МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ПОВОЛЖСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра ИиСП

Отчет
по лабораторной работе № 5

по дисциплине «Машинно-зависимые языки программирования»
Вариант 1

Выполнил: ст. гр. ПС-11
Ложкин С.А.
Проверил: доцент, доцент кафедры ИиСП Баев А.А.

г. Йошкар-Ола
2023

Цель работы: научиться работать с устройствами вывода (светодиодами, 7-сегментными индикаторами) и изменять состояние программы, используя устройства ввода (кнопка), работать с прерываниями, используя языки программирования C и ассемблер, после чего запустить программу на локальном схематичном окружении.

Задания на лабораторную работу:

1. Сделать на C и asm мигание светодиода;
2. Сделать на C и asm анимацию на 7-сегментном индикаторе;
3. Сделать на C и asm две анимации на 7-сегментном индикаторе с использованием кнопки;
4. Сделать на C и asm секундомер, используя 7-сегментный индикатор, кнопку и прерывания
5. Сделать на C и asm секундомер, используя два 7-сегментных индикатора, кнопку и прерывания.
6. Сделать на C гирлянду с тремя режимами работы

1. Теоретические сведения

Для работы с устройствами вывода необходимо первоначально настроить пины на вход или выход через регистры типа DDRx (в них номер бита на 1 - на выход, 0 - на вход). Для настройки пинов на вход дополнительно необходимо включить подтягивающий к питанию резистор к этому выводу. Это можно настроить в регистрах типа PORTx (в них номер бита на 1 - подача напряжения на выход, если этот пин настроен на вход, то включится подтягивающий резистор к пину, при записи 0 - напряжения на пине не будет)

Устройства ввода как раз работают с пинами, настроенными на вход. Их значения можно считывать нативно через пины и потом их обрабатывать, что делается не постоянно, если в программе есть задержка, или же использовать прерывания, которые обрабатываются всегда. Настройка прерываний осуществляется с помощью регистров EIMSK, чтобы определить, какой пин будет наблюдать за внешинй прерыванием. Регистр EICRA позволяет настроить условия генерации прерываний для каждого из возможных для этого пинов. После настройки прерывания необходимо не забыть включить их.

Что касается настройки окружения и микроконтроллера в Proteus, то необходимо подключать каждый элемент к соответствующему пину или группе пинов, которую вы хотите использовать, и на землю. Реальная цепь подразумевает наличие резисторов для уменьшения силы тока и напряжения, чтобы не повредить периферию, но в случае с моделированием этим можно пренебречь и сосредоточиться только на работоспособности программы и системы в целом.

2. **Практическая часть**

Мигание светодиода

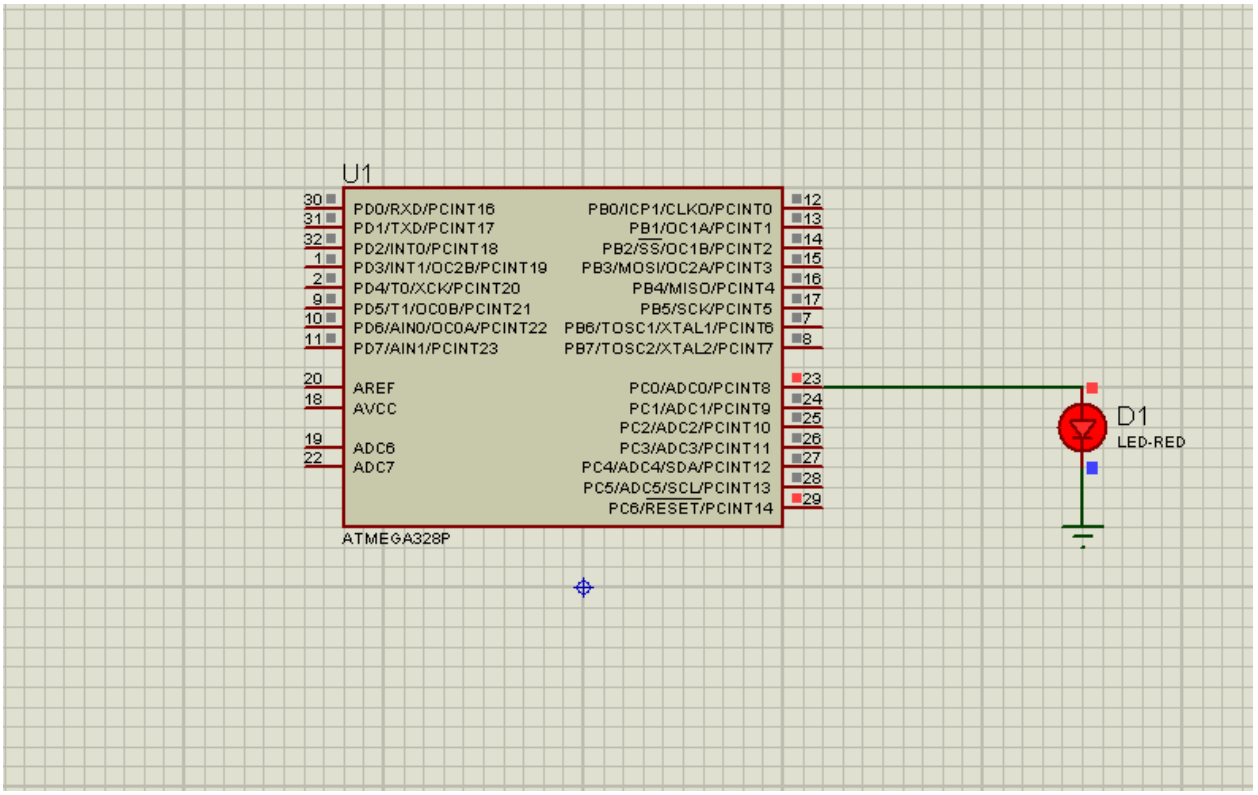
Код на С:

```
#define F_CPU 1000000UL
#include <avr/io.h>
#include <util/delay.h>

int main(void)
{
	DDRC |= (1 << PORTC0);

	int dir = 1;
	while (1)
	{
		if (dir)
		{
			PORTC |= (1 << PORTC0);
			dir = 0;
		}
		else
		{
			PORTC &= ~(1 << PORTC0);
			dir = 1;
		}
		_delay_ms(250);
	}
}
```

Код на asm:

```
setup:
        sbi DDRC,0
        ldi r16,1
        ldi r17,0

loop:
        or r16,r17
        ; if
        breq clear_bit
        ; then
        sbi PORTC,0
        ldi r16,0x00
        rjmp delay
        ;
clear_bit:
        ; else
        cbi PORTC,0
        ldi r16,0x01


delay:
        ldi     r30,35
        ldi     r31,244
delay_loop:
        sbiw    r30,1
        brne    delay_loop


rjmp loop
```

Сборка в Proteus:



7-сегментный индикатор

Код на C:

```
#define F_CPU 1000000UL
#include <avr/io.h>
#include <util/delay.h>

int main(void)
{
	DDRB = 0xFF;
	while(1)
	{
		PORTB = (1 << 0);
		_delay_ms(250);
		PORTB = (1 << 1);
		_delay_ms(250);
```

```
            PORTB = (1 << 2);
            _delay_ms(250);
            PORTB = (1 << 3);
            _delay_ms(250);
            PORTB = (1 << 4);
            _delay_ms(250);
            PORTB = (1 << 5);
            _delay_ms(250);
        }
}
```

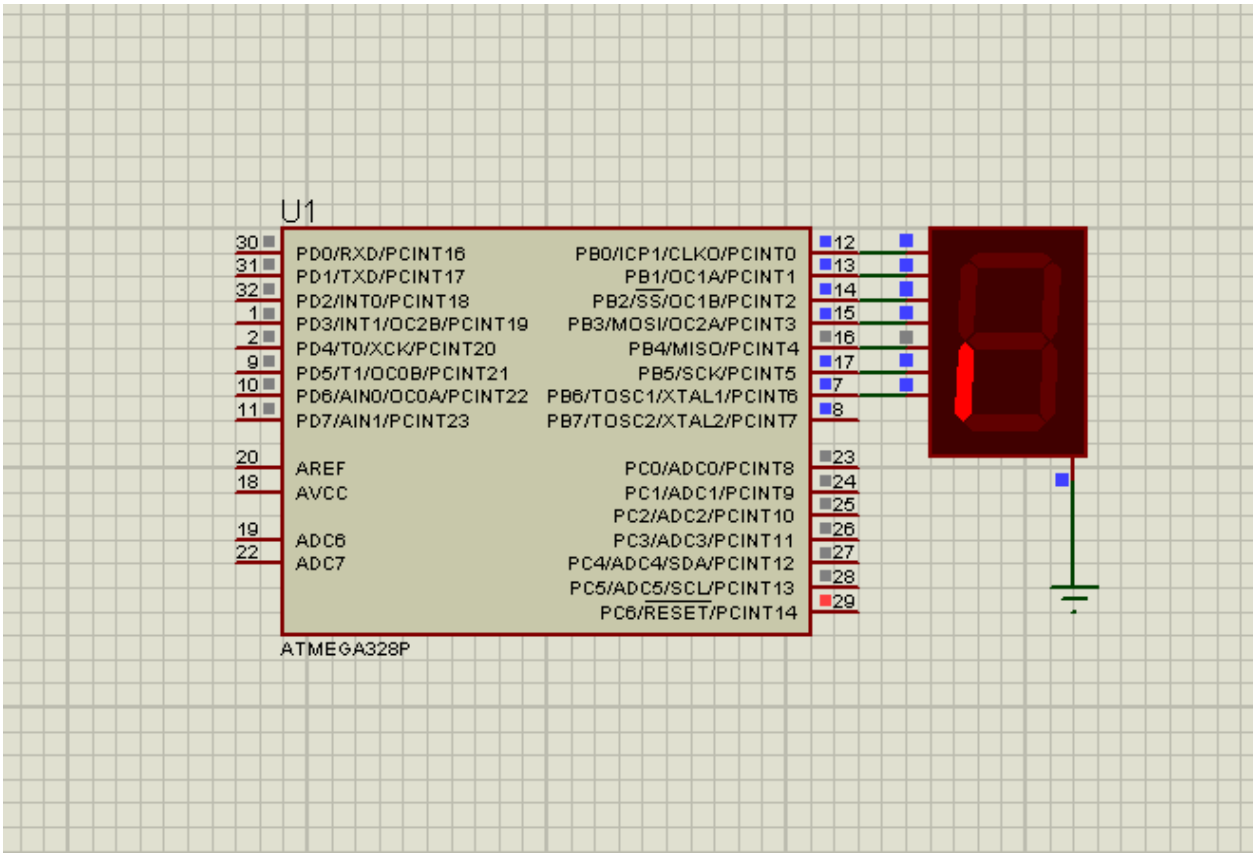
Код на asm:

```
setup:
        ldi r16,0xFF
        out     DDRB,r16    ; весь DDRB на выход
        ldi     r16, 0b00000001   ; 1 (1 << 0)
        ldi     r17, 0b00000010   ; 2 (1 << 1)
        ldi     r18, 0b00000100   ; 4 (1 << 2)
        ldi     r19, 0b00001000   ; 8 (1 << 3)
        ldi     r20, 0b00010000   ; 16(1 << 4)
        ldi     r21, 0b00100000   ; 32(1 << 5)
main:
        out     PORTB, r16 ; PORTB (1 << 0)
        //задержка
        ldi     r30, 0x23     ; 35
        ldi     r31, 0xF4     ; 244
delay1:
        sbiw r30, 0x01       ; вычиаем сразу их двух
        brne delay1
```

```
        out     PORTB,r17  ; PORTB (1 << 1)
        //задержка
        ldi     r30, 0x23     ; 35
        ldi     r31, 0xF4     ; 244
delay2:
        sbiw r30, 0x01       ; вычиаем сразу их двух
        brne delay2


        out     PORTB,r18  ; PORTB (1 << 2)
        //задержка
        ldi     r30, 0x23     ; 35
        ldi     r31, 0xF4     ; 244
delay3:
        sbiw r30, 0x01       ; вычиаем сразу их двух
        brne delay3


        out     PORTB,r19  ; PORTB (1 << 3)
        //задержка
        ldi     r30, 0x23     ; 35
        ldi     r31, 0xF4     ; 244
delay4:
        sbiw r30, 0x01       ; вычиаем сразу их двух
        brne delay4


        out     PORTB,r20  ; PORTB (1 << 4)
        //задержка
        ldi     r30, 0x23     ; 35
```

```
        ldi     r31, 0xF4     ; 244
delay5:
        sbiw r30, 0x01        ; вычиаем сразу их двух
        brne delay5


        out     PORTB,r21  ; PORTB (1 << 5)
        //задержка
        ldi     r30, 0x23     ; 35
        ldi     r31, 0xF4     ; 244
delay6:
        sbiw r30, 0x01        ; вычиаем сразу их двух
        brne delay6

rjmp main
```

Сборка в Proteus:



Подключение кнопки

Код на C:

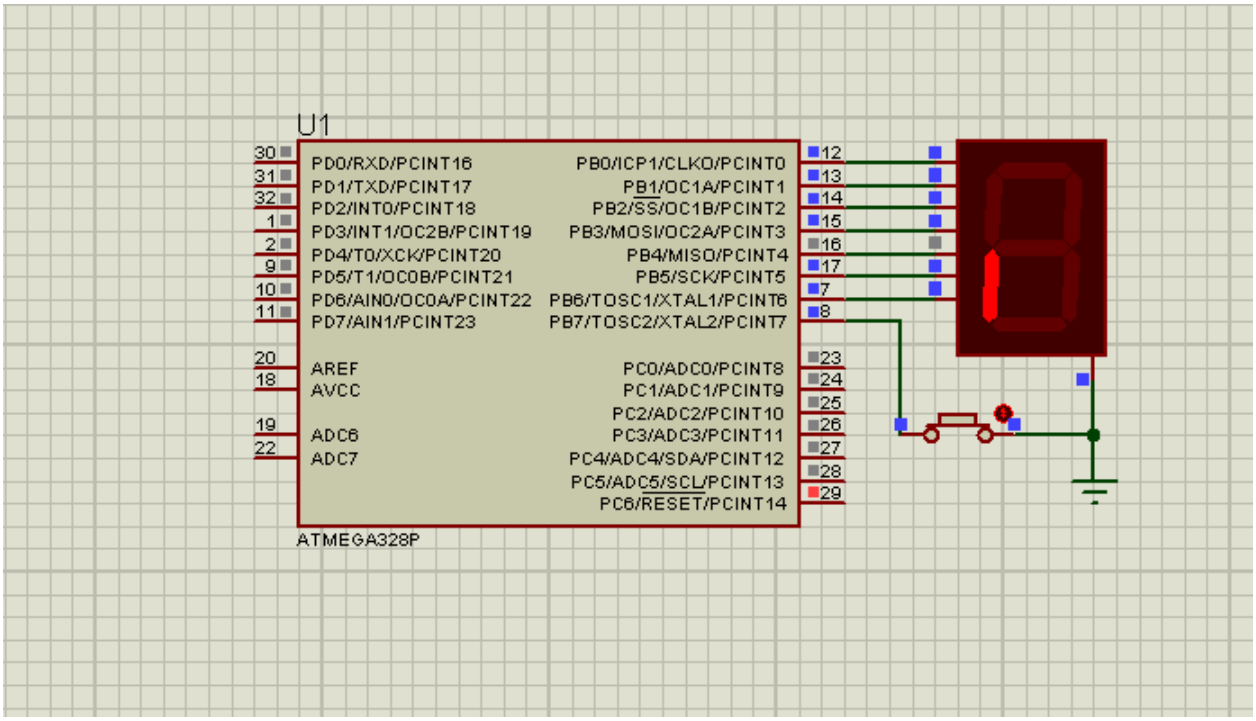
```
#define F_CPU 1000000UL
#include <avr/io.h>
#include <util/delay.h>

int main(void)
{
	DDRB = 0xFF & ~(1 << PINB7);
	PORTB |= (1 << PINB7);
	int button = 0;

	while(1)
	{
		for(int i = 0; i < 6; i++)
		{
			button = PINB & (1 << PINB7);
			if(button != 0)
			{
				PORTB = (1 << i);
			}
			else
			{
				PORTB = (0x20 >> i);
			}
			PORTB |= (1 << PINB7);
			_delay_ms(250);
		}
	}
}
```

Код на asm:

```
.def I = r17
.def TEMP = r18
.def SHIFT_I = r19



reset:
   rjmp setup
setup:
        ldi TEMP,0b01111111
        out DDRB,TEMP ; все на выход, 7 бит на вход
        sbi PORTB,7 ; подтягиваем резистор к 7 биту

loop:
        ldi I,0

        for_loop:
        ; пропуск если кнопка не нажата
                sbis PINB,7
                rjmp button_pressed

        ; не нажата
                ldi TEMP,1
        ;сдвиг
                mov SHIFT_I,I
                inc SHIFT_I
        shift_not_pressed:
                lsl TEMP
                dec SHIFT_I
                brne shift_not_pressed
```

```
            lsr TEMP
            rjmp outer

        ;нажата
        button_pressed:
        ;сдвиг
            ldi TEMP,0b01000000
            mov SHIFT_I,I
            inc SHIFT_I
        shift_pressed:
            lsr TEMP
            dec SHIFT_I
            brne shift_pressed

        outer:
            out PORTB,TEMP
            sbi PORTB,7 ; для кнопки

            ; задержка
                ldi     r30, 0x23     ; 35
                ldi     r31, 0xF4     ; 244
            delay:
                sbiw r30, 0x01        ; вычиаем из слова(r31:r30)
                brne delay
            inc I
            cpi I,6
            brlo for_loop ; I < 6 => переход
rjmp loop
```

Сборка в Proteus:



Прерывания

Код на C:

```
#define F_CPU 1000000UL


#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>



uint8_t segments[] =
{
	0b00111111, // 0 - A, B, C, D, E, F
	0b00000110, // 1 - B, C
	0b01011011, // 2 - A, B, D, E, G
	0b01001111, // 3 - A, B, C, D, G
	0b01100110, // 4 - B, C, F, G
	0b01101101, // 5 - A, C, D, F, G
```

```
        0b01111101, // 6 - A, C, D, E, F, G
        0b00000111, // 7 - A, B, C
        0b01111111, // 8 - A, B, C, D, E, F, G
        0b01101111, // 9 - A, B, C, D, F, G
};
volatile int is_button_pressed = 0;
volatile int counter = 0;

ISR(INT0_vect)
{
        if(is_button_pressed == 0)
        {
                is_button_pressed = 1;
        }
        else
        {
                is_button_pressed = 0;
                counter = 0;
        }
}

void setup()
{
        DDRB = 0xFF;
        PORTD |= (1 << PD2);
        EIMSK |= (1 << INT0);
        EICRA |= (1 << ISC01);
        sei();
}
```

```
int main(void)
{
	setup();
	while(1)
	{
		if(is_button_pressed == 0)
		{
			if(counter >= 10)
			{
				counter = 0;
			}
			PORTB = segments[counter++];
			_delay_ms(1000);
		}
	}
}
```

Код на asm:

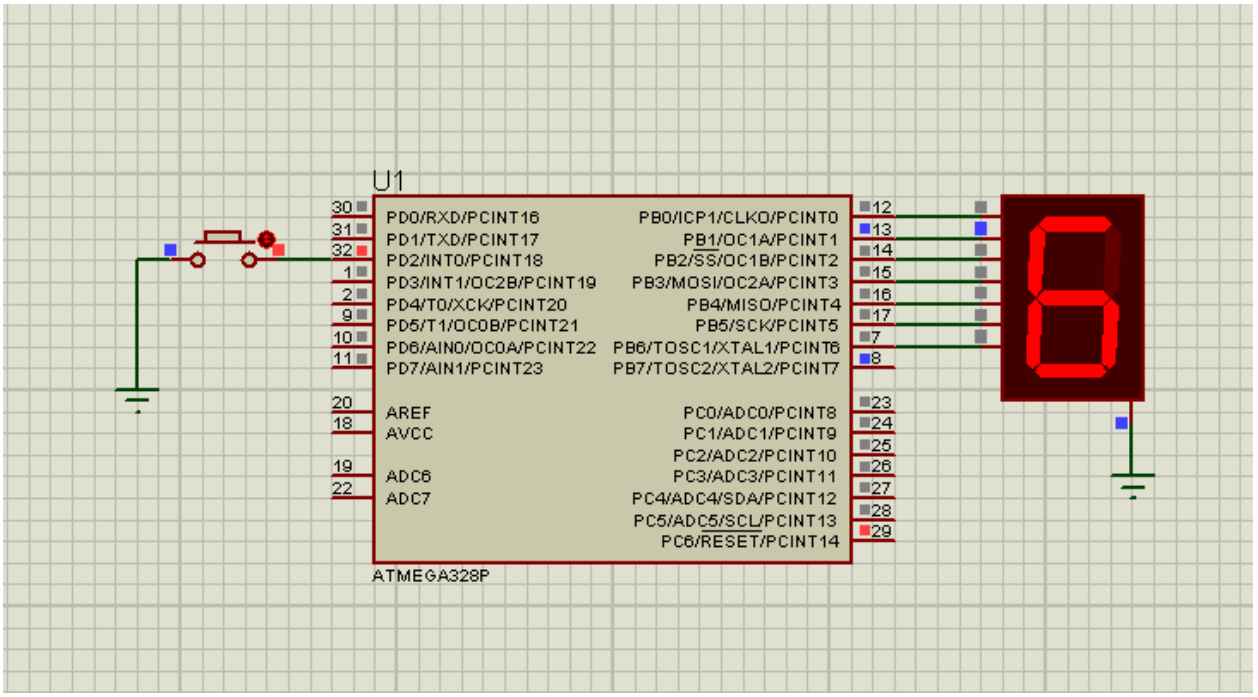
```
.equ ARR_SIZE = 10
.def IS_BUTTON_PRESSED = r16
.def COUNTER = r17
.def TEMP = r18
.def NUM_ITEMS = r19



.dseg

segments: .BYTE ARR_SIZE



.cseg
```

```
.org 0x0000
rjmp main

.org INT0addr
rjmp button_handler

button_handler:
        cpi IS_BUTTON_PRESSED,0
        brne first_press
        ldi IS_BUTTON_PRESSED,1
        rjmp handler_finish

    first_press:
    ; сбросился и остановился
        ldi COUNTER,0
        ldi IS_BUTTON_PRESSED,0

    handler_finish:
reti



reset:
    rjmp main
main:
    ldi ZH,High(segments_flash * 2)
    ldi ZL,Low(segments_flash * 2)
    ldi YH,High(segments)
    ldi YL,Low(segments)
    ldi NUM_ITEMS,ARR_SIZE
```

```
arr_copy:
        lpm TEMP,Z+
        st Y+,TEMP
        subi NUM_ITEMS,1
        brne arr_copy

; установка
        ldi TEMP,0xFF
        out DDRB,TEMP
        sbi PORTD,2
        ; найстройка прерываний
        sbi EIMSK,INT0
        lds r24,EICRA
        ori r24,0x02
        sts EICRA,r24
        sei
        ; настройка указателя
        ldi YH,High(segments)
        ldi YL,Low(segments)
        ldi TEMP,0

loop:
                cpi IS_BUTTON_PRESSED,0
                brne loop

                cpi COUNTER,10
                brlo outer
                ldi COUNTER,0

        outer:
```

```
        ; установка числа в регистр
                add YL,COUNTER
                adc YH,TEMP
                ; вывод
                ld r24,Y
                out PORTB,r24
                ; возвращение указателя обратно
                sub YL,COUNTER
                sbc YH,TEMP

                inc COUNTER

        ; задержка на 1 сек
                ldi     r18, 0x3F ; 63
                ldi     r24, 0x0D
                ldi     r25, 0x03
        delay_loop:
                subi    r18, 0x01
                sbci    r24, 0x00
                sbci    r25, 0x00
                brne    delay_loop
rjmp loop

.cseg
segments_flash:  .db  0b00111111,  0b00000110,  0b01011011,  0b01001111,
0b01100110, 0b01101101, 0b01111101, 0b00000111, 0b01111111, 0b01101111
```

Сборка в Proteus:



Секундомер

Код на C:

```
#define F_CPU 1000000UL

#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>



uint8_t segments[] =
{
	0b00111111, // 0 - A, B, C, D, E, F
	0b00000110, // 1 - B, C
	0b01011011, // 2 - A, B, D, E, G
	0b01001111, // 3 - A, B, C, D, G
	0b01100110, // 4 - B, C, F, G
	0b01101101, // 5 - A, C, D, F, G
```

```
        0b01111101, // 6 - A, C, D, E, F, G
        0b00000111, // 7 - A, B, C
        0b01111111, // 8 - A, B, C, D, E, F, G
        0b01101111, // 9 - A, B, C, D, F, G
};
volatile int counter = 0;



void reset_timer()
{
        PORTB = segments[0];
        PORTC = segments[0];
        PORTD &= ~(1 << PIND3);
}

ISR(INT0_vect)
{
        counter = 0;
        reset_timer();
}

void draw_stopwatch(int counter)
{
        PORTB = segments[counter / 10];
        PORTC = segments[counter % 10];
        PORTD = ((segments[counter % 10] & 0b01000000)  >> 3) | 0b00000100; //
маска для индикатора
}
```

```
void setup(void)
{
	DDRB = 0b01111111;
	DDRC = 0b11111111;
	DDRD = 0b00001000; // PIND3 & PIND2

	PORTD |= (1 << PIND2);

	EIMSK |= (1 << INT0);
	EICRA |= (1 << ISC01);
	sei();
}

int main(void)
{
	setup();

	while(1)
	{
		if(counter == 100)
		{
			counter = 0;
		}

		draw_stopwatch(counter);
		_delay_ms(1000);
		counter++;
	}
}
```

Код на asm:

```
.equ ARR_SIZE = 10
.def COUNTER = r17
.def TEMP = r18
.def NUM_ITEMS = r19
.def ADAPTED_COUNTER = r20

.dseg
segments: .BYTE ARR_SIZE

.cseg
.org 0x0000
rjmp main

.org INT0addr
rjmp button_handler

button_handler:
        ldi COUNTER,0
        out PORTB,TEMP
        out PORTC,TEMP
        cbi PORTD,3
reti



reset:
        rjmp main
main:
        ldi ZH,High(segments_flash * 2)
        ldi ZL,Low(segments_flash * 2)
```

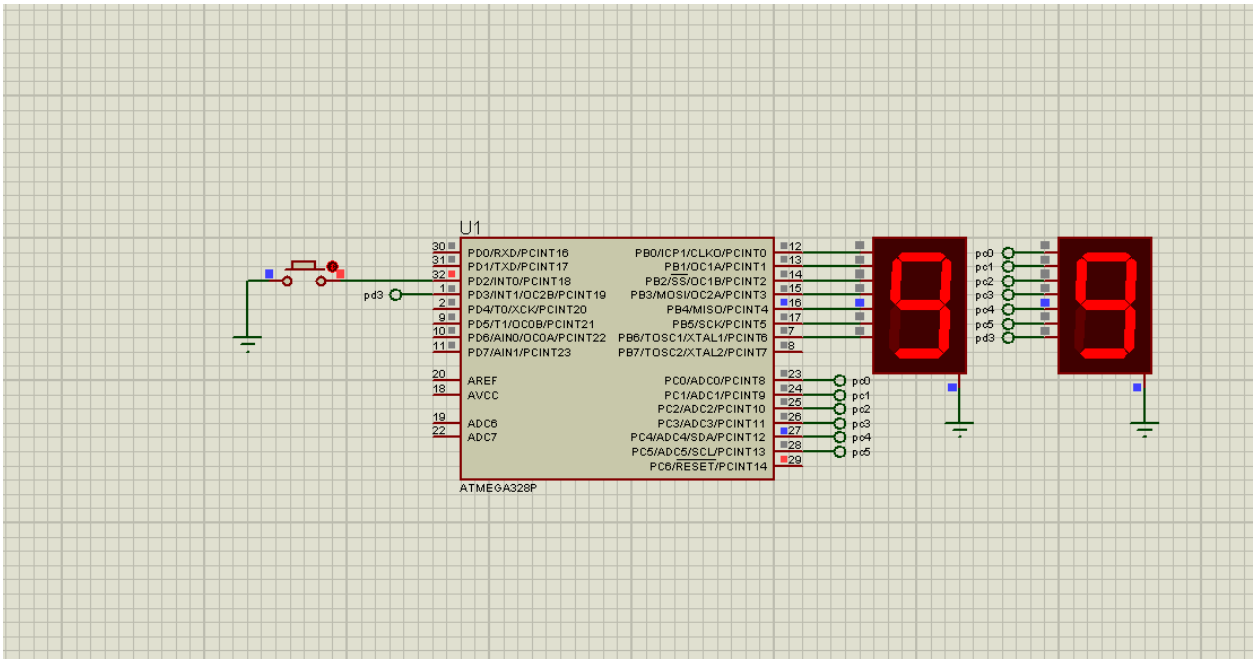
```
        ldi YH,High(segments)
        ldi YL,Low(segments)
        ldi NUM_ITEMS,ARR_SIZE
arr_copy:
        lpm TEMP,Z+
        st Y+,TEMP
        subi NUM_ITEMS,1
        brne arr_copy

; установка
        ldi TEMP,0b01111111
        out DDRB,TEMP
        out DDRC,TEMP
        cbi DDRD,2
        sbi DDRD,3
        sbi PORTD,2
        ; найстройка прерываний
        sbi EIMSK,INT0
        lds r24,EICRA
        ori r24,0x02
        sts EICRA,r24
        sei
        ; настройка указателя
        ldi YH,High(segments)
        ldi YL,Low(segments)
        ldi TEMP,0
loop:

                cpi COUNTER,100
                brlo outer
```

```
        ldi COUNTER,0

    outer:

        rcall print_dozens
        rcall print_digits

        inc COUNTER

    ; задержка на 1 сек
        ldi     r23, 0x3F ; 63
        ldi     r24, 0x0D
        ldi     r25, 0x03
    delay_loop:
        subi    r23, 0x01
        sbci    r24, 0x00
        sbci    r25, 0x00
        brne    delay_loop
rjmp loop

.cseg
segments_flash: .db 0b00111111, 0b00000110, 0b01011011, 0b01001111,
0b01100110, 0b01101101, 0b01111101, 0b00000111, 0b01111111, 0b01101111


find_dozens:
        mov ADAPTED_COUNTER,COUNTER
        clr r22
    dozens_loop:
        cpi ADAPTED_COUNTER,10
```

```
            brlo find_dozens_finish

            subi ADAPTED_COUNTER,10
            inc r22
        rjmp dozens_loop

        find_dozens_finish:
        mov ADAPTED_COUNTER,r22
ret


find_digits:
            mov ADAPTED_COUNTER,COUNTER
        digits_loop:
            cpi ADAPTED_COUNTER,10
            brlo find_digits_finish ; переход если < 10
            subi ADAPTED_COUNTER,10
        rjmp digits_loop

        find_digits_finish:
ret

print_7_bit:
        andi r24,0b01000000
        ; сдвигаю под пин d3
        lsr r24
        lsr r24
        lsr r24
        ; для кнопки
        ori r24,0b00000100
```

```
        out PORTD,r24
ret


print_dozens:
        ; установка десятков
                rcall find_dozens
                add YL,ADAPTED_COUNTER
                adc YH,TEMP
                ; получили сегмент
                ld r24,Y
                out PORTB,r24
                ; возвращение указателя обратно
                sub YL,ADAPTED_COUNTER
                sbc YH,TEMP
ret

print_digits:
        ; установка единиц
                rcall find_digits
                add YL,ADAPTED_COUNTER
                adc YH,TEMP
                ; получили сегмент
                ld r24,Y
                out PORTC,r24
                rcall print_7_bit
                ; возвращение указателя обратно
                sub YL,ADAPTED_COUNTER
                sbc YH,TEMP
ret
```

Сборка в Proteus:



Гирлянда

Код на C:

```
#define F_CPU 1000000UL

#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>

#include <stdint.h>



/**
Режимы работы:
0. Накопление слева направо
1. Накопление в центр
2. Уменьшение налево
*/
```

```
volatile uint8_t state = 0;
volatile uint8_t i = 0;
volatile uint8_t off_lights_counter = 18;

volatile uint32_t garland_value = 0;



void reset_garland()
{
	garland_value = 0;
	PORTB &= 0b11000000;
	PORTC &= 0b11000000;
	PORTD &= 0b11000100;
}
void fill_garland()
{
	garland_value = 0xFFFFF;
	PORTB |= 0b00111111;
	PORTC |= 0b00111111;
	PORTD |= 0b01111111;
}
void draw_garland()
{
	PORTB = garland_value & 0b00111111;
	PORTC = (garland_value >> 6) & 0b00111111;

	int temp = (garland_value >> 12) & 0b00111111;
	PORTD = ((temp & 0b11111100) << 1) | (1 << PIND2) | (temp & 0b00000011);
```

```
}

void generate_accumulation_to_right()
{
	garland_value &= ~(1ul << (i - 1));	// убираю предыдущий
	garland_value |= (1ul << i);		// рисую текущий

	i++;
	if(off_lights_counter == 0)
	{
		i = 0;
		off_lights_counter = 18;
		reset_garland();
	}
	if(i == off_lights_counter)
	{
		i = 0;
		off_lights_counter--;
	}
}

void generate_accumulation_to_center()
{
	//
	// правая сторона
	//
	garland_value &= ~(0x20000 >> (i - 1)); // 1 на 17 бите
	garland_value |= (0x20000 >> i);
	//
	// левая сторона
```

```
        //
        garland_value &= ~(1ul << (i - 1));
        garland_value |= (1ul << i);


        i++;
        if (off_lights_counter == 0)
        {
                i = 0;
                off_lights_counter = 18;
                reset_garland();
        }
        if (i == off_lights_counter / 2)
        {
                i = 0;
                off_lights_counter -= 2;
        }
}

void generate_decreasing_to_left()
{
        garland_value |= (0x20000 >> (i - 1));
        garland_value &= ~(0x20000 >> i); // 0b10000000

        i++;
        if(off_lights_counter == 0)
        {
                i = 0;
                off_lights_counter = 18;
                fill_garland();
```

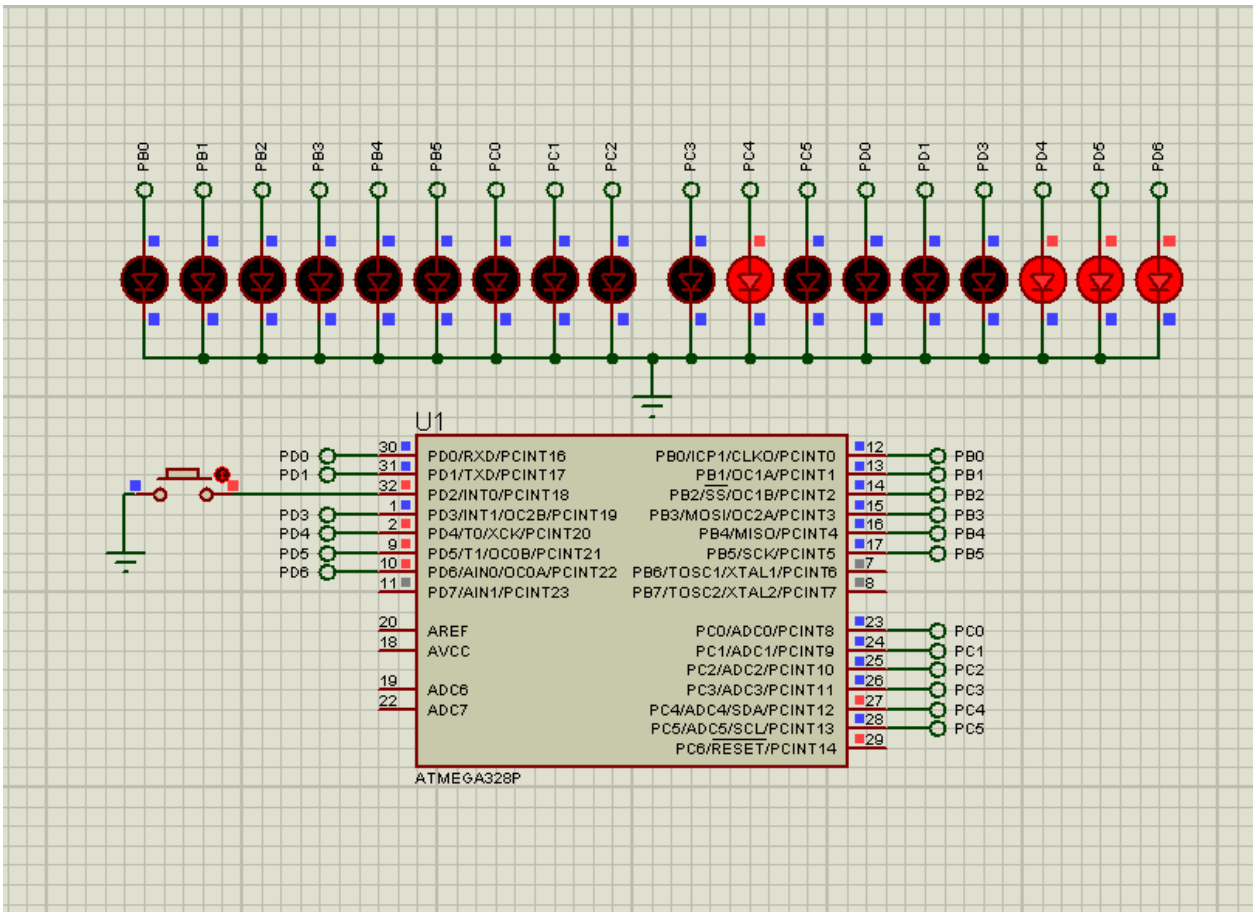
```
        }
        if(i == off_lights_counter)
        {
                i = 0;
                off_lights_counter--;
        }
}

ISR(INT0_vect)
{
        i = 0;
        off_lights_counter = 18;
        reset_garland();



        state++;
        if(state == 3)
                state = 0;
        if (state == 2) fill_garland();
}

void setup(void)
{
        DDRB = 0b00111111;     // все 6
        DDRC = 0b00111111;     // + 6
        DDRD = 0b01111011;     // PIND2 на прерывание по изменению состояения и ещё 6 на гирлянду

        PORTD |= (1 << PIND2);
```

```
    EIMSK |= (1 << INT0);
    EICRA |= (1 << ISC01);
    sei();
}

int main(void)
{
    setup();

    while (1)
    {
        switch (state)
        {
            case 0:
                generate_accumulation_to_right();
                break;
            case 1:
                generate_accumulation_to_center();
                break;
            case 2:
                generate_decreasing_to_left();
        }
        draw_garland();
        _delay_ms(200);
    }
}
```

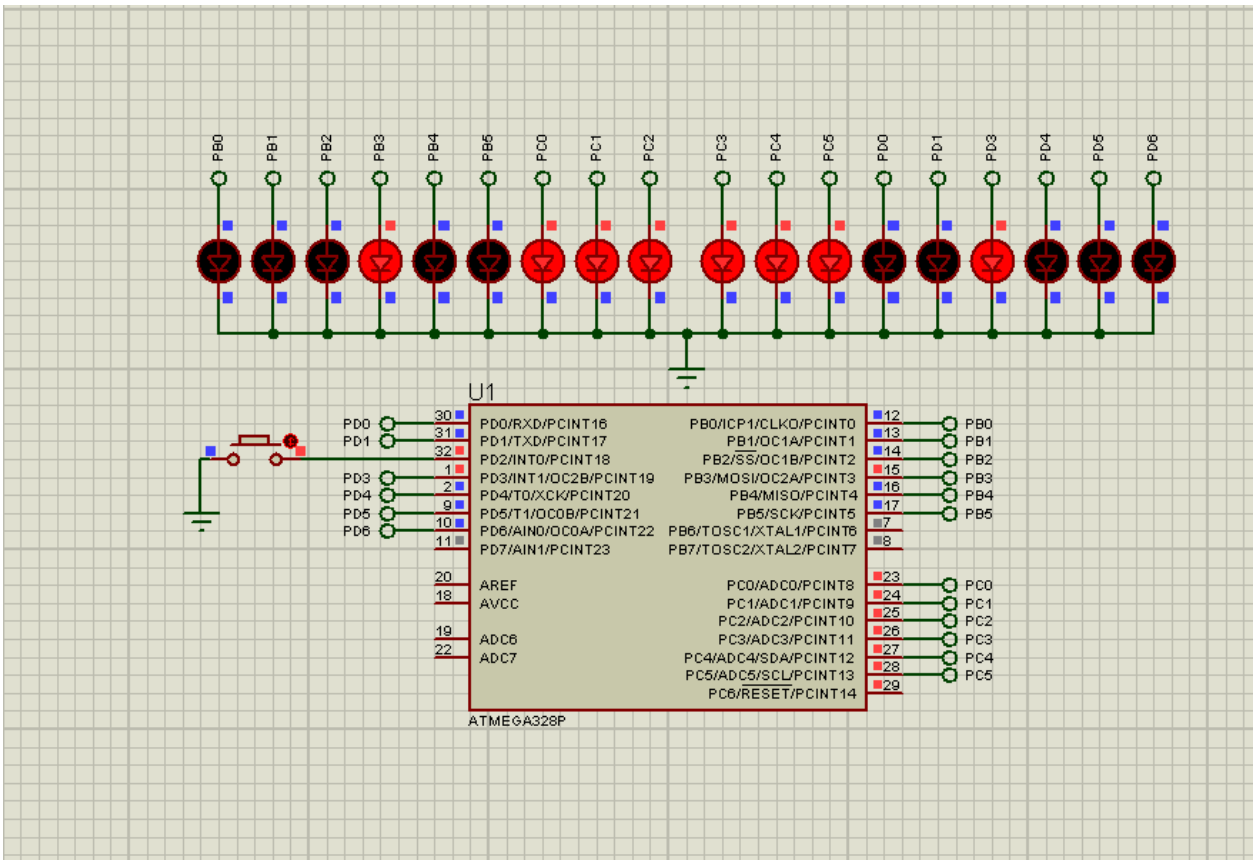
Сборка в Proteus:

Ниже представлены все режимы работы

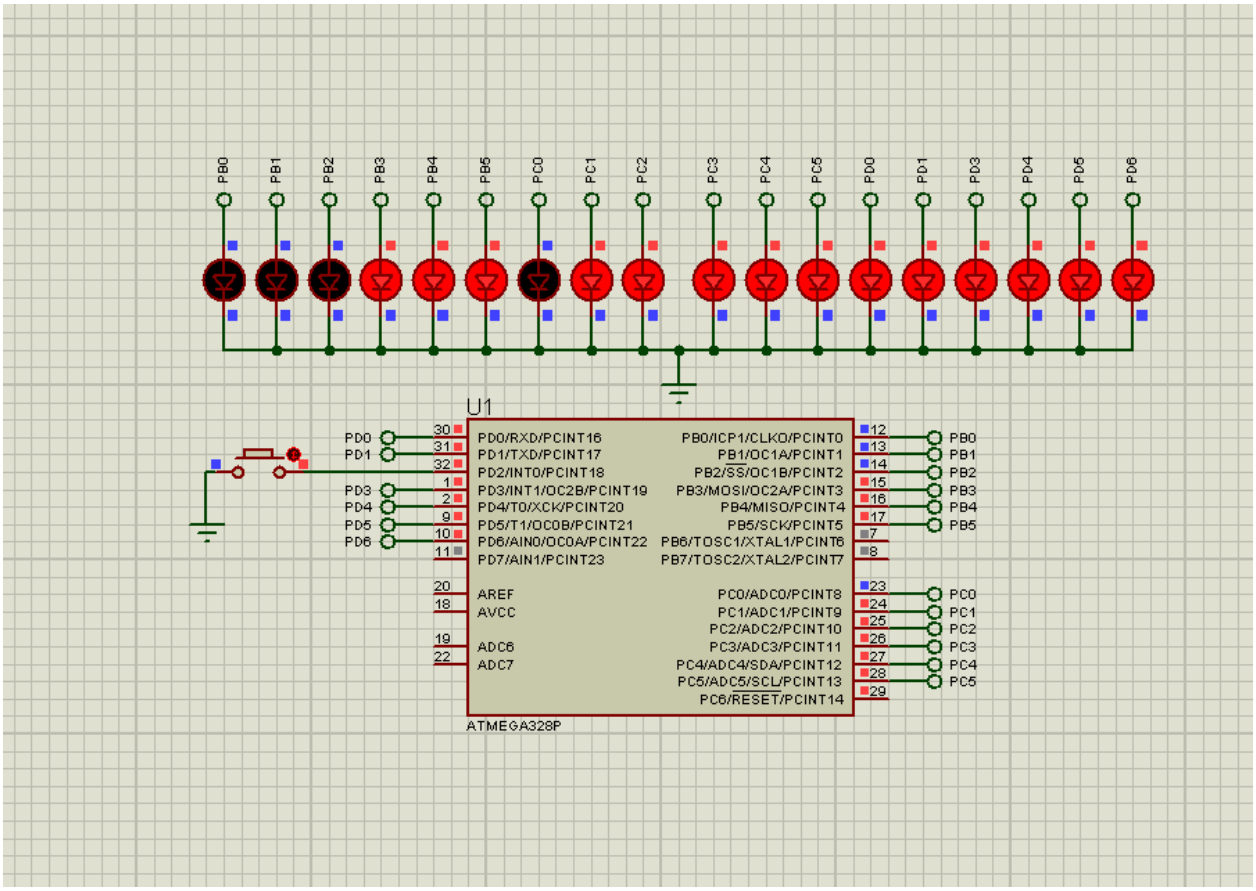
Накопление с правой стороны



Накопление в центре



Уменьшение с левой стороны


PB0
PB1
PB2
PB3
PB4
PB5
PC0
PC1
PC2
PC3
PC4
PC5
PD0
PD1
PD3
PD4
PD5
PD6
U1
PD0
PD1
PD3
PD4
PD5
PD6
30
31
32
1
2
9
10
11
PD0/RXD/PCINT16
PD1/TXD/PCINT17
PD2/INT0/PCINT18
PD3/INT1/OC2B/PCINT19
PD4/T0/XCK/PCINT20
PD5/T1/OC0B/PCINT21
PD6/AIN0/OC0A/PCINT22
PD7/AIN1/PCINT23
20
18
19
22
AREF
AVCC
ADC6
ADC7
PB0/ICP1/CLKO/PCINT0
PB1/OC1A/PCINT1
PB2/SS/OC1B/PCINT2
PB3/MOSI/OC2A/PCINT3
PB4/MISO/PCINT4
PB5/SCK/PCINT5
PB6/TOSC1/XTAL1/PCINT6
PB7/TOSC2/XTAL2/PCINT7
PC0/ADC0/PCINT8
PC1/ADC1/PCINT9
PC2/ADC2/PCINT10
PC3/ADC3/PCINT11
PC4/ADC4/SDA/PCINT12
PC5/ADC5/SCL/PCINT13
PC6/RESET/PCINT14
12
13
14
15
16
17
7
8
23
24
25
26
27
28
29
PB0
PB1
PB2
PB3
PB4
PB5
PC0
PC1
PC2
PC3
PC4
PC5
ATMEGA328P

Выводы: в результате выполнения данной лабораторной работы я научился писать программы по взаимодействию с устройствами ввода/вывода, запускать их на локальном схематичном окружении и применять полученные знания на практике.