Obsah

Komunikační rozhraní `npa_r04_rock`

Platí pro verzi NPA R04: 2.15.beta2 · Vytvořeno: 2026-06-17 · Aktualizováno: 2026-07-16

Tento dokument popisuje všechna komunikační rozhraní jednotky NPA R04 — jak ta, na kterých jednotka **poskytuje data** (servery), tak ta, přes která **sama komunikuje** s okolím (klienti), a hardwarová sběrnicová rozhraní na desce ROCK.

> **Detailní popisy TCP serverů podle portu** jsou v `tcp/`: - `tcp/port-502-modbus-tcp.md` — Modbus TCP (kompletní register mapa) - `tcp/port-2404-iec104.md` — IEC 60870-5-104 server (NPA jako RTU) - `tcp/port-7777-json-rpc.md` — JSON-RPC API - `tcp/port-8080-http.md` — HTTP/HTML webserver - `tcp/README.md` — přehled TCP portů
>
> **Cloudoví klienti:** - `config-upload.md` — odesílání konfigurace na server (`POST /config`)

Přehled:

#	Rozhraní	Směr	Protokol / médium	Port / zařízení	Zdroj
1	Modbus TCP server	server	Modbus/TCP	konfig. `ns_netPort` (typ. 502)	`nmodbusrv.cpp`
2	Modbus RTU slave	server	Modbus/RTU (RS-485)	`/dev/ttyS2`	`nmodbusrv.cpp`
3	Modbus klient (drivery zařízení)	klient	Modbus TCP / RTU / přes VPN-ID	`dd_port` (502) / `/dev/ttyS2`	`device/devicemodbus.`
4	HTTP/HTML webserver	server	HTTP/1.x	TCP **8080**	`server.cpp`, `nhtml.cpp`
5	JSON-RPC server	server	HTTP + JSON	TCP **7777**	`njsrv.cpp`
6	JSON-RPC klient	klient	HTTP + JSON	cíl `:7777`	`njcli.cpp`
7	Remote log (cloud)	klient	HTTP POST + JSON	`http://<vpn-srv>/measuring`	`nremotelog.cpp`
8	Modules discovery	klient	HTTP POST + JSON	`http://<vpn-srv>/moduleslist`	`nmodules.cpp`
9	VPN (tunel)	médium	OpenVPN tun	rozhraní `tun_npa_r04`	`util/nutils.cpp`

#	Rozhraní	Směr	Protokol / médium	Port / zařízení	Zdroj
10	RS-485 nízkoúrovňový	obojí	sériová linka	`/dev/ttyS2`	`nrs485.cpp`
11	I²C (binární I/O, FRAM, D/A)	obojí	I²C	sběrnice 0	`ni2c*.cpp`, `nfram.cpp`, `nda.cpp`
12	GPIO (relé, tranzistor, LED, watchdog)	obojí	GPIO/sysfs (mraa)	`/dev/watchdog` ad.	`niopins.cpp`, `nbinout.cpp`
13	IEC 60870-5-104 server	server	IEC 104 (APCI/ASDU)	konfig. `i104_port` (typ. 2404)	`niec104srv.cpp`

---

1. Modbus TCP server (`NModbusSrv`)

Hlavní rozhraní pro nadřazené systémy (SCADA, RTU distributora). Implementováno přes `QModbusTcpServer`, singleton `NModbusSrv::GetNModbusSrv()`.

- **Naslouchá** na `0.0.0.0`, port = `mC2["ns_netPort"]`, Modbus server address (Unit ID) = `mC2["ns_netId"]` (výchozí 1).
- **Mapa registrů** (`setMap`):
  - Coils: `0 … G_NUM_NOIL` (`G_NUM_NOIL = 5`)
  - Discrete Inputs: `0 … 0xE`
  - Input Registers: `0 … 60000`
  - Holding Registers: `0 … 60000`
- Obsah registrů se průběžně zrcadlí z `NState` ve funkci `Loop()` (volaná z 50 ms a 500 ms tick smyček v `main.cpp`). Zápisy klienta se zpracují v `updateRegisters()` přes signál `dataWritten`.

Konvence pořadí slov

Hodnoty se zapisují pomocnými funkcemi, které vynucují pořadí registrů: `WriteMTUint16/32/64`, `WriteMTFloat` (BE), `WriteMTFloatLE`, `WriteMTDoubleLE`, `WriteMT2bit` (dvoubitový stav OFF/ON do jednoho registru), `WriteFloatToIR`, `WriteUint64ToIR`/`WriteInt64ToIR`. Float/double se ukládá po 16bit slovech, LE varianty prohazují slova.

Rozložení adresního prostoru (mapování „NPA-R04_MODBUS-tabulka”)

Rozsah	Význam
`0x04 … 0x07`	inicializační „magic” `0x1234`
`1000 … 1203`	data z RTU pro NPA (el. i neel. veličiny, zápisem se plní `RtuMeter`)
`2000 …`	předávací místo obchodního měření (TML): stavy spínačů (2bit), I/U/P/Q/cosφ/f, energie E_del/E_con po fázích, …
`2500 …`	suma rozpadových míst, systémové chyby, povely a stavy regulace
`2535 / 2547`	**zápis** – globální BESS nabíjení / vybíjení
`2538`	**zápis** – globální povel činného výkonu výrobny (SPG, stupně P1–P4)
`2541`	**zápis** – globální povel jalového výkonu Q (QLC)
`2544`	**zápis** – globální povel účiníku cosφ (SPFG)
`2550 / 2552`	**zápis** – blokování zapnutí výrobny / BESS (sekvence galv. odpojení)
`2554`	**zápis** – galvanické odpojení zdrojů (GD_Power_supply)

Rozsah	Význam
2572 / 2576	sumy přes lokace (FVE/BESS/MVE/KGJ/BPS/VTE/DA/HE/TUV/AKU/BackUp/Wallbox)
2634	**zápis** – Power_export_limit (float)
2636	**zápis** – DS_power_value
3000 … 3000+200·20	bloky rozpadových míst (DS); offsety: +44 fs_DS_power_value, +55 Total_stop
40000 … 40000+200·20	DS zrcadlená z RTU (plní ExtDs)
50000 + 200·dd_modbusAddr	**per-device blok** každého zařízení s nastaveným dd_modbusAddr (0–9): Us12/23/31, U1–3, I1–3, P/Q/S po fázích i sumy (vše float LE)

Adresy s poznámkou **zápis** jsou holding registry, jejichž změna klientem vyvolá akci (povel regulaci, uložení stavu) v updateRegisters().

2. Modbus RTU slave (NModbuSrv, RTU větev)

Aktivuje se pouze když mC2["ns_485_m-s"] == "ns_485_slave". Sdílí stejnou mapu registrů jako TCP server (QModbusRtuSerialSlave).

- **Port:** /dev/ttyS2
- **Unit ID:** mC2["ns_485IdSl"] (výchozí 1)
- **Rychlost:** mC2["ns_485Spd"] (výchozí 115200)
- **Parita:** mC2["ns_485Parity"] → none (0) / odd (3) / even (2)
- **Data bity:** 8, **Stop bity:** mC2["ns_485Sbit"] (výchozí 1)

Pozn.: RTU server a RTU master (bod 3) sdílejí fyzicky /dev/ttyS2. Jednotka je v daném okamžiku buď master (čte zařízení), nebo slave (ns_485_m-s).

---

3. Modbus klient – drivery zařízení (DeviceModbus)

Bázová třída pro všechny Modbus drivery (FVE měniče, BESS, elektroměry, KGJ, …). Instancuje se v Devices::InitDevice() podle řetězce dd_type. Každý driver běží na vlastním mLoopTimer (výchozí mReadInterval = 500 ms).

Tři režimy komunikace (dd_comm / dd_ip_npaid):

1. **Modbus TCP** (QModbusTcpClient)
   - IP = dd_ip, port = dd_port (výchozí 502), Unit ID = dd_idNet (výchozí 1)
2. **Modbus RTU master** (QModbusRtuSerialMaster, statický, sdílený všemi RTU drivery)
   - Port /dev/ttyS2; parita/rychlost/stop bity z **ns_485*** konfigurace (stejně jako RTU slave), data bity 8
   - Přístup na linku je arbitrážován mutexem (mChannels, TryLockComm/UnLockComm); při neúspěšném zámku se loop zkrátí na KS_INTERVAL = 42 ms
3. **Přes VPN-ID** (dd_id_npaid) — NpaIdConnect()
   - dd_npaid → přes NModules (bod 8) se zjistí IP vzdálené NPA jednotky
   - přes NJCli::GetModbusSrvParam (bod 6) se získá mb_port/mb_id cílového Modbus serveru a naváže se TCP spojení

Společné: setTimeout(5000), setNumberOfRetries(3). Instrukce se řadí do zásobníku mInstr (DM_GetWord/Int32/Int64, DM_WriteWord/Float/UInt32/WordMulti) a vykonávají sekvenčně (SendInstr/onReadReady/onWriteFinished). Pomocné čtecí/zápisové funkce řeší endianitu a 10^{-}exponent (SaveWord10Exp, GetFloatLE, SaveInt64BE, …).

Podporované `dd_type`

em300, bmr_pla33_rtu/bmr_pla33_tcp, tme, mve_langer, solaredge_tcp/_rtu/_meter, simulator, rtu_meter, npa_slave_tml, npa_slave_ds, bess_gres, aku_tuv, ext_ds, janitza, da_output, quido_eth4, goodwe, gw_15_50kw_bess, gw_15_50kw_inverter, kgj_tedom, sungrow, logger1000, smean, pro380, sermatec, goldensigma, sma, eme319, eme319ds, deyesun, pulzni, sinexel.

> Nový typ zařízení: implementovat potomka Device/DeviceModbus, zaregistrovat .cpp/.h v npa_r04_rock.pro a přidat větev do Devices::InitDevice().

---

4. HTTP/HTML webserver (Server + NHtml)

Surový QTcpServer (server.cpp) poskytující konfigurační a monitorovací webové rozhraní.

- **Naslouchá:** 0.0.0.0:8080
- **Spojení:** NHttpSocket (nhttpsocket.cpp) parsuje HTTP požadavek; odpovědi skládá NHtml ze šablon v html/<lang>/ (jazyky cz/, en/, plus v2/)
- **Relace/autentizace:** cookie NsessionId, stav drží NSession (login, úroveň oprávnění, „EA" uživatel); login přes login.html (POST), odhlášení logout.html
- **Hlavičky odpovědí:** HtmlHead (text/html), JsonHead, JsHead, CssHead

Endpointy (GET, příkaz = 4. segment cesty)

- **Stránky:** index.html, index2.html, info.html, net_settings.html, basic_settings.html, factory_settings.html, settings.html, regulation.html, locations_dis.html, devices_detection.html, io_settings.html, param.html, config.html, data.html, state.html, log.html, debug.html, public.html, modbus_registers.html, reboot.html, restart.html, login.html, logout.html
- **JSON data (AJAX):** get_index_data, get_bin, get_da, getstate, getid, update_gaude, ping
- **Statika:** style.css, npa.css, fonts.css, main.js, main1.js, gauge.min.js
- **Konfigurace lokací/zařízení:** lo_add, lo_edit, lo_save, lo_delete, lo_add_device, lo_delete_device, dd_add, dd_add_tcp, dd_add_485, dd_edit, dd_save, dd_delete, setting_next, io_select_location, re_add_accu
- **Ruční ovládání / testy:** dd_test_out, dd_xch_out, outtogle, releclick, tranzistorclick, rs485click, setda, settestbatt, instr, clear_fram
- **Povely regulace (ruční):** setglobalpower, setgloballevelp, setgloballevelq, setgloballevelfi, setglobalbesspower, setglobalbesslevelch, setglobalbessleveldisch, setlocationpower, setlocationlevelp, setlocbesspower, setlocbesslevelch, setlocbessleveldisch, setdevpower, setdevlevelp, setdevbesspower, setdevbesslevelch, setdevbessleveldisch
- **Systém:** lang, password, save, restart_save, reboot.html, restart.html

POST

Větev tokens[0] == "POST" — ukládání formulářů konfigurace (Save(post) → zápis do NConfig a NConfig::Save() do /data/NPA_R04.cnf), přihlašování a změna hesla.

---

5. JSON-RPC server (NJSrv)

QTcpServer na portu **7777** (NJSrv::Init()), strojové API mezi NPA jednotkami i pro nástroje.

- **Omezení přístupu:** jen ze sítě 10.80.0.0/255.255.0.0 (mNet/mMask) — jiná IP je okamžitě odpojena.
- **Transport:** požadavek je HTTP s tělem JSON; server čte do \r\n\r\n, dle Content-Length načte tělo, odpoví HTTP/1.1 200 OK, Content-Type: application/json, Connection: close.

- **Dispatch:** `Req2Ans()` podle pole `req`. Odpověď vždy obsahuje `time` (UTC) a `res` (`"ok"` / `"command not found"`).

req	Parametry	Odpověď
`get_modbus_srv_param`	—	`mb_port` (`ns_netPort`), `mb_id` (`ns_netId`)
`set_time_log_sec`	`mTimeLogSec` (int > 0)	nastaví periodu remote logu

Smoke test: `test/test_njsrv.sh` (curl na port 7777 živé jednotky).

6. JSON-RPC klient (`NJCli`)

Odchozí protějšek bodu 5 — používá se při VPN-ID Modbus připojení (bod 3).

- `GetJ(ip, req, param)` — POST JSON na `http://<ip>:7777/`, `Content-Type: application/json`, synchronní (vlastní `QEventLoop`).
- `GetModbusSrvParam(ip)` — vrátí `mb_port`/`mb_id` vzdálené jednotky.

---

7. Remote log / cloud (`NRemoteLog`)

Periodicky (a při změnové události) odesílá měřené hodnoty do cloudu/serveru. Gated polem `mC2["ns_cloud"]` (musí být ≠ `"false"`).

- **Cíl:** IP serveru se zjistí z VPN rozhraní (`GetVpnNpaServerIp()`, bod 9).
- **Upload:** `POST http://<vpn-srv>/measuring`, `Content-Type: application/json`, tělo = JSON snímek stavu (přes `GetNetManager()`).
- **Inicializace / poslední stav:** `POST http://<vpn-srv>/getlastmeas`.
- Změny se detekují porovnáním `NState::mM` vůči `mSavedM`. Periodu lze měnit přes `SetTimeLogSec()` (i přes JSON-RPC `set_time_log_sec`).
- **Odeslání konfigurace:** při každém `NConfig::Save()` se navíc po bezpečném zápisu na disk odešle celý config (`POST http://<vpn-srv>/config`) – viz `config-upload.md`.

8. Modules discovery (`NModules`)

Zjišťování ostatních NPA jednotek v cloudu (pro VPN-ID Modbus připojení).

- **`GetModules()`** — `POST http://<vpn-srv>/moduleslist` → JSON pole modulů (`id`, `ip`, …).
- **`GetModuleIp(id)`** — vyhledá IP modulu podle `id` (pro `dd_npaid`).

9. VPN (tunel)

Veškerá cloudová komunikace (body 6–8) běží přes VPN tunel.

- **Rozhraní:** `tun_npa_r04` (`VPN_NPA_IFACE` v `util/nutils.h`).
- **`GetVpnNpaServerIp()`** vrátí IP serveru z adresního rozsahu tohoto rozhraní; když rozhraní není validní, cloudová odesílání se přeskakují.

---

10. RS-485 nízkoúrovňový (`NRs485`)

Singleton arbitrující fyzickou linku RS-485 (`/dev/ttyS2`) mimo Modbus stack — používá se např. pro vysílání sériového čísla / identifikace jednotky.

- Otevírá `/dev/ttyS2` přímo (`open`, `write`), neblokující.

- `SetConnect(bool)` přepíná směr; odesílá `mSN` (sériové číslo z `NConfig`).
- RTU Modbus drivery i RTU slave server **musí** arbitrovat přes sdílenou linku, neotevírají port samostatně.

11. I²C – binární I/O, FRAM, D/A (`NI2c`, `NI2cBoard0/1`, `NFram`, `NDA`)

Hardwarová sběrnice na desce ROCK. Přístup přes `mraa::I2c`, sběrnice **0** (`I2C_BUS = 0` v `nbinout.h`). `NBinOut` vytváří a v 50 ms smyčce obsluhuje:

- **`NI2cBoard0`** — I²C adresa `0x27` (nová deska; stará `0x23`) — expandér binárních I/O.
- **`NI2cBoard1`** — I²C adresa `0x26` (nová deska; stará `0x22`) — expandér binárních I/O.
- **`NFram`** — I²C adresa `0x50`, 16bit adresování paměti (zápis 2 B adresy + data); ukládá perzistentní stav/čítače.
- **`NDA`** (`nda.cpp`) — D/A převodníky (až 4 kanály, `mDACount`), typ kanálu dle `mC2["sTypDA<n>"]` (`nula` vs. proudový/napěťový rozsah).

12. GPIO – relé, tranzistor, LED, watchdog (`NIOPins`)

Přímé GPIO piny (mraa) obsluhované v 500 ms smyčce (`NIOPins::Loop`):

- **Relé** (`mRele.write`) a **tranzistorový výstup** (`mTranzistor.write`) — ovladatelné i z webu (`releclick`, `tranzistorclick`).
- **LED** — zelená (`mGLed`), režimová R-LED (`mRLed`, `WriteRState`).
- **Wi-Fi pin** (`mWiFiPin`) — čtení stavu / spínání AP.
- **Watchdog** — `mWatchdog.write(mWatchdogState)`, periodicky kopáno; HW watchdog na `/dev/watchdog` se otevírá při zapnutém `G_WATCHDOG` (viz `globals.h`).

---

13. IEC 60870-5-104 server (`NIec104Srv`)

Jednotka jako **RTU (řízená stanice)** protokolu IEC 104 pro dispečerské řízení distributora / SCADA. Vlastní implementace dle normy (bez externí knihovny), singleton `NIec104Srv::GetNIec104Srv()`, tick z 500 ms smyčky.

- **Naslouchá** na `0.0.0.0`, port `mC2["i104_port"]` (výchozí 2404); ve výchozím stavu vypnuto (`i104_enable`).
- **Max. 2 masteři** — ACL dle IP/masky, práva per master (monitorování × monitorování + povely), každý s vlastní frontou událostí a stavem STARTDT.
- **Monitorovací směr:** datové body z `NState` (konfigurovatelná mapa `iec104_points`) — spontánní přenos s integrálním delta kritériem, generální dotaz, cyklický přenos; typy M_SP_TB_1 / M_DP_TB_1 / M_ME_TF_1 (CP56 v UTC).
- **Povelový směr:** C_SC_NA_1 / C_DC_NA_1 / C_SE_NC_1 → `NState::SetGlobal…` se zdrojem `"iec104"`; direct operate i select-before-operate (per master).
- **Konfigurace na webu** (menu *IEC 104*): server, masteři, datové body, profily distributorů (ČEZ / EG.D / PRE / obecný), export IOA tabulky (CSV).

Detailní uživatelská dokumentace: `tcp/port-2404-iec104.md`. Technický návrh a požadavky distributorů: `zadani/iec104-navrh.md`, `zadani/iec104-pozadavky-ds.md`.

Souhrn portů a zařízení

Prostředek	Hodnota	Konfigurovatelné
Modbus TCP server	port `ns_netPort`, Unit ID `ns_netId`	ano (`/data/NPA_R04.cnf`)

Prostředek	Hodnota	Konfigurovatelné
IEC 60870-5-104 server	port `i104_port` (typ. 2404), CA `i104_ca`, gate `i104_enable`	ano (web, menu IEC 104)
Modbus RTU (slave i master)	`/dev/ttyS2`, `ns_485Spd`/`ns_485Parity`/`ns_485Sbit`/`ns_485IdSl`	ano
HTTP/HTML	TCP 8080	ne (pevně)
JSON-RPC	TCP 7777, jen `10.80.0.0/16`	ne (pevně)
Cloud (measuring/moduleslist)	`http://<vpn-srv>/…`, gate `ns_cloud`	částečně
VPN rozhraní	`tun_npa_r04`	ne (pevně)
I²C	sběrnice 0; `0x26`, `0x27`, FRAM `0x50`	ne (HW)
Watchdog	`/dev/watchdog` (`G_WATCHDOG`)	compile-time

Pozn.: v PC-debug režimu (`DEBUG_PC=1`) je watchdog i login vypnutý a logování jde na stdout (viz `globals.h`).