

ČASOVAČ

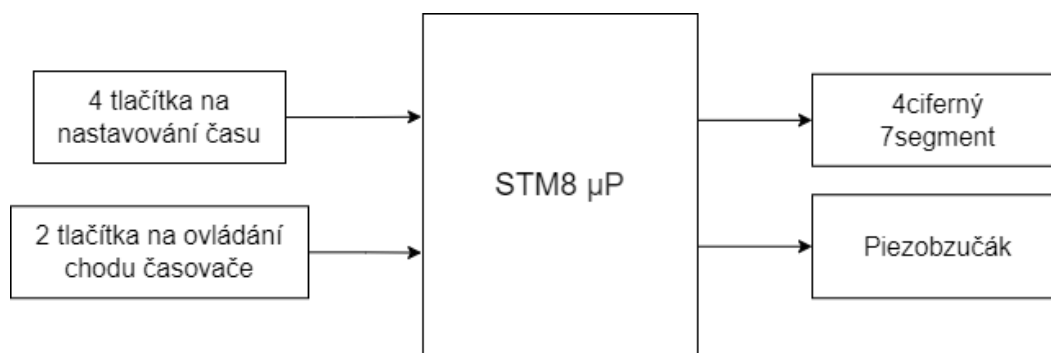
1. Slovní popis

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

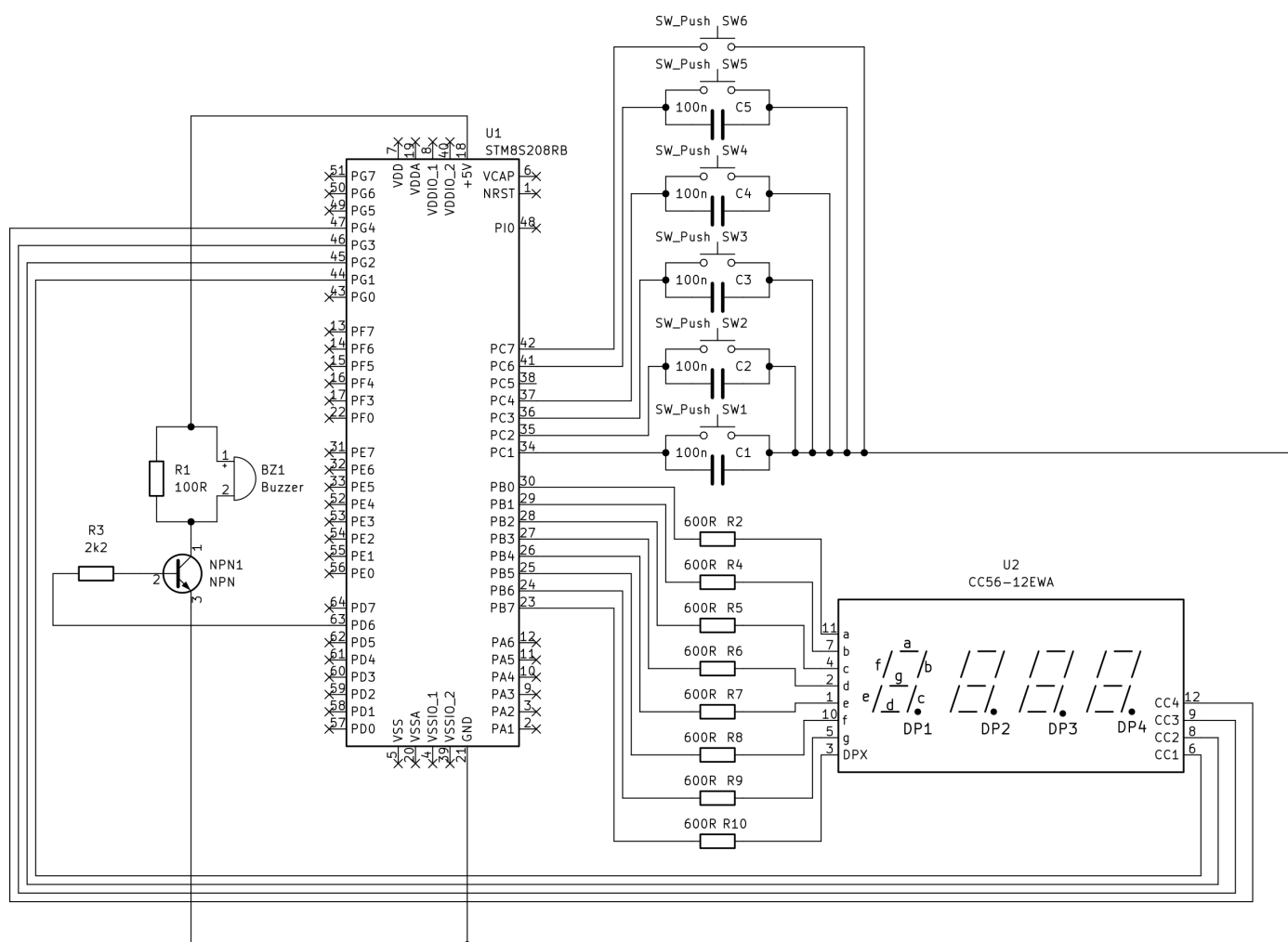
Tlačítko 6 – tlačítko pro resetování odpočtu a vynulování číselných hodnot na 4ciferném 7segmentovém displeji.

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. Výpis programu

```
1  #include "stm8s.h"
2
3  // 7segmenty
4  #define SEG1 (GPIO_PIN_1)
5  #define SEG2 (GPIO_PIN_2)
6  #define SEG3 (GPIO_PIN_3)
7  #define SEG4 (GPIO_PIN_4)
8
9  // Tlačítka
10 #define BUTTON1 (GPIO_PIN_1)
11 #define BUTTON2 (GPIO_PIN_2)
12 #define BUTTON3 (GPIO_PIN_3)
13 #define BUTTON4 (GPIO_PIN_4)
14 #define BUTTON5 (GPIO_PIN_6)
15 #define BUTTON6 (GPIO_PIN_7)
16
17 // Funkce pro vypínání a zapínání 7segmentů
18 #define SEGOFF(SEG) (GPIO_WriteHigh(GPIOG, SEG))
19 #define SEGON(SEG) (GPIO_WriteLow(GPIOG, SEG))
20
21 // Funkce pro kontrolu zmáčknutí tlačítka
22 #define buttonPressed(BUTTON) (GPIO_ReadInputPin(GPIOC, BUTTON)==RESET)
23
24 // Funkce pro výpočet délky arraye
25 #define len(arr) sizeof(arr)/sizeof(arr[0])
26
27 // Funkce
28 void delay(uint32_t iterations);
29 void adjustTime();
30 void changeState();
31 void reset();
32 void buzz(int timeUnit);
33
34 // Globální proměnné - potřebné pro interrupt
35 uint8_t state = 0; // Stav časovače (0 = neaktivní, 1 = odpočítává)
36 uint8_t segValues[4] = {0, 0, 0, 0}; // Číselné odnoty na 4dig 7segment
37 GPIO_Pin_TypeDef segButtons[4] = {BUTTON1, BUTTON2, BUTTON3, BUTTON4}; // Tlačítka pro nastavování času
38 int secs; // Počet sekund
39 int mins; // Počet minut
40 uint16_t buzzCount = 0; // Počítání bzučení
41
42 // Přerušení
43 INTERRUPT_HANDLER(EXTI_PORTC_IRQHandler, 5)
44 {
45     if (state == 0) {adjustTime();} // Pokud je stav odpočtu neaktivní, je možné nastavit čas na odpočet
46
47     if (buttonPressed(BUTTON5)) {changeState();} // Změna stavu odpočtu při zmáčknutí tlačítka 5
48
49     if (buttonPressed(BUTTON6)) {reset();} // Resetování odpočtu při zmáčknutí tlačítka 6
50 }
51
52 void main(void)
53 {
54     // Deinitializace potřebných modulů
55     GPIO_DeInit;
56     TIM4_DeInit;
57     EXTI_DeInit;
58
59     CLK_HSIPrescalerConfig(CLK_PRESCALER_HSIDIV1); // FREQ MCU 16MHz
60
61     // Inicializace potřebných portů na STM8
62     GPIO_Init(GPIOC, GPIO_PIN_ALL, GPIO_MODE_IN_PU_IT);
63     GPIO_Init(GPIOG, GPIO_PIN_ALL, GPIO_MODE_OUT_PP_LOW_SLOW);
64     GPIO_Init(GPIOD, GPIO_PIN_6, GPIO_MODE_OUT_PP_LOW_SLOW);
65     GPIO_Init(GPIOB, GPIO_PIN_ALL, GPIO_MODE_OUT_PP_LOW_SLOW);
66
67     // Nastavení přerušení
68     EXTI_SetExtIntSensitivity(EXTI_PORT_GPIOC, EXTI_SENSITIVITY_FALL_ONLY);
69     ITC_SetSoftwarePriority(ITC_IRQ_PORTC, ITC_PRIORITYLEVEL_0);
70     enableInterrupts();
71
72     // Nastavení časování
73     TIM4_TimeBaseInit(TIM4_PRESCALER_128, 250);
74     TIM4_Cmd(ENABLE);
75 }
```

```

76 // Lokální proměnné - nepotřebné pro interrupt
77 GPIO_Pin_TypeDef SEGS[8] = {SEG1, SEG2, SEG3, SEG4}; // 7segmenty
78 uint8_t nums[10] = {0b00111111, 0b00000110, 0b01011011,
79                     0b01001111, 0b01100110, 0b01101101,
80                     0b01111101, 0b00000111, 0b01111111,
81                     0b01100111}; // Čísla zobrazované na 4dig 7segmentu
82 uint32_t timeUnit = 0; // Jednotka pro počítání času (500 = 1 sekunda)
83
84 // Výpočet sekund a minut z čísel na 4dig 7segmentu
85 secs = segValues[2] * 10 + segValues[3];
86 mins = segValues[0] * 10 + segValues[1];
87
88 while (1)
89 {
90     // Odpočítávání zadaného času
91     if (state == 1) {
92
93         // Kontrola odpočtu
94         uint8_t segSum = 0; // Součet hodnot na 4dig 7segmentu (0 -> odpočet ukončen)
95         for(uint8_t i = 0; i < len(segValues); i++) {
96             segSum += segValues[i];
97         }
98
99         if(TIM4_GetFlagStatus(TIM4_FLAG_UPDATE)==SET) {
100             TIM4_ClearFlag(TIM4_FLAG_UPDATE);
101             timeUnit++;
102
103             // Pokud časovač dokončil odpočet, 10 sekund bzuč, pak nastav stav odpočtu na neaktivní
104             if (segSum == 0) {
105                 if (timeUnit == 500) {
106                     timeUnit = 0;
107                     buzzCount++;
108                 }
109                 buzz(timeUnit);
110             }
111         }
112
113         if (segSum != 0) {
114             // Pokud uběhla sekunda, odečti jednu sekundu z odpočtu
115             if (timeUnit == 500) {
116                 timeUnit = 0;
117                 secs--;
118             }
119
120             // Pokud uběhla minuta, odečti jednu minut z odpočtu
121             if (secs<0) {
122                 secs = 59;
123                 if (mins != 0) {
124                     mins--;
125                 }
126             }
127
128             // Převed sekundy a minuty zpět na číselné hodnoty na 4dig 7segmentu
129             segValues[3] = secs % 10;
130             segValues[2] = secs / 10;
131             segValues[1] = mins % 10;
132             segValues[0] = mins / 10;
133         }
134     }
135
136     // Zobrazení číselných hodnot na 4dig 7segmentu
137     for (uint8_t i = 0; i < len(SEGS); i++) {
138         SEGON(SEGS[i]);
139         GPIO_Write(GPIOB, nums[segValues[i]]);
140         if (i == 1) {GPIO_WriteHigh(GPIOB, GPIO_PIN_7);} // Zobrazení tečky mezi minutami a sekundami
141         delay(150);
142         SEGOFF(SEGS[i]);
143     }
144 }
145
146

```

```

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     }
170     // Přepočtení upravených číselných hodnot 4dig 7segmentu na minuty a sekundy
171     mins = segValues[0] * 10 + segValues[1];
172     secs = segValues[2] * 10 + segValues[3];
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(GPIO, 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         // Chvilí ticho, chvíli bzučí
202         if (timeUnit == 100) {
203             GPIO_WriteReverse(GPIO, 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(GPIO, GPIO_PIN_6);
208         state = 0;
209     }
210 }
211

```

5. Zhodnocení, závěr

Projekt úspěšně a bez komplikací funguje. Jediným problémem může být spotřeba vodičů a zdoluhavé zapojování vodičů na nepájivé pole a piny STM8. Při implementaci projektu jsem si osvěžil znalosti jazyku C a naučil se lépe ovládat komunikaci mikroprocesoru se vstupními a výstupními periferiemi.