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

Факультет информатики и
вычислительной техники

Кафедра ИСП

Отчет

по лабораторной работе № 6

по дисциплине «Машинно-зависимые языки программирования»

Вариант 1

Выполнил: студент группы ПС-11
Щеглов Г.С

Проверил: Баев А.А.

г. Йошкар-Ола
2024

Цель работы: научиться работать с таймерами и интерфейсом SPI.

Задания на лабораторную работу:

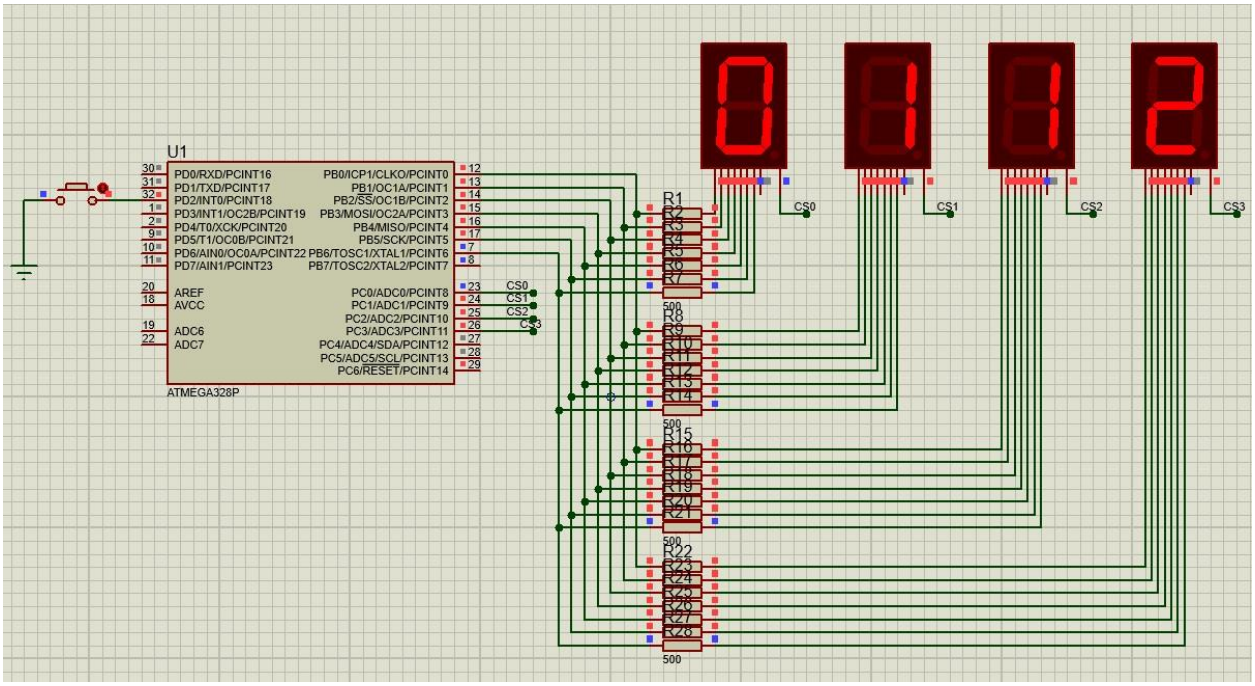
1. Реализовать динамическую индикацию
2. Реализовать подключение индикаторов с помощью регистров
3. Реализовать передачу данных на регистры при помощи SPI

1. Теоретические сведения

Учебное пособие - ПРИМЕНЕНИЕ МИКРОКОНТРОЛЛЕРОВ В РАДИОТЕХНИЧЕСКИХ И БИОМЕДИЦИНСКИХ СИСТЕМАХ

2. Практическая часть

Схема динамической индикации:



Код на С:

```
#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
};

void InitPorts(void);
void Send_Data(uint8_t data, uint8_t ind);
void InitTimer0(void);
void Bin2Dec(uint16_t data);

volatile uint16_t cnt = 0;
volatile uint8_t switch_state = 0;
```

```
volatile uint8_t bcd_buffer[] = {0, 0, 0, 0};

int main(void)
{
	InitPorts();
	InitTimer0();
	EIMSK |= (1 << INT0);  // Включить INT0
	EICRA |= (1 << ISC01); // Настройка INT0 на прерывание по спаду
	sei();                 // Глобальное разрешение прерываний

	while (1)
	{
		if (switch_state == 0)
		{
			Bin2Dec(cnt);
			if (cnt < 9999)
			{
				cnt++;
			}
			else
			{
				cnt = 0;
			}
			_delay_ms(100);
		}
	}
}

//-------------------------------------------
ISR(TIMER0_COMPA_vect)
{
	Send_Data(bcd_buffer[3], 0);
	Send_Data(bcd_buffer[2], 1);
	Send_Data(bcd_buffer[1], 2);
	Send_Data(bcd_buffer[0], 3);
}

ISR(INT0_vect)
{
	if (switch_state == 0)
	{
		switch_state = 1;
	}
	else
	{
		switch_state = 0;
		cnt = 0;
	}
}
```

```
void InitPorts(void)
{
	DDRB = 0xFF;
	DDRC = (1 << PC0 | 1 << PC1 | 1 << PC2 | 1 << PC3);
	PORTC = 0x0F;
	DDRD = (0 << PD2);
	PORTD |= (1 << PD2);
}

void Send_Data(uint8_t data, uint8_t ind)
{
	PORTC = 0x0F & ~(1 << ind);
	PORTB = segments[data];
	_delay_ms(5);
	PORTB = 0;
	PORTC = 0x0F;
}

void InitTimer0(void)
{
	TCCR0A = (1 << WGM01); // режим CTC - Clear Timer on Compare
	TCCR0B = (1 << CS02 | 1 << CS00); // prescaler = sys_clk / 1024
	TCNT0 = 0x00; // начальное значение счетчика
	OCR0A = 16; // порог срабатывания
	TIMSK0 |= (1 << OCIE0A); // включение прерывания при достижении
порога A
}

void Bin2Dec(uint16_t data)
{
	bcd_buffer[3] = (uint8_t)(data / 1000);
	data = data - bcd_buffer[3] * 1000;
	bcd_buffer[2] = (uint8_t)(data / 100);
	data = data - bcd_buffer[2] * 100;
	bcd_buffer[1] = (uint8_t)(data / 10);
	data = data - bcd_buffer[1] * 10;
	bcd_buffer[0] = (uint8_t)(data);
}
```

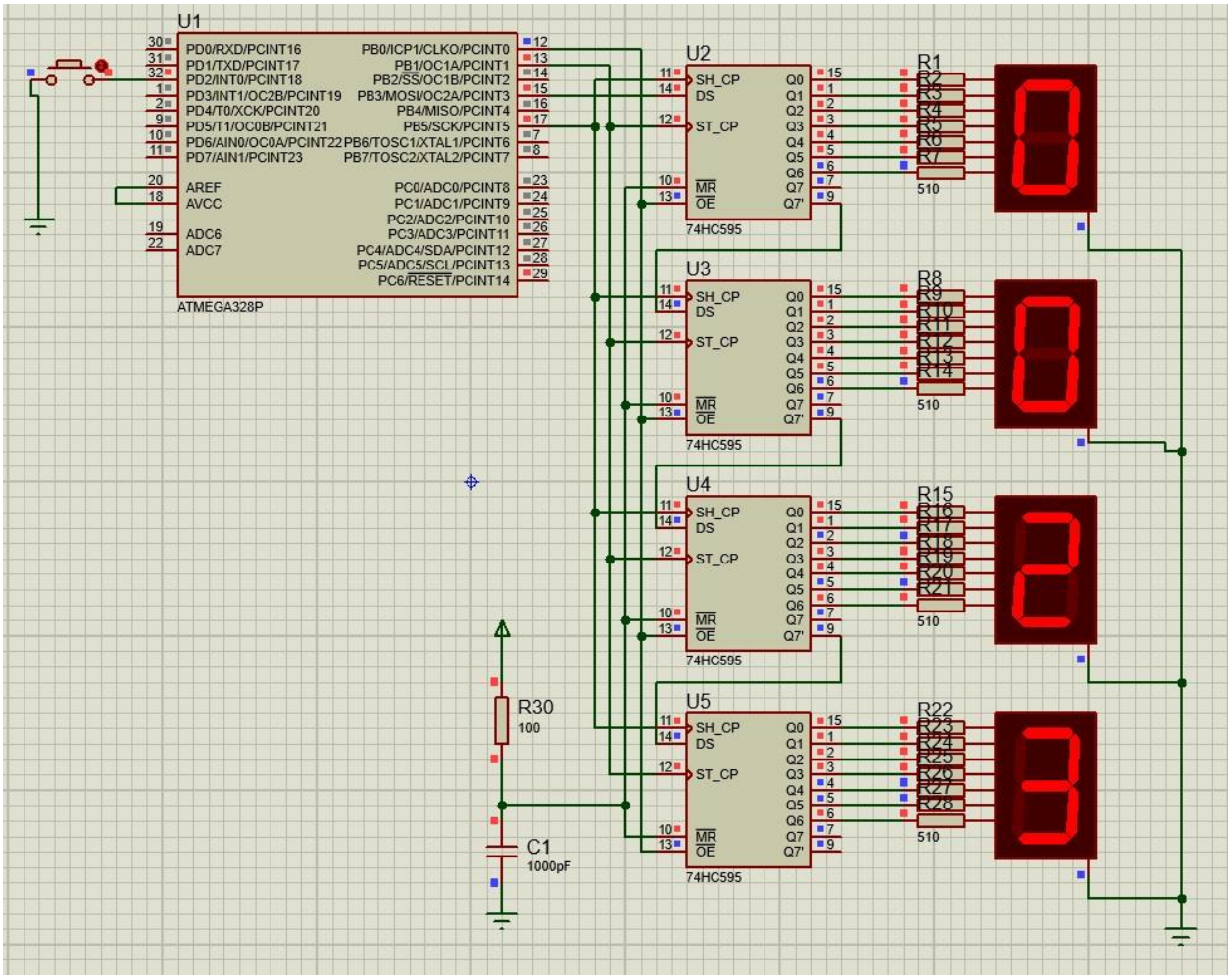
Оптимизированный код:

```
#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
};

void InitPorts(void);
void Send_Data(uint8_t data, uint8_t ind);
void InitTimer0(void);
void Bin2Dec(uint16_t data);

volatile uint16_t cnt = 0;
volatile uint8_t switch_state = 0;
volatile uint8_t bcd_buffer[] = {0, 0, 0, 0};

int main(void) {
      InitPorts();
      InitTimer0();
      EIMSK |= (1 << INT0);
      EICRA |= (1 << ISC01);
      sei();

      while (1) {
            if (!switch_state) {
                  Bin2Dec(cnt);
                  cnt = (cnt < 9999) ? cnt + 1 : 0;
                  _delay_ms(100);
            }
      }
}

ISR(TIMER0_COMPA_vect) {
      Send_Data(bcd_buffer[3 - (PORTC & 0x03)], (PORTC & 0x03));
      PORTC = (PORTC & 0xFC) | ((PORTC + 1) & 0x03);
}

ISR(INT0_vect) {
      switch_state = !switch_state;
      if (!switch_state) cnt = 0;
}
void InitPorts(void) {
      DDRB = 0xFF;
      DDRC = 0x0F;
      PORTC = 0x0F;
      DDRD &= ~(1 << PD2);
      PORTD |= (1 << PD2);
}
```

```
void Send_Data(uint8_t data, uint8_t ind) {
	PORTC = 0x0F & ~(1 << ind);
	PORTB = segments[data];
	PORTB = 0;
	PORTC = 0x0F;
}

void InitTimer0(void) {
	TCCR0A = (1 << WGM01);
	TCCR0B = (1 << CS02) | (1 << CS00);
	TCNT0 = 0x00;
	OCR0A = 16;
	TIMSK0 |= (1 << OCIE0A);
}

void Bin2Dec(uint16_t data) {
	bcd_buffer[0] = data % 10;  // Последняя цифра
	bcd_buffer[1] = (data / 10) % 10;  // Вторая цифра
	bcd_buffer[2] = (data / 100) % 10;  // Третья цифра
	bcd_buffer[3] = (data / 1000) % 10;  // Первая цифра
}
```

Схема подключения индикаторов с помощью регистров:



Код на С:

```
#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
};
void InitPorts(void);
```

```
void send_data(uint8_t data, uint8_t ind);
void InitTimer0(void);
void Bin2Dec(uint16_t data);
void InitTimer1(void);
void StartTimer1(void);
void StopTimer1(void);
void SendData(uint8_t data);
void DisplayData(uint16_t data);
volatile uint16_t cnt = 0;
volatile uint8_t switch_state = 0;
volatile uint8_t bcd_buffer[] = {0,0,0,0};
int main(void)
{
	InitPorts();
	InitTimer1();
	EIMSK |= (1<<INT0); //разрешить прерывание  INT0
	EICRA |= (1<<ISC01);//Запуск по заднему фронту  INT0
	sei();
	//Разрешение прерываний
	PORTB &= ~(1<<PB0); //OE = low (active)
	DisplayData(0);
	while(1)
	{ }
}
//-------------------------------------------
ISR(TIMER1_COMPA_vect){
	DisplayData(cnt);
	if(cnt<9999){
		cnt++;
		}else{
		cnt=0;
	}
}
ISR(INT0_vect){
	if(switch_state == 0){
		switch_state = 1;
		StartTimer1();
		}else{
		StopTimer1();
		DisplayData(cnt);



		switch_state = 0;
		cnt = 0;
	}
}
//-------------------------------------------
void InitPorts(void){
```

```
	DDRB = (1<<PB0|1<<PB1|1<<PB3|1<<PB5);
	DDRD &= ~(1<<PD2);
	PORTD |= (1<<PD2);
}
void InitTimer1(void){
	TCCR1A = 0;
	TCCR1B = (1<<CS11|1<<CS10|1<<WGM12);
	TCNT1 = 0;
	OCR1A = 15624;
}
void StartTimer1(void){
	TCNT1 = 0;
	TIMSK1 |= (1<<OCIE1A);
}
void StopTimer1(void){
	TIMSK1 &= ~(1<<OCIE1A);
}
void Bin2Dec(uint16_t data){
	bcd_buffer[3] = (uint8_t)(data/1000);
	data = data % 1000;
	bcd_buffer[2] = (uint8_t)(data/100);
	data = data % 100;
	bcd_buffer[1] = (uint8_t)(data/10);
	data = data % 10;
	bcd_buffer[0] = (uint8_t)(data);
}
void SendData(uint8_t data){
	for(uint8_t i=0; i<8; i++){
		PORTB &= ~(1<<PB5);
		//CLK low
		if(0x80 & (data<<i)){
			PORTB |= 1<<PB3; //DAT high
			} else {
			PORTB &= ~(1<<PB3);
			//DAT low
		}
		PORTB |= (1<<PB5);
	}
	//CLK high
}
void DisplayData(uint16_t data){
	Bin2Dec(data);
	PORTB &= ~(1<<PB1);
	//clk_out = 0
	SendData(segments[bcd_buffer[0]]);
	SendData(segments[bcd_buffer[1]]);
	SendData(segments[bcd_buffer[2]]);
	SendData(segments[bcd_buffer[3]]);
	PORTB |= (1<<PB1);
	//clk_out = 1
```

```
}
```

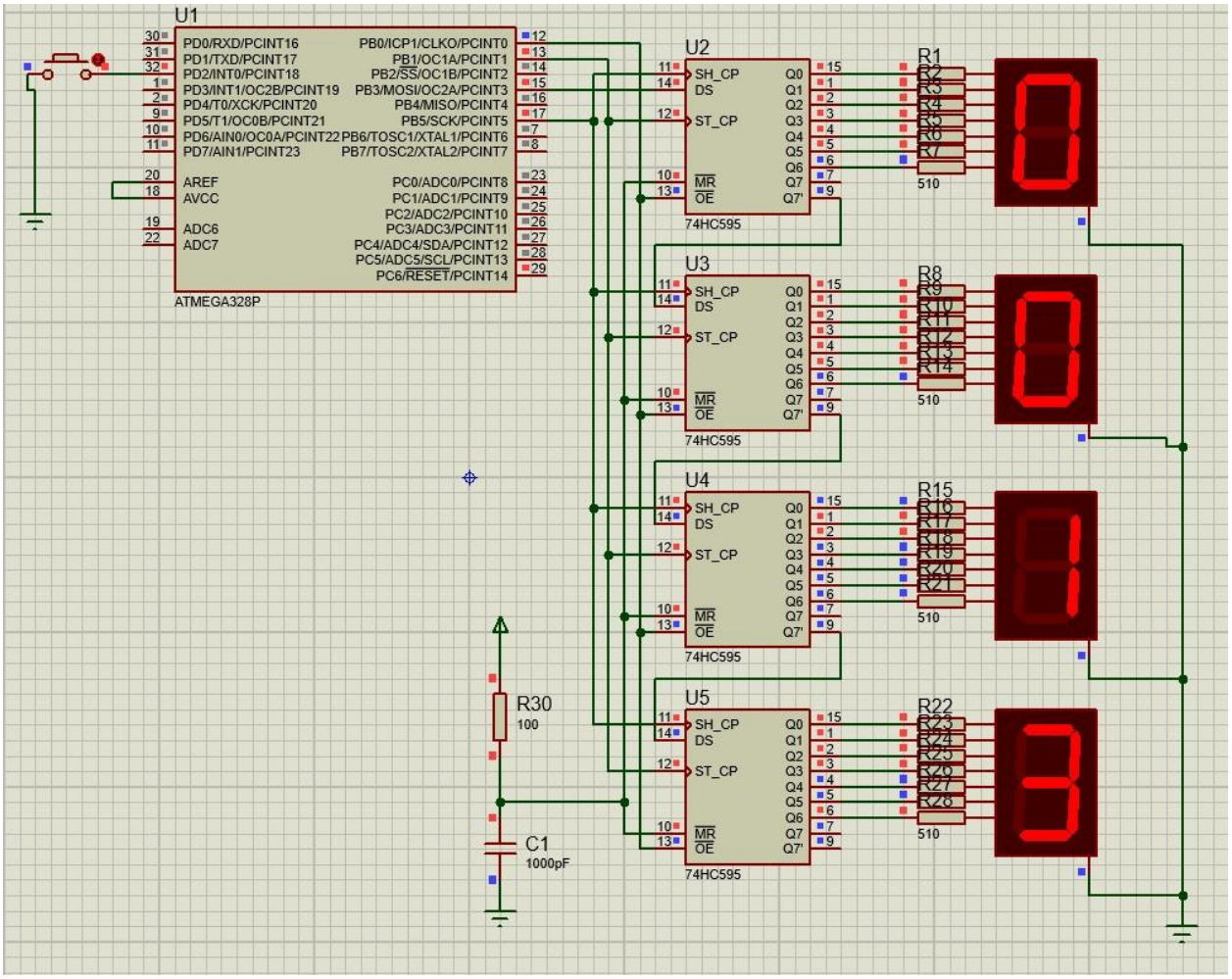
Оптимизированный код:

```
#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 uint16_t cnt = 0;
volatile uint8_t switch_state = 0;
volatile uint8_t bcd_buffer[] = {0, 0, 0, 0};

void InitPorts(void);
void SendData(uint8_t data);
void DisplayData(uint16_t data);
void InitTimer1(void);
void StartTimer1(void);
void StopTimer1(void);
void Bin2Dec(uint16_t data);

int main(void) {
      InitPorts();
      InitTimer1();
      EIMSK |= (1 << INT0);  // Разрешить прерывание INT0
      EICRA |= (1 << ISC01); // Запуск по заднему фронту INT0
      sei();                 // Разрешение прерываний
      PORTB &= ~(1 << PB0);  // OE = low (active)
      DisplayData(0);

      while (1) {}
}

//-----------------------------------------
ISR(TIMER1_COMPA_vect) {
```

```
	DisplayData(cnt);
	cnt = (cnt < 9999) ? cnt + 1 : 0;
}


ISR(INT0_vect) {
	switch_state ^= 1;

	if (switch_state) {
		StartTimer1();
		} else {
		StopTimer1();
		DisplayData(cnt);
		cnt = 0;
	}
}
//-------------------------------------------
void InitPorts(void) {
	DDRB = (1 << PB0) | (1 << PB1) | (1 << PB3) | (1 << PB5);
	DDRD &= ~(1 << PD2);
	PORTD |= (1 << PD2);
}

void InitTimer1(void) {
	TCCR1A = 0;
	TCCR1B = (1 << CS11) | (1 << CS10) | (1 << WGM12);
	TCNT1 = 0;
	OCR1A = 15624;
}

void StartTimer1(void) {
	TCNT1 = 0;
	TIMSK1 |= (1 << OCIE1A);
}

void StopTimer1(void) {
	TIMSK1 &= ~(1 << OCIE1A);
}

void Bin2Dec(uint16_t data) {
	bcd_buffer[3] = data / 1000;
	bcd_buffer[2] = (data / 100) % 10;
	bcd_buffer[1] = (data / 10) % 10;
	bcd_buffer[0] = data % 10;
}
void SendData(uint8_t data) {
	for (uint8_t i = 0; i < 8; i++) {
		PORTB &= ~(1 << PB5); // CLK low
		if (data & (1 << (7 - i))) {
			PORTB |= (1 << PB3); // DAT high
```

```
                } else {
                PORTB &= ~(1 << PB3); // DAT low
            }
            PORTB |= (1 << PB5); // CLK high
        }
}

void DisplayData(uint16_t data) {
      Bin2Dec(data);
      PORTB &= ~(1 << PB1); // clk_out = 0
      for (uint8_t i = 0; i < 4; i++) {
            SendData(segments[bcd_buffer[i]]);
      }
      PORTB |= (1 << PB1); // clk_out = 1
}
```

Схема передачи данных на регистры по SPI:



Код на С:

```
#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
};
void InitPorts(void);
```

```
void send_data(uint8_t data, uint8_t ind);
void InitTimer0(void);
void Bin2Dec(uint16_t data);
void InitTimer1(void);
void StartTimer1(void);
void StopTimer1(void);
void SendData(uint8_t data);
void DisplayData(uint16_t data);
volatile uint16_t cnt = 0;
volatile uint8_t switch_state = 0;
volatile uint8_t bcd_buffer[] = {0,0,0,0};
int main(void)
{
	InitPorts();
	InitTimer1();
	EIMSK |= (1<<INT0); //разрешить прерывание INT0
	EICRA |= (1<<ISC01);//Запуск по заднему фронту INT0
	sei();
	//Разрешение прерываний
	PORTB &= ~(1<<PB0); //OE = low (active)
	DisplayData(0);
	while(1)
	{ }
}
//-------------------------------------------
ISR(TIMER1_COMPA_vect){
	DisplayData(cnt);
	if(cnt<9999){
		cnt++;
		}else{
		cnt=0;
	}
}
ISR(INT0_vect){
	if(switch_state == 0){
		switch_state = 1;
		StartTimer1();
		}else{
		StopTimer1();
		DisplayData(cnt);
		switch_state = 0;
		cnt = 0;
	}
}
//-------------------------------------------
void InitPorts(void){
	DDRB = (1<<PB0|1<<PB1|1<<PB3|1<<PB5);
	DDRD &= ~(1<<PD2);
	PORTD |= (1<<PD2);
}
```

```
void InitTimer1(void){
	TCCR1A = 0;
	TCCR1B = (1<<CS11|1<<CS10|1<<WGM12);
	TCNT1 = 0;
	OCR1A = 15624;
}
void StartTimer1(void){
	TCNT1 = 0;
	TIMSK1 |= (1<<OCIE1A);
}
void StopTimer1(void){
	TIMSK1 &= ~(1<<OCIE1A);
}
void Bin2Dec(uint16_t data){
	bcd_buffer[3] = (uint8_t)(data/1000);
	data = data % 1000;
	bcd_buffer[2] = (uint8_t)(data/100);
	data = data % 100;
	bcd_buffer[1] = (uint8_t)(data/10);
	data = data % 10;
	bcd_buffer[0] = (uint8_t)(data);
}
void SendData(uint8_t data){
	for(uint8_t i=0; i<8; i++){
		PORTB &= ~(1<<PB5);
		//CLK low
		if(0x80 & (data<<i)){
			PORTB |= 1<<PB3; //DAT high
			} else {
			PORTB &= ~(1<<PB3);
			//DAT low
		}
		PORTB |= (1<<PB5);
	}
	//CLK high
}
void DisplayData(uint16_t data){
	Bin2Dec(data);
	PORTB &= ~(1<<PB1);
	//clk_out = 0
	SendData(segments[bcd_buffer[0]]);
	SendData(segments[bcd_buffer[1]]);
	SendData(segments[bcd_buffer[2]]);
	SendData(segments[bcd_buffer[3]]);
	PORTB |= (1<<PB1);
	//clk_out = 1
}
void InitSPI(void){
	DDRB |= (1<<PB3 | 1<<PB5);//настроить MOSI и CLK как выходы
	SPSR |= (1<<SPI2X);  //Fclk = Fosc/2
```

```
      SPCR = (1<<SPE | 1<<MSTR); //SPI включен, мастер,
      //MSB первый, CPOL=0, CPHA=0
      PORTB &= ~(1<<PB3 | 1<<PB5);
      //инициализация: DAT=0, CLK=0
}
void SPI_send (uint8_t data){
      SPDR = data;
      while(!(SPSR & (1<<SPIF)));
}
```

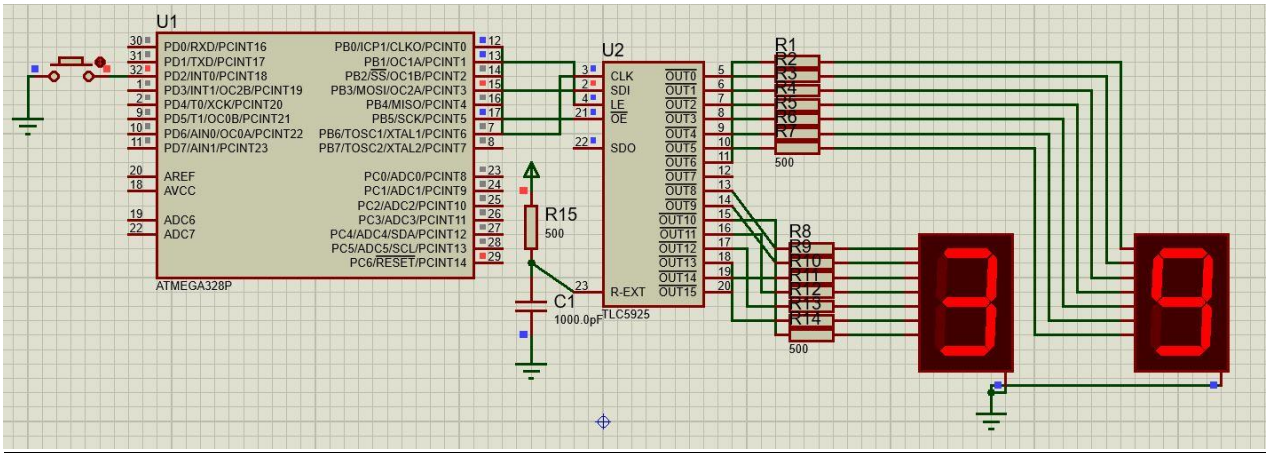
Оптимизированный код:

```
#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 uint16_t cnt = 0;
volatile uint8_t switch_state = 0;
volatile uint8_t bcd_buffer[] = {0, 0, 0, 0};

void InitPorts(void);
void SendData(uint8_t data);
void DisplayData(uint16_t data);
void InitTimer1(void);
void StartTimer1(void);
void StopTimer1(void);
void Bin2Dec(uint16_t data);

int main(void) {
      InitPorts();
      InitTimer1();
      EIMSK |= (1 << INT0);  // Разрешить прерывание INT0
      EICRA |= (1 << ISC01); // Запуск по заднему фронту INT0
      sei();                 // Разрешение прерываний
      PORTB &= ~(1 << PB0);  // OE = low (active)
```

```
	DisplayData(0);

	while (1) {}
}

//-------------------------------------------
ISR(TIMER1_COMPA_vect) {
	DisplayData(cnt);
	cnt = (cnt < 9999) ? cnt + 1 : 0;
}


ISR(INT0_vect) {
	switch_state ^= 1;

	if (switch_state) {
		StartTimer1();
		} else {
		StopTimer1();
		DisplayData(cnt);
		cnt = 0;
	}
}
//-------------------------------------------
void InitPorts(void) {
	DDRB = (1 << PB0) | (1 << PB1) | (1 << PB3) | (1 << PB5);
	DDRD &= ~(1 << PD2);
	PORTD |= (1 << PD2);
}

void InitTimer1(void) {
	TCCR1A = 0;
	TCCR1B = (1 << CS11) | (1 << CS10) | (1 << WGM12);
	TCNT1 = 0;
	OCR1A = 15624;
}

void StartTimer1(void) {
	TCNT1 = 0;
	TIMSK1 |= (1 << OCIE1A);
}

void StopTimer1(void) {
	TIMSK1 &= ~(1 << OCIE1A);
}

void Bin2Dec(uint16_t data) {
	bcd_buffer[3] = data / 1000;
	bcd_buffer[2] = (data / 100) % 10;
	bcd_buffer[1] = (data / 10) % 10;
```

```
	bcd_buffer[0] = data % 10;
}
void SendData(uint8_t data) {
	for (uint8_t i = 0; i < 8; i++) {
		PORTB &= ~(1 << PB5); // CLK low
		if (data & (1 << (7 - i))) {
			PORTB |= (1 << PB3); // DAT high
			} else {
			PORTB &= ~(1 << PB3); // DAT low
		}
		PORTB |= (1 << PB5); // CLK high
	}
}

void DisplayData(uint16_t data) {
	Bin2Dec(data);
	PORTB &= ~(1 << PB1); // clk_out = 0
	for (uint8_t i = 0; i < 4; i++) {
		SendData(segments[bcd_buffer[i]]);
	}
	PORTB |= (1 << PB1); // clk_out = 1
}

void InitSPI(void) {
	DDRB |= (1 << PB3) | (1 << PB5);
	SPSR |= (1 << SPI2X);
	SPCR = (1 << SPE) | (1 << MSTR);
	PORTB &= ~(1 << PB3 | 1 << PB5);
}

void SPI_send(uint8_t data) {
	SPDR = data;
	while (!(SPSR & (1 << SPIF)));
}
```

Задание из методички - Вариант 1



Код на C:

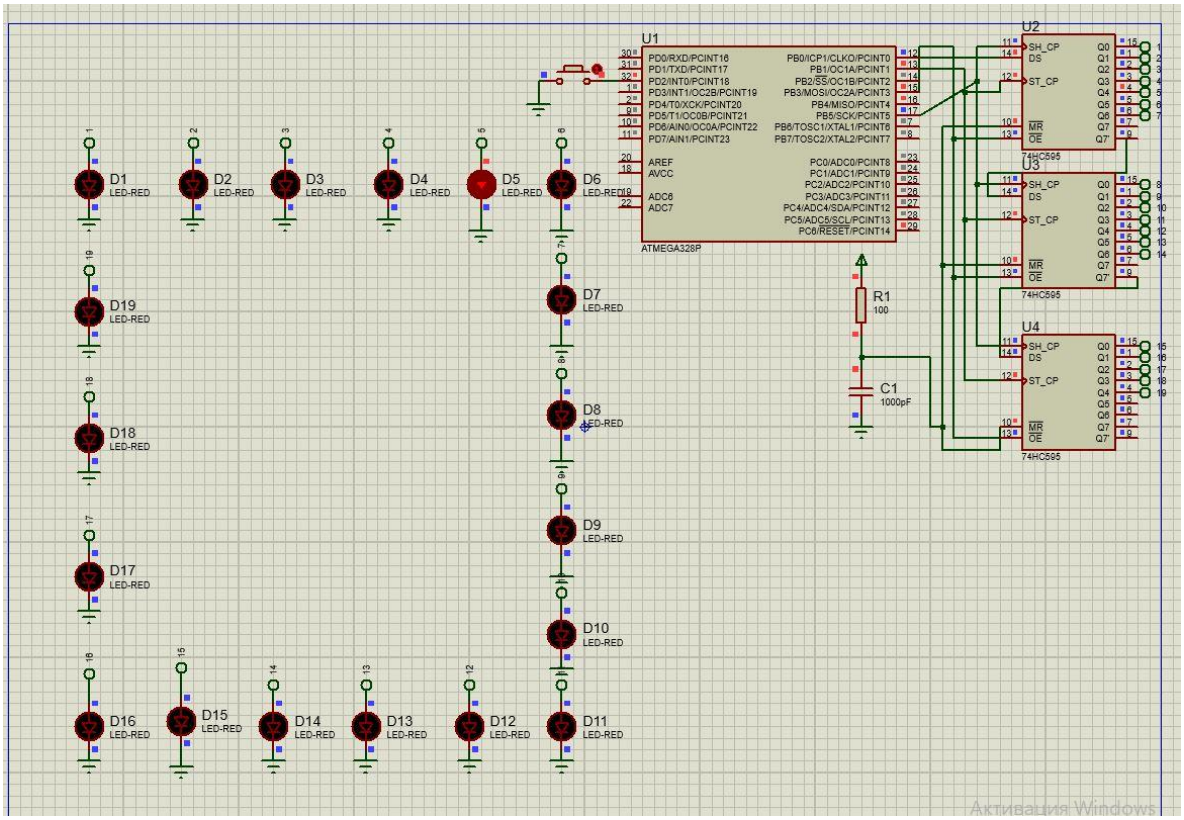
```c
#define F_CPU 1000000UL
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>

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 uint8_t cnt = 0;
volatile uint8_t switch_state = 0;

void InitPorts() {
      DDRB |= (1 << PB0) | (1 << PB1) | (1 << PB3) | (1 << PB5);
      PORTB &= ~(1 << PB0);
      DDRD &= ~(1 << PD2);
      PORTD |= (1 << PD2);
```

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

void InitTimer1() {
	TCCR1A = 0;
	TCCR1B = (1 << WGM12) | (1 << CS11) | (1 << CS10);
	TCNT1 = 0;
	OCR1A = 7811;
	TIMSK1 |= (1 << OCIE1A);
}

void StartTimer1() {
	TCNT1 = 0;
	TIMSK1 |= (1 << OCIE1A);
}

void StopTimer1() {
	TIMSK1 &= ~(1 << OCIE1A);
}

void InitSPI() {
	DDRB |= (1 << PB3) | (1 << PB5);
	SPSR |= (1 << SPI2X);
	SPCR = (1 << SPE) | (1 << MSTR);
	PORTB &= ~(1 << PB3 | 1 << PB5);
}

void SPI_send(uint8_t data) {
	SPDR = data;
	while (!(SPSR & (1 << SPIF)));
}

void DisplayData(uint8_t num) {
	uint8_t tens = num / 10;
	uint8_t ones = num % 10;
	PORTB &= ~(1 << PB1);
	SPI_send(segments[tens]);
	SPI_send(segments[ones]);
	PORTB |= (1 << PB1);
}

ISR(TIMER1_COMPA_vect) {
	cnt++;
	if (cnt > 99) cnt = 0;
	DisplayData(cnt);
}

ISR(INT0_vect) {
	switch_state = !switch_state;
```

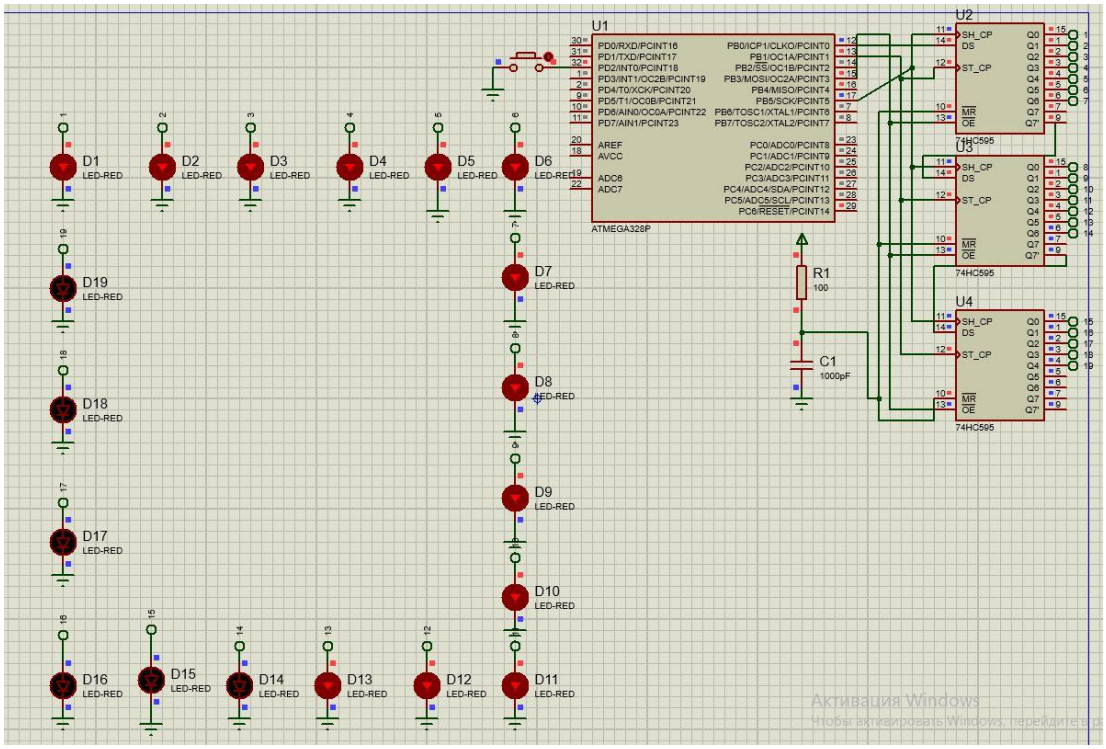
```
      if (switch_state) {
            StartTimer1();
            } else {
            StopTimer1();
            DisplayData(cnt);
            cnt = 0;
      }
}
int main() {
      InitPorts();
      InitTimer1();
      InitSPI();
      sei();
      DisplayData(0);
      while (1) {}
}
```

Задание от преподавателя: Вариант 26. Форма – квадрат. Эффекты 1,8,4

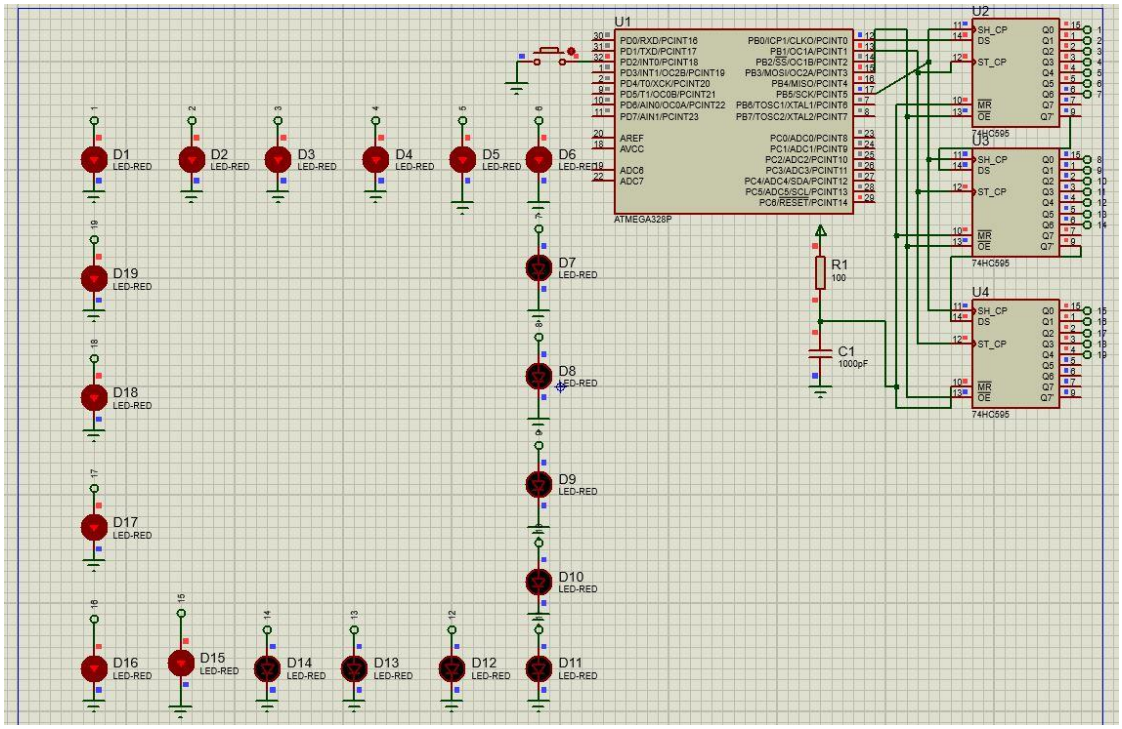
Эффект 1 (Пробежка в обе стороны 0->1->2->...-> count-1 -> count-2 -> ... -> 0):



Эффект 8 (Пробежка по 2 диода. В конечном положении должно светится заданное число диодов (эффект отскока):



Эффект 4 (Накопление и убывание к центру):



Код на C:

```c
#define F_CPU 16000000UL
#include <avr/io.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
};

void InitPorts(void);
void InitSPI(void);
void SPI_send(uint8_t data);
void InitTimer1(void);
void StartTimer1(void);
void StopTimer1(void);
void Bin2Dec(uint16_t data);
void DisplayData(uint16_t data);
void switch_mode(void);
void frame_creation(void);
void frame_output(void);
void InitTimer0(void);

volatile uint16_t cnt = 0;
volatile uint8_t switch_state = 0;
volatile uint8_t bcd_buffer[] = {0, 0, 0, 0};
volatile uint8_t update_frame_flag = 0;

volatile uint32_t frame = 0;
uint8_t step = 0;
uint8_t mode = 0;
uint8_t direction = 1;
const uint32_t bit = 1;
const uint8_t bounce_size = 2;
uint8_t bounce_flag = 0;

ISR(TIMER1_COMPA_vect) {
```

```
    DisplayData(cnt);
    if (cnt < 9999) {
        cnt++;
    } else {
        cnt = 0;
    }
}

ISR(TIMER0_COMPA_vect) {
    update_frame_flag = 1;
}

ISR(INT0_vect) {
    if (!(PIND & (1 << PD2))) {
        switch_mode();
    }
}

int main(void) {
    InitPorts();
    InitSPI();
    InitTimer1();
    InitTimer0();
    EIMSK |= (1 << INT0);
    EICRA |= (1 << ISC01);
    sei();

    PORTB &= ~(1 << PB0);
    DisplayData(0);

    while (1) {
        if (update_frame_flag) {
            update_frame_flag = 0;
            frame_creation();
            frame_output();
        }
    }
}

//-------------------------------------------
void InitPorts(void) {
    DDRB = (1 << PB0 | 1 << PB1 | 1 << PB3 | 1 << PB5);
    DDRD &= ~(1 << PD2);
    PORTD |= (1 << PD2);
}

void InitTimer1(void) {
    TCCR1A = 0;
    TCCR1B = (1 << CS11 | 1 << CS10 | 1 << WGM12);
    TCNT1 = 0;
```

```
    OCR1A = 6;
}

void InitTimer0(void) {
    TCCR0A = (1 << WGM01);
    TCCR0B = (1 << CS02) | (1 << CS00);
    OCR0A = 156;
    TIMSK0 |= (1 << OCIE0A);
}

void StartTimer1(void) {
    TCNT1 = 0;
    TIMSK1 |= (1 << OCIE1A);
}

void StopTimer1(void) {
    TIMSK1 &= ~(1 << OCIE1A);
}

void Bin2Dec(uint16_t data) {
    bcd_buffer[3] = (uint8_t)(data / 1000);
    data %= 1000;
    bcd_buffer[2] = (uint8_t)(data / 100);
    data %= 100;
    bcd_buffer[1] = (uint8_t)(data / 10);
    bcd_buffer[0] = (uint8_t)(data % 10);
}

void DisplayData(uint16_t data) {
    Bin2Dec(data);
    PORTB &= ~(1 << PB1);
    SPI_send(segments[bcd_buffer[3]]);
    SPI_send(segments[bcd_buffer[2]]);
    SPI_send(segments[bcd_buffer[1]]);
    SPI_send(segments[bcd_buffer[0]]);
    PORTB |= (1 << PB1);
}

void InitSPI(void) {
    DDRB |= (1 << PB3 | 1 << PB5);
    SPSR |= (1 << SPI2X);
    SPCR = (1 << SPE | 1 << MSTR);
    PORTB &= ~(1 << PB3 | 1 << PB5);
}

void SPI_send(uint8_t data) {
    SPDR = data;
    while (!(SPSR & (1 << SPIF)));
}
```

```
void switch_mode(void) {
    mode++;
    if (mode > 2) {
        mode = 0;
    }
    frame = 0;
    step = 0;
    direction = 1;
    bounce_flag = 0;
}

void frame_creation(void) {
    switch (mode) {
        case 0:
            frame = bit << step;
            if (direction) {
                step++;
                if (step > 20) {
                    step = 19;
                    direction = 0;
                }
            } else {
                step--;
                if (step == 0) {
                    direction = 1;
                }
            }
            break;

        case 1:
        if (bounce_flag) {
              frame = 0;
              bounce_flag = 0;
              step = direction ? bounce_size : (18 - bounce_size);
              } else {
              frame |= (bit << step) | (bit << (step + 1));
              if (direction) {
                    step += 2;
                    if (step > 20) {
                          bounce_flag = 1;
                          direction = 0;
                    }
                    } else {
                    step -= (step == 0) ? 0 : 2;
                    bounce_flag = step == 0;
                    direction = step == 0 ? 1 : direction;
              }
        }
            break;
```

```
        case 2:
            if (step < 11) {
                frame |= (bit << step) | (bit << (20 - step));
            } else if (step < 20) {
                uint8_t center = 10, offset = step - 11;
                frame &= ~((bit << (center - offset)) | (bit << (center
+ offset)));
            }
            if (++step == 20) {
                step = 0;
                frame = 0;
            }
            break;
    }
}

void frame_output(void) {
    PORTB &= ~(1 << PB1);
    SPI_send((frame >> 16) & 0xFF);
    SPI_send((frame >> 8) & 0xFF);
    SPI_send(frame & 0xFF);
    PORTB |= (1 << PB1);
}
```

Выводы: мы научились работать с таймерами и интерфейсом SPI