Лабораторная работа № 9. Моделирование схем с микроконтроллерами в программе Proteus.Isis

Программирование микроконтроллеров в программе CodeVisionAVR.

Ход работы

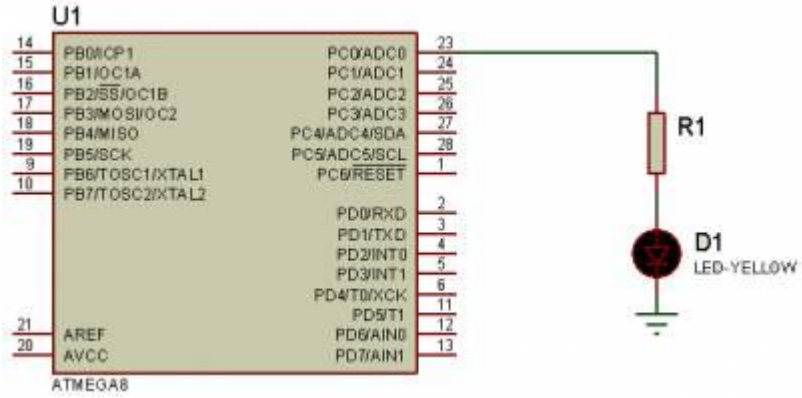
1. Создание схемы в Proteus.Isis.

Схема для эксперимента приведена на рис.1.

Наш светодиод рассчитан примерно на силу тока примерно в 20 мА. При этом на светодиоде падает примерно 2 В. Остаётся: 5 В (напряжение VCC) – 2 = 3 В.

По закону Ома I = U/R. Тогда R= 150 Ом.

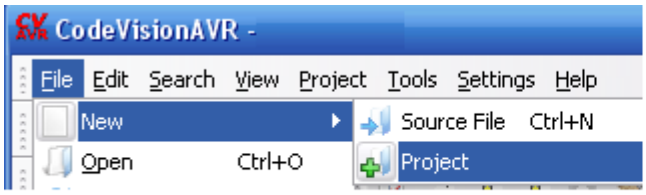
Вычисленное значение R и вводим в качестве параметра Resistance свойств R1.

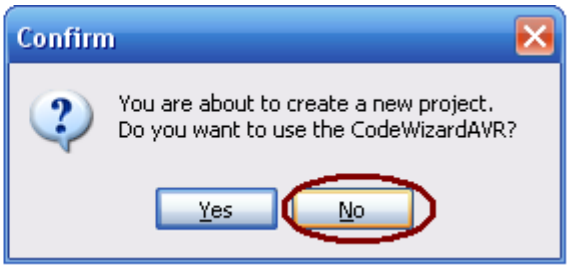


2. Создание программного кода в *CodeVisionAVR*

Для того, чтобы микроконтроллер работал, ему нужен программный файл – «прошивка». Создадим программу в среде программирования для микроконтроллеров AVR CodeVisionAVR. Откройте в главном меню CodeVisionAVR в группе HP InfoTech .

Создаём проект без мастера.

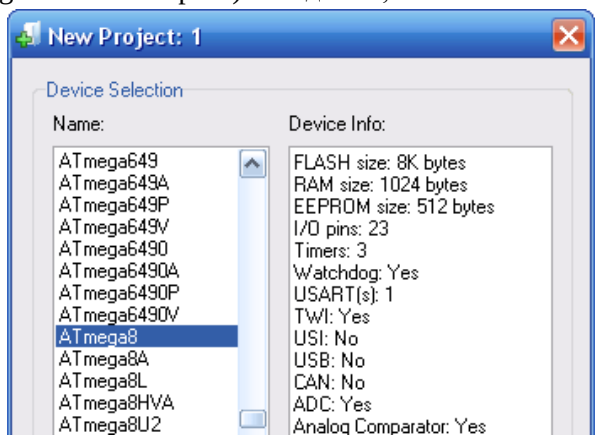


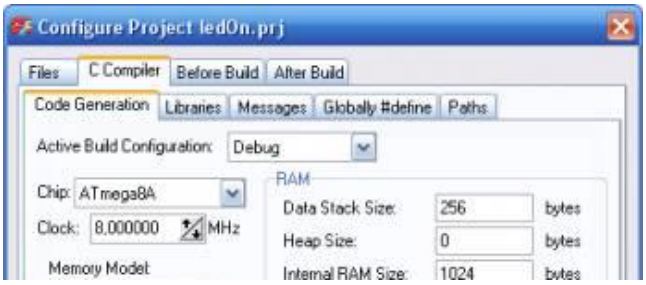


Сохраните проект в своей папке.

Выбираем наш chip ATMega8A.

После создания проекта откроется окно его конфигурации (его всегда можно открыть так: Project → Configure → C Compiler). Убедимся, что тактовая частота (Clock) = 8 MHz, остальное тоже оставляем дефолтное.





Комментарии (неисполняемый код) пишутся после двух слешей: // Управление портом C

Подключаем заголовочные файлы директивами

#include <io.h> //(ввод/вывод) и
<delay.h> // (задержка).

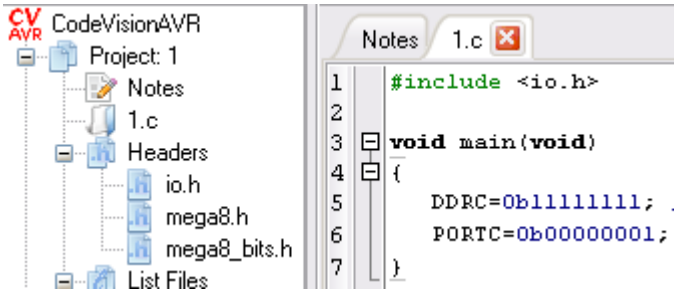
Пишем главную функцию

```
void main(void)
 {
DDRC=0b11111111; // Data Direction Registr - регистр-переключатель направления данных порта С.
```

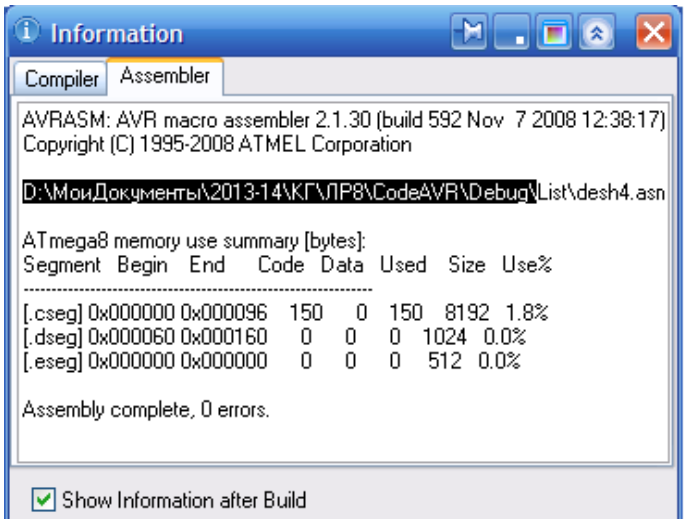
К DDR подключены пины порта, он определяет их направление выдачи сигнала.
1 – пин настроен на вывод сигнала.
0 (по умолчанию) – на вход (считывание состояние).
0b – в двоичной системе.
Все единицы – значит, все пины порта настроены на вывод сигнала.
Выбор направления данных можно задать и 16-ричными цифрами, тогда получим равносильный оператор DDRC = 0xFF (вспомним тетрады: 1111(=F) 1111(=F)).
Итак, пишем инструкции в программе.
Для начала научимся подавать напряжение:

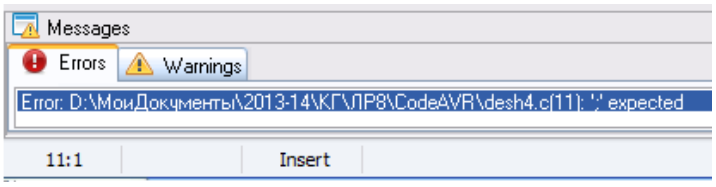
```
PORTC=0b00000001; // подаём на нулевой (младший) пин логическую единицу (5 В), а на остальных оставляем логический 0.
```

Пока все, закрываем программу
}
Вот что, по минимуму, должно получиться:



Компилируем код (**F9**). Если всё хорошо, увидим диалог с информацией о программе, если нет – то, кроме диалога и список ошибок в нижней части окна программы.

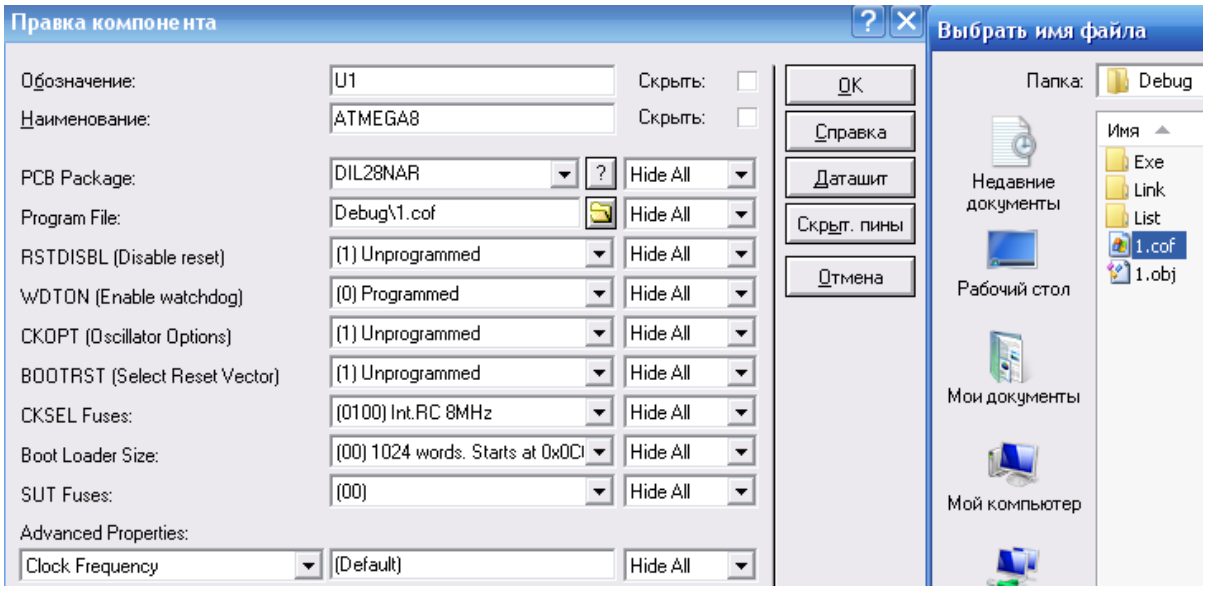




Исправим (если они есть) ошибки (дабл-клик по описанию ошибки выделит некорректную строку, либо следующую за ней, если предыдущую забыли закрыть точкой с запятой) и построим (по **Ctrl+F9**) hex-файл прошивки (попутно создаются и отладочные файлы в других форматах, напр. cof – для отладки программ CVAVR).
Бинарный файл .hex (который исполняется МК) будет в папке \Debug\Exe, а .cof – в \Debug.

3. Прошивка МК в Proteus.Isis.

В свойствах МК ATMega8 изменяем только CKSEL Fuses – выставляем в 0100, что задаёт тактирование МК от внутреннего RC-генератора на частоте 8MHz, и указываем в качестве ProgramFile созданный в CodeVisionAVR .cof–файл (выбор .cof -файла вместо .hexпозволит вести отладку по исходному коду):



Всё, схема должна заработать – должен загореться светодиод.

4. После того, как мы научились зажигать светодиод, сможем его и потушить.
Для этого в исходном коде добавьте команду задержки после включения

delay_ms(1000); // пауза в 1 сек.

Но для того, чтобы она заработала, необходимо подключить библиотеку, в которой она прописана. Для этого в начале кода добавьте

#include <delay.h>

После задержки отключите светодиод сбросом в 0 младшего разряда порта С командой

PORTC=0b00000000;

```
1   #include <io.h>
2   #include <delay.h>
3
4   void main(void)
5   {
6       DDRC=0b11111111; ,
7       PORTC=0b00000001;
8       delay_ms(1000);
9       PORTC=0b00000000;
10  }
```

Перепрошейте МК и проверьте в Протеусе работу схемы – светодиод должен включиться на секунду и потухнуть.

5. Мигание светодиода можно организовать в бесконечном цикле

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

Перенесите в него команды установки напряжения и задержки (конфигурирование направления данных оставьте перед циклом), добавьте задержку после гашения светодиода и проверьте результат (мигание) в Протеусе.

Notes 1.c

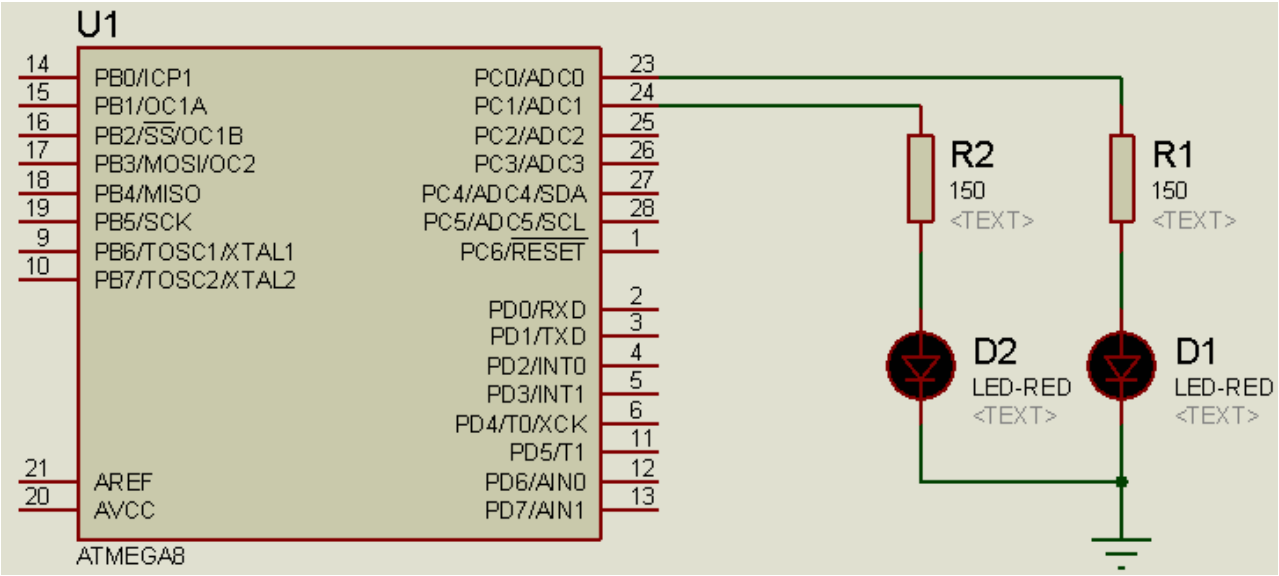
```
1  #include <io.h>
2  #include <delay.h>
3
4  void main(void)
5  {
6     DDRC=0bllllllll;
7
8  while (1)
9   {
10    PORTC=0b00000001;
11    delay_ms(1000);
12    PORTC=0b00000000;
13    delay_ms(1000);
14   }
15 }
```

6. Добавьте в схему (на пин DC.1, что соответствует разряду 1 порта С) ещё один светодиод и помигайте ими поочерёдно (один – зажигается, другой – гаснет, через секунду – наоборот).

Для этого нужно всего лишь изменить предпоследнюю команду на

```
PORTC=0b00000010;
```

Соответственно отредактируйте схему в Протеусе и проведите в нём эмуляцию.

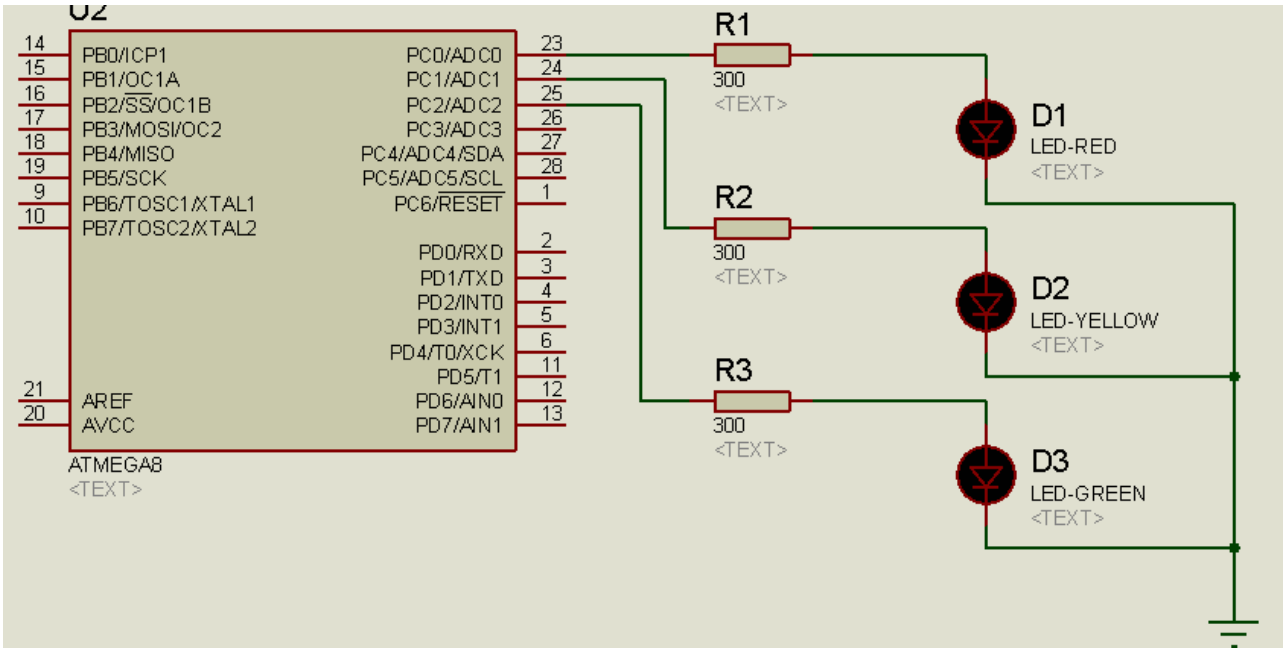


7. Добавьте в схему третий светодиод и создайте программу "**Светофор**" – мигания по схеме, близкой к реальному светофору.

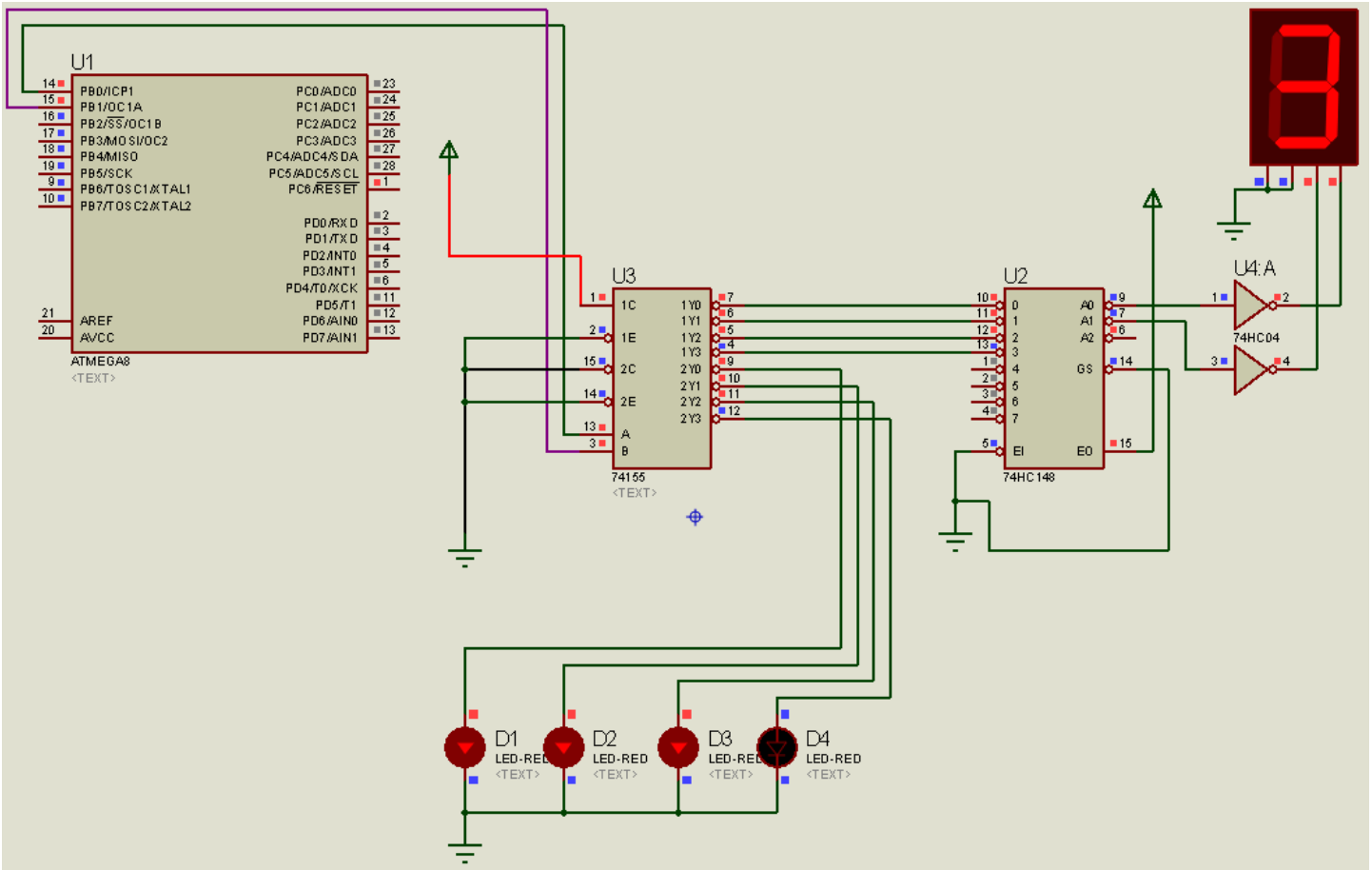
В простейшем случае код цикла будет таким:

```
while (1)
  {
    PORTC=0b00000001;
    delay_ms(1000);
    PORTC=0b00000010;
    delay_ms(500);
    PORTC=0b00000100;
    delay_ms(1000);
    PORTC=0b00000010;
    delay_ms(500);
  }
```

Схема:



8. Смоделируем дешифратор 2→4 на ИМС 74155 (представляющей собой сдвоенные дешифраторы 2→4). Выходы одного дешифратора подадим на 4 светодиода, а второго – через шифратор 74148 – на 7-сегментный индикатор с встроенным дешифратором:

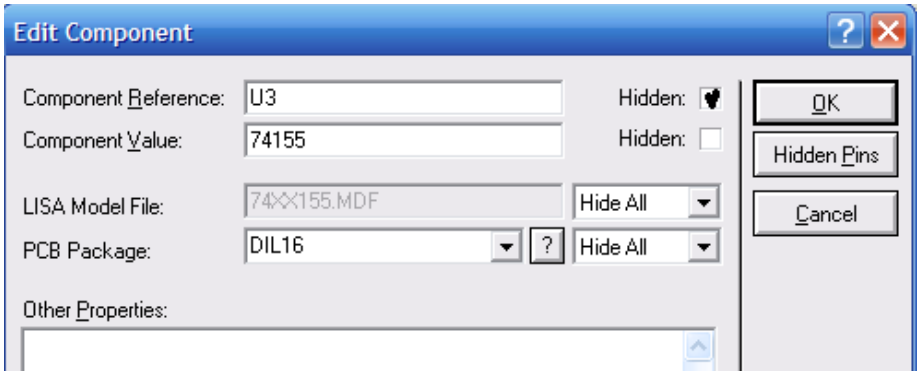


Программный код:

```
#include <io.h>
#include <delay.h>
void main(void)
{
DDRB =255;
PORTB = 0;
while (1)
{
     PORTB++;
     if (PORTB>3) PORTB=0;
        delay_ms(500);
}
}
```

Светодиоды будут поочерёдно гаснуть, а на индикаторе высвечиваться цифры от 0 до 3.

Кстати, чтобы под компонентом не отображался навязчивый <Text>, введите пробел в Other свойствах компонента, а чтобы убрать идентификаторы ИМС, поставьте галочку в свойствах Hidden:



9. Самостоятельное задание:

1) Измените направление перемещения бегущего огня на противоположное; на двустороннее
2) Сделайте, чтобы индикатор показывал другие цифры;
3) В схеме задания 7 измените алгоритм мигания светофора на более приближённый к реальному.