ČASOVAČ

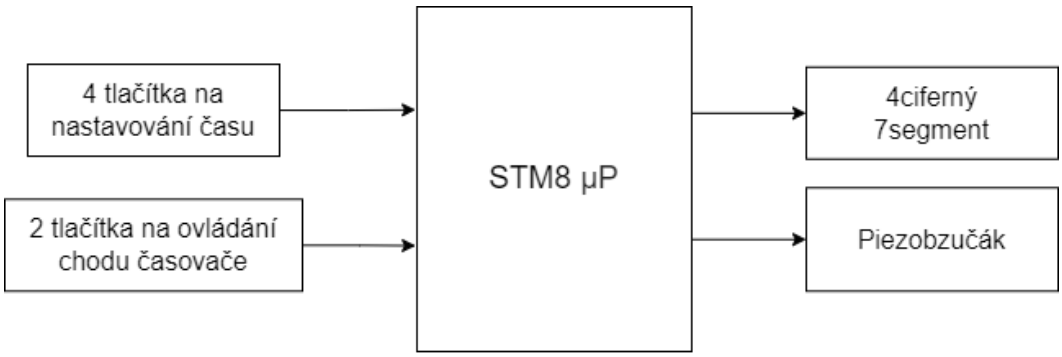
1. Slovní popis

Časovač je zařízení, které provádí odpočet nastavené časové hodnoty k nule a upozorní, jakmile je odpočet ukončen.

Funkce vytvořeného časovače:

- Uživatel pomocí 4 tlačítek nastaví čas pro odpočet.
- Odpočet je spuštěn pátým tlačítkem, které slouží i pro pozastavení při odpočtu.
- Resetování času na nulu během odpočtu je realizováno šestým tlačítkem.
- Po dokončení odpočtu zabzučí piezobzučák po dobu 10ti sekund, přičemž bzučení jde dříve zastavit tlačítkem na reset.

2. Blokové schéma



Vstupy:

Tlačítko 1 – tlačítko na nastavení časové hodnoty v řádu desítek minut

Tlačítko 2 – tlačítko na nastavení časové hodnoty v řádu jednotek minut

Tlačítko 3 – tlačítko na nastavení časové hodnoty v řádu desítek sekund

Tlačítko 4 – tlačítko na nastavení časové hodnoty v řádu jednotek sekund

Tlačítko 5 – tlačítko pro aktivaci a pozastavení odpočtu

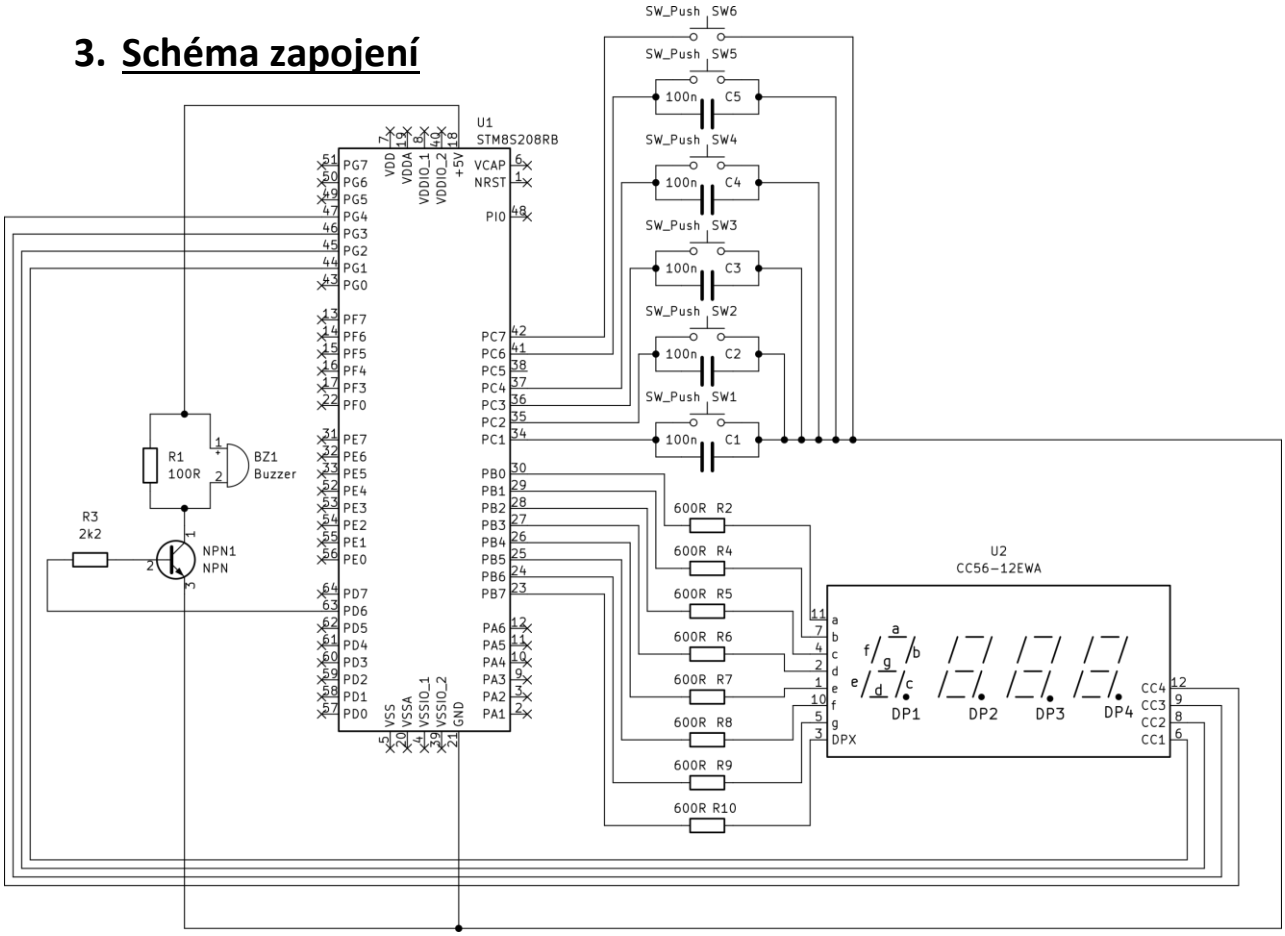
Tlačítko 6 – tlačítko pro resetování odpočtu a vynulování číselných hodnot na 4ciferném 7segmentovém displeji.

Výstupy:

Piezobzučák – generováním zvukového tónu signalizuje dokončení odpočtu

4ciferný 7segment – Zobrazuje odpočítávanou časovou hodnotu: první dvojice čísel zleva ukazuje počet minut, druhá dvojice čísel pak ukazuje počet sekund. Maximální nastavitelný čas tak je 99 minut a 59 sekund.

3. Schéma zapojení



4. Návrh časování

Chceme, aby doba přetečení registru $T_{ARR} = 2\ ms$, abychom mohli po pěti stech přetečení registru přičíst 1s. Podle této podmínky navrhneme a nastavíme limit přetečení registru ARR_{limit}.

$f_{hlavní}$ = 16 MHz

$předdělička_{hlavní}$ = 1

$předdělička_{TIM4}$ = 128

frekvence časovače TIM4 f_{TIM4} = ?

$$f_{TIM4} = \frac{f_{hlavní}/předdělička_{hlavní}}{předdělička_{TIM4}} = \frac{16000\,000/1}{128} = 125000\ Hz$$

Doba přičtení do registru $T_{přičtení}$ = ?

$$T_{přičtení} = \frac{1}{f_{TIM4}} = \frac{1}{125000} = 8 \times 10^{-6}\ s$$

T_{ARR} = 2 ms

Limit přetečení registru ARR_{limit} = ?

$$ARR_{limit} = \frac{T_{ARR}}{T_{přičtění}} = \frac{2 \times 10^{-3}}{8 \times 10^{-6}} = 250$$

$$500 \times T_{ARR} = 500 \times 2 \times 10^{-3} = 1\ s$$

5. Program

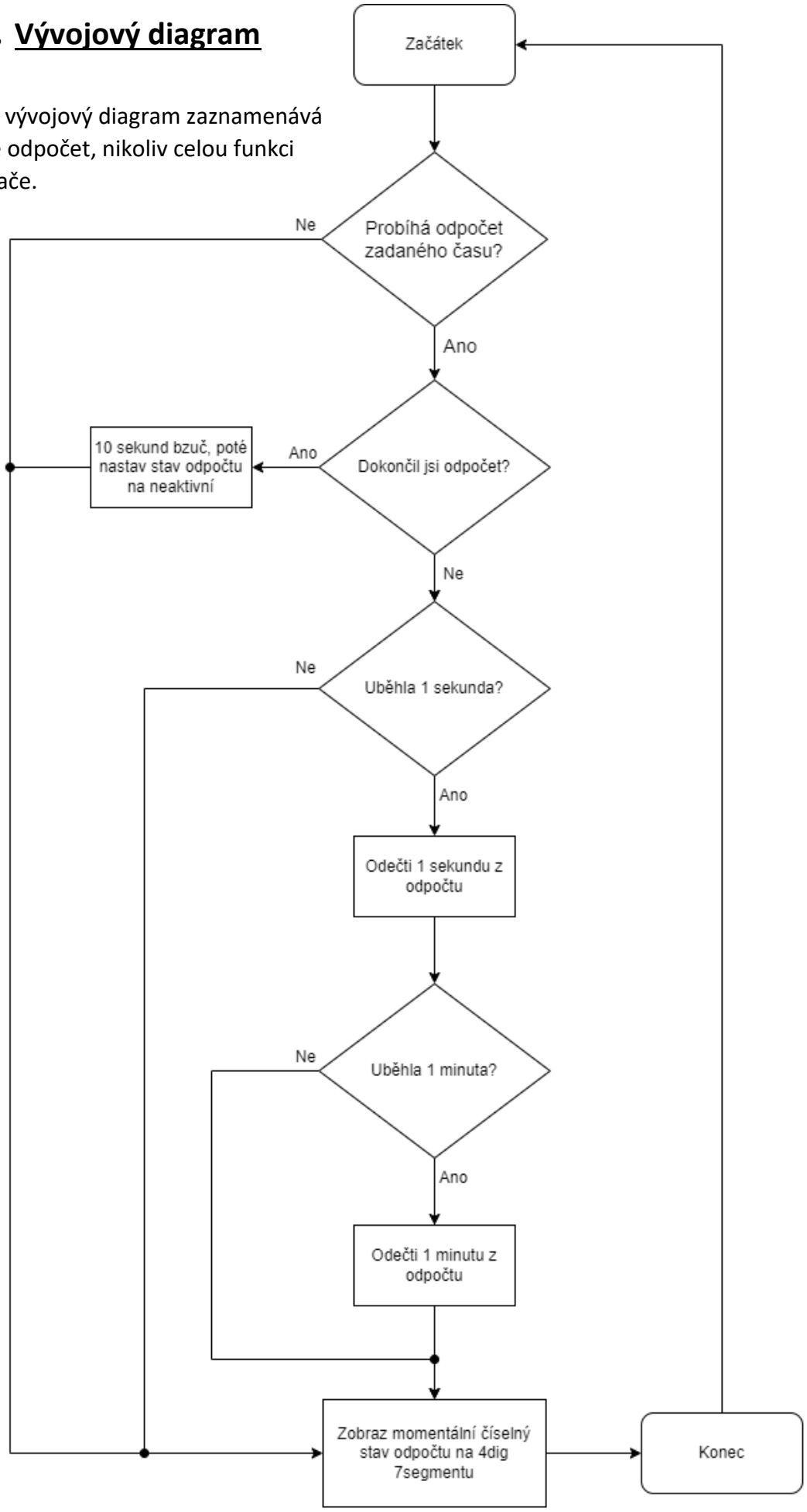
```c
#include "stm8s.h"

// 7segmenty
#define SEG1 (GPIO_PIN_1)
#define SEG2 (GPIO_PIN_2)
#define SEG3 (GPIO_PIN_3)
#define SEG4 (GPIO_PIN_4)

// Tlačítka
#define BUTTON1 (GPIO_PIN_1)
#define BUTTON2 (GPIO_PIN_2)
#define BUTTON3 (GPIO_PIN_3)
#define BUTTON4 (GPIO_PIN_4)
#define BUTTON5 (GPIO_PIN_6)
#define BUTTON6 (GPIO_PIN_7)

// Funkce pro vypínání a zapínání 7segmentů
#define SEGOFF(SEG) (GPIO_WriteHigh(GPIOG, SEG))
#define SEGON(SEG) (GPIO_WriteLow(GPIOG, SEG))

// Funkce pro kontrolu zmáčknutí tlačítka
#define buttonPressed(BUTTON) (GPIO_ReadInputPin(GPIOC, BUTTON)==RESET)

// Funkce pro výpočet délky arraye
#define len(arr) sizeof(arr)/sizeof(arr[0])

// Funkce
void delay(uint32_t iterations);
void adjustTime();
void changeState();
void reset();
void buzz(int timeUnit);

// Globální proměnné - potřebné pro interrupt
uint8_t state = 0; // Stav časovače (0 = neaktivní, 1 = odpočítává)
uint8_t segValues[4] = {0, 0, 0, 0}; // Číselné odnoty na 4dig 7segment
GPIO_Pin_TypeDef segButtons[4] = {BUTTON1, BUTTON2, BUTTON3, BUTTON4}; // Tlačítka pro nastavování času
int secs; // Počet sekund
int mins; // Počet minut
uint16_t buzzCount = 0; // Počítání bzučení

// Přerušení
INTERRUPT_HANDLER(EXTI_PORTC_IRQHandler, 5)
{
    if (state == 0) {adjustTime();} // Pokud je stav odpočtu neaktivní, je možné nastavit čas na odpočet

    if (buttonPressed(BUTTON5)) {changeState();} // Změna stavu odpočtu při zmáčknutí tlačítka 5

    if (buttonPressed(BUTTON6)) {reset();} // Resetování odpočtu při zmáčknutí tlačítka 6
}

void main(void)
{
    // Deinicializace potřebných modulů
    GPIO_DeInit;
    TIM4_DeInit;
    EXTI_DeInit;

    CLK_HSIPrescalerConfig(CLK_PRESCALER_HSIDIV1); // FREQ MCU 16MHz

    // Inicializace potřebných portů na STM8
    GPIO_Init(GPIOC, GPIO_PIN_ALL, GPIO_MODE_IN_PU_IT);
    GPIO_Init(GPIOG, GPIO_PIN_ALL, GPIO_MODE_OUT_PP_LOW_SLOW);
    GPIO_Init(GPIOD, GPIO_PIN_6, GPIO_MODE_OUT_PP_LOW_SLOW);
    GPIO_Init(GPIOB, GPIO_PIN_ALL, GPIO_MODE_OUT_PP_LOW_SLOW);

    // Nastavení přerušení
    EXTI_SetExtIntSensitivity(EXTI_PORT_GPIOC, EXTI_SENSITIVITY_FALL_ONLY);
    ITC_SetSoftwarePriority(ITC_IRQ_PORTC, ITC_PRIORITYLEVEL_0);
    enableInterrupts();

    // Nastavení časování
    TIM4_TimeBaseInit(TIM4_PRESCALER_128, 250);
    TIM4_Cmd(ENABLE);

```

```
    // Lokální proměnné - nepotřebné pro interrupt
    GPIO_Pin_TypeDef SEGS[8] = {SEG1, SEG2, SEG3, SEG4}; // 7segmenty
    uint8_t nums[10] = {0b00111111, 0b00000110, 0b01011011,
                        0b01001111, 0b01100110, 0b01101101,
                        0b01111101, 0b00000111, 0b01111111,
                        0b01100111}; // Čísla zobrazované na 4dig 7segmentu
    uint32_t timeUnit = 0; // Jednotka pro počítání času (500 = 1 sekunda)

    // Výpočet sekund a minut z čísel na 4dig 7segmentu
    secs = segValues[2] * 10 + segValues[3];
    mins = segValues[0] * 10 + segValues[1];

    while (1)
    {
        // Odpočítávání zadaného času
        if (state == 1) {

            // Kontrola odpočtu
            uint8_t segSum = 0; // Součet hodnot na 4dig 7segmentu (0 -> odpočet ukončen)
            for(uint8_t i = 0; i < len(segValues); i++) {
                segSum += segValues[i];
            }

            if(TIM4_GetFlagStatus(TIM4_FLAG_UPDATE)==SET) {
                TIM4_ClearFlag(TIM4_FLAG_UPDATE);
                timeUnit++;

                // Pokud časovač dokončil odpočet, 10 sekund bzuč, pak nastav stav odpočtu na neaktivní
                if (segSum == 0) {
                    if (timeUnit == 500) {
                        timeUnit = 0;
                        buzzCount++;
                    }
                    buzz(timeUnit);
                }
            }

            if (segSum != 0) {
                // Pokud uběhla sekunda, odečti jednu sekundu z odpočtu
                if (timeUnit == 500) {
                    timeUnit = 0;
                    secs--;
                }

                // Pokud uběhla minuta, odečti jednu minut z odpočtu
                if (secs<0) {
                    secs = 59;
                    if (mins != 0) {
                        mins--;
                    }
                }

                // Převeď sekundy a minuty zpět na číselné hodnoty na 4dig 7segmentu
                segValues[3] = secs % 10;
                segValues[2] = secs / 10;
                segValues[1] = mins % 10;
                segValues[0] = mins / 10;
            }
        }

        // Zobrazení číselných hodnot na 4dig 7segmentu
        for (uint8_t i = 0; i < len(SEGS); i++) {
            SEGON(SEGS[i]);
            GPIO_Write(GPIOB, nums[segValues[i]]);
            if (i == 1) {GPIO_WriteHigh(GPIOB, GPIO_PIN_7);} // Zobrazení tečky mezi minutami a sekundami
            delay(150);
            SEGOFF(SEGS[i]);
        }
    }
}
```

```
147  // Pozastavení
148  void delay(uint32_t iter)
149  {
150      for(uint32_t i = 0; i < iter; i++);
151  }
152
153  // Změna číselných hodnot, které se mají odpočítávat
154  void adjustTime() {
155      for (uint8_t i = 0; i < len(segButtons); i++) {
156          if (buttonPressed(segButtons[i])) {
157              // Pokud bylo tlačítko zmáčknuto, přičti 1 k číselné hodnotě na daném 7segmentu
158              segValues[i]++;
159              if (i == 2) { // Třetí 7segment je nastavitelný pouze do pěti
160                  if (segValues[i] > 5) {
161                      segValues[i] = 0;
162                  }
163              } else { // Ostatní 7segmenty jsou nastavitelné do devíti
164                  if (segValues[i] > 9) {
165                      segValues[i] = 0;
166                  }
167              }
168          }
169          // Přepočtení upravených číselných hodnot 4dig 7segmentu na minuty a sekundy
170          mins = segValues[0] * 10 + segValues[1];
171          secs = segValues[2] * 10 + segValues[3];
172      }
173  }
174
175  // Změna stavu odpočtu - je možné odpočet spustit nebo pozastavit
176  void changeState() {
177      if (state == 0) {
178          state = 1;
179          buzzCount = 0;
180      } else {
181          state = 0;
182      }
183  }
184
185  // Resetování odpočtu
186  void reset() {
187      state = 0;
188      GPIO_WriteLow(GPIOD, GPIO_PIN_6); // Zastavení bzučení, pokud je v provozu
189
190      // Vynulování času na 4dig 7segmentu
191      for (uint8_t i = 0; i < len(segValues); i++) {
192          segValues[i] = 0;
193      }
194      mins = segValues[0] * 10 + segValues[1];
195      secs = segValues[2] * 10 + segValues[3];
196  }
197
198  // 10ti-sekundové bzučení
199  void buzz(int timeUnit) {
200      if (buzzCount < 10) {
201          // Chvíli ticho, chvíli bzučí
202          if (timeUnit == 100) {
203              GPIO_WriteReverse(GPIOD, GPIO_PIN_6);
204          }
205      // Po 10tisekundách se se zastaví bzučení a stav odpočtu se nastaví na neaktivní
206      } else if (buzzCount == 10) {
207          GPIO_WriteLow(GPIOD, GPIO_PIN_6);
208          state = 0;
209      }
210  }
```

6. Vývojový diagram

Tento vývojový diagram zaznamenává pouze odpočet, nikoliv celou funkci časovače.



7. Zhodnocení, závěr

Projekt úspěšně a bez komplikací funguje. Uživatel je schopný nastavit čas, začít odpočítávat a kdykoliv odpočet zastavit nebo resetovat. Při realizaci jsem na programově jednoduchou aplikaci zaznamenal velkou spotřebu vodičů a zdlouhavé zapojování vodičů na nepájivé pole a piny STM8. Připomenul jsem si znalosti jazyku C a naučil se lépe ovládat komunikaci mikroprocesoru se vstupními a výstupními periferiemi.

Jindřich Machka, 3.A, 2021/2022