```c
/***********************************************************************
 *
 * Seven-segment display library for AVR-GCC.
 * ATmega328P (Arduino Uno), 16 MHz, AVR 8-bit Toolchain 3.6.2
 *
 * Copyright (c) 2019-2020 Tomas Fryza
 * Dept. of Radio Electronics, Brno University of Technology, Czechia
 * This work is licensed under the terms of the MIT license.
 *
 **********************************************************************/

/* Includes ----------------------------------------------------------*/
#define F_CPU 16000000
#include <util/delay.h>
#include "gpio.h"
#include "segment.h"

/* Variables ---------------------------------------------------------*/
// Active-low digit a-f
uint8_t segment_value[] = {
    // abcdefgDP
    0b01111111,     // Digit a
    0b10111111,     // Digit b
    0b11011111,     // Digit c
    0b11101111,     // Digit d
    0b11110111,     // Digit e
    0b11111011,     // Digit f
};

// Active-high position 0 to 3
uint8_t segment_position[] = {
    // p3p2p1p0....
    0b00010000,     // Position 0
    0b00100000,     // Position 1
    0b01000000,     // Position 2
    0b10000000      // Position 3
};


/* Function definitions ----------------------------------------------*/
void SEG_init(void)
{
    /* Configuration of SSD signals */
    GPIO_config_output(&DDRD, SEGMENT_LATCH);
    GPIO_config_output(&DDRD, SEGMENT_CLK);
    GPIO_config_output(&DDRB, SEGMENT_DATA);
}

/*-------------------------------------------------------------------*/
void SEG_update_shift_regs(uint8_t segments, uint8_t position)
{
    uint8_t bit_number;
    segments = segment_value[segments];     // 0, 1, ..., 5
    position = segment_position[position];  // 0

    // Pull LATCH, CLK, and DATA low
    GPIO_write_low(&PORTD, SEGMENT_LATCH);
    GPIO_write_low(&PORTD, SEGMENT_CLK);
    GPIO_write_low(&PORTB, SEGMENT_DATA);

    // Wait 1 us
    _delay_us(1);

    // Loop through the 1st byte (segments)
    // a b c d e f g DP (active low values)
    for (bit_number = 0; bit_number < 8; bit_number++)
    {
        // Output DATA value (bit 0 of "segments")
        if ((segments & 1) == 0)
            GPIO_write_low(&PORTB, SEGMENT_DATA);
        else
            GPIO_write_high(&PORTB, SEGMENT_DATA);

        // Wait 1 us
        _delay_us(1);

        // Pull CLK high
        GPIO_write_high(&PORTD, SEGMENT_CLK);

        // Wait 1 us
        _delay_us(1);

        // Pull CLK low
        GPIO_write_low(&PORTD, SEGMENT_CLK);

        // Shift "segments"
        segments = segments >> 1;
    }

    // Loop through the 2nd byte (position)
    // p3 p2 p1 p0 . . . . (active high values)
    for (bit_number = 0; bit_number < 8; bit_number++)
    {
        // Output DATA value (bit 0 of "position")
        if ((position % 2) == 0)
            GPIO_write_low(&PORTB, SEGMENT_DATA);
        else
            GPIO_write_high(&PORTB, SEGMENT_DATA);

       // Wait 1 us
       _delay_us(1);

       // Pull CLK high
       GPIO_write_high(&PORTD, SEGMENT_CLK);

       // Wait 1 us
```

```c
            _delay_us(1);

            // Pull CLK low
            GPIO_write_low(&PORTD, SEGMENT_CLK);

             // Shift "position"
             position = position >> 1;
        }

        // Pull LATCH high
        GPIO_write_high(&PORTD, SEGMENT_LATCH);

        // Wait 1 us
        _delay_us(1);
    }

    /*-------------------------------------------------------------------- */
    /* SEG_clear */
    void SEG_clear(void)
    {
        uint8_t bit_number, segments = 0b11111111, position = 0;

        // Pull LATCH, CLK, and DATA low
        GPIO_write_low(&PORTD, SEGMENT_LATCH);
        GPIO_write_low(&PORTD, SEGMENT_CLK);
        GPIO_write_low(&PORTB, SEGMENT_DATA);

        // Wait 1 us
        _delay_us(1);

        // Loop through the 1st byte (segments)
        // a b c d e f g DP (active low values)
        for (bit_number = 0; bit_number < 8; bit_number++)
        {
            // Output DATA value (bit 0 of "segments")
            if ((segments & 1) == 0)
                GPIO_write_low(&PORTB, SEGMENT_DATA);
            else
                GPIO_write_high(&PORTB, SEGMENT_DATA);

            // Wait 1 us
            _delay_us(1);

            // Pull CLK high
            GPIO_write_high(&PORTD, SEGMENT_CLK);

            // Wait 1 us
            _delay_us(1);

            // Pull CLK low
            GPIO_write_low(&PORTD, SEGMENT_CLK);

            // Shift "segments"
            segments = segments >> 1;
        }

        // Loop through the 2nd byte (position)
        // p3 p2 p1 p0 . . . . (active high values)
        for (bit_number = 0; bit_number < 8; bit_number++)
        {
            // Output DATA value (bit 0 of "position")
            if ((position % 2) == 0)
                GPIO_write_low(&PORTB, SEGMENT_DATA);
            else
                GPIO_write_high(&PORTB, SEGMENT_DATA);

            // Wait 1 us
            _delay_us(1);

            // Pull CLK high
            GPIO_write_high(&PORTD, SEGMENT_CLK);

            // Wait 1 us
            _delay_us(1);

            // Pull CLK low
            GPIO_write_low(&PORTD, SEGMENT_CLK);

            // Shift "position"
            position = position >> 1;
        }

        // Pull LATCH high
        GPIO_write_high(&PORTD, SEGMENT_LATCH);

        // Wait 1 us
        _delay_us(1);

    }

    /*-------------------------------------------------------------------- */
    /* SEG_clk_2us */
    void SEG_clk_2us(void)
    {
        // Wait 1 us
        _delay_us(1);

        // Pull CLK high
        GPIO_write_high(&PORTD, SEGMENT_CLK);

        // Wait 1 us
        _delay_us(1);

        // Pull CLK low
        GPIO_write_low(&PORTD, SEGMENT_CLK);
    }
```