SRNE ASP48120SH3 (Direct Hybrid Inverter) + mbpoll Command Reference

Version: 1.1 **Prepared For**: Field & Deployment Teams **Platform**: Raspberry Pi / Linux **Tool**: mbpoll / pymodbus

1. Purpose

Complete reference for the SRNE ASP48120SH3 three-phase hybrid inverter over Modbus RTU: what can be READ, what can be WRITTEN, and the tested limits. Validated against the live unit at Nansana on 2026-07-21 (slave address 1).

2. Hardware & Software Requirements

- SRNE ASP48120SH3, RS485 to USB converter (CH340), Raspberry Pi / Linux PC.
- Wiring: A(converter) to WiFi-port RJ45 pin 7 (RS485-A), B to pin 8 (RS485-B), GND to pin 2. The WiFi port carries the monitoring bus; the RS485/CAN port is for the battery BMS.
- Install: `sudo apt install mbpoll` and/or `pip install pymodbus`.

3. Serial parameters and function codes

- 9600 baud, 8 data bits, No parity, 1 stop bit (8N1). Slave address 1.
- READ: function code 03 (holding registers). 16-bit registers with scale factors (not 32-bit floats).
- WRITE: BOTH FC06 (write single) and FC16 (write multiple) are supported and tested. Note: the CHINT meters reject FC06, the inverter does NOT.
- mbpoll read flags: `-b 9600 -P none -s 1 -t 4 -0`.

4. Reading values (exhaustive, all ranges respond)

Every documented range answers FC03. Ranges (SRNE V1.7 segment map, confirmed on this unit):

Product info (P00), 0x0000 to 0x0049

- 0x000B machine type = 4 (integrated inverter-controller)
- 0x0014 software version = 925
- 0x001A RS485 address = 1 (read only)

Controller / battery (P01), 0x0100 to 0x0122

- 0x0100 SOC (%), 0x0101 battery V (x0.1), 0x0102 battery I (x0.1 signed, + charge / - discharge), 0x0103 device temp (reads 0, unused)
- 0x0104 to 0x0106 DC load V/I/P (0, no DC load on this unit), 0x0107 to 0x0111 PV panels 1/2 (0, no PV), 0x010B charge state (1 = quick charge)

Inverter (P02), 0x0200 to 0x0237 (the main live telemetry, see the register-map page)

- state 0x0210, bus 0x0212, grid+inverter phase A 0x0213 to 0x0218, load 0x0219 to 0x021F, heat-sinks 0x0220 to 0x0223, PBus/NBus 0x0228/0x0229, 3-phase B/C V/I 0x022A to 0x022F, 3-phase B/C load 0x0230 to 0x0237.

Settings / parameters, 0xE001 to 0xE02D (writable, see section 5). Raw values present; battery voltage thresholds around 0xE006 to 0xE00E appear as 48V-system values (register/10 x 4), e.g. 144 = 57.6 V. Exact per-register meaning is TBD from the ASP settings doc.

Settings F, 0xF000 to 0xF01F: additional parameters (raw values present).

Control commands, 0xDF00 to 0xDF09: write-only, read back as 0.

Example reads (mbpoll):

```
# battery block 0x0100 (dec 256), 4 regs
mbpoll -m rtu -b 9600 -P none -s 1 -t 4 -0 -r 256 -c 4 -a 1 -1 <port>
# inverter block 0x0210 (dec 528), 32 regs
mbpoll -m rtu -b 9600 -P none -s 1 -t 4 -0 -r 528 -c 32 -a 1 -1 <port>
# settings 0xE001 (dec 57345), 45 regs
mbpoll -m rtu -b 9600 -P none -s 1 -t 4 -0 -r 57345 -c 45 -a 1 -1 <port>
```

5. Writing values (tested 2026-07-21)

The inverter accepts BOTH FC06 and FC16 and enforces a permission model:

- Read-only registers (telemetry, product info, e.g. 0x001A) REJECT writes with exception code 7 ("parameter is read only"). Tested.

- Settings registers (0xE0xx) ACCEPT writes (both FC06 and FC16), with no password or run-state lock on the tested registers (0xE015, 0xE01F, 0xE024). Tested via no-op write-back (wrote the current value, confirmed unchanged).
- Control commands (0xDF00 to 0xDF09) are write-only actions. NOT fired on the live unit because it feeds customers. Documented below.

Exception code table (SRNE V1.7): 1 illegal function, 2 illegal data address, 3 illegal data value, 4 operation failed, 5 password error, 7 read only, 8 cannot change while running, 9 password protection, 10 length error, 11 permission denied.

Control command registers (DANGEROUS on a live system, documented only):

- 0xDF00 power ON/OFF: 1 = on, 0 = off. OFF cuts inverter output, do NOT fire while it feeds customers.
- 0xDF01 reset: 1 = reset.
- 0xDF02 restore defaults: 0xAA wipes all settings.
- 0xDF03 clear current alarm: 1.
- 0xDF04 clear statistics: 1.
- 0xDF05 clear history: 1.
- 0xDF08 sleep/run: 0x5A5A sleep, 0xA5A5 run.

How to write:

```
# mbpoll FC06 (write single): value as the last argument
mbpoll -m rtu -b 9600 -P none -s 1 -t 4 -0 -r <dec_addr> -a 1 <port> <value>


# pymodbus
client.write_register(address=REG, value=V, device_id=1)     # FC06
client.write_registers(address=REG, values=[V], device_id=1) # FC16
```

6. Tested limits summary

- CAN: read every range over FC03; write settings 0xE0xx with FC06 or FC16; read-only registers are protected (code 7).
- CAUTION: control commands 0xDF00 to 0xDF09 are live actions (power, reset, restore). Documented, NOT fired on the customer-feeding unit. Only fire with an outage window and explicit sign-off.
- UNKNOWN / next: exact meaning of each 0xE0xx and 0xF0xx setting (needs the ASP settings doc or careful change-and-revert testing offline); machine-state codes other than 2 (verify by observing transitions).

7. Preparing for OpenEMS integration

The validated telemetry map is a declarative table in `src/inverter/registers.py` (nfe-modbus-energy-logger), grouped by OpenEMS nature (ElectricityMeter grid/backup, SymmetricEss battery, OffGridBatteryInverter state). It maps 1:1 to a GoodWe-style defineModbusProtocol (FC3ReadRegistersTask + word elements + SCALE_FACTOR). Control writes map to FC16WriteRegistersTask on the 0xDF00 area and 0xE0xx settings.

Revision #3
Created 2026-07-20 20:35:02 UTC by hillary.arinda
Updated 2026-07-20 20:44:50 UTC by hillary.arinda