

```
1  /*****
2  *
3  * Stopwatch with LCD display output.
4  * ATmega328P (Arduino Uno), 16 MHz, AVR 8-bit Toolchain 3.6.2
5  *
6  * Copyright (c) 2017-2020 Tomas Fryza
7  * Dept. of Radio Electronics, Brno University of Technology, Czechia
8  * This work is licensed under the terms of the MIT license.
9  *
10 *****/
11
12 /* Includes ----- */
13 #include <avr/io.h>          // AVR device-specific IO definitions
14 #include <avr/interrupt.h>    // Interrupts standard C library for AVR-GCC
15 #include "timer.h"           // Timer library for AVR-GCC
16 #include "lcd.h"             // Peter Fleury's LCD library
17 #include <stdlib.h>           // C library. Needed for conversion function
18
19 /* Variables ----- */
20 // Custom character definition using https://omerk.github.io/lcdchangen/
21 uint8_t customChar[8*6] = {
22     0b00100,
23     0b01110,
24     0b11111,
25     0b11111,
26     0b11111,
27     0b00100,
28     0b01110,
29     0b11111,
30
31     //progressBar1
32     0b10000,
33     0b10000,
34     0b10000,
35     0b10000,
36     0b10000,
37     0b10000,
38     0b10000,
39     0b10000,
40
41     //progressBar2
42     0b11000,
43     0b11000,
44     0b11000,
45     0b11000,
46     0b11000,
47     0b11000,
48     0b11000,
49     0b11000,
50
51     //progressBar3
52     0b11100,
53     0b11100,
```

```
54     0b11100,
55     0b11100,
56     0b11100,
57     0b11100,
58     0b11100,
59     0b11100,
60
61     //progressBar4
62     0b11110,
63     0b11110,
64     0b11110,
65     0b11110,
66     0b11110,
67     0b11110,
68     0b11110,
69     0b11110,
70
71     //progressBar5
72     0b11111,
73     0b11111,
74     0b11111,
75     0b11111,
76     0b11111,
77     0b11111,
78     0b11111,
79     0b11111
80 };
81
82 uint8_t running_text[] = "    I like Digital electronics!\n";
83
84 /* Function definitions ----- */
85 /**
86  * Main function where the program execution begins. Update stopwatch
87  * value on LCD display when 8-bit Timer/Counter2 overflows.
88  */
89 int main(void)
90 {
91     // Initialize LCD display
92     lcd_init(LCD_DISP_ON);
93
94     // Set pointer to beginning of CGRAM memory
95     lcd_command(1 << LCD_CGRAM);
96     for (uint8_t i = 0; i < 8*6; i++)
97     {
98         // Store all new chars to memory line by line
99         lcd_data(customChar[i]);
100     }
101     // Set DDRAM address
102     lcd_command(1 << LCD_DDRAM);
103
104     // Display custom characters
105     lcd_putc(0);
106     lcd_gotoxy(15, 0);
```

```
107     lcd_putc(0);
108
109     // Put string(s) at LCD display
110     lcd_gotoxy(1, 0);
111     lcd_puts("00:00.0");
112     lcd_gotoxy(11, 0);
113     lcd_putc('a');
114     lcd_gotoxy(1, 1);
115     lcd_putc('b');
116     lcd_gotoxy(11, 1);
117     lcd_putc('c');
118
119     // Configure 8-bit Timer/Counter2 for Stopwatch
120     // Enable interrupt and set the overflow prescaler to 4 ms
121     TIM2_overflow_4ms();
122     TIM2_overflow_interrupt_enable();
123
124
125     // Configure 8-bit Timer/Counter0 for Stopwatch
126     // Enable interrupt and set the overflow prescaler to 4 ms
127     TIM0_overflow_4ms();
128     TIM0_overflow_interrupt_enable();
129
130
131     // Enables interrupts by setting the global interrupt mask
132     sei();
133
134     // Infinite loop
135     while (1)
136     {
137         /* Empty loop. All subsequent operations are performed exclusively
138          * inside interrupt service routines ISRs */
139     }
140
141     // Will never reach this
142     return 0;
143 }
144
145 /* Interrupt service routines ----- */
146 /**
147  * ISR starts when Timer/Counter0 overflows.
148  * 5 x 50 x 4ms = 1s
149  * (5 x 4ms = 20ms) for one part of bar
150  */
151 ISR(TIMER0_OVF_vect)
152 {
153     static uint8_t number_of_overflows = 0;
154     static uint8_t nth_cell = 0;    // nth cell bar
155     static uint8_t part_of_bar = 0; // part of bar
156
157     number_of_overflows++;
158     if (number_of_overflows >= 5)
159     {
```

```

160     // Do this every 5 x 4 ms = 20 ms
161     number_of_overflows = 0;
162
163     part_of_bar++;
164     if(part_of_bar >= 5)
165     {
166         part_of_bar = 0;
167         nth_cell++;
168         if(nth_cell > 9)
169         {
170             nth_cell = 0;
171             lcd_gotoxy(1, 1);
172             lcd_puts(" "); //reset
173         }
174     }
175     lcd_gotoxy(nth_cell+1, 1); //position of bar
176     lcd_putc(part_of_bar+1); //part of bar from memory
177 }
178 }
179
180
181 /**
182  * ISR starts when Timer/Counter2 overflows.
183  * 10 x 25 x 4ms = 1s
184  * (25 x 4 ms = 100 ms) for one tenth of a second
185  */
186 ISR(TIMER2_OVF_vect)
187 {
188     static uint8_t number_of_overflows = 0;
189     static uint8_t tens = 0; // Tenths of a second
190     static uint8_t secs = 0; // Seconds
191     static uint32_t secs_secpow = 0; // Seconds to the second power
192     static uint8_t mins = 0; // Minutes
193     char lcd_string[5] = " "; // String for converting numbers by itoa ↗
194     ()
195
196     number_of_overflows++;
197     if (number_of_overflows > 25)
198     {
199         // Do this every 25 x 4 ms = 100 ms
200         number_of_overflows = 0;
201
202         // WRITE YOUR CODE HERE
203         tens++;
204         if(tens > 9){ //10 x 0.1s = 1.0s
205             tens = 0;
206             secs++;
207         }
208         if(secs > 59){ //60 x 1s = 1min 0s
209             secs = 0;
210             mins++;
211         }

```

```
212     if(mins > 59) //60 x 1min = (1h) 0min
213         mins = 0;
214
215     secs_secpow = secs*secs;
216
217
218     // tenths of seconds
219     itoa(tens, lcd_string, 10); // Convert decimal value to string
220     lcd_gotoxy(7, 0);
221     lcd_puts(lcd_string);
222
223     // seconds
224     if(secs < 10){
225         lcd_gotoxy(4, 0);
226         lcd_putc('0');
227         lcd_gotoxy(5, 0);
228     }else
229         lcd_gotoxy(4, 0);
230     itoa(secs, lcd_string, 10); // Convert decimal value to string
231     lcd_puts(lcd_string);
232
233     // minutes
234     if(mins < 10){
235         lcd_gotoxy(1, 0);
236         lcd_putc('0');
237         lcd_gotoxy(2, 0);
238     }else
239         lcd_gotoxy(1, 0);
240     itoa(mins, lcd_string, 10); // Convert decimal value to string
241     lcd_puts(lcd_string);
242
243     // seconds to the second power
244     if(secs == 0){
245         lcd_gotoxy(11, 0);
246         lcd_puts(" "); //reset
247     }
248     lcd_gotoxy(11, 0);
249     itoa(secs_secpow, lcd_string, 10); // Convert decimal value to
250     string
251     lcd_puts(lcd_string);
252 }
```