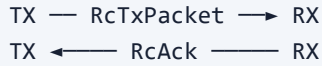
RC Ultimate

Recap původního protokolu, jeho rozšíření a nové servisní vrstvy

Vývoj service vrstvy od v3/v10 · aktuální doplněk v13 · aktuální beta · stav k 8. 9. 2026

> Máme nyní spolehlivý malý servisní protokol vedle hlavního řízení. Hlavní řízení běží dál pravidelně a servisní komunikace pouze využívá vyhrazené bajty uvnitř každé běžné RF výměny.

1. Hlavní real-time protokol



TX posílá řízení a pomocné kanály, stav přepínačů, flags, sequence counter, servisní request a CRC16. RX vrací telemetrii, skutečný stav přepínačů, flags, diagnostiku, servisní odpověď a CRC8.

Hlavní ovládání a servis jsou fyzicky ve stejných RF paketech, ale logicky jsou oddělené. Když servis nic nedělá, jeho čtyři bajty jsou nulové.

Wire paket	Původně	Nyní
TX → RX	18 B	18 B
RX → TX	9 B	13 B
Service request	4 B	4 B
Service response	1 B	4 B

TX paket nenarostl. Symetrická service odpověď původně zvětšila RX ACK o tři bajty a v10 přidal další bajt pro RX core teplotu. Přechod service ACK z 9 na 12 B odpovídal pozorovanému zvýšení doby obsluhy rádia přibližně z 1920 na 2020 µs. Následující protokoly v11–v13 zachovaly 13B ACK a byly prakticky provozovány na aktuální TX/RX dvojici; cílený thermal fault-injection test zůstává otevřený.

2. Původní servisní protokol

Původní request už měl čtyři bajty:

```
byte 0: [transaction ID | operation]
byte 1: task
byte 2: parameter
byte 3: value
```

RX však odpovídal pouze jedním bajtem:

```
[transaction ID | status]
```

TX tedy poznal, ke které transakci odpověď patří a zda skončila například `OK`, `BUSY` nebo `BAD_VALUE`. Odpověď ale neuměla vrátit task, parameter, skutečnou hodnotu ani vícebajtová data.

Bylo to dostatečné pro jednoduché povely typu „nastav toto na 75“ nebo „zapni/vypni“, ale ne pro spolehlivé čtení aktuální konfigurace z RX.

Původní stop-and-wait

1. TX vytvořil request s transakčním ID.
2. Stejný request opakoval v každém RF cyklu.
3. RX provedl příkaz pouze jednou.
4. Na duplikáty vracel uloženou odpověď.
5. TX skončil po terminálním statusu se správným ID.

Už tento základ chránil příkazy s vedlejším účinkem před vícenásobným provedením způsobeným opakováním RF paketů.

3. Symetrický čtyřbajtový protokol

Dnes mají request i response stejnou fyzickou strukturu:

Byte 0 TID + operation/status	**Byte 1** task	**Byte 2** parameter	**Byte 3** value

Request TX → RX nese transaction ID 1 až 15, operaci, task, parametr a hodnotu nebo řídicí údaj streamu. Response RX → TX vrací stejné transaction ID, status, potvrzený task, potvrzený parametr a hodnotu.

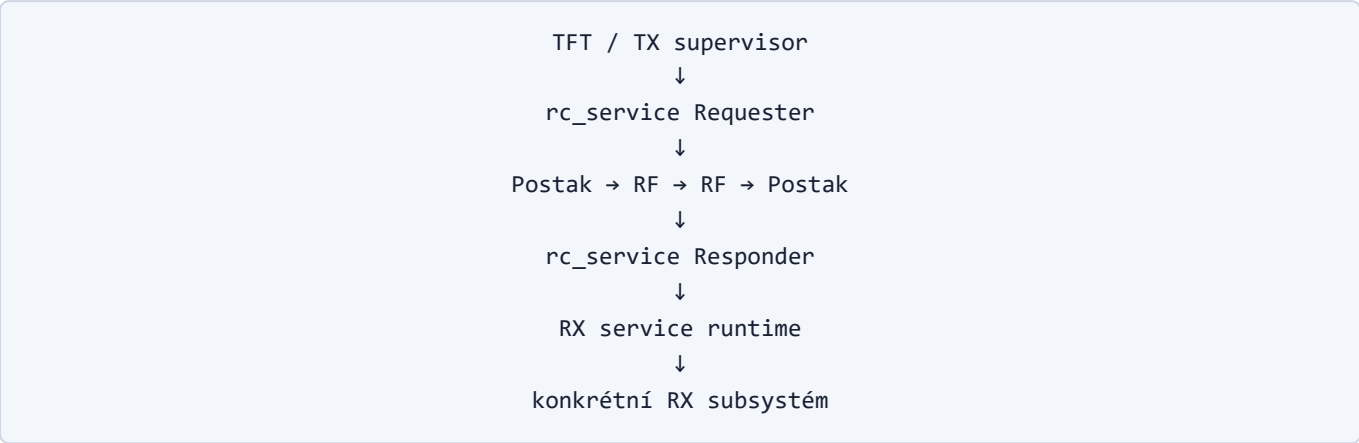
TX přijme odpověď jen tehdy, pokud souhlasí transaction ID, task i parameter. Opožděná odpověď předchozího příkazu se proto nemůže snadno vydávat za odpověď na nový příkaz.

```
Service transaction ID: 1 → 2 → 3 → … → 15 → 1
ID 0: idle
```

> Hlavní osmibitový `RcTxPacket::seq` je jiný čítač: počítá RF pakety hlavně pro diagnostiku. Čtyřbitové service transaction ID označuje konkrétní servisní konverzaci.

4. Nová servisní vrstva

Samotné čtyři bajty jsou pouze wire slot. Knihovna `rc_service` nad nimi staví spolehlivou komunikační vrstvu.



Requester na TX

- povolí jen jednu aktivní transakci;
- přidělí transaction ID a vloží request pošťákovi;
- opakuje request, dokud nepřijde odpověď;
- kontroluje ID, task a parameter;
- automaticky řídí vícebajtové přenosy;
- hlídá timeout a počet pokusů.

Sdílené limity TX i RX jsou 1500 ms nebo 64 dokončených RF cyklů. Pak TX transakci skončí statusem `TIMEOUT` nebo `RETRY_LIMIT`.

Responder na RX

- validuje request a rozpozná duplikát;
- spustí konkrétní handler a uloží jeho výsledek;
- na opakovaný paket vrátí stejný výsledek bez nového provedení;
- odmítne změnu obsahu uvnitř běžící transakce;
- řídí příjem a odesílání fragmentů.

Pokud TX pošle desetkrát stejný `SAVE_CONFIG`, protože čeká na odpověď, RX zapíše NVS pouze jednou. Další duplikáty dostanou výsledek z responder cache.

Recovery opuštěné session

RX autonomně zahodí rozpracovaný stream po 1500 ms nebo po rozpětí 64 hlavních RF packet sequence čísel. Expirace vymaže také duplicate-response cache. Po 2000 ms bez nového přijatého RF paketu se resetuje celý responder. RX přitom rozlišuje skutečně nový paket od opakovaného čtení starého snapshotu z Postaka, takže starý `STREAM_BEGIN` session znovu nevytvoří. Výpočet 2000ms linkového stáří saturuje případ, kdy jiný task právě publikoval timestamp novější než lokálně zachycený čas; mezitaskové pořadí tak nevyvolá falešný reset responderu. Wire `CANCEL` zůstává rychlý úklid, ale zotavení na něm nezávisí.

5. Operace a statusy

Operace	Význam
`GET`	Přečti aktuální hodnotu z RX.
`WRITE`	Změň hodnotu v RAM.
`WRITE_SAVE`	Změň a persistentně ulož, pokud to daný task dovoluje.
`EXECUTE`	Proveď akci bez klasické konfigurační hodnoty.
`STREAM_BEGIN`	Zahaj vícebajtový přenos.
`STREAM_DATA_0..3`	Přenes konkrétní fragment.
`CANCEL`	Ukonči rozpracovaný stream.

Mezi běžné statusy patří `OK`, `BUSY`, `ERROR`, `BAD_VALUE`, `BAD_OPERATION`, `UNKNOWN_TASK`, `NOT_SUPPORTED` a `TRANSACTION_MISMATCH`. Statusy `TIMEOUT` a `RETRY_LIMIT` vytváří lokálně TX; RX je nevysílá.

6. Vícebajtový přenos

Fyzický frame stále přenese pouze jeden datový bajt na RF krok. Servisní vrstva ale umí jednu logickou hodnotu dlouhou až čtyři bajty.

Jednobajtová hodnota

Servo hodnoty od −120 do +120 používají kompaktní cestu:

```
TX: WRITE, SERVO_SETUP, FRONT_DIFF_ON, 78
RX: OK,    SERVO_SETUP, FRONT_DIFF_ON, 78
```

Vícebajtový WRITE

Servo +130 % použije dvoubajtový little-endian `int16`:

```
130 = 0x0082
data: 0x82, 0x00

TX: STREAM_BEGIN, WRITE, délka 2
RX: STREAM_BEGIN, potvrzena délka 2

TX: STREAM_DATA_0, 0x82
RX: potvrzení fragmentu 0

TX: STREAM_DATA_1, 0x00
RX: OK
```

RX hodnotu aplikuje až po přijetí a validaci celého obsahu. Nikdy tedy neaplikuje jen polovinu `int16`.

Vícebajtový GET

```
TX: GET daného parametru
RX: STREAM_BEGIN, odpověď bude mít 2 bajty

TX: žádost o fragment 0
RX: fragment 0 = 0x82

TX: žádost o fragment 1
RX: fragment 1 = 0x00 + finální OK
```

Každý krok je stop-and-wait. Další fragment se neposílá, dokud není potvrzen předchozí. Celý stream drží stejné transaction ID, task a parameter.

7. Aktuálně zapojené servisní tasky

System

- ping;
- výběr TX AP pro příští RX setup;
- výběr společné client Wi-Fi.

Smart drive

- automatické blinkry;
- střed blinkrů odvozený ze steering trimu;
- RAM nebo explicitní persistentní zápis.

Light control

- pomocné příspěvky levého a pravého blinkru;
- brzda a zpátečka;
- `GET` / `WRITE` runtime LIGHTBAR PWM.

LIGHTBAR PWM je RAM override. Nemění jednotlivé `dim_duty` hodnoty ani RX NVS a po restartu RX zmizí. TFT si při každém otevření editoru načte skutečnou efektivní hodnotu z RX.

Servo setup

- přední diferenciál OFF a ON;
- zadní diferenciál OFF a ON;
- převodovka OFF/LOW a ON/HIGH;
- samostatný příkaz `SAVE_CONFIG`.

Hodnoty jsou v rozsahu −150 až +150 % a RX je převádí:

```
pulse_us = 1500 + procenta × 5

-60 %  → 1200 µs
  0 %  → 1500 µs
+78 %  → 1890 µs
+150 % → 2250 µs
```

Winch control

- podepsaná runtime rychlost −100 až +100 %;
- TFT krok 5 %, záporně OUT a kladně IN;
- potvrzovaný RAM-only `GET` / `WRITE` bez použití proporcionálního AUX kanálu;
- potvrzovaný arm při otevření menu a STOP následovaný disarmem při jeho zavření;
- disarm drží pouze `WINCH_PWM` signál LOW;
- automatický STOP a disarm při failsafe, setupu, restartu RX nebo aplikaci konfigurace.

Battery control

- `GET` uloženého stavu RX Battery Guardu;
- `WRITE_SAVE` pro vědomé trvalé zapnutí nebo vypnutí guardu;
- `EXECUTE/SLEEP_NOW` pouze během potvrzeného cutoff okna;
- RX autonomní timeout, ztráta service linku a nouzový voltage floor mají vždy přednost.

Brake ABS

- `BRAKE_ABS` (`0x15`) pro režim `OFF/LINEAR/FIXED` ;
- frekvence pulzní obálky 5–50 Hz, lineární duty min/max a pevné duty 0–100 %;
- `GET` skutečné RX RAM hodnoty, potvrzený RAM-only `WRITE` a jediný explicitní `EXECUTE/SAVE_CONFIG` ;
- stávající 4B service slot; TX paket 18 B a RX ACK 13 B se nezměnily.

Log control

- `LOG_CONTROL` (`0x16`) pro TFT kartu `Setup/Logs` ;
- `GET/ENABLED` vrací skutečný persistentní stav RX loggeru;
- `WRITE_SAVE/ENABLED` logger zapne nebo vypne a změnu uloží;
- `EXECUTE/ERASE` potvrzeně smaže RX incident logy;
- společné smazání pokračuje k TX failsafe/crash logům až po `OK` z RX, takže `BUSY` nebo nedostupný RX ponechá TX logy beze změny.

Čtení, obě persistentní přepnutí i společné mazání byly prakticky potvrzeny na aktuální TX/RX S3 Mini V2 stable v3 dvojici. Hlavní wire rámce ani čtyřbajtový service slot se nezměnily.

Megatransfer / Wi-Fi provisioning

`MEGATRANSFER` (`0x20`) používá 4B service rámec jako řídicí rovinu a během aktivního FS přenáší osm datových bajtů v osových polích. RX přijme data až po fyzicky potvrzeném servo hard-disarmu. První objekt je client Wi-Fi profil s CRC32; nejprve se aplikuje v RAM a do NVS se uloží až po připojení a získání IP. Happy path na čistém RX byl prakticky ověřen.

Průběh práce ve vzdálené TFT kartě

Při první chybě přednačítání se dávka zastaví, takže nedostupný RX nevyvolá řetězec dalších timeoutů. Čtení nic nezapisuje do TX flash; při novém otevření se cache příslušné karty zneplatní a obnoví. Pokud NVS zápis selže nebo se potvrzení nedoručí, TX neukáže `SAVED`, konfigurace zůstane označená jako dirty a uložení lze zopakovat.

8. Co servisní protokol nedělá

Přes servis neposíláme věci, které musí reagovat v každém řídicím cyklu:

- steering, throttle a aux kanály;
- běžné přepínače;
- failsafe a stav linky;
- základní telemetrii.

Ty zůstávají součástí hlavní 200Hz linky. Kdyby servis čekal, timeoutoval nebo přenášel několik fragmentů, běžné řízení pokračuje dál.

Vyhrazený megatransfer je bezpečnostně omezená výjimka: osová pole použije jako data pouze v aktivním FS po servo hard-disarmu. RX profile JSON backup/restore se přes RF service neposílá; používá HTTP cestu přes TX WebSetup.

Servis je určen pro setup, pomalé příkazy, readback konfigurace, potvrzené změny, explicitní NVS save a drobnou obousměrnou diagnostiku.

9. Navazující I2C světla a obnova rádia

STM8 0.8 je nyní aktivní jedenáctikanálový světelný modul s I2C protokolem v3. Servisní LIGHT_CONTROL dál mění logický stav a parametry na RX; výsledný stav publikuje RX supervisor pro lokální výstupy i rx_light_i2c. Blokový CONFIG/COMMIT a periodický STATE/STATUS mezi RX a STM8 jsou samostatný lokální protokol, nikoli další task ve 4B RF service slotu.

External profil má 16 persistentních pozic, skutečný počet a PWM schopnosti určuje INFO. RX vlastní konfiguraci i nejnovější stav; I2C manager vlastní dostupnost a obnovu sběrnice. Při reconfig a reconnectu se čeká na potvrzení generace před obnovením živých světel. Podrobnosti: I2C vrstva v aktuálním protokolu.

Kontrola nRF po 200 ms ticha a případný reinit patří přímo RF tasku. Neobnovují stáří dat ani nemažou Pošťáka nebo service stav. Další kontrola/pokus po nejméně 1 s; nejde o service příkaz ani o nový RF wire formát.

Uživatel potvrdil Wi-Fi megatransfer včetně chybových větví, bateriové rozhodování a OTA okraje. Otevřené zůstávají thermal a CSV testy a oprava OTA nabídky při pozdním připojení RX do Setup Wi-Fi.

Výsledek

Původní model:

> „Nastav hodnotu a řekni mi jen OK/FAIL.“

Aktuální model:

> „Přečti skutečnou hodnotu, přenes až čtyři bajty,
> změň ji pouze jednou, potvrď přesně který příkaz jsi zpracoval
> a persistentní zápis proveď jen na explicitní potvrzení.“

Čtyřbajtový service slot, deterministické RF pakety a jednoduchý stop-and-wait princip zůstaly zachované. Výsledkem je malá obdoba diagnostického a konfiguračního transportu nad existující RC linkou: otevřená pro další setup položky, ale stále výrazně jednodušší než plný CAN/ISO-TP protokol.

Aktuální diagnostická verze: `RC_PROTO_VER = 13` · aktuální beta · stav 8. 9. 2026
Zdrojový wire kontrakt: `lib/common/include/rc_protocol.h`
Pravidla servisní vrstvy: `lib/rc_service/README.md`