procedure:

```
auth commission 1234
out status                 # confirm the loop reads closed
cal start trim4
# read the reference meter, e.g. 4.031 mA
cal enter 4.031
cal commit                 # persists; returns to AUTO on success

cal start trim20
cal enter 19.962
cal commit
sys nvm                    # wait for the write to drain
out trim                   # confirm both offsets
```

The same trim from two places

SIGNAL has no front-panel buttons, so the trim is a console or HART operation. Modbus can read the offsets but not set them.

	Console	HART
Trim zero	`cal start trim4`	**command 45** — *trim loop current zero*
Trim span	`cal start trim20`	**command 46** — *trim loop current gain*
Apply a reading	`cal enter <measured>`	the measured value is the command's argument
Nudge	`cal nudge <delta>`	—
Save	`cal commit`	the command commits on acceptance
Read back	`out trim`	command 161

HART calls the span end **gain** rather than span; they are the same trim. A correction outside the ±0.25 mA limit answers **rc 3** when the card reads high and **rc 4** when it reads low, rather than a bare error — so a master can tell an operator which way the card is out of spec. That is a repair signal, not a retry.

Over Modbus the offsets are readable at **IR 48-51** and clearing them is an armed action. There is deliberately no Modbus write path that *sets* a trim: a trim needs a reference meter and an operator reading it, which a polling master cannot supply.

To discard trims entirely: `out trim clear` (commission).

Field calibration (`fcal`) — CAP

Command	Tier	Effect
`fcal show`	operator	Enabled flag, lab water cut %, saved pF at its temperature with the raw water cut, linear scale/offset, oil-table state
`fcal save`	commission	**Pin the reading at the sample (step 1).** Stores the current capacitance and temperature as the point the lab result will be fitted to. Refuses at 0 pF — that value doubles as the "nothing pinned" sentinel
`fcal set <wc>`	commission	**Apply the lab result (step 3).** Fits water content `wc` % (0 .. 98) to the **pinned** reading. Refuses if nothing is pinned
`fcal adj <d>`	commission	Step the lab water cut by `d` % (−5 .. +5). Also requires a pinned sample
`fcal zero`	commission	Make the current reading read 0 % water cut. The one-visit exception: pins and applies in the same breath, so it needs no prior `fcal save`
`fcal clear`	commission	Disable field calibration, reset scale/offset, restore the pristine oil table
`probe clear`	commission	Reset the probe capacitance offset to 0 pF

> **Field calibration is a two-visit procedure.** `fcal save` pins the reading at the sample; `fcal set` fits the laboratory result to **that pinned reading** days later, not to whatever is live when it runs. This is what makes the second visit independent of the process condition.
>
> `fcal set` and `fcal adj` **refuse** without a pin — "*no saved sample point - run `fcal save` at the sample, then apply the lab result*". The saved capacitance doubles as the "nothing pinned" sentinel, which is why `fcal save` itself refuses a 0 pF reading rather than storing one.
>
> Both halves are reachable over HART (**144** pins, **143** applies), so a master is not restricted to half the procedure — see the device-specific command table.

Mechanics. *apply_lab* inverse-solves the oil dielectric constant by **bisection, 20 iterations**, searching DC in **[1.5, 3.5]** and accepting in **[1.5, 4.0]**. It then rebuilds the working oil table as the pristine baseline scaled by `targetDC / baseDC` at the saved temperature, selects it, and zeroes scale/offset. **Dielectric-constant mode and linear mode are mutually exclusive.**

Every mutating `fcal` command is **atomic**: it refuses up front if the persistence buffer is busy, and rolls the live calibration back if either the derivation or the persist fails.

Probe geometry and medium — CAP

Command	Tier	Effect
`probe show`	operator	OD / ID / length in mm, the `geometry_set` flag, and the probe offset
`probe set <od> <id> <len>`	commission	Set geometry in mm. **Rejects ID ≥ OD.** Bounds: OD 10–200, ID 5–190, length 50–2000. Persists `REC_PROBE_OFFSET`; atomic-or-fail
`medium show`	operator	Oil dielectric table (temperature | DC) plus pristine / calibrated-oil state
`medium set <dc>`	commission	Install a **flat one-point** medium at **25.0 °C** with dielectric `dc` (1.2 – 10). **Clears the field calibration first.** Persists `REC_FIELDCAL`

Command	Tier	Effect
`medium table <e0> <e25> <e50> <e75> <e100>`	commission	Write the whole **five-point** oil curve at **0 / 25 / 50 / 75 / 100 °C** in one shot, each 1.2 – 10. **Clears the field calibration first.** Persists `REC_FIELDCAL`; atomic-or-fail

Use `medium table`, not `medium set`, to enter a customer's oil. `medium set` installs a *flat* medium — one point, no temperature slope — so entering a lab dielectric with it silently discards the oil's temperature dependence, which is a measurement error rather than a missing feature. `medium set` is for a genuinely flat medium only.

Three ways in, and they differ in what they can express:

Path	Points	Temperatures
`medium set <dc>`	1	flat, at 25 °C
`medium table …`	5	the fixed grid 0 / 25 / 50 / 75 / 100 °C
Calibration import (`oilTable` / `baseOilTable` in the interchange JSON)	up to **32**	arbitrary

The console grid is fixed at those five temperatures; an arbitrary curve at other temperatures has to arrive over the import path. Between points the card interpolates linearly, and outside the table it clamps to the end values.

> The card carries **one** probe and **one** medium — one curve, not a library of media.

Both writes are repeatable: re-running either simply replaces the curve, and each write is atomic, rolling back if the persist fails. Two consequences worth knowing before commissioning:

- **Every medium write clears the field calibration**, deliberately — a field cal derived against the previous oil is meaningless against a new one. Re-run `fcal` afterwards, not before.
- **There is no console command that restores the factory default curve.** Clearing the field calibration restores the *pristine base*, which is whatever was last written, not the shipped default. To go back, re-enter it: `medium table 2.30 2.25 2.20 2.15 2.10`.

Calibration export

```
cal export capcal
cal export probeoffset
cal export fieldcal
```

Serialises the **current live** calibration — not the stored record — as its interchange JSON, streamed in **192-byte chunks every 25 ms**. Synthetic span anchors are never exported.

This is the mechanism by which calibration is carried across a firmware cutover. See Firmware Upgrade and Migration.

Power loss during calibration

- The record store is **commit-then-supersede**: abort or power loss at any point leaves the **previous** calibration intact.
- A `cal commit` only **queues** the flash write; the record commits asynchronously over the next few seconds of store ticks. **Power loss before the flash commit leaves the old record.** The CLI says so: "*committed (record persists in background; `sys nvm` shows progress)*".

- **Consequence for any procedure:** after a calibration commit, wait for `sys nvm` to show the write drained — state idle, `writes_ok` incremented — before removing power.
- Live trim previews are **not** persistent. A power cut during a trim flow loses the preview and leaves the last committed trim.
- Every mutating commit is **atomic-or-fail**: if serialisation or the write queue fails, the in-RAM calibration is rolled back and the command reports the failure. Nothing is left half applied.

HART Interface

Physical layer

Parameter	Value
Modem	AD5700 on SERCOM4
Signalling	Bell 202 FSK on the 4-20 mA loop
Baud	1200
Framing	8 data bits, **odd** parity, 1 stop bit (11 bits on the wire per byte)
HART revision	**5**
Control pins	RESET released HIGH; nRTS deasserted HIGH (carrier **not** keyed); DUPLEX driven LOW; CD is an input

Loop gating — the structural fix

While the loop is open the modem is held in reset and received bytes are dropped at the door. nRTS never asserts.

"Loop OK" means the `aout`-owned `loop` variable reads **closed** *and* its quality is **good**.

This is the structural fix for the previous generation's boot-storm and open-loop screaming demodulator. It also means **the first HART diagnostic step is always** `out status` — if the loop is open, no HART symptom is meaningful.

Reply timing

Parameter	Value
Reply due	`rx_complete + sta_ms`, default **20 ms**
Carrier settle	After keying nRTS the first byte is held **8 ms** — streaming immediately mangles the frame head at the far demodulator (bench-verified)
Carrier drop	Only after the TX shifter is empty

Declared revision

The meter **declares HART revision 6** as supplied. Revision 7 is a factory setting (`cfg set hart.rev 7`, factory tier) and is switched on per order, because declaring 7 is a claim about conformance rather than a preference.

Everything revision 7 needs is built either way; only the declared number and the shapes that depend on it change. **Command 0 answers 12 bytes at revision 6 and 22 at revision 7** — the extra ten carry the slave-to-master preamble count, the last device variable code, the configuration change counter, extended device status, manufacturer id, distributor and device profile. Those fields are held at either revision, so switching a meter between them loses nothing.

Implemented universal and common-practice commands

Universal: 0, 1, 2, 3, 6, 7, 8, 9, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 38, 48 — that is the complete universal set (4 and 5 are Reserved by the specification).

Common practice: 34, 35, 40, 45, 46, 49, 59.

Device-specific: 143, 144, 147, 148 (field calibration), and the 140–151 family.

Anything not in those lists answers **response code 64, command not implemented**.

> **Command 42 (device reset) is deliberately NOT implemented.** The Common Practice Command Specification is explicit — "*this command must not be used in new device types*" — and gives it no request or response bytes at all. A master cannot reboot the meter over HART. Use `sys reboot` on the console. Earlier documentation described command 42 rebooting both cards; that behaviour was removed in `0.17.32-sig-c56`.

Device variables (commands 8 and 9)

Command 9 reads up to eight device variables in one transaction, each with its own status. Asking for fewer than eight is not an error.

Code	Variable	Units code	Classification
0	Water cut (PV)	149 vol%	88 volume per volume
1	Process temperature (SV)	32 °C	64 temperature
2	Capacitance (TV)	153 pF	82 capacitance
3	**Reserved** (QV) — not specified in this issue	251 none	0 not classified
4	Dielectric constant	251 none	0 not classified
5	PCB temperature	32 °C	64 temperature
6	MCU temperature	32 °C	64 temperature
7	Loop current	39 mA	84 current

Code 3 is answered only when the attached CAP speaks heartbeat v3. Against an older CAP the code reads **unsupported** rather than zero, so "no such field" and a genuine zero cannot be confused.

The value is dimensionless, hence units **251** ("none") and classification **0** — the same pair the dielectric constant uses. It is deliberately not given a unit that names a quantity.

The standardized codes of Common Table 34 are answered as well: **244** percent of range, **245** loop current, **246–248** PV/SV/TV. **249** (quaternary variable) follows code 3.

Command 3 is deliberately not extended. Its dynamic set stays loop current + PV + SV + TV, 19 bytes. Widening a standard command's reply has conformance consequences, and the length would vary at runtime with which CAP is fitted — read device variable 3 with command 9 or 33 instead.

An unsupported variable is answered, not refused. It comes back as not-a-number with units 250 ("not used") and status `0x30` (bad, constant), so one absent code in a list does not cost a master the variables the meter does have.

Per-variable status carries the same quality model as everything else. Good reads good; a value at its limit reads *poor accuracy, high limited*; stale, faulted or never-measured reads **bad**, never a number with a clean status.

Command 8 reports the classifications of the four dynamic variables: **88, 64, 82, 0**. Zero for the quaternary variable is the specified answer for a dynamic variable a device does not have, not a placeholder.

Configuration changed, and the change counter

The meter keeps a **configuration change counter** (`cfg get hart.cfg_changes`) and a configuration-changed indication **per master** — the primary's and the secondary's are tracked separately, because each has to see it once.

The counter is incremented by every command the specification lists as affecting it — **6, 17, 18, 19, 22** (universal) and **34, 35, 45, 46, 49, 59** (common practice) — and by any console or Modbus configuration write. **Calibration commits count too**: the specification counts "every user action that changes the device's configuration or calibration". Command 40, entering fixed current mode, does **not** count.

Command 38 clears the indication for the master that sends it, and returns the counter in every case. Sent with the master's own copy of the counter, it answers **response code 9** if they disagree — meaning something changed again while that master was not looking. Both the indication and the counter survive a power cycle.

> The counter is never reset by any HART command. It can be zeroed at the factory (`cfg set hart.cfg_changes 0` , factory tier) so a meter leaves manufacture at a known value.

Additional device status (command 48)

Command 48 answers **nine bytes**. The length is not a function of the HART revision: the reply is truncated after the last status byte a device supports, and bytes 9–13 are required only of a device with **more than one analog channel**. This meter has one.

Byte	Content
0	Device-specific status — see below
1–5	Device-specific status, unused, always zero
6	Extended device status (Common Table 17)
7	Device operating mode (Common Table 14) — always 0
8	Standardized status 0 (Common Table 29)

Device-specific status, byte 0:

Bit	Meaning
`0x01`	CAP has gone quiet — the measurement is stale
`0x02`	CAP reports a fault
`0x04`	Loop open
`0x08`	Loop current held fixed
`0x10`	A queued command to CAP failed
`0x20`	*Retired* — always 0
`0x40`	*Retired* — always 0
`0x80`	*Retired* — always 0

Extended device status uses `0x02` , *device variable alert*, whenever the water cut is not good. Standardized status 0 uses `0x20` , *environmental conditions out of range*, when the MCU die temperature leaves −40 … +105 °C.

"More status available", and how a master clears it

The specification requires each manufacturer to state **which** command-48 bits raise the "more status available" bit. For this meter:

Byte	Raises the bit
0 — device-specific	**all bits**
1–5 — device-specific	unused
6 — extended device status	**all bits**
7 — device operating mode	**no** — an enumeration, not a fault
8 — standardized status 0	**all bits**

The bit means **"my command-48 data has moved since you last acknowledged it"**, and it is tracked per master.

Reading command 48 does not clear it. To clear it, a master reads command 48 and then sends those same nine bytes back as a command 48 *request*. On an exact match the meter resets that master's bit; a mismatch or a short request leaves it alone, and the reply carries the current values either way.

> **This matters for bus loading.** A master that never acknowledges leaves the bit permanently set, and hosts are built to issue command 48 whenever they see it — in the specification's words, communication bandwidth is then effectively cut in half.

A fault **clearing** also sets the bit: the data has changed from what the master acknowledged, and that is news too.

Output, range and trim commands

These are standard HART commands, so a generic master or DD host already knows them by number.

Cmd	Behaviour
15	Read PV output information, **18 bytes**: alarm select (**1**, low — the loop alarms at 3.6 mA), transfer function (**0**, linear), range units code, upper range value = `pv.range`, lower range value = **0**, damping, write protect (**251**, not implemented), reserved (**250**, not used), PV analog channel flags (**0** — the channel is an output, not an ADC input). Damping is SIGNAL-local, so it is always known — see the note below
34	Write PV damping, big-endian f32 seconds, range-checked **0 .. 60**. **SIGNAL-local**: a plain write of this card's own `meas.damping`, so **rc 0 means DONE**, not accepted, and command 15 reads the same field straight back. **rc 3/4** out of range, **rc 32** the record store is busy (retriable), **rc 5** short payload
35	Write PV range values: units code, then upper and lower range value as big-endian f32. The **lower range value must be 0** — the span on this device runs from zero water cut by construction — and a non-zero one is refused (**rc 9** too high, **rc 10** below zero). The upper value is range-checked against `pv.range`'s own limits, **0.1 .. 100** (**rc 11** too high, **rc 12** too low, **rc 13** out of limits); **rc 18** for a units code that is neither the device's own nor 250 ("not used")
40	Enter fixed current mode at the given big-endian f32 mA, or **exit it with 0**. Accepts **3.5 .. 21 mA**; **rc 3/4** outside that, **rc 32** while a calibration flow owns the output
45	Trim the 4 mA end against a current the operator measured, big-endian f32 mA, accepted band **3.0 .. 5.0**. Requires fixed mode at ≈4 mA first (**rc 9** otherwise); **rc 32** if the record store is busy or a console owns the calibration engine

Cmd	Behaviour
46	Trim the 20 mA end the same way, accepted band **18.5 .. 21.5**, requires fixed mode at ≈20 mA
49	Write the PV transducer serial number (the **probe's**, 24-bit) — `cfg set probe.serial`. Read it back inside command 14
59	Write the number of response preambles, **5 .. 20**. **rc 3/4** outside that band

A write that is refused **echoes nothing** — the reply carries the value only on success, so a master cannot read its own request back out of a reply that rejected it.

> **Response codes are per command.** The same number means different things on different commands: on command 35, code 12 is "upper range value too low", while on command 15 it is undefined. A master that reads these against one shared table will name the wrong fault.

> **Fixed current mode set over HART does not time out.** The console's `out test` reverts to tracking after 30 s; command 40 holds until a master sends command 40 with 0 mA. **An engineer who fixes the loop over HART and walks away leaves it fixed.** The reason is command 45/46: a silent revert partway through a trim would have the operator measuring one current while the card trims against another. Every reply carries the **"loop current fixed"** status bit while the loop is held, so the condition is always visible to a master, and `hart status` reports it on the console.

The loop trim sequence. The measured value in commands 45 and 46 only means something if the card was driving the nominal current when the meter was read, which is why fixed mode is a precondition rather than something the trim command establishes for itself:

1. **40** with `4.0` — the loop goes to 4 mA and stays there
2. Read the loop with a meter in series
3. **45** with what the meter said
4. **40** with `20.0`, read the meter, **46** with the reading
5. **40** with `0.0` — back to tracking the measurement

The card **stays in fixed mode after a trim**, deliberately: the master put the loop there and only command 40 takes it out, so step 3 can be followed by another meter reading to verify the trim actually landed.

Damping is SIGNAL-local, and commands 15 and 34 are a matched pair. Command 34 writes this card's own `meas.damping` ; command 15 reads the same field back. A write either succeeds or returns a response code saying why — there is no "accepted, verify later" step, and no state in which the damping is unknown.

What this replaced. Until firmware 0.17.35-sig-c59 the damping lived on CAP, so command 34 queued a command over the link and confirmed asynchronously. Command 15 answered from a mirror that was written only when CAP *confirmed* a value, so a refused write left the old value visible on read-back; a background poll refreshed that mirror every 30 s; an unconfirmed mirror answered command 15 with HART's not-a-number pattern rather than a zero, because zero is a legal damping meaning *off*; and three device-specific status bits reported never-confirmed, stale and unsettled.

Every one of those safeguards was correct for the problem it had. Moving the damping onto SIGNAL removed the problem instead. **Status bits `0x20` , `0x40` and `0x80` are retired rather than reused** — none of the three conditions can arise, so a master reading 0 there is reading the truth, and an old device description is not misled.

Device-specific commands

Cmd	Behaviour
140	**Response code 64, "not implemented"** (provisioning, a later phase)
143	Calibrate to a lab water cut %. Payload: big-endian f32, range-checked **0 .. 98**. **Requires a prior 144** — CAP refuses it with "no saved sample point" if nothing is pinned. Cannot be answered synchronously — queues `fcal set <wc>` for asynchronous dispatch to CAP and answers **rc 0, accepted**. **rc 32 (busy)** if the single pending slot is occupied; **rc 5** if the payload is short; **rc 2** if out of range
144	**Pin the sample**: queues `fcal save` to CAP, same accepted/busy contract, no payload. This is step 1 of field calibration and exists on HART precisely because 143 refuses without it. CAP can refuse (a 0 pF reading cannot be pinned); the refusal travels the usual route — 147 raises "more status available" and `hart status` carries the reason
148	Zero the field calibration: queues `fcal zero` to CAP, same accepted/busy contract, no payload
145, 146	**rc 64, "not implemented"** — there is no single-shot CAP equivalent
147	Read the CAP calibration summary from a cache: **4 big-endian f32** = lab water cut, saved pF, saved temperature °C, calculated water cut. The cache is refreshed by a background `fcal show` poll every **30 s** over the link. The "more status available" bit `0x10` is raised if the cache is missing or older than **95 s**, **or if the last 143/144 failed on CAP**
151	Diagnostics: `sensorStatus` (0 good / 1 stale / 2 fault, derived from the `wcut` quality), `errorCode` 0, `sensorVoltage` 0.0 (not applicable), `internalTemp` = SIGNAL's MCU die temperature, `measurementCount` = heartbeats consumed from CAP

143 and 144 are asynchronous by necessity. They must cross the internal link to CAP, which cannot complete inside a HART reply window. The "accepted" response means the request was queued, not that the calibration succeeded. **Always verify with command 147, or with `cap fcal show` on the console.** There is exactly **one** pending slot — a second request while one is in flight gets rc 32.

How a failure after "accepted" reaches you. CAP's verdict on the queued command is latched by SIGNAL, so a calibration that CAP refused cannot look like a successful one:

Where	What you see
Command 147	The "more status available" bit `0x10` is raised, and stays raised until a later device command succeeds
`hart status`	The `devcmd:` line — `FAILED`, **CAP's own error text verbatim**, how long ago, and a cumulative failure count

A reply that could not be parsed, and a reply that never arrived at all, both count as failures. Silence about a calibration is not success.

What HART deliberately does not carry. Field calibration is reachable over HART because a lab water cut is a number the host legitimately has. The **capacitance calibration table is not, and will not be** — `capcal add`, `del`, `clear`, `save` and `load` are console commands only. `capcal add <pF>` is only meaningful with a known capacitance standard connected to the probe, and `save` / `load` / `clear` overwrite or discard a working table whose state is known only to the person at the console. No HART command number is assigned to any of them, and none will be.

Nor does it carry CAP's own configuration. Probe geometry, the medium dielectric and the capacitance calibration table have no HART commands on this card, and none are planned — the command numbers that were once reserved for them have been withdrawn permanently. This is the same boundary the Modbus map draws, and for the same reason: those values live on CAP, and a mirrored copy that could silently go stale is what this firmware's honest-data rule forbids. Set and read them over the console, through the `cap` proxy or on CAP directly.

Over HART a master can read the measurements, the calibration summary and the diagnostics, read and write the range and the damping, field-calibrate, trim the loop, drive the output, and reset the pair. That is the complete list for this release.

Variable assignment

HART variable	Source	Note
PV	`wcut`	**Unit code 49** — the previous generation's PV unit code, kept for application compatibility
SV	`temp_proc`	Process temperature
TV	`cap_pf`	Probe capacitance
Loop current	`out_ma`	
Loop state	`loop`	Closed / open

Identity defaults

Item	Default
Expanded device type	`0x26E1`
Device id	`0x000001`
Poll address	0
Preambles	5
HART revision	5
Device revision	1
Software revision	16

Item	Default
Hardware revision	1
Tag	`ZTWCM` (also the fallback when the tag is empty)

Identity writes persist

Commands **6, 17, 18, 22 and 59** call the identity-changed hook, which mirrors tag, device id, poll address and preamble count into the configuration shadow and best-effort persists `REC_CFG`. Poll address and preamble count take effect **live**, with no reboot.

Stale data and HART

When the link mirror ages CAP's data to `stale`, HART raises the **"primary variable out of limits"** status alongside the 3.6 mA loop alarm. The values themselves keep their source qualities — SIGNAL does not invent a quality.

Modbus (SIGNAL)

SIGNAL runs a Modbus RTU slave on the external RS485 port — the same pair as the console, separated by the traffic classifier described in External port traffic classifier.

Parameter	Value
Port	External RS485 (port 1). **Not** the inter-card link
Slave address	**1**
Baud / framing	As the console: **115200** default, 8 data bits, no parity, 1 stop bit
Function codes	FC02 discrete inputs, FC03/FC06/FC16 holding registers, FC04 input registers
FC01 coils	Not implemented — answers exception 1
Float encoding	Two consecutive registers, **high word first**

`mb map` **on the card is the authority.** It prints every register this build implements and marks the absent ones. The tables below describe the map's shape and, more usefully, *which parts of it SIGNAL does not have*.

Input registers (FC04, read-only)

Measurements occupy four registers each: **value f32, quality, reserved.**

Registers	Content	On SIGNAL
0–3	Water cut %	● mirrored from CAP
4–7	Capacitance pF	● mirrored from CAP
8–11	Dielectric constant	● mirrored from CAP
12–15	Process temperature °C	● mirrored from CAP
16–19	CAP board temperature °C	● mirrored from CAP
20–23	SIGNAL MCU temperature °C	● local
24–27	Loop output mA	● local
28–31	Percent of range	**absent**
32–39	Field calibration: lab %, calculated %, saved pF, saved °C	● from the 30 s cache
40–47	Probe geometry: od, id, length, offset	**absent** — CAP-owned
48–51	Output trim offsets: zero (`trim4`), span (`trim20`)	● local
52–53	Uptime seconds (u32)	●
54–56	Firmware major, minor, patch	●
61	HART configuration change **counter** (u16)	●
62	Declared HART revision, 6 or 7 (u16)	●
64–67	Action queue: state, last opcode, last result, completion count	●
68–69	**Reserved**, u32, high word first — not part of this issue's published specification	
80–89	Capacitance calibration row window	**absent** — CAP-owned

Quality codes match the card's internal model exactly: `0` init, `1` good, `2` stale, `3` over range, `4` under range, `5` fault.

> **Read the value and its quality in one transaction.** They are adjacent for exactly this reason. The earlier layout put qualities far from their values, which let a master pair a fresh reading with a stale quality — a silent way to publish a number the card had already disowned.

Percent of range is absent, deliberately. SIGNAL does not hold it as a measured value, and publishing a figure computed on the fly would produce a number that nothing else on the card agrees with. Compute it from the loop current and the span if you need it.

Holding registers (FC03/06/16)

Registers	Content	On SIGNAL
0–15	Eight f32 configuration slots, declared per product	**Slot 0 (HR 0–1) =** `pv.range`, **slot 1 (HR 2–3) =** `meas.damping`; slots 2–7 absent
16–24	Action block: opcode, arming key, three arguments, trigger	●
25	Capacitance calibration row select	**absent**
26–41	`id.longtag`, 32 characters	●
42–57	`id.message`, 32 characters	●
58–65	`id.descriptor`, 16 characters	●
66–69	`id.tag`, 8 characters	●
70–71	`unit.serial` (u32) — the **assembly**	●
72–73	`probe.serial` (u32) — the **sensor**	●
74	`hart.polladdr`	●
75	`hart.preambles`	●
76–85	Action **wide** arguments, five f32 (staged)	● (see below)

A configuration write goes through the same path as `cfg set` on the console — same range check, same commission tier, same persistence. An absent slot reads 0 and refuses writes with exception 2.

The identity block (26–75) is everything HART can write, reachable from Modbus too — the same seven keys, so a unit commissioned over the fieldbus and one commissioned over the loop end up identical. Earlier issues of this manual listed only the long tag and called it the only string on the map; SIGNAL has declared all four string slots and both serials since the identity work landed (`zt-100c-signal/src/composition.c:3696-3703`).

Strings are two characters per register, **first character in the high byte**, NUL-padded. A write must cover **the whole slot, starting exactly at its base** — anything narrower is exception 2, and FC06 therefore cannot reach a string slot at all. There is no such thing as half a tag. Trailing spaces and NULs are stripped as padding, so a master with a fixed-width field gets back what it meant rather than a tag with fifteen trailing blanks.

The wide argument window (76–85) carries five f32 for opcodes that take more than the staged block's three. It is written whole — ten registers from register 76, or exception 2 — and staged in its own transaction, then fired by the ordinary trigger at 24. Only one opcode uses it today, 24 `medium table`, and **on SIGNAL that opcode is refused** like every other CAP-owned write: the registers accept the staging, and the action reports NOT SUPPORTED because SIGNAL's registry has no `medium table`. The medium belongs to CAP, which has no Modbus at all — see the CAP boundary below.

Discrete inputs (FC02, read-only)

Bit	Meaning	On SIGNAL
0	Loop closed	●
1	CAP link fresh	●
2	Record store mounted	●
3	HART modem out of reset	●
4	Field calibration enabled	**absent** — CAP-owned
5	Output in fixed mode	●
6	Factory calibrated	**absent** — CAP-owned
7	Crash record present	●

The CAP boundary, and what it means for an integrator

This is the single most important thing to design around. **SIGNAL's Modbus map covers what SIGNAL owns plus what it already mirrors for the 4-20 mA loop. It does not extend to CAP's own configuration.**

Concretely, a Modbus master on the ZT-100-C **can**: read every measurement with its quality, read the field-calibration summary and the output trims, write the 4-20 mA span, and drive the output and its trim flows through the action block.

It **cannot** read or write probe geometry, the capacitance calibration table, the medium dielectric, or the two CAP-owned status flags. The registers are absent rather than zero-filled, and the corresponding action opcodes are refused with a result of *not supported* and the text **"CAP-owned; no such command on SIGNAL"**.

That refusal is a designed boundary and not a defect: those values live on CAP, and a mirrored copy that could silently go stale is exactly what this firmware's honest-data rule forbids. **Use the console for them** — `cap probe show`, `cap medium show`, `cap capcal show` through the proxy, or CAP's own console directly.

> **Note for a future revision.** The absences above are the current extent of the Modbus surface on this card, not a permanent product boundary. The caching mechanism that backs the field-calibration registers is the same one that would back the others. Design against `mb map` on the card in front of you rather than against this table.

Reading an action result

The action block is staged and then triggered, so a master writes the opcode, arming key and arguments first and the trigger last. Poll IR 64–67 for the outcome.

The trap is worth stating: **IR 64 returns to `done` after every action**, so a master that polls only the state cannot tell its own completion from the one before it. IR 67 is a completion counter — sample it before triggering, and treat the result as yours only once it has advanced.

Inter-Card Link

Frame format

```
[SOF 0x7E][dst][src][ctrl][len][payload …][crc_hi][crc_lo]
```

Item	Value
CRC	CRC16-CCITT, poly `0x1021`, init `0xFFFF`, over `dst..payload`
Maximum payload	**200 bytes**
Overhead	7 bytes
Addresses	`0x00` = broadcast; 1–254 usable; 255 reserved
Frame types	`LK_CLAIM 0x1`, `LK_CLAIM_NAK 0x2`, `LK_HEARTBEAT 0x3`, `LK_REQ 0x4`, `LK_REP 0x5`, `LK_EVT 0x6`

> **This is a clean break from the previous generation's intercom protocol. There is no interoperability** between a ZT-100-C card on this firmware and one on the previous firmware. Cards must be upgraded as a pair.

Engine behaviour

- **Masterless.** Addresses are claimed by UID hash. Heartbeats run at **1 Hz**. Peers are advisory — a card alone on the bus claims its address and runs.
- **One outstanding request at a time** per link engine. This is what limits `cap` proxy concurrency.
- CAP raises its remote-exec session to **commission tier**; SIGNAL's link session stays at **operator tier**.

SIGNAL's own heartbeat

SIGNAL's heartbeat carries **no payload** (length 0). It advertises presence only, so CAP can never mistake it for enclosure measurement data.

Link mirror and staleness (SIGNAL)

Behaviour	Detail
Quality source	Every mirrored write carries the **source quality byte from CAP's payload**. SIGNAL never invents a quality
TTL	**2500 ms** default; SIGNAL leaves it at the default
On TTL expiry, or unknown payload version	**All mirrored variables go** `stale`
Downstream effect	`aout` drives the 3.6 mA alarm; HART raises PV-out-of-limits
Recovery	**Automatic** on the next good heartbeat — no operator action
Peer adoption	`peer_addr = 0` → adopt the **first** peer whose heartbeat parses as a **known** enclosure heartbeat version, v2 or v3 (one CAP per enclosure)

Expected behaviour on a bench pull test: removing CAP's power makes SIGNAL's mirrored variables go stale **within about 2.5 s** and the loop go to 3.6 mA.

Diagnosing the link

```
link status        # on either card: peers, addresses, counters
cap                # on SIGNAL: peer count, fresh/STALE, heartbeats, bad payloads
sys rs485          # port counters, echo_drops, tx_dropped
```

`link status` also reports the peer-reply buffer on both cards:

```
  reply:   dropped=0 overruns=0 hi=418/1024
```

Field	Meaning
`dropped`	Bytes lost because the **1024-byte** reply buffer was full
`overruns`	Whole exchanges abandoned for that overrun. A truncated reply is discarded, never delivered half-complete
`hi`	High-water mark, for sizing. A `hi` approaching 1024 means a command is returning close to the maximum reply the link will carry

A non-zero `dropped` or `overruns` means a proxied command (`cap …`) returned more than the link can assemble — not a wiring fault.

The heartbeat LED flash count is the no-tools version: **1 + peer count**. Two flashes on both cards is a healthy pair.

Persistence and Record Store

Store layout

Item	Value
Medium	Raw QSPI sectors — **not a filesystem**
Device	GD25Q16, 2 MB, 4 KB sectors, 256-byte pages
Layout	**12 slots × 2 sectors = 8 KB slots**, base sector 0
Ownership	The whole QSPI belongs to the record store
Payloads	JSON

Record ids

Id	Record	CAP	SIGNAL
1	`REC_CAPCAL`	●	—
2	`REC_PROBE_OFFSET`	●	—
3	`REC_FIELDCAL`	●	—
4	`REC_CFG`	●	●
5	`REC_CRASHLOG`	●	●
6	`REC_AOUT_TRIM`	—	●

CAP deliberately keeps ids 1–5 identical to the single-card ET-25001 product so that calibration exports and imports interchange between the two card types.

Write semantics

- **Writes are asynchronous.** A `cfg set` or a calibration commit only *queues* the flash write; the record lands over the next few seconds of store ticks. Progress is visible via `sys nvm`.
- **One shared payload buffer** serves every writer and the export path — 8 KB on CAP, 4 KB on SIGNAL. A writer that finds it busy **fails honestly** ("*nvm busy — retry in a few seconds*") and changes nothing.
- **Defaults-only operation:** if the flash chip is absent or the store fails to mount, the card still runs on defaults. `cfg set` stays RAM-only (the persist hook is attached only when the store mounted) and `sys nvm` reports `flash: ABSENT (defaults-only)`.
- Boot order: defaults first, then the persisted overlay.

`sys nvm`

Reports flash presence, mount state, store state, `writes_ok`, `writes_failed`, `crc_rejects`, torn-slot counts, and which records loaded at boot. **Use it after every commit** to confirm the write drained before power is removed.

Diagnostics, Forensics and Watchdog

LED vocabulary — CAP and SIGNAL

Both cards carry a **D13 red heartbeat LED** and an onboard **APA102 RGB "DotStar"**. Both are driven only by the `monitor` unit.

Heartbeat LED:

State	Pattern
Healthy	**Burst of (1 + link peer count) short flashes**, 150 ms on / 150 ms off, then a 500 ms pause. Alone = 1 flash, one peer = 2
Stall latched (about to watchdog)	Fast toggle, **100 ms** interval

RGB LED — precedence order, highest first:

1. Flash / blink override (brightness 20)
2. Stall latched → **red fast flash**, 100 ms on/off (brightness 15)
3. Steady mode colour (brightness 12)
4. Idle → **green breathing**, RGB (0,255,0), brightness ramping 2 → 10 → 2 over 500 ms

State	Colour
Idle / healthy	Green breathing
Progress stall latched	Red fast flash
Calibration auto-abort (**CAP only**)	**Purple (160, 0, 200) steady for 1500 ms**

> **CAP and SIGNAL implement no mode colours and no button feedback blinks.** Their RGB LED shows only green breathing, red fast flash and — CAP only — the purple auto-abort blip. **Do not document or expect any other colour on these cards.** Colour vocabularies in earlier documents belong to the single-card ET-25001 product.

The RGB LED is written only when the rendered colour actually changes, and is switched off on unit stop.

> **Neither indicator is visible from outside an assembled IME 9080 enclosure.** Both are diagnostic aids for a technician with the enclosure open — no indication reaches an operator in the field, so nothing in the installation may depend on seeing them.

Watchdog and stall attribution

- The hardware watchdog is armed **last** in boot, at **8000 ms** requested. The actually armed value is reported by `sys uptime` as `wdt: <n> ms timeout`.
- The `monitor` **unit is the only watchdog feeder**, and it feeds strictly while the scheduler says it is safe to.
- Each unit declares a **progress deadline**: **250 ms** for `meas`, **500 ms** for everything else. Missing it is a **stall**.

- A stall must be **continuously present for 1000 ms** before it latches — a single transient is forgiven.
- Once latched: the monitor writes a forensics stall marker **naming the unit**, stops feeding the watchdog, the heartbeat LED goes fast (100 ms) and the RGB LED goes red fast-flash. **The card visibly signals "about to reset" in its final seconds.**

`sys stallinject` (factory tier) demonstrates exactly this.

Forensics across resets

- On a fault the platform handler writes PC / LR / SP / CFSR / HFSR plus a magic tag into 8 backup registers, then resets.
- On the next boot the forensics collector reads the reset cause and the backup registers and **clears the tag** — evidence is consumed exactly once. The composition writes a JSON crash record via `REC_CRASHLOG` and announces it in the boot banner.
- Reset causes reported: `por`, `bod`, `ext`, `wdt`, `syst`, `other`.
- Boot banner format: `<product> <version> | reset: <cause>`, plus `| <crash summary>` when evidence is present.
- A WDT reset with **no** stall marker means a hard in-poll hang the scheduler could not attribute. It is recorded as a **hard hang** with no PC.

`sys boot`

Card	Prints
CAP	Reset cause, crash summary, **and the last persisted crash record as raw JSON**
SIGNAL	Reset cause and crash summary only — **not** the stored record

Capture `sys boot` from both cards before power-cycling a unit that has misbehaved. Evidence is consumed once.

Fault injection

Command	Tier	Effect
`sys faultinject`	factory	Forces a hardfault by reading unmapped address `0x20080000`
`sys stallinject`	factory	Forces a scheduler stall, demonstrating the full stall → LED → watchdog path

Both are demonstration tools for verifying that forensics and the watchdog work on a given unit. **Do not run them on a commissioned instrument.**

Supply rail — not observable

The 24 V rail is not detectable in firmware on this hardware revision. A rail watch was built and removed: I²C parts back-power through the bus pull-ups, and the AD5420 rides hold-up charge and answers readback long after switch-off — so any firmware "rail present" test would lie. The stated fix is an ADC divider on a future PCB revision.

Consequences: there is no supply reading, no low-supply alarm, and no brown-out diagnostic on the console. Supply problems must be measured at the terminals.

Firmware Upgrade and Migration

Per-card upgrade

Firmware is built and flashed **per card**. Version strings bump independently. After any upgrade, record both:

```
sys version
cap sys version
```

The cutover from the previous generation

> **At the cutover the QSPI record store is reformatted. Data written by the previous-generation firmware is NOT read in place.**

The previous generation stored its data in an old filesystem layout that this firmware does not read and that is destroyed on the first record write. Calibration therefore does **not** survive the cutover by itself.

Calibration migrates by JSON interchange, using the export path:

1. On the old firmware, export the calibration data.
2. Import it into the new firmware over the console / application path.
3. Verify with `cap capcal show`, `cap probe show`, `cap fcal show` and `out trim`.
4. Confirm the records persisted with `sys nvm` before removing power.

On this firmware the export side is:

```
cal export capcal
cal export probeoffset
cal export fieldcal
```

which streams the **current live** calibration as interchange JSON in 192-byte chunks every 25 ms. Synthetic span anchors are never exported.

The specific tooling, the import path, and the step-by-step cutover procedure for the ZT-100-C are **[TBC]**. Do not attempt a production cutover until they are documented.

> Records live on QSPI and survive a reflash of the same firmware generation. It is only the generation cutover that reformats.

Factory Procedures

Card-level bring-up

CAP:

```
sys version                # confirm zt-100c-cap 0.16.37-cap-b38
sys boot                   # reset cause should be por on a fresh power-up
unit list                  # all five units running
sys nvm                    # flash present and mounted
meas read                  # probe, reference, temperatures, all with qualities
```

Expected at bring-up with the probe in air and a PT100 connected:

Variable	Expectation
`ad0` / `ad1`	`good`, plausible values (see `capcal show` for the raw columns)
`cap_pf`	`good`
`temp_proc`	`good`, near ambient. `fault` **means the PT100 is not connected** — the card will substitute 25 °C
`temp_pcb`	`good`, near ambient
`wcut`	`good` once a calibration table exists

SIGNAL:

```
sys version                # confirm zt-100c-signal 0.17.35-sig-c59
unit list                  # all eight units running
sys nvm
sys rs485                  # both ports, classifier counters
out status                 # loop state and output state
hart status
```

Capacitance calibration rig (production)

This is the `capcal` interactive editor procedure on CAP. Connect to CAP's USB console.

The production rig uses **8 known capacitors**:

0, 47, 100, 220, 470, 750, 1000 and 1500 pF.

The point count is operator-chosen up to a table capacity of **16**. Capacitor tolerance and dielectric type for the production rig are **[TBC]** — specify before the rig is built.

```
auth commission 1234
capcal clear                # empty the WORKING table (live cal untouched)

# with no capacitor fitted (open circuit):
capcal add 0                # ~6 s capture, streams [n/16]

# fit each reference capacitor in turn, allowing the reading to settle:
capcal add 47
capcal add 100
capcal add 220
capcal add 470
capcal add 750
capcal add 1000
capcal add 1500

capcal show                 # review: Known pF | VRatio | AD0 | AD1 | Board Temp
capcal save                 # commit working -> live; requires >= 2 points
sys nvm                     # WAIT for the write to drain before power-down
```

Checks before saving:

Check	Why
Every row's **AD0 / AD1 are plausible**	The raw columns are shown precisely so a bad probe or rig connection is caught here
Board temperature is consistent across all rows	A drifting board temperature during the run smears the compensation
Points exist at or below 10 pF and at or above 1000 pF	Otherwise synthetic span anchors are superimposed and an "incomplete calibration" warning is raised at save
No "**incomplete calibration**" warning at save	If it appears, add real points at the range ends

Acceptance criteria for the resulting table (monotonicity, linearity error, permitted board-temperature spread during the run) are **[TBC]** for this firmware.

Probe zero (meter zeroing)

Performed with the probe fitted, empty and dry. These are CAP commands — run them on CAP's own console, or prefix each with `cap` from SIGNAL.

```
auth commission 1234
probe show                  # confirm geometry matches the fitted probe
probe set 52.5 42.16 328    # if it does not (od id len, mm)
cal start capzero
cal enter                   # step 1/2 acknowledgement
# step 2/2 captures for ~6 s
cal commit
sys nvm
```

To discard a probe offset: `probe clear`.

> **From SIGNAL the session must authenticate itself first.** `cap auth commission 1234`, then the same sequence with a `cap` prefix. Forwarded sessions used to start at commission automatically; since firmware 0.16.37-cap-b38 they do not, because `cap` is an operator command and that made the SIGNAL console a route to commission-tier work on CAP with no code at all.

Zero and span trim (SIGNAL)

Requires a calibrated reference meter in a **closed** loop.

The trim flows are **operator** tier since 0.17.35-sig-c59 — a meter verifies what the card drives, it does not define what the card believes — so no `auth` line is needed. `out status` still is: the loop must be closed.

```
out status                  # the loop MUST read closed
cal start trim4             # output goes to a fixed 4.000 mA immediately
cal enter <measured mA>     # accepted band 3.0 .. 5.0
cal commit
cal start trim20            # fixed 20.000 mA
cal enter <measured mA>     # accepted band 18.5 .. 21.5
cal commit
sys nvm
out trim                    # record both offsets
```

Re-run to converge — the correction is additive, so a second pass drives the residual to zero.

To reset: `out trim clear`.

Output verification

```
auth commission 1234
out test 4
out test 12
out test 20
out test 3.6                # verify the alarm current
out test 20.4               # verify the saturation-high current
out auto
```

Each `out test` holds for **30 seconds** then reverts by itself. In FIXED mode the saturation rewrite is suppressed, so out-of-span bench values are driven as commanded.

Pair integration test

With both cards powered and the internal link connected:

Step	Command	Expect
1	Heartbeat LED on both cards	**2 flashes** per burst
2	`cap` on SIGNAL	Peer count 1, CAP fresh, heartbeats climbing, all mirrored values with qualities
3	`cap meas read`	CAP's measurement set
4	`out status`	Loop closed, state `normal`, current matching the mapping law for the current `pv.range`
5	**Pull CAP's power**	Within about **2.5 s**: SIGNAL's mirrored values go `stale`, `out status` state goes `alarm`, the loop reads **3.6 mA**, and the heartbeat LED on SIGNAL drops to **1 flash**
6	**Restore CAP's power**	Values return to `good` and the loop resumes tracking **automatically**, with no operator action

Step 5 and 6 are the acceptance test for the safety-relevant change in this firmware. Run them on every unit.

HART verification

With a closed loop and a HART master:

Step	Expect
Command 0	Identity: device type `0x26E1`, HART rev 5, device rev 1, sw rev 16, hw rev 1
Command 1	PV = water cut, unit code **49**
Command 2	Loop current and percent of range
Command 3	PV / SV / TV = water cut / process temperature / capacitance
Command 151	Diagnostics; `measurementCount` climbing (heartbeats consumed)
Open the loop	**HART stops answering entirely** — the modem is held in reset. This is correct
Close the loop	HART resumes

The HART test rig needs a defined loop impedance — nominally **250 Ω**. That resistor is **not fitted on the SIGNAL card**: the only series element in the loop output is R201, 33 Ω. The rig must supply it, as must a real installation.

Records to keep per unit

Item	Source
Firmware versions, both cards	`sys version`, `cap sys version`
Capacitance calibration table	`capcal show` and `cal export capcal`
Probe geometry and offset	`probe show`, `cal export probeoffset`
Field calibration state	`fcal show`, `cal export fieldcal`
Output trim offsets	`out trim`
Configuration	`cfg list` on both cards
Store health	`sys nvm` on both cards

Troubleshooting

Diagnostic entry points

```
cap                # SIGNAL: link health + every mirrored value with quality
out status         # SIGNAL: loop state, output state, AD5420 status, control writes
hart status        # SIGNAL: modem/engine state, last 143/144 outcome, cmd-147 cache
sys rs485          # port counters, classifier verdicts, echo_drops, tx_dropped
link status        # link peers, counters, reply-buffer drops/overruns/high-water
cap meas read      # CAP: the whole measurement set with qualities
sys nvm            # store health
sys boot           # reset cause + crash evidence
sys uptime         # uptime and armed watchdog timeout
unit list          # composed units and their state
```

Loop at 3.6 mA

Discriminator	Command	Reading	Meaning
First	`out status`	state `sat-lo`	The water cut really is below zero — probe or probe-zero problem
		state `alarm`	The source is stale or faulted — continue
Then	`cap`	CAP **STALE**, peer count 0	The link is down or CAP is dead
		CAP fresh but `wcut` `fault`	The measurement itself failed — see below
Then	heartbeat LED, both cards	1 flash on either	The cards cannot hear each other
Then	`sys rs485`, `link status`	rising error counters	Bus noise, wiring or termination

Remember the timing: the loop reaches alarm about **2.5 s** after CAP stops broadcasting, and recovers automatically. **A loop that alarms and recovers repeatedly means an intermittent internal link**, not an intermittent process.

Loop at 20.4 mA

Water cut above `pv.range`. Compare `cap meas read` against `cfg get pv.range`.

`wcut` quality `fault`

Check	Command	Look for
Are the raw ADC readings good?	`cap meas read`	`ad0` / `ad1` **`good`** means the front end is alive and the maths failed. `fault` means a driver or bus error
Is there a calibration table?	`cap capcal show`	Point count and `factory_calibrated`
Is the table plausible?	`cap capcal show`	Implausible AD columns mean a bad probe or rig connection
Is the geometry right?	`cap probe show`	Compare against the fitted probe

`wcut` quality `underrange`

Either a genuine sub-zero reading, or a **bridge-domain error** (AD1 ≤ AD0 — the probe is reading below the bridge zero). In the bridge-domain case the value is published with a 0 pF clamp. Check the probe connection and the probe zero calibration.

`temp_proc` quality `fault`

The PT100 is missing or faulty. The card substitutes a fixed **25 °C** and keeps measuring, retrying the sensor every 500 ms.

Do not accept this as normal — the meter is running uncompensated. Check the PT100 wiring at the CAP card and the 3-wire configuration.

RS485 console silent or garbled

Order	Check
1	**Swap A and B** (terminals 7 and 6). This is the single most likely fix
2	Baud rate — **115200** by default, not 9600. 8N1, no flow control
3	Terminal **local echo** on — the firmware never echoes keystrokes, so a silent terminal proves nothing
4	Correct terminals — 8/7 is the external console; 3/4 is the internal link
5	`sys rs485` on SIGNAL via USB — check the classifier verdict. Console text is only answered when it is classified `TEXT`: an `UNKNOWN` burst is counted and **dropped**, and a `MODBUS` burst goes to the Modbus slave instead. Either way your line is **not answered as a command**
6	`tx_dropped` climbing — a reply exceeded the TX ring plus the 512-byte pending tail
7	Another master on the bus

Never diagnose this by matching wire colours.

HART not responding

Order	Check
1	`out status` **— is the loop closed?** While the loop is open the modem is held in reset and nothing will answer
2	`hart status` — modem and engine state (the `devcmd:` and `cache:` lines concern calibration, not connectivity)
3	Poll address — `cfg get hart.polladdr`, default 0
4	Master framing — 1200 baud, 8 data bits, **odd** parity, 1 stop bit
5	Loop resistance — HART needs a defined loop impedance, nominally **250 Ω**, and it is **not fitted on the card**. Confirm the loop actually contains one
6	Is the command actually implemented? 140, 145 and 146 answer rc 64 by design

HART calibration command "accepted" but nothing changed

Commands 143 and 144 are **asynchronous**. "Accepted" means queued — the work happens on CAP, one link round trip away. Verify with command 147 or `cap fcal show`. If a request returned **rc 32 (busy)** the single pending slot was occupied — wait and retry.

Run `hart status` and read the `devcmd:` line. If CAP refused the command, that line names the reason in CAP's own words (no saved reading, dielectric out of band, store busy) and counts the

failures. The same condition raises the "more status available" bit on command 147, so a master polling 147 sees it too.

Note that command 147 reads a cache refreshed every 30 s; a reading older than 95 s also raises that bit. The `cache:` line of `hart status` distinguishes the two: `EMPTY` or a large `age=` means staleness, while a valid recent cache with a `FAILED` `devcmd:` means the calibration itself was refused. A rising `ovf=` means the summary reply overflowed its accumulator and was deliberately discarded rather than published half-parsed.

`cap <command>` fails or times out

Cause	Symptom
Link down	`cap` with no arguments shows peer count 0
Slot busy	A second concurrent `cap` is refused immediately. A single `cap` defers one ~300 ms retry first
Reply too long	Replies are capped at about 400 bytes
Streaming attempted	Streams do not stream through the proxy — use CAP's USB console
More than 5 tokens	Only 5 positional tokens are rejoined

`nvm busy`

One shared payload buffer serves every writer and the export path. A writer that finds it busy fails honestly and **changes nothing**. Wait a few seconds and repeat; watch `sys nvm`.

Card resetting repeatedly

Red fast-flashing RGB plus a fast heartbeat = a latched stall, watchdog reset imminent. After the reset:

```
sys boot          # reset cause + crash summary; CAP also prints the stored JSON record
sys uptime        # armed watchdog timeout
unit list         # which unit is not progressing
```

A `wdt` reset with a stall marker names the offending unit. A `wdt` reset with **no** stall marker is a hard in-poll hang the scheduler could not attribute. **Capture this evidence before power-cycling — it is consumed exactly once.**

Known items to watch on real hardware

Item	Status
`out_ma` quality flapping to `fault` while the loop is fine	Flagged as a risk: it would mean the single-frame AD5420 status readback returns stale bits. The fix would be a two-frame readback or debounce. Whether it occurs on production silicon is unrecorded — **report it if seen**
Trim at exactly 4.0 / 20.0 mA clamping one-sided within the 4-20 range	Documented limitation. Revisit if the reference meter disagrees
SERCOM0 external RS485 and the HART SERCOM4 mux	A hardware note exists against one **old board revision**: "RX causes USB disconnect". Whether current boards are affected is **[TBC]**

ZT-100-C Digital Water Cut Meter — Zelentech