

```

1  /*****
2  *
3  * Seven-segment display library for AVR-GCC.
4  * ATmega328P (Arduino Uno), 16 MHz, AVR 8-bit Toolchain 3.6.2
5  *
6  * Copyright (c) 2019-2020 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 #define F_CPU 16000000
14 #include <util/delay.h>
15 #include "gpio.h"
16 #include "segment.h"
17
18 /* Variables ----- */
19 // Active-low digit a-f
20 uint8_t segment_value[] = {
21     // abcdefgDP
22     0b01111111,    // Digit a
23     0b10111111,    // Digit b
24     0b11011111,    // Digit c
25     0b11101111,    // Digit d
26     0b11110111,    // Digit e
27     0b11111011,    // Digit f
28 };
29
30 // Active-high position 0 to 3
31 uint8_t segment_position[] = {
32     // p3p2p1p0....
33     0b00010000,    // Position 0
34     0b00100000,    // Position 1
35     0b01000000,    // Position 2
36     0b10000000,    // Position 3
37 };
38
39
40 /* Function definitions ----- */
41 void SEG_init(void)
42 {
43     /* Configuration of SSD signals */
44     GPIO_config_output(&DDRD, SEGMENT_LATCH);
45     GPIO_config_output(&DDRD, SEGMENT_CLK);
46     GPIO_config_output(&DDRB, SEGMENT_DATA);
47 }
48
49 /*----- */
50 void SEG_update_shift_regs(uint8_t segments, uint8_t position)
51 {
52     uint8_t bit_number;
53     segments = segment_value[segments];    // 0, 1, ..., 5

```

```
54     position = segment_position[position]; // 0
55
56     // Pull LATCH, CLK, and DATA low
57     GPIO_write_low(&PORTD, SEGMENT_LATCH);
58     GPIO_write_low(&PORTD, SEGMENT_CLK);
59     GPIO_write_low(&PORTB, SEGMENT_DATA);
60
61     // Wait 1 us
62     _delay_us(1);
63
64     // Loop through the 1st byte (segments)
65     // a b c d e f g DP (active low values)
66     for (bit_number = 0; bit_number < 8; bit_number++)
67     {
68         // Output DATA value (bit 0 of "segments")
69         if ((segments & 1) == 0)
70             GPIO_write_low(&PORTB, SEGMENT_DATA);
71         else
72             GPIO_write_high(&PORTB, SEGMENT_DATA);
73
74         // Wait 1 us
75         _delay_us(1);
76
77         // Pull CLK high
78         GPIO_write_high(&PORTD, SEGMENT_CLK);
79
80         // Wait 1 us
81         _delay_us(1);
82
83         // Pull CLK low
84         GPIO_write_low(&PORTD, SEGMENT_CLK);
85
86         // Shift "segments"
87         segments = segments >> 1;
88     }
89
90     // Loop through the 2nd byte (position)
91     // p3 p2 p1 p0 . . . . (active high values)
92     for (bit_number = 0; bit_number < 8; bit_number++)
93     {
94         // Output DATA value (bit 0 of "position")
95         if ((position % 2) == 0)
96             GPIO_write_low(&PORTB, SEGMENT_DATA);
97         else
98             GPIO_write_high(&PORTB, SEGMENT_DATA);
99
100        // Wait 1 us
101        _delay_us(1);
102
103        // Pull CLK high
104        GPIO_write_high(&PORTD, SEGMENT_CLK);
105
106        // Wait 1 us
```

```
107     _delay_us(1);
108
109     // Pull CLK low
110     GPIO_write_low(&PORTD, SEGMENT_CLK);
111
112     // Shift "position"
113     position = position >> 1;
114 }
115
116 // Pull LATCH high
117 GPIO_write_high(&PORTD, SEGMENT_LATCH);
118
119 // Wait 1 us
120 _delay_us(1);
121 }
122
123 /*----- */
124 /* SEG_clear */
125 void SEG_clear(void)
126 {
127     uint8_t bit_number, segments = 0b11111111, position = 0;
128
129     // Pull LATCH, CLK, and DATA low
130     GPIO_write_low(&PORTD, SEGMENT_LATCH);
131     GPIO_write_low(&PORTD, SEGMENT_CLK);
132     GPIO_write_low(&PORTB, SEGMENT_DATA);
133
134     // Wait 1 us
135     _delay_us(1);
136
137     // Loop through the 1st byte (segments)
138     // a b c d e f g DP (active low values)
139     for (bit_number = 0; bit_number < 8; bit_number++)
140     {
141         // Output DATA value (bit 0 of "segments")
142         if ((segments & 1) == 0)
143             GPIO_write_low(&PORTB, SEGMENT_DATA);
144         else
145             GPIO_write_high(&PORTB, SEGMENT_DATA);
146
147         // Wait 1 us
148         _delay_us(1);
149
150         // Pull CLK high
151         GPIO_write_high(&PORTD, SEGMENT_CLK);
152
153         // Wait 1 us
154         _delay_us(1);
155
156         // Pull CLK low
157         GPIO_write_low(&PORTD, SEGMENT_CLK);
158
159         // Shift "segments"
```

```
160     segments = segments >> 1;
161 }
162
163 // Loop through the 2nd byte (position)
164 // p3 p2 p1 p0 . . . . (active high values)
165 for (bit_number = 0; bit_number < 8; bit_number++)
166 {
167     // Output DATA value (bit 0 of "position")
168     if ((position % 2) == 0)
169         GPIO_write_low(&PORTB, SEGMENT_DATA);
170     else
171         GPIO_write_high(&PORTB, SEGMENT_DATA);
172
173     // Wait 1 us
174     _delay_us(1);
175
176     // Pull CLK high
177     GPIO_write_high(&PORTD, SEGMENT_CLK);
178
179     // Wait 1 us
180     _delay_us(1);
181
182     // Pull CLK low
183     GPIO_write_low(&PORTD, SEGMENT_CLK);
184
185     // Shift "position"
186     position = position >> 1;
187 }
188
189 // Pull LATCH high
190 GPIO_write_high(&PORTD, SEGMENT_LATCH);
191
192 // Wait 1 us
193 _delay_us(1);
194 }
195
196
197 /*----- */
198 /* SEG_clk_2us */
199 void SEG_clk_2us(void)
200 {
201     // Wait 1 us
202     _delay_us(1);
203
204     // Pull CLK high
205     GPIO_write_high(&PORTD, SEGMENT_CLK);
206
207     // Wait 1 us
208     _delay_us(1);
209
210     // Pull CLK low
211     GPIO_write_low(&PORTD, SEGMENT_CLK);
212 }
```