«Змейка». Таблица состояний:

№ диода / Шаг	7	6	5	4	3	2	1	0
1	0	1	1	1	0	0	0	0
2	0	0	1	1	1	0	0	0
3	0	0	0	1	1	1	0	0
4	0	0	0	0	1	1	1	0
5	0	0	0	0	0	1	1	1
6	0	0	0	0	1	1	1	0
7	0	0	0	1	1	1	0	0
8	0	0	1	1	1	0	0	0
9	0	1	1	1	0	0	0	0
10	1	1	1	0	0	0	0	0

```
#include <avr/io.h>
#include <avr/interrupt.h>

#define INITIAL_TIMER_VALUE 60000

uint8_t pattern = 0b11111000;
uint8_t direction = 0; // Направление движения змейки
uint8_t step = 0;

ISR(TIMER1_OVF_vect)
{
	PORTC = pattern;

	if (direction == 0) {
		pattern = (pattern << 1) | (pattern >> 7); // Сдвиг вправо
		} else {
		pattern = (pattern >> 1) | (pattern << 7); // Сдвиг влево
	}

	step++;

	if(step > 4) {
		direction = !direction;
		step = 0;
	}

	TCNT1 = INITIAL_TIMER_VALUE;
}


int main(void) {
	DDRC=0xFF;
	TCCR1B|=(1<<CS12)|(1<<CS10);
	TIMSK|=(1<<TOIE1);
```

```
        TCNT1 = INITIAL_TIMER_VALUE;
        sei();

        while(1)
        {
        }
}
```

«Полицейская мигалка». Таблица состояний:

Шаг \ № диода	7	6	5	4	3	2	1	0
1	1	1	1	1	0	0	0	0
2	0	0	0	0	1	1	1	1

```
#include <avr/io.h>
#include <avr/interrupt.h>

#define INITIAL_TIMER_VALUE 60000

uint8_t pattern = 0b11110000;

ISR(TIMER1_OVF_vect)
{
        PORTC = pattern;
        pattern = pattern^0b11111111;
        TCNT1 = INITIAL_TIMER_VALUE;
}


int main(void) {
        DDRC=0xFF;
        TCCR1B|=(1<<CS12)|(1<<CS10);
        TIMSK|=(1<<TOIE1);
        TCNT1=63000;
        sei();

        while(1)
        {
        }
}
```

«Светофор». Таблица состояний:

№ диода / Шаг	7	6	5	4	3	2	1	0
1	0	0	0	0	0	1	1	1
2	0	0	1	1	1	0	0	0
3	1	1	0	0	0	0	0	0

```
#include <avr/io.h>
#include <avr/interrupt.h>

#define INITIAL_TIMER_VALUE 60000

uint8_t svetofor = 0;
uint8_t flag = 1;

void update() {
	if (flag) {
		svetofor++;
		} else {
		svetofor--;
	}
}

ISR(TIMER1_OVF_vect) {
	switch (svetofor) {
		case 0: flag = 1; PORTC = 0b11111000; update(); break;
		case 1: PORTC = 0b11000111; update(); break;
		case 2: flag = 0; PORTC = 0b00111111; update(); break;
	}

	TCNT1 = INITIAL_TIMER_VALUE;
}


int main(void) {
	DDRC=0xFF;
	TCCR1B|=(1<<CS12)|(1<<CS10);
	TIMSK|=(1<<TOIE1);
	TCNT1=INITIAL_TIMER_VALUE;
	sei();
	while(1)
	{
	}
}
```

«Край-край с смещением к центру». Таблица состояний:

№ диода / Шаг	7	6	5	4	3	2	1	0
1	0	0	0	0	0	0	0	1
2	1	0	0	0	0	0	0	0
3	0	0	0	0	0	0	1	0
4	0	1	0	0	0	0	0	0
5	0	0	0	0	0	1	0	0
6	0	0	1	0	0	0	0	0
7	0	0	0	0	1	0	0	0
8	0	0	0	1	0	0	0	0

```
#include <avr/io.h>
#include <avr/interrupt.h>

#define INITIAL_TIMER_VALUE 60000


uint8_t triangle = 0b11111110;
uint8_t step = 7;

ISR(TIMER1_OVF_vect)
{
	PORTC = triangle;
	if (step % 2 == 1){
		triangle = (triangle<<step)|(triangle>>(8-step));
		} else {
		triangle=(triangle>>step)|(triangle<<(8-step));
	}

	if (step == 0){
		step = 8;
		triangle = 0b11111110;
	}

	step--;

	TCNT1 = INITIAL_TIMER_VALUE;
}


int main(void) {
	DDRC=0xFF;
	TCCR1B|=(1<<CS12)|(1<<CS10);
	TIMSK|=(1<<TOIE1);
	TCNT1=INITIAL_TIMER_VALUE;
	sei();
```

```
        while(1)
        {
        }
}
```

«Вывод кодов чисел 1-15 в младшую тетраду и их зеркальное представление в старшей тетраде». Таблица состояний:

Шаг \ № диода	7	6	5	4	3	2	1	0
1	1	0	0	0	0	0	0	1
2	0	1	0	0	0	0	1	0
3	1	1	0	0	0	0	1	1
4	0	0	1	0	0	1	0	0
5	1	0	1	0	0	1	0	1
6	0	1	1	0	0	1	1	0
7	1	1	1	0	0	1	1	1
8	0	0	0	1	1	0	0	0
9	1	0	0	1	1	0	0	1
10	0	1	0	1	1	0	1	0
11	1	1	0	1	1	0	1	1
12	0	0	1	1	1	1	0	0
13	1	0	1	1	1	1	0	1
14	0	1	1	1	1	1	1	0
15	1	1	1	1	1	1	1	1

```
#include <avr/io.h>
#include <avr/interrupt.h>

#define INITIAL_TIMER_COUNT 63000
#define MAX_NUM 16

uint8_t num = 1;

unsigned char get_low_nibble(unsigned char num) {
        return num & 0x0F;
}

unsigned char reverse_nibble(unsigned char nibble) {
        unsigned char reversed_nibble = 0;
        for (int i = 0; i < 4; i++) {
                unsigned char bit = (nibble >> i) & 1;
                reversed_nibble |= bit << (3 - i);
        }
        return reversed_nibble;
}
```

```
unsigned char combine_nibbles(unsigned char low_nibble, unsigned char
reversed_nibble) {
	return (reversed_nibble << 4) | low_nibble;
}

ISR(TIMER1_OVF_vect)
{
	if (num == MAX_NUM) {
		num = 0;
		} else {
		uint8_t low_nibble = get_low_nibble(num);
		PORTC = combine_nibbles(low_nibble, reverse_nibble(low_nibble))
^ 0xFF;
	}

	num++;

	TCNT1 = INITIAL_TIMER_COUNT;
}

int main(void) {
	DDRC = 0xFF;
	TCCR1B |= (1 << CS12) | (1 << CS10);
	TIMSK |= (1 << TOIE1);
	TCNT1 = INITIAL_TIMER_COUNT;
	sei();

	while(1) {
	}
}
```

«Вывод кодов чисел 1-15 в младшую тетраду и их побитовая инверсия в старшей тетраде». Таблица состояний:

№ диода / Шаг	7	6	5	4	3	2	1	0
1	1	1	1	0	0	0	0	1
2	1	1	0	1	0	0	1	0
3	1	1	0	0	0	0	1	1
4	1	0	1	1	0	1	0	0
5	1	0	1	0	0	1	0	1
6	1	0	0	1	0	1	1	0
7	1	0	0	0	0	1	1	1
8	0	1	1	1	1	0	0	0
9	0	1	1	0	1	0	0	1
10	0	1	0	1	1	0	1	0
11	0	1	0	0	1	0	1	1
12	0	0	1	1	1	1	0	0
13	0	0	1	0	1	1	0	1
14	0	0	0	1	1	1	1	0
15	0	0	0	0	1	1	1	1

```
#include <avr/io.h>
#include <avr/interrupt.h>

#define INITIAL_TIMER_VALUE 60000

uint8_t get_low_nibble(uint8_t num) {
	return num & 0x0F;
}

uint8_t reverse_nibble(uint8_t nibble) {
	return nibble ^ 0b1111;
}

uint8_t combine_nibbles(uint8_t low_nibble, uint8_t reversed_nibble) {
	return (reversed_nibble << 4) | low_nibble;
}

volatile uint8_t num = 1;

ISR(TIMER1_OVF_vect)
{
	if (num == 16) num = 0;
	else {
		uint8_t low_nibble = get_low_nibble(num);
		PORTC = combine_nibbles(low_nibble, reverse_nibble(low_nibble))
^ 0xFF;
	}
```

```
        num++;

        TCNT1 = INITIAL_TIMER_VALUE;
}

int main(void) {
        DDRC = 0xFF;
        TCCR1B |= (1 << CS12) | (1 << CS10);
        TIMSK |= (1 << TOIE1);
        TCNT1 = INITIAL_TIMER_VALUE;
        sei();

        while(1) {
        }
}
```

Вопросы по лабораторной работе №1

4.1 Задача предделителя тактовой частоты в микроконтроллерах.

Предделитель тактовой частоты в микроконтроллерах - это функциональный блок, который позволяет делить входную тактовую частоту на более низкую выходную частоту. Это может быть полезно, например, для уменьшения частоты тактового сигнала, чтобы уменьшить энергопотребление или синхронизировать работу различных компонентов микроконтроллера.

Предделитель может быть настроен на определенное деление, обычно путем программирования специальных регистров микроконтроллера. Например, если у нас есть микроконтроллер с тактовой частотой 16 МГц, и нам нужно установить частоту 1 МГц для работы с периферийными устройствами, мы можем использовать предделитель, чтобы поделить исходную частоту на 16 и получить нужную частоту.

4.2 Как задается режим работы портов микроконтроллера?

Режим работы портов микроконтроллера задается через специальные регистры управления портами (Port Control Registers). Эти регистры позволяют настраивать порты микроконтроллера на вход или выход.

Вот пример того, как можно настроить порт в качестве входа или выхода на микроконтроллере AVR

```
// Настройка порта D на вход
DDRD &= ~(1 << PD2); // Сброс бита PD2
```

```
// Настройка порта D на выход
DDRD |= (1 << PD3); // Установка бита PD3
```

4.3 Как изменится значение переменной, если необходимо вызывать прерывание каждые 2 секунды?

Для настройки прерывания каждые 2 секунды на микроконтроллере ATmega16, необходимо использовать таймер. Вот пример, как можно настроить таймер на ATmega16 для вызова прерывания каждые 2 секунды:

```
#include <avr/io.h>
#include <avr/interrupt.h>

volatile uint16_t counter = 0;

void timer1_init() {
	TCCR1B |= (1 << WGM12); // Выбираем режим CTC
	OCR1A = 31249; // Устанавливаем значение для достижения интервала 2 секунды при тактовой частоте 16 МГц
	TIMSK |= (1 << OCIE1A); // Разрешаем прерывание при совпадении с OCR1A
	TCCR1B |= (1 << CS12) | (1 << CS10); // Устанавливаем предделитель на 1024
}

ISR(TIMER1_COMPA_vect) {
	counter++; // Увеличиваем значение переменной каждый раз при вызове прерывания
}

int main() {
	timer1_init();
	sei(); // Разрешаем прерывания

	while(1) {    }

	return 0;
}
```