```
/*/*****************************************************************************
 *
 * Blink a LED and use the function from the delay library.
 * 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.
 *
 *****************************************************************************/

/* Defines -----------------------------------------------------------*/
#define LED_GREEN   PB5     // AVR pin where green LED is connected
#define SHORT_DELAY 100      // Delay in miliseconds
#define LONG_DELAY 300
#define SPACE_DELAY 500
#ifndef F_CPU
#define F_CPU 16000000      // CPU frequency in Hz required for delay func
#endif

/* Includes ----------------------------------------------------------*/
#include <util/delay.h>     // Functions for busy-wait delay loops
#include <avr/io.h>         // AVR device-specific IO definitions

/* Variables ---------------------------------------------------------*/

/* Function prototypes -----------------------------------------------*/

/* Functions ---------------------------------------------------------*/
/**
 * Toggle one LED and use the function from the delay library.
 */
int main(void)
{
    // Set pin as output in Data Direction Register
    // DDRB = DDRB or 0010 0000
    DDRB = DDRB | (1<<LED_GREEN);

    // Set pin LOW in Data Register (LED off)
    // PORTB = PORTB and 1101 1111
    PORTB = PORTB & ~(1<<LED_GREEN);

    // Infinite loop
    while (1)
    {
```

```
        // Pause several miliseconds

        _delay_ms(SPACE_DELAY);



        // Invert LED in Data Register

        // PORTB = PORTB xor 0010 0000


        // D

        PORTB = PORTB ^ (1<<LED_GREEN);     // -
        _delay_ms(LONG_DELAY);
        PORTB = PORTB ^ (1<<LED_GREEN);
        _delay_ms(SHORT_DELAY);

        PORTB = PORTB ^ (1<<LED_GREEN);     // .
        _delay_ms(SHORT_DELAY);
        PORTB = PORTB ^ (1<<LED_GREEN);
        _delay_ms(SHORT_DELAY);

        PORTB = PORTB ^ (1<<LED_GREEN);     // .
        _delay_ms(SHORT_DELAY);
        PORTB = PORTB ^ (1<<LED_GREEN);
        _delay_ms(SPACE_DELAY);


        // E

        PORTB = PORTB ^ (1<<LED_GREEN);     // .
        _delay_ms(SHORT_DELAY);
        PORTB = PORTB ^ (1<<LED_GREEN);
        _delay_ms(SPACE_DELAY);


        // 2

        PORTB = PORTB ^ (1<<LED_GREEN);     // .
        _delay_ms(SHORT_DELAY);
        PORTB = PORTB ^ (1<<LED_GREEN);
        _delay_ms(SHORT_DELAY);

        PORTB = PORTB ^ (1<<LED_GREEN);     // .
        _delay_ms(SHORT_DELAY);
        PORTB = PORTB ^ (1<<LED_GREEN);
        _delay_ms(SHORT_DELAY);

        PORTB = PORTB ^ (1<<LED_GREEN);     // -
        _delay_ms(LONG_DELAY);
        PORTB = PORTB ^ (1<<LED_GREEN);
        _delay_ms(SHORT_DELAY);

        PORTB = PORTB ^ (1<<LED_GREEN);     // -
        _delay_ms(LONG_DELAY);
        PORTB = PORTB ^ (1<<LED_GREEN);
        _delay_ms(SHORT_DELAY);

        PORTB = PORTB ^ (1<<LED_GREEN);     // -
        _delay_ms(LONG_DELAY);
        PORTB = PORTB ^ (1<<LED_GREEN);
        _delay_ms(SPACE_DELAY);

    }



    // Will never reach this

    return 0;

}



/* Interrupt routines ----------------------------------------------*/
```