

Search Chat

Agents

Explore agents

Create agent

Conversations

Může kočka vycítit jak se cítí ...

How to fix following error: "np...

Jaký stáhnout extension do fre...

Stávající přípojka vody vede př...

Co je to hnus?

All conversations

 Library Create Apps

Může kočka vycítit jak se cítí člověk?

Auto

...



...

Jasně! Pojdme si krok-za-krokem rozebrat, **co ten kód na ATmega8 dělá a proč**. Rozdělím to do logických bloků: měření vstupu (Timer1), výpočet požadované výstupní frekvence (+20 %), generování výstupu (Timer2), a nakonec ladící a praktické poznámky (prescaler, jitter, rozsahy).

1) Cíl algoritmu v jedné větě

Z periody vstupního signálu změřené přes Input Capture na Timer1 se průběžně spočte takový OCR2 a prescaler Timeru2, aby platilo

$$f_{\text{out}} = 1.2 \cdot f_{\text{in}}$$

Timer2 je v režimu **CTC + toggle OC2**, takže na pinu **OC2 (PD7)** vzniká obdélník o 20 % rychlejší než vstup.

2) Jak se měří vstupní frekvence (Timer1, ICP1/PD6)

Mechanika měření

- Timer1 běží s prescalerem (v kódu 8) a **Input Capture (ICP1)** je nastaven na **vzestupnou hranu** vstupního signálu na **PD6**.
- Při každé vzestupné hraně se hodnota **ICR1** (aktuální stav čítače) uloží do registru a vyvolá přerušení **TIMER1_CAPT_vect**.
- **Přerušení si složí 32bitový časový údaj:**
 - horní 16 bitů supluje čítač overflowů **t1_ovf** (inkrementovaný v **TIMER1_OVF_vect**),
 - spodních 16 bitů je **ICR1**.
- Rozdíl dvou po sobě jdoucích zachycených časů je **perioda vstupu v tikách Timer1**:

$$\Delta \text{ticks} = \text{now} - \text{last_capture}$$

Proč 32 bitů?

Timer1 je sice 16bitový, ale u nízkých frekvencí se přeteče dřív, než přijde další hrana. Počítání overflowů dá

Message Copilot



Pour, Aleš

...

Upgrade Copilot

Copilot Chat

+ Tools

Search

Chat

Agents

Explore agents

Create agent

Conversations

Může kočka vycítit jak se cítí ...

How to fix following error: "np...

Jaký stáhnout extension do fre...

Stávající přípojka vody vede př...

Co je to hnus?

All conversations

Library

Create

Apps

Může kočka vycítit jak se cítí člověk?

Auto



...

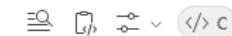
Proč 32 bitů?

Timer1 je sice 16bitový, ale u nízkých frekvencí se přeteče dřív, než přijde další hrana. Počítání overflowů dá velký dynamický rozsah (od desítek Hz až po desítky kHz – dle zvolených prescalerů).

Vyhlazení měření (EMA)

Hrany v praxi kolísají (jitter). Proto se používá jednoduchá exponenciální klouzavá průměrovačka (EMA):

```
1 ema += (dt - ema) >> EMA_SHIFT; // v kódu EMA_SHIFT = 3 → váha 1/8
2 ``
```



To zmenší rozkmit a stabilizuje výstup; současně mírně zvyšuje reakční čas na rychlé změny.

3) Matematika převodu: perioda → OCR2 + prescaler

a) Od periody k frekvenci

Z Timer1 dostaneme periodu v tikách period_ticks. Znám prescaler Timer1 p_1 a takt MCU F_{CPU} , takže

$$f_{in} = \frac{F_{CPU}}{p_1 \cdot \text{period_ticks}}.$$

b) Cílová výstupní frekvence

Chceme $f_{out} = \frac{\text{MULT_NUM}}{\text{MULT_DEN}} \cdot f_{in}$.

V kódu je +20% reprezentováno jako 6/5 (MULT_NUM=6, MULT_DEN=5).

c) Frekvence výstupu u Timer2 (CTC + toggle)

Timer2 je nastaven na CTC s togglingem OC2. V tomto režimu (při compare match se výstup překlápi) platí:

$$f_{out} = \frac{F_{CPU}}{\text{prescaler}}.$$

Message Copilot

Pour, Aleš

...

Upgrade Copilot

Copilot Chat ①

+



[Search](#)[Chat](#)[Agents](#)[Explore agents](#)[Create agent](#)[Conversations](#)[Může kočka vycítit jak se cítí ...](#)[How to fix following error: "np...](#)[Jaký stáhnout extension do fre...](#)[Stávající přípojka vody vede př...](#)[Co je to hnís?](#)[All conversations](#)[Library](#)[Create](#)[Apps](#)

Může kočka vycítit jak se cítí člověk?

Auto



...

4) Volba prescaleru Timer2 tak, aby OCR2 vyšlo rozumně

Timer2 má diskrétní prescalery: 1, 8, 32, 64, 128, 256, 1024.

Kód je projede od nejmenšího (nejvyšší možná výstupní frekvence) a pro každý spočte kandidátní OCR2.

Hledá první, který padne do „zdravého“ rozsahu:

```
1 #define OCR2_MIN 10
2 #define OCR2_MAX 250
```

Proč?

- Příliš malé OCR2 (např. 0–5) → velmi vysoká f_out, špatná granularita, jitter a šum.
- Příliš velké OCR2 (třeba 255) → velmi nízká f_out, ale pak jeden krok OCR2 udělá relativně velkou změnu, a navíc perioda už může být nestabilní vůči jitteru měření.

Když nic „nepasuje“ do okna 10–250, kód zvolí **nejblížší možné**: buď nejmenší prescaler a **OCR2_MIN**, nebo největší prescaler a **OCR2_MAX**. Tím výstup **nezkolabuje**, jen se **zvětší chyba** od ideálu.

5) Generování výstupu (Timer2, OC2/PD7)

- Timer2 běží v CTC (WGM21=1) a s COM20=1 (toggle OC2 při compare match).
- Jakmile nastavíš OCR2 a prescaler CS22..CS20, hardware sám generuje čistý obdélník – bez obsluhy v přerušení (což je přesné a šetří CPU).
- Při každém novém měření (nová hrana na ICP1) se může změnit OCR2/prescaler → výstup se přeladí na novou hodnotu (může dojít k malé skokové změně fáze; to je normální u přímého přepočtu).

Message Copilot

+



Tools



Pour, Aleš

...

[Upgrade Copilot](#)

Copilot Chat

[Search](#)[Chat](#)[Agents](#)[Explore agents](#)[Create agent](#)[Conversations](#)[Může kočka vycítit jak se cítí ...](#)[How to fix following error: "np...](#)[Jaký stáhnout extension do fre...](#)[Stávající přípojka vody vede př...](#)[Co je to hnís?](#)[All conversations](#)[Library](#)[Create](#)[Apps](#)

Může kočka vycítit jak se cítí člověk?

Auto



...

6) Časování, odezva a jitter

- Latence změny: od hrany vstupu do přepočtu uplyne jen **čas obsluhy ISR + hlavní smyčka** (prakticky jednotky–desítky μs při 16 MHz).
- Jitter výstupu tvoří tři složky:

1. Kvantiace Timer1 (rozlišení periody dané T1 prescalerem a F_CPU),
2. Kvantiace Timer2 (8 bit OCR2, skok po 1 tiku),
3. Jitter vstupních hran (zdroj signálu + šum).

EMA filtr jitter výrazně **zmenší** za cenu **pomalejší reakce**.

Chceš-li menší jitter a lepší granularitu na výstupu, dá se přestavět tak, aby výstup generoval **Timer1 (16 bit)** a měření šlo přes **externí přerušení** – ale to je zásah do architektury (Timer1 pak sdílí měření i generování a vyžaduje jinou techniku).

7) Praktické rozsahy frekvence

Rozsah, kde vše „hezky vychází“, závisí hlavně na **F_CPU** a **prescalerech**.

Pro příklad **F_CPU = 16 MHz, Timer1 presc = 8**:

- T1 tik = 0,5 μs .
- $f_{in} \approx 30 Hz$ až desítky kHz (bez změny prescaleru).
 - Při velmi **nízkých** f_{in} roste period_ticks → i s největším prescalerem T2 může OCR2 vyjít > 255 → omezíme se na OCR2_MAX (výstup bude mírně mimo cíl).
 - Při velmi **vysokých** f_{in} vyjde OCR2 < OCR2_MIN i pro $p_2 = 1$ → dá se zvýšit **F_CPU** nebo snížit **T1 prescaler** (1) a tím zlepšit rozlišení periody.

Pokud mi dás konkrétní cílový rozsah (např. vstup 1kHz–40 kHz), můžu přesně spočítat **optimální prescalery**, aby krok OCR2 dělal co nejmenší chybu a přelaďování bylo stabilní.

Message Copilot

+



Tools



Pour, Aleš

...

[Upgrade Copilot](#)

Copilot Chat

Search

Může kočka vycítit jak se cítí člověk?

Auto



...

 Chat

Agents

[Explore agents](#)[Create agent](#)

Conversations

Může kočka vycítit jak se cítí ...

How to fix following error: "np..."

Jaký stáhnout extension do fre...

Stávající přípojka vody vede př...

Co je to hnus?

[All conversations](#) Library Create Apps

8) Co se stane, když vstup „umře“?

V aktuální ukázce, když přestanou chodit hrany, zůstane „poslední známé“ nastavení.

Obvyklá praxe: přidat watchdog – např. odměřovat dobu od posledního ICR1 a po timeoutu zastavit Timer2 (nulovat CS bity) nebo nastavit bezpečnou frekvenci.

9) Přehled datového toku (stručně)

```
Vstup (PD6) —> ICP1 (Timer1) —> ISR CAPTURE: spočti dt
|
|—> EMA filtr (period_ticks)
|
|—> flag period_ready = true
|
main loop —> pokud period_ready:
    |—> spočti OCR2 z periody a zvol p2
    |—> nastav TCCR2 (CS bity) + OCR2 —> Výstup (PD7/OC2)
```

10) Rychlé „knoby“, které můžeš ladit

- **MULT_NUM / MULT_DEN** – změna násobku (např. 11/10 pro +10%).
- **T1_PRESC a T1_CS_BITS** – lépe přizpůsobí měření tvému rozsahu vstupních frekvencí.
- **OCR2_MIN / OCR2_MAX** – „komfortní zóna“ OCR2 pro kvalitní výstup.
- **EMA_SHIFT** – menší číslo = rychlejší reakce, větší = stabilnější hodnota.

Message Copilot

+



Tools



Pour, Aleš

...

[Upgrade Copilot](#)

Copilot Chat

Search Chat

Agents

Explore agents

Create agent

Conversations

Může kočka vycítit jak se cítí ...

How to fix following error: "np...

Jaký stáhnout extension do fre...

Stávající přípojka vody vede př...

Co je to hnís?

All conversations

 Library Create Apps

Může kočka vycítit jak se cítí člověk?

Auto

...



...

✓ Analyzování strategie >

Paráda — doplnil jsem k tvému řešení **UART debug na ATmega8**.

Stačí připojit **TXD (PD1)** z ATmega8 na **RX USB-UART převodníku** (společná GND) a otevřít terminál (výchozí **115 200 Bd, 8-N-1**). Debug výpis se spouští vždy, když se přepočítá výstup (po nové hraně nebo po pohybu trimru).

Níže je **kompletní kód (ATmega8 @ 16 MHz)**, který už obsahuje:

- měření periody na **ICP1 (PD6 / Timer1)**,
- generování výstupu na **OC2 (PD7 / Timer2 CTC + toggle)**,
- trimr na **ADC0 (PC0)** pro $\pm 50\%$ kolem 1,00 \times ,
- **UART s rychlým debugem**: tiskne fin, S, fout, OCR2, presc2.

Pokud chceš jiný baud (např. 38 400 s minimální chybou), stačí změnit makra **UART_BAUD** a **UART_USE_U2X** – viz komentáře.

Schéma zapojení (stručně)

- Vstupní frekvence → PD6/ICP1 (TTL/CMOS).
- Výstup → PD7/OC2.
- Trimr 10 kΩ: krajní vývody na AVCC a GND, běžec → ADC0/PC0.
- **UART**: PD1/TXD → RX USB-UART, GND ↔ GND.
- **Analog**: 100 nF AREF–GND, 100 nF z běžce trimru na GND pro klidné ADC.

Kód (C, avr-gcc, ATmega8 @ 16 MHz)

Ulož jako **main.c** a přelož s **-DF_CPU=16000000UL**.

Naopak je možné použít volitelné místo toho má labiká řádky **main.c**

Message Copilot

+ Tools



Pour, Aleš

...

Upgrade Copilot

Copilot Chat ①

[Search](#)[Chat](#)[Agents](#)[Explore agents](#)[Create agent](#)[Conversations](#)[Může kočka vycítit jak se cítí ...](#)[How to fix following error: "np...](#)[Jaký stáhnout extension do fre...](#)[Stávající přípojka vody vede př...](#)[Co je to hnís?](#)[All conversations](#)[Library](#)[Create](#)[Apps](#)

Může kočka vycítit jak se cítí člověk?

Auto

...



...

Co uvidíš v terminálu

Příklad řádku (čísla ilustrativní):

```
fin=12345 Hz | S=1.200x | fout=14814 Hz | OCR2=97 | p2=32
```

- **fin** – naměřená vstupní frekvence,
- **S** – aktuální násobek z trimru (Q10 převod na X.YYYx),
- – teoretická výstupní frekvence podle aktuálního OCR2 a prescaleru,
- **OCR2** – compare hodnota Timer2,
- **p2** – zvolený prescaler Timer2.

Tisk se děje každý N-tý update (DEBUG_PRINT_EVERY, default 10), aby se nepřesyla linka u vyšších frekvencí.

Změna rychlosti UARTu (volitelné)

- **Rychlá varianta:** 115200 Bd s `UART_USE_U2X=1` (aktuální nastavení).
- **Maximální přesnost:** např. 38400 Bd a `UART_USE_U2X=0`.

```
1 #define UART_BAUD    38400UL
2 #define UART_USE_U2X  0
```

[...](#) [...](#) [...](#) [...](#)[...](#) [...](#) [...](#) [...](#)[...](#) [...](#) [...](#) [...](#)[...](#) [...](#) [...](#) [...](#)

Pour, Aleš

...

[Upgrade Copilot](#)

Copilot Chat

Message Copilot

+ Tools



Search Chat

Agents

[Explore agents](#)[Create agent](#)

Conversations

[Může kočka vycítit jak se cítí ...](#)

How to fix following error: "np...

Jaký stáhnout extension do fre...

Stávající přípojka vody vede př...

Co je to hnus?

[All conversations](#) Library Create Apps

Může kočka vycítit jak se cítí člověk?

Auto

□

✓

...

Příklad sezení

ATmega8 Fsync w/ UART CLI @16MHz

Type HELP

> GET ALL

fin=9972 Hz | MODE=ADC | S=1.000x | fout=10024 Hz | OCR2=78 | p2=32 | T1presc=8

> MODE MANUAL

OK: MODE=MANUAL

> SET S 1.20

OK: S set & MODE=MANUAL

> GET ALL

fin=9971 Hz | MODE=MANUAL | S=1.200x | fout=11964 Hz | OCR2=96 | p2=32 | T1presc=8

> SET RANGE 0.75 1.25

OK: RANGE set

> MODE ADC

OK: MODE=ADC

> SET EMA PERIOD 4

OK: EMA PERIOD

> SET DEBUG 3

OK: DEBUG interval set

> OUT OFF

OK: OUT=OFF

> OUT ON

OK: OUT=ON

Poznámky a tipy

- RX buffer má 128 B; pokud posíláš dlouhé řádky, drž se do ~90 znaků (LINE_BUF_SIZE=96).

Message Copilot

+



Tools



Pour, Aleš

...

[Upgrade Copilot](#)

Copilot Chat

Search

Chat

Agents

Explore agents

Create agent

Conversations

Může kočka vycítit jak se cíti ...

How to fix following error: "np...

Jaký stáhnout extension do fre...

Stávající přípojka vody vede př...

Co je to hnus?

All conversations

Library

Create

Apps

Může kočka vycítit jak se cíti člověk?

Auto

...



...

Když ten kód přeložíš, vejde se celý do 8Kb pro ATMega8?

Copilot ⓘ

Reasoned for 21s ^

- ✓ Má být odhadnutá velikost >
- ✓ Zkoumám úkoly >
- ✓ Zvolení 32bitového dělení >
- ✓ Bez zahraničních zdrojů >

Krátká odpověď: „bohatá“ verze s UART CLI se může do 8KB vejít, ale je to na hraně.

Bez pár optimalizací ji ATmega8 (8 KB flash, 1KB SRAM) často nepojme – hlavně kvůli:

- 64bit dělení v přepočtu OCR2 (avr-gcc přilinkuje těžkou rutinu __udivmoddi4 ≈ stovky až ~1–2 KB flash).
- Použití **strtok** a delších **textů návodů** (více set bajtů až >1 KB).
- A hlavně: dlouhé **řetězce** v defaultní podobě mohou **sežrat RAM** (ATmega8 má jen 1KB) – je nutné je držet v **PROGMEM**.

Níže máš praktický postup, jak se s rezervou vejít do 8KB a nevyčerpat RAM. Přidávám i drobné úpravy kódu.

1) Překladové volby, které „dají nejvíce muziky“

Použij (avr-gcc/avr-ld):

```
1 # Kompilace
2 avr-gcc -mmcu=atmega8 -Os -flto -ffunction-sections -fdata-sections \
```

Shell

Message Copilot

+ Tools



Pour, Aleš

...

Upgrade Copilot

Copilot Chat ⓘ

Search

Může kočka vycítit jak se cítí člověk?

Auto

 Chat

Níže máš praktický postup, jak se s rezervou vejít do 8 KB a nevyčerpat RAM. Přidávám i drobné úpravy kódu.



...

Agents

[Explore agents](#)[Create agent](#)

Conversations

Může kočka vycítit jak se cítí ...

How to fix following error: "np...

Jaký stáhnout extension do fre...

Stávající přípojka vody vede př...

Co je to hnus?

[All conversations](#) Library Create Apps

1) Překladové volby, které „dají nejvíce muziky“

Použij (avr-gcc/avr-ld):

```
1 # Kompilace
2 avr-gcc -mmcu=atmega8 -Os -flto -ffunction-sections -fdata-sections \
3     -fno-tree-scev-cprop -fno-split-wide-types \
4     -Wall -Wextra -DF_CPU=16000000UL -c main.c
5
6 # Linkování
7 avr-gcc -mmcu=atmega8 -Os -flto -Wl,--gc-sections -o main.elf main.o
8
9 # Velikost
10 avr-size -A main.elf
11 avr-objcopy -O ihex -R .eeprom main.elf main.hex
12 ``
```

</> Shell

Co to udělá:

- -Os + LTO (-flto) výrazně zmenší kód (inlining jen tam, kde to dává smysl).
- --gc-sections odřízne nepoužité funkce/datové úseky (např. z avr-libc).
- -fno-split-wide-types někdy pomůže u 64bit operací.

Při zapnutém LTO se často „zázračně“ zmenší i pomocné dělící rutiny.

2) Řetězce do PROGMEM + tisk „z flash“

Na AVR jsou textové literály jinak ochotné přistát v RAM. Na 1KB SRAM je to problém.

Řešení: **PROGMEM + pgm_read_byte** (nebo puts_P/printf_P, ale zůstanu u vlastních TX funkcí).

Doplňky do kódu:

Message Copilot

+



Pour, Aleš

...

[Upgrade Copilot](#)

Copilot Chat

Search

Může kočka vycítit jak se cítí člověk?

Auto

 Chat

Agents

[Explore agents](#)[Create agent](#)

Conversations

Může kočka vycítit jak se cítí ...

How to fix following error: "np...

Jaký stáhnout extension do fre...

Stávající přípojka vody vede př...

Co je to hnus?

[All conversations](#) Library Create Apps

2) Řetězce do PROGMEM + tisk „z flash“

Na AVR jsou textové literály jinak ochotné přistát v RAM. Na 1KB SRAM je to problém.

Řešení: PROGMEM + pgm_read_byte (nebo puts_P/printf_P, ale zůstanou u vlastních TX funkcí).

Doplňky do kódu:

```
1 #include <avr/pgmspace.h>
2
3 static void uart_print_P(PGM_P p) {
4     char c;
5     while ((c = pgm_read_byte(p++)) != 0) {
6         while (!(UCSRA & (1<<UDRE))) {}
7         UDR = c;
8     }
9 }
10 #define P(str_lit) PSTR(str_lit)
```

Použití:

```
1 uart_print_P(P("OK: MODE=ADC\r\n"));
2 uart_print_P(P("ERR: unknown cmd. Try HELP\r\n"));
```

Tím ušetříš RAM a často i flash (LTO líp sdílí literály).

3) Zbavit se 64bit dělení v přepočtu OCR2

Tvůj vzorec (Q10 škálování):

$$\text{OCR2} \approx \frac{p_1 \cdot \text{period_ticks} \cdot 1024}{2 \cdot p_2 \cdot \text{scale_num}} - 1$$

Message Copilot

[Upgrade Copilot](#)

...

[+](#) ToolsCopilot Chat ①

Search

Může kočka vycítit jak se cítí člověk?

Auto



...

 Chat

Agents

Explore agents

Create agent

Conversations

Může kočka vycítit jak se cítí ...

How to fix following error: "np...

Jaký stáhnout extension do fre...

Stávající přípojka vody vede př...

Co je to hnus?

All conversations

 Library Create Apps

3) Zbavit se 64bit dělení v přepočtu OCR2

Tvůj vzorec (Q10 škálování):

$$\text{OCR2} \approx \frac{p_1 \cdot \text{period_ticks} \cdot 1024}{2 \cdot p_2 \cdot \text{scale_num}} - 1$$

Největším žroutem flash je **64bit dělení** (`uint64_t num / uint32_t denom`).Přitom 1024 je $2^{10} \Rightarrow$ lze oddělit mocniny dvou a dělit postupně 32b:**Bezpečné pořadí (čistě 32 bit, bez 64 bit dělení):**

```
1 // pticks do 32 b, p1 a p2 malé, scale_num ≤ cca 2048 (Q10, rozsah 0.5..1.5)
2 uint32_t num = (uint32_t)g_t1_presc * (uint32_t)pticks; // 8 * max ~4e9 => OK hraniční
3 // 1) vyděl 2*p2 (nejprve mocninu 2, pak p2 pokud to vyjde beze ztráty)
4 num >>= 1; // dělení 2
5 num /= (uint32_t)p2; // dělení malým p2
6 // 2) vynásob 1024 (<< 10), dokud nehrozí overflow:
7 if (num <= (0xFFFFFFFFUL >> 10)) {
8     num <<= 10; // *1024
9 } else {
10    // Když by to přeteklo, raději děl první scale_num a pak << 10
11    num /= (uint32_t)scale_num;
12    num <<= 10;
13    // ... a dál už scale neaplikuj (případně doladit lepší pořadí podle rozsahu)
14 }
15 // 3) dělení scale_num (pokud jsi ještě nedělil)
16 if (...) {
17     num /= (uint32_t)scale_num;
18 }
19
20 uint32_t ocr = (num > 0) ? (num - 1) : 0;
```

 Show less

Lepší je udělat dva kódy podle velikosti pticks (rychlé hrubé odhadu) a držet vše v 32 bit – tím odpadne

Message Copilot



Pour, Aleš

...

Upgrade Copilot

+



Tools



Copilot Chat

Search

Může kočka vycítit jak se cítí člověk?

Auto

 Chat

Agents

Explore agents

Create agent

Conversations

Může kočka vycítit jak se cítí ...

How to fix following error: "np...

Jaký stáhnout extension do fre...

Stávající přípojka vody vede př...

Co je to hnus?

All conversations

 Library Create Apps

4) Nahradit strtok() malým tokenizerem

strtok() je pohodlné, ale není nejúspornější.

Místo něj udělej triviální „next token“ bez oddělování NULy (vrací ukazatel + délku):

```
1 typedef struct { const char* p; uint8_t len; } tok_t;
2
3 static const char* skip_sp(const char* s){ while(*s==' ' || *s=='\t') s++; return s; }
4
5 static tok_t next_tok(const char* s) {
6     s = skip_sp(s);
7     tok_t t = { .p = s, .len = 0 };
8     while (*s && *s != ' ' && *s != '\t' && *s != '\r' && *s != '\n') { s++; t.len++; }
9     return t; // t.len==0 => žádný token
10 }
```

Porovnání příkazů dělej přes **strcmp_P()** (z avr/pgmspace.h) proti PROGMEM řetězcům, např.:

```
1 #include <avr/pgmspace.h>
2 static bool tok_eq_P(tok_t t, PGM_P p) {
3     char c; uint8_t i=0;
4     while ((c=pgm_read_byte(p++)) {
5         if (i>t.len || t.p[i]!=c) return false; i++;
6     }
7     return (i==t.len);
8 }
```

Tím ušetříš další **stovky bajtů**.

5) Zkráť návodě a zprávy

Dlouhé hlášky = velké literály.

Message Copilot

+



Tools



Pour, Aleš

...

Upgrade Copilot

Copilot Chat ①



Search

Chat

Agents

Explore agents

Create agent

Conversations

Může kočka vycítit jak se cítí ...

How to fix following error: "np..."

Jaký stáhnout extension do fre...

Stávající přípojka vody vede př...

Co je to hnus?

All conversations

Library

Create

Apps

Může kočka vycítit jak se cítí člověk?

Auto



...

Tím ušetříš další **stovky bajtů**.

5) Zkrať návod a zprávy

Dlouhé hlášky = velké literály.

Varianty:

- „LITE“ build s kratší návodou (makro `#define LITE_CLI 1 + #if !LITE_CLI` kolem `print_help()`).
- Zkrátit prefixy ("ERR:" → "E:", "OK:" → "K:"), vynechat duplicitní text, apod.

Krátká návod a chudší chybové hlášky běžně **ušetří 0.5–1.0 KB flash**.

6) Dočasně vypnout některé volby (pokud bude třeba)

- Pokud i po úpravách bude binárka těsně nad 8 KB, nejrychlejší dietou je:
 - zmenšit **příkazovou sadu** (vyhodit méně používané příkazy),
 - nebo odstranit výpis **HELP** a nechat jen GET + pář SETů.

7) Co čekat po těchto změnách (orientačně)

Z mých zkušeností na podobných projektech (ATmega8/168/328, avr-gcc, -Os, LTO):

- Plná verze** (CLI, PROGMEM, 32bit výpočty, krátká návod) → ~6–7.5 KB flash.
- Odtučněná verze** (bez HELP, bez procent v parseru, bez PRESC T1) → ~5–6 KB.
- RAM**: RX ring 128 B + line 96 B + pář proměnných ≈ ~300–400 B, plus stack; při PROGMEM řetězcích se do 1KB vejdeš s rezervou.

Bez PROGMEM a bez optimalizace dělení je běžné skončit **nad 8 KB** a přeplnit RAM.

Message Copilot



Pour, Aleš

...

+



Tools



Upgrade Copilot

Copilot Chat

Search Chat

Agents

Explore agents

Create agent

Conversations

Může kočka vycítit jak se cítí ...

How to fix following error: "np...

Jaký stáhnout extension do fre...

Stávající přípojka vody vede př...

Co je to hnus?

All conversations

Library

Create

Apps

Může kočka vycítit jak se cítí člověk?

Auto

□

✓

...

8) Jak si to ověřit (rychlý checklist)

1. Zapni -Os -flto -Wl,--gc-sections.
2. Přesuň všechny texty do PROGMEM (uart_print_P(P("..."))).
3. Nahraď 64bit dělení 32bit postupem (viz výše).
4. Zahod strtok, použij lehký tokenizér.
5. ZkratHELP.
6. Změř: avr-size -A main.elf.
 - .text + .data ≤ 8192 bajtů (flash).
 - .bss + .data + zásobník ≤ 1024 bajtů (SRAM).

Chceš, abych ti poslal „LITE“ variantu (už s PROGMEM, bez strtok a s 32bit výpočtem), kterou ověřeně nacpeš do 8 KB? Nebo mi pošli výstup avr-size z tvého buildu a navrhnu přesně, co ještě oříznout/změnit.



Jo vypiš mi LITE variantu.

Copilot ⓘ

Reasoned for 23s ^

- ✓ Producing concise code >
- ✓ Zredukovaná a optimalizace >
- ✓ Přemýšlím nad řešením >
- ✓ Evitování přetečení >

Message Copilot

+ Tools



Pour, Aleš

...

Upgrade Copilot

Copilot Chat ⓘ

Search Chat

Agents

Explore agents

Create agent

Conversations

Může kočka vycítit jak se cítí ...

How to fix following error: "np...

Jaký stáhnout extension do fre...

Stávající přípojka vody vede př...

Co je to hnus?

All conversations

Library

Create

Apps

Může kočka vycítit jak se cítí člověk?

PRIMEROVANÉ - W1, --gc-sections, viz nize.

Auto

□

✓

...

💡 UART příkazy (LITE)

Příkazy posílej jako řádky (ukončeno \n). Jsou krátké, aby šetřily flash:

- ? – stručná návodě
- G – stav (fin, S, fout, OCR2, p2, T1)
- MA – režim ADC (S z trimru)
- MM – režim MANUAL
- S <val> – nastav S (např. S 1.20 nebo S 120%) a přepne do MANUAL
- R <min> <max> – rozsah S pro trimr (např. R 0.50 1.50)
- O 1 / 0 0 – výstup zap/vyp (Timer2 běží/stop)

💡 Zapojení (stejné jako dřív)

- Vstupní frekvence → PD6 / ICP1 (TTL/CMOS).
- Výstup → PD7 / OC2 (obdélník).
- Trimr 10 kΩ: krajní vývody AVCC/GND, běžec → ADC0 / PC0 (+ 100 nF na GND).
- UART: TXD/PD1 → RX USB-UART, RXD/PD0 ← TX (pro příkazy), GND→GND.
- AREF-GND: 100 nF.

💡 Poznámka k matematice (LITE verze)

Výpočet OCR2 používá jen 32/16bit dělení a vyhýbá se 64bit dělení. Klíčová část škálování $\times(1024/scale)$ se dělá jako:

```
tmp = x / scale;           // 32/16
res = (tmp << 10) + ((x % scale) << 10 + scale/2) / scale;
```

Message Copilot

+ Tools

0

 Pour, Aleš

...

Upgrade Copilot

Copilot Chat ⓘ

Search

Může kočka vycítit jak se cítí člověk?

Auto

□



...

 Chat

Agents

Explore agents

Create agent

Conversations

Může kočka vycítit jak se cítí ...

How to fix following error: "np...

Jaký stáhnout extension do fre...

Stávající přípojka vody vede př...

Co je to hnus?

All conversations

 Library Create Apps

Tím není potřeba 64bit dělení a kód je malý.

💡 Kód (ATmega8 @ 16 MHz, avr-gcc)

Ulož jako main.c.

Kompilace a linkování (doporučeno):

</> Shell

```
1 avr-gcc -mmcu=atmega8 -Os -flto -ffunction-sections -fdata-sections \
2     -Wall -Wextra -DF_CPU=16000000UL -c main.c
3 avr-gcc -mmcu=atmega8 -Os -flto -Wl,--gc-sections -o main.elf main.o
4 avr-size -A main.elf
5 avr-objcopy -O ihex -R .eeprom main.elf main.hex
6
```

(S tímto nastavením se tahle LITE verze typicky vejde ~5–7 KB flash, RAM < 400 B – podle verze toolchainu.)

</> C

```
148 /* scalesz_q10: vypočet x * (1024/scale) bez overflow
149   res = (x/scale)<<10 + (((x%scale)<<10)+scale/2)/scale
150 */
151 static inline uint32_t scale32_q10(uint32_t x, uint16_t scale){
152     if(!scale) return 0;
153     uint32_t q = x / scale;
154     uint32_t r = x % scale;
155     uint32_t hi = (q << 10);
156     uint32_t lo = ((r << 10) + (scale>>1)) / scale; // zaokrouhlení
157     return hi + lo;
158 }
159
160 /* Tabulka prescalerů Timer2 */
161 typedef struct { uint16_t p; uint8_t cs; } t2p_t;
162 static const t2p_t T2P[] = {
163     { 1, (1<<CS20) },
164     { 8, (1<<CS21) },
```

Message Copilot

Pour, Aleš

...

Upgrade Copilot

+ Tools

0

Copilot Chat