

1.

Digit	A	B	C	D	E	F	G	DP
0	0	0	0	0	0	0	1	1
1	1	0	0	1	1	1	1	1
2	0	0	1	0	0	1	0	1
3	0	0	0	0	1	1	0	1
4	1	0	0	1	1	0	0	1
5	0	1	0	0	1	0	0	1
6	0	1	0	0	0	0	0	1
7	0	0	0	1	1	1	1	1
8	0	0	0	0	0	0	0	1
9	0	0	0	1	1	0	0	1

Common Cathode

LED dané číslice mají společnou katodu. Když na jednu katodu připojíme úroveň L (tím vybereme danou číslici) a na jednu anodu úroveň H, tak se rozsvítí jeden segment.

Common Anode

LED dané číslice mají společnou anodu. Když na jednu anodu připojíme úroveň H (tím vybereme danou číslici) a na jednu katodu úroveň L, tak se rozsvítí jeden segment.

2.

```
/******
 *
 * 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.
 *
 *****/
#define F_CPU 16000000

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

/* Variables -----*/

// Active-low digit 0 to 9
uint8_t segment_value[] = {
    //abcdefgDP
    0b00000011, // Digit 0
    0b10011111, // Digit 1
    0b00100101, // Digit 2
    0b00001101, // Digit 3
    0b10011001, // Digit 4
    0b01001001, // Digit 5
    0b01000001, // Digit 6
    0b00011111, // Digit 7
    0b00000001, // Digit 8
    0b00011001 // Digit 9
};

// 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, ..., 9
    position = segment_position[position]; // 0, 1, 2, 3
}
```

```

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

// Pull LATCH high
GPIO_write_high(&PORTD, SEGMENT_LATCH);
// Wait 1 us
_delay_us(1);
}

```

```

/*****
 *
 * Decimal counter with 7-segment output.
 * ATmega328P (Arduino Uno), 16 MHz, AVR 8-bit Toolchain 3.6.2
 *
 * Copyright (c) 2018-2020 Tomas Fryza
 * Dept. of Radio Electronics, Brno University of Technology, Czechia
 * This work is licensed under the terms of the MIT license.
 *
 *****/

/* Includes -----*/
#include <avr/io.h>           // AVR device-specific IO definitions
#include <avr/interrupt.h>    // Interrupts standard C library for AVR-GCC
#include "timer.h"           // Timer library for AVR-GCC
#include "segment.h"         // Seven-segment display library for AVR-GCC

uint8_t singles = 0, decimals = 0;

/* Function definitions -----*/
/**
 * Main function where the program execution begins. Display decimal
 * counter values on SSD (Seven-segment display) when 16-bit
 * Timer/Counter1 overflows.
 */
int main(void)
{
    // Configure SSD signals
    SEG_init();

    // Test of SSD: display number '3' at position 0
    SEG_update_shift_regs(3, 0);

    /* Configure 16-bit Timer/Counter1
     * Set prescaler and enable overflow interrupt */
    TIM1_overflow_262ms();
    TIM1_overflow_interrupt_enable();
    /* Configure 8-bit Timer/Counter0
     * Set prescaler and enable overflow interrupt */
    TIM0_overflow_4ms();
    TIM0_overflow_interrupt_enable();
    // Enables interrupts by setting the global interrupt mask
    sei();
    // Infinite loop
    while (1)
    {
        /* Empty loop. All subsequent operations are performed exclusively
         * inside interrupt service routines ISRs */
    }

    // Will never reach this
    return 0;
}

```

```

/**
 * ISR starts when Timer/Counter1 overflows. Increment decimal counter
 */

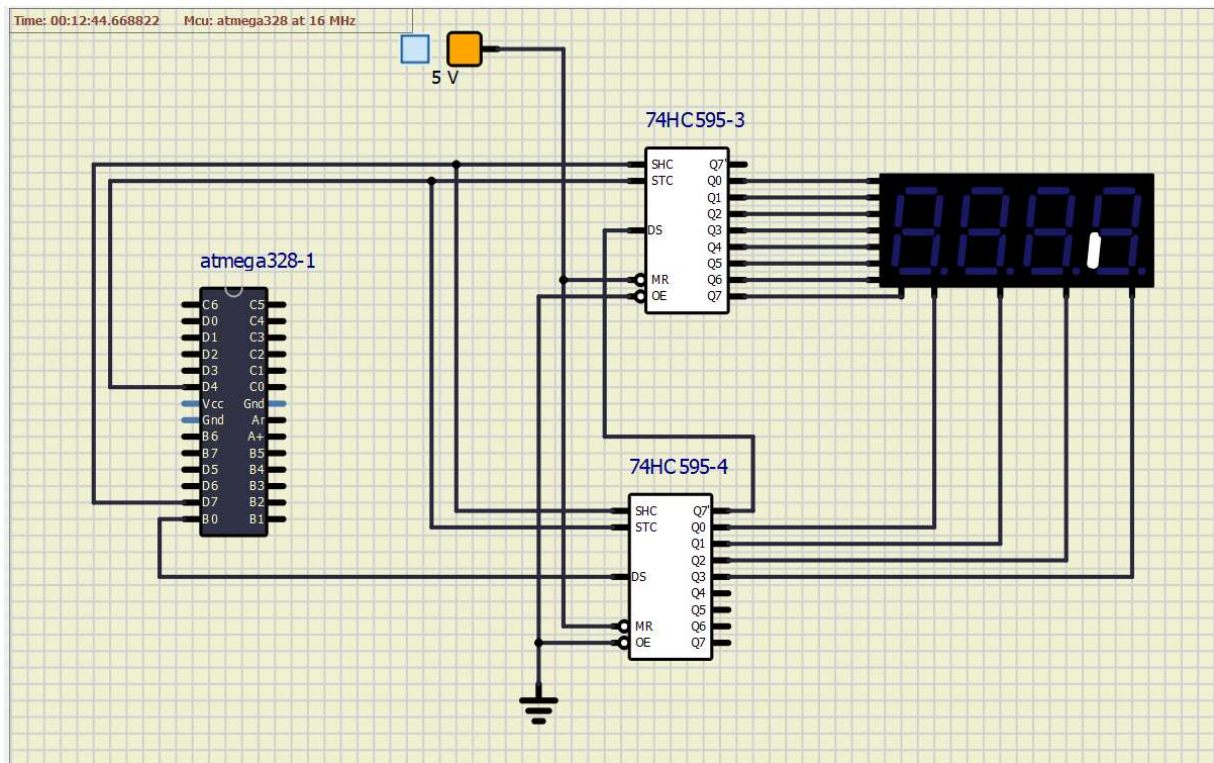
ISR(TIMER1_OVF_vect)
{
    // WRITE YOUR CODE HERE
    singles++;
    if (singles>9)
    {
        singles = 0;
        decimals++;
    }

    if (decimals>5)
    {
        decimals=0;
    }
}

/* Interrupt service routines -----*/
/**
 * ISR starts when Timer/Counter0 overflows. Display value on SSD.
 */

ISR(TIMER0_OVF_vect)
{
    static uint8_t pos = 0;
    if (pos == 0)
    {
        SEG_update_shift_regs(singles,pos);
        pos = 1;
    }
    else
    {
        SEG_update_shift_regs(decimals,pos);
        pos = 0;
    }
}

```



3.

snake (segment)	A	B	C	D	E	F	G	DP
0	0	1	1	1	1	1	1	1
1	1	0	1	1	1	1	1	1
2	1	1	0	1	1	1	1	1
3	1	1	1	0	1	1	1	1
4	1	1	1	1	0	1	1	1
5	1	1	1	1	1	0	1	1

```

/*****
 *
 * Decimal counter with 7-segment output.
 * ATmega328P (Arduino Uno), 16 MHz, AVR 8-bit Toolchain 3.6.2
 *
 * Copyright (c) 2018-2020 Tomas Fryza
 * Dept. of Radio Electronics, Brno University of Technology, Czechia
 * This work is licensed under the terms of the MIT license.
 *
 *****/

/* Includes -----*/
#include <avr/io.h>           // AVR device-specific IO definitions
#include <avr/interrupt.h>    // Interrupts standard C library for AVR-GCC
#include "timer.h"           // Timer library for AVR-GCC
#include "segment.h"         // Seven-segment display library for AVR-GCC

uint8_t snake = 0;

/* Function definitions -----*/
/**
 * Main function where the program execution begins. Display decimal
 * counter values on SSD (Seven-segment display) when 16-bit
 * Timer/Counter1 overflows.
 */
int main(void)
{
    // Configure SSD signals
    SEG_init();

    /* Configure 16-bit Timer/Counter1
     * Set prescaler and enable overflow interrupt */
    TIM1_overflow_262ms();
    TIM1_overflow_interrupt_enable();
    /* Configure 8-bit Timer/Counter0
     * Set prescaler and enable overflow interrupt */
    TIM0_overflow_4ms();
    TIM0_overflow_interrupt_enable();
    // Enables interrupts by setting the global interrupt mask
    sei();
    // Infinite loop
    while (1)
    {

```

```

        /* Empty loop. All subsequent operations are performed exclusively
        * inside interrupt service routines ISRs */
    }

    // Will never reach this
    return 0;
}

/* Interrupt service routines -----*/
/**
ISR(TIMER0_OVF_vect)
{
    static uint8_t pos = 0;
    SEG_update_shift_regs(snake,pos);
}

ISR(TIMER1_OVF_vect)
{
    // WRITE YOUR CODE HERE
    snake++;
    if (snake>5) snake = 0;
}

```


3.