

Obsah

Zkoušky dle RfG (testovací sekvence činného výkonu)	1
Sekvence	1
Základ výkonu – instalovaný / fyzický (od 2.15.beta17)	1
Výběr rozpadových míst	2
Co se stane po startu	2
Pozastavit / Spustit	2
Ukončení a přerušení	3
Spouštění – HW nebo SW	3
Funkce binárních vstupů (globální, libovolné zařízení se vstupy)	3
Funkce binárních výstupů – kód kroku	3
Přiřazení vstupů a výstupů na stránce	4
Kde je vidět průběh	4
Omezení (zapíše se jako varování do logu zkoušky při startu)	4
Soubory	4
Endpointy (HTTP, admin a vyšší, odpověď JSON)	5
Klíče v configu (mC2)	5
Zdrojové soubory	5

Zkoušky dle RfG (testovací sekvence činného výkonu)

Platí pro verzi NPA R04: 2.15.beta17 · Vytvořeno: 2026-09-08 · Aktualizováno: 2026-10-06 (základ výkonu, Pozastavit/Spustit)

Pro ověření souladu výroby s požadavky RfG (Requirements for Generators) potřebuje provozovatel nechat výrobu projít **předepsanou sekvencí kroků činného výkonu** a na každém kroku setrvat zadanou dobu. NPA to umí spustit z binárních vstupů (4 DI) nebo ze softwaru (stránka **Zkoušky dle RfG**, jen admin a vyšší), průběh zobrazuje a aktuální krok signalizuje na 4 binárních výstupech jako 4bitový kód.

Sekvence

č.	kroky [% výkonu, výchozí instalovaného]	setrvání na kroku
1	100, 90, 80, 70, 60, 50, 40, 30, 20, 10, 30, 60, 100	190 s
2	100, 60, 30, 0, 30, 60, 100	90 s
3	100, 60, 30, 100, 0, 100	70 s

Toto jsou **výchozí hodnoty**. Na stránce lze kroky i časy upravit (kroky = celá čísla 0--100 oddělená čárkou, nejvýše 50; čas 1--3600 s). Prázdné pole = výchozí hodnota (na stránce zobrazená šedě). Neplatný zápis se při startu nahradí výchozí sekvencí a zapíše se do logu zkoušky. Tlačítko **výchozí** u sekvence vyplní výchozí kroky, výchozí čas setrvání i základ výkonu *instalovaný* (uloží se tlačítkem *uložit*).

Základ výkonu – instalovaný / fyzický (od 2.15.beta17)

U každé sekvence je šoupátko, z čeho se počítají procenta kroků (klíč rfg_sN_base):

- **instalovaný** (výchozí, prázdné = instalovaný): % instalovaného výkonu místa (lo_insPower). Na měniče se krok přepočte: % × instalovaný / fyzický, oříznuto na 0--100 %. Příklad: 120 kW instalovaných, 100 kW měničů → krok 50 % = 60 % měničů, krok 90 % = 108 % → 100 % měničů. Místo bez vyplněného lo_insPower jede z fyzického.
- **fyzický**: % fyzického výkonu místa = součet dd_Power měničů; povel jde na měniče beze změny.

Základ se ukládá k povelu místa jako PowerLimitBase (/data/state.cnf), přepočet dělá společná funkce NState::ProcMistaNaFyzicky - tatáž jako u stupňů P0--P4 a Px místa (viz řízení-vykonu.md, přepínač základu u Rozpadového místa). Kód DO nese vždy procento kroku sekvence, ne přepočtenou hodnotu. Při startu se do logu zapíše varování, když místo nemá instalovaný výkon, nebo když je instalovaný větší než fyzický (od kolika % se krok ořízne).

Výběr rozpadových míst

Na stránce je v kartě *nastavení* seznam vytvořených rozpadových míst (název, typ, instalovaný výkon) a u každého **fajfka „zařadit do sekvence“**. Výchozí stav = fajfka u všech míst, takže zkoušku lze udělat za celou výrobní, nebo odškrtnutím ostatních míst samostatně po rozpadových místech.

- Fajfka se ukládá do objektu místa jako lo_rfgEn (true / false); **chybějící klíč = zařazeno** (nově vytvořené místo je automaticky ve zkoušce).
- Množina zařazených míst se **zapamatuje při startu** sekvence (uložené povelů saved_limits v /data/rfg_test.json). Změna fajfek během běžící zkoušky se uplatní až u další; hlídání cizího povelu i návrat povelů se týkají jen zapamatovaných míst.
- Místo bez fajfky zkouška vůbec nemění – jede dál podle běžné regulace.
- Bez jediného zařazeného místa se start odmítne (zadne rozpadove misto neni zarazene do zkousky).
- Do logu zkoušky se při startu zapíše zarazena mista: ... a případně vynechana mista:
- Výkon výrobní v průběhu: jsou-li zařazená všechna místa, je to měření DS celé výrobní a % z celkového instalovaného výkonu; jinak součet měření DS zařazených míst a % z jejich instalovaného výkonu (místo bez vlastního DS se do součtu nepočítá). U sekvence se základem *fyzický* se % počítá z fyzického výkonu zařazených míst.

Co se stane po startu

1. Zapamatují se aktuální povelů (PowerLimit) zařazených rozpadových míst.
2. Každému zařazenému rozpadovému místu se zapíše **pevný povel** = procento kroku (NState::SetLocationRegPower). Ten má v řetězu NState::PozadovanyVykon přednost před výstupem PI regulace, takže regulace podle distribuce / přetoku je po dobu zkoušky obehita. Týká se jen měničů FVE (zařízení s pwr); **BESS se nedotýká**.
3. Po uplynutí doby kroku se zapíše další krok. **Čas setrvání se počítá od zápisu povelu kroku**, ne od ustálení výkonu (rampy měničů tedy čas kroku zkracují). Po posledním kroku se **vrátí původní povelů míst** a zkouška je DOKONČENA.

Pozastavit / Spustit

Tlačítko sekvence mění text podle stavu: **Start sekvence x** → během běhu této sekvence **Pozastavit** → po pozastavení **Spustit**.

- **Pozastavit** platí okamžitě: odpočet kroku se zastaví, výrobní **drží aktuální krok**, DO dál nesou jeho kód. Stav je POZASTAVENA, v tabulce kroků „pozastaveno, zbývá ...“.
- **Spustit** pokračuje **zbývajícím časem** kroku.
- Pozastavit/Spustit funguje i u sekvence spuštěné z HW (nic nespouští). Během pauzy platí vše jako za běhu: povel z jiného zdroje zkoušku přeruší, rozepnutí RfG-enable (HW) ji zastaví, restart NPA vrátí původní povelů.
- **Stop = zrušení sekvence** (i pozastavené) a návrat původních povelů míst; znovu lze sekvenci spustit jen od začátku.

Stav zkoušky (fáze, sekvence, krok, uložené povelů míst, log) je v /data/rfg_test.json. Zápis kroků do měničů jde přes běžný mechanismus driverů (akt_proc), tj. respektuje nastavené rampy zdroje (dd_rampUpEn/DnEn; snížení kroku obejde rampu dolů přes dd_rampIgnDsr, zvýšení se rampuje) a vypnuté ovládání zařízení (dd_ctrlEnable=false – takové zařízení povel nedostane).

Ukončení a přerušení

Důvod	Výsledek	Co se stane
poslední krok uplynul tlačítko Stop (zrušit sekvenci, i pozastavenou)	DOKONČENA (ok) ZASTAVENA (stop)	původní povely míst vráceny původní povely míst vráceny
DI RfG-enable rozepnut (HW režim)	ZASTAVENA (stop)	původní povely míst vráceny
povel z jiného zdroje během zkoušky (dispečink P0--P4 / procenta, ruční povel místa, Modbus, IEC 104)	PŘERUŠENA (abort)	místa, která jiný povel nezměnil, se vrátí; místo změněné cizím povelům zůstává tak, jak ho nastavil - povel dispečinku má přednost
restart NPA během zkoušky	PŘERUŠENA (restart)	po startu se jen vrátí původní povely míst; zkouška se neobnovuje (povel místa je v /data/state.cnf, bez návratu by výroba zůstala omezená na posledním kroku)

Start další sekvence během běžící zkoušky se ignoruje (zapíše se do logu).

Spouštění - HW nebo SW

Volba **Spouštění sekvencí** na stránce (klíč rfg_src):

- **SW** (výchozí): tlačítka *Start sekvence 1/2/3* na stránce; binární vstupy se nepoužívají (jejich stav se jen zobrazuje).
- **HW**: binární vstupy s funkcemi níže; tlačítka Start jsou zakázána. **Stop** na stránce funguje v obou režimech.

Funkce binárních vstupů (globální, libovolné zařízení se vstupy)

Funkce	Význam
RfG-enable	povolení zkoušek - úroveň : 1 = povoleno; rozepnutí během zkoušky = stop
RfG-seq1 ... RfG-seq3	start sekvence 1--3 - náběžná hrana (0 → 1), jen když je RfG-enable = 1

Vstupy se čtou každých 500 ms přes NState::GetBinValue (OR přes všechny kanály s danou funkcí, negace io_inp_neg se uplatní). Hrana se detekuje v automatu, takže stačí puls delší než ~1 s.

Funkce binárních výstupů - kód kroku

Funkce	Význam
RfG-bit0 ... RfG-bit3	bit 0 (LSB) ... bit 3 (MSB) kódu aktuálního kroku

Kód = zaokrouhleno(procento kroku / 10):

kód (bit3 bit2 bit1 bit0)	výkon
0000	0 %
0001	10 %
0010	20 %
0011	30 %
...	...
1010	100 %

Mimo běžící zkoušku je kód **0000**. Krok, který není násobkem 10 (např. 55 %), se zaokrouhlí (→ 0110 = 60 %). Na kód se jako u ostatních funkcí uplatní negace výstupu (io_out_neg) a testovací režim výstupů zařízení.

Přiřazení vstupů a výstupů na stránce

Karta *Přiřazení vstupů a výstupů* vypisuje pro každou funkci (RfG-enable, RfG-seq1..3, RfG-bit0..3) zařízení a pin, kde je přiřazena (NHtmL::RfgIoTable, prochází input[] / output[] všech zařízení), včetně poznámky o negaci. Nepřiřazená funkce je vypsána **červeně „není přiřazen“**. Tabulka se vytváří při načtení stránky.

Kde je vidět průběh

Během běžící zkoušky se **na všech stránkách** webu (nahore, nad obsahem) zobrazuje červené varování *Probíhá zkouška dle RfG - ...* s odkazem na stránku zkoušek (NHtmL::RfgWarning, vkládá se do menu*.html přes {{rfg_warning}}). U admina a vyšších se každých 5 s obnovuje z rfg_status, u uživatele odpovídá stavu při načtení stránky.

Stránka **Zkoušky dle RfG** (menu, admin a vyšší) se obnovuje každou sekundu (rfg_status): stav, sekvence a krok i/N, cílový výkon, zbývající čas kroku a uplynulý čas, zařazená rozpadová místa (vynechaná přeškrtnutě, při běhu s výkonem místa), výkon výroby z měření DS v kW a v % instalovaného výkonu, stav DI, kód DO, tabulka kroků se zvýrazněním aktuálního a log posledních 20 událostí. Jeden řádek stavu je i na stránce **Info** (*Zkouška dle RfG (poslední)*) a v diagnostice (get_diag). Start, kroky a konec se zapisují do trvalého logu (/log/npa_run.log, prefix rfg_test:).

Omezení (zapiší se jako varování do logu zkoušky při startu)

- Rozpadové místo s regulací **Bez regulace** (lo_regDev = none): povel místa se na něm neuplatní (řetěz PozadovanyVykon ho bere dřív).
- **Limit výstupního výkonu místa** (lo_maxOutPower): strop se uplatní i na krok zkoušky (HW ochrana kabeláže).
- Zařízení s **vlastním povel** (PowerLimit / stupeň nastavený přímo na zařízení): má přednost před povel míst.
- **Náhrada při výpadku elektroměru** (fs_Elmeter0M): při výpadku TML má přednost.

Soubory

Soubor	Obsah
/data/rfg_test.json	stav zkoušky (fáze, sekvence, krok, termín kroku, základ base, pauza paused / pause_left, uložené povel míst, výsledek, log)
/data/state.cnf	povel míst (PowerLimit + PowerLimitBase) - během zkoušky hodnota a základ kroku

Endpointy (HTTP, admin a vyšší, odpověď JSON)

Endpoint	Akce
rfg_tests.html rfg_status	stránka (HTML) stav: phase (""/running/finished), paused (pozastaveno, phase zůstává running), base (inst/phys), base_text, src, seq, step, steps, target_pct, step_sec, step_left, elapsed, total_sec, started, finished, result (ok/stop/abort/restart), result_text, di (enable, seq1..3: 0/1, -1 = nepřirazen), do_code, do_bits, p_plant_kw, p_inst_kw (výkon základu: instalovaný nebo fyzický dle base), p_plant_pct (jen ze zařazených míst), locations[] (uuid, name, type, active, p_kw, p_inst_kw), all_active, pct[], text, log[]
rfg_start?seq=<1..3>	start sekvence (jen v režimu SW); {"res": "error", "msg": ...} když už běží / neplatná sekvence / HW režim
rfg_stop	zrušení běžící (i pozastavené) zkoušky a návrat povelů
rfg_pause	pozastavení běžící zkoušky (drží krok, odpočet stojí)
rfg_resume	pokračování pozastavené zkoušky zbývajícím časem kroku
rfg_save (POST)	uložení nastavení (rfg_*) a fajfek míst (rfg_lo_<uuid>=true → lo_rfgEn v objektu místa) a znovuvykreslení stránky

Klíče v configu (mC2)

Klíč	Význam
rfg_src	sw (výchozí, prázdné = sw) / hw
rfg_s1_pct, rfg_s2_pct, rfg_s3_pct	kroky sekvence, čárkami oddělená procenta; prázdné = výchozí
rfg_s1_sec, rfg_s2_sec, rfg_s3_sec	setrvání na kroku [s]; prázdné = výchozí (190 / 90 / 70)
rfg_s1_base, rfg_s2_base, rfg_s3_base	základ procent kroků: inst (výchozí, prázdné) / phys
lo_rfgEn (v objektu místa)	true / false = místo zařazené do zkoušky; chybí = zařazeno

Zdrojové soubory

Část	Soubor / symbol
Stavový automat, DI, návrat povelů, stav na disku	nrfgtest.cpp → NRfgTest (Start, Stop, Pause, Resume, Loop, ApplyStep, CheckLimitsIntact, RestoreLimits, DoCode)
Výběr míst	nrfgtest.cpp → LocationEnabled, ActiveLocations, SaveLocationEnable; tabulka fajfek NHtml::RfgLocationTable
Tick 500 ms	main.cpp → Test1s
Kód kroku na DO	nstate.cpp → GetBinOutStateOnDevice (větev RfG-bit)

Část	Soubor / symbol
Routy	server.cpp → rfg_tests.html, rfg_status / rfg_start / rfg_stop, POST rfg_save
Web	html/v2/rfg_tests.html (vykresluje NHtml::RfgTests, tabulka I/O NHtml::RfgIoTable, tabulka míst NHtml::RfgLocationTable), menu menuadmin/menuguru/menusasa.html, info.html
Volby funkcí I/O	html/v2/device_edit_input.html, device_edit_output.html, device_ edit_outputTest.html