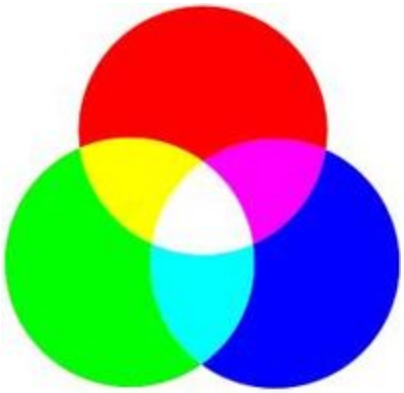
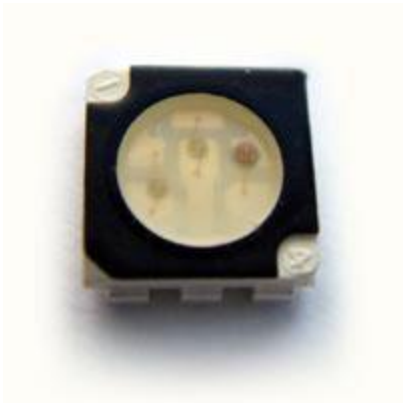
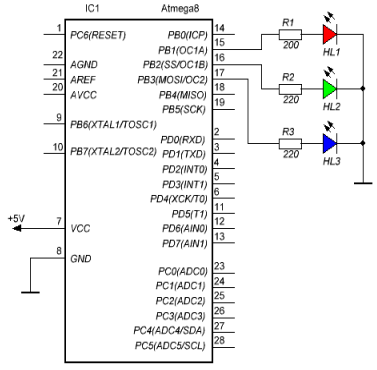
Зажигаем RGB светодиод. Программная реализация ШИМ

Цель: Изучить принцип построения программной Широтно-импульсной модуляции на основе микроконтроллера.

Для отображения всей палитры видимых оттенков теоретически достаточно иметь три цвета. Это — так называемый RGB-синтез (Red — красный, Green — зеленый, Blue — синий). Но в реальности трех цветов бывает недостаточно. Смешивание цветов может происходить одним из трех способов. Светодиоды могут раздельно располагаться в светильнике, а цвета смешиваются уже непосредственно на освещаемом объекте. Преимущества такого подхода — простота конструкции и высокая светоотдача. Недостаток — наличие нескольких разноцветных теней для объектов, расположенных близко к светильнику. Более сложный вариант — оптическая система, смешивающая лучи от разных светодиодов. Наконец, выпускаются специальные RGB-светодиоды, объединяющие в одном корпусе три кристалла разных цветов. Использование RGB-светодиодов позволяет создать очень тонкий осветительный прибор, дающий равномерный свет без заметных зон затенения.







Аноды RGB светодиода подключаем к линиям 1,2,3 порта B, катоды соединяем с минусом. Чтобы получить разнообразные палитры цвета на аноды будем подавать ШИМ сигнал в определенной последовательности. В этом примере мы специально используем программный ШИМ, хотя на ATmega8 можно без проблем получить аппаратный ШИМ на 3 канала. Программный ШИМ можно использовать в случаях нехватки таймеров/счетчиков и по другим причинам. Для генерации ШИМ определенной частоты используем прерывание по переполнению 8-ми битного таймера T0(TIMER0_OVF_vect). Так как предделитель не используем, частота переполнения таймера будет равна 31250 Гц. В обработчике прерывания исходя из значений в переменных pwm_r, pwm_g, pwm_b переключаются ножки порта B. Цветовые эффекты настраиваются с помощью функций, где задается время свечения светодиода. В тестовой программе сначала загораются красный, зеленый, синий, белый цвета, а потом начинается цикл с переходами цвета.

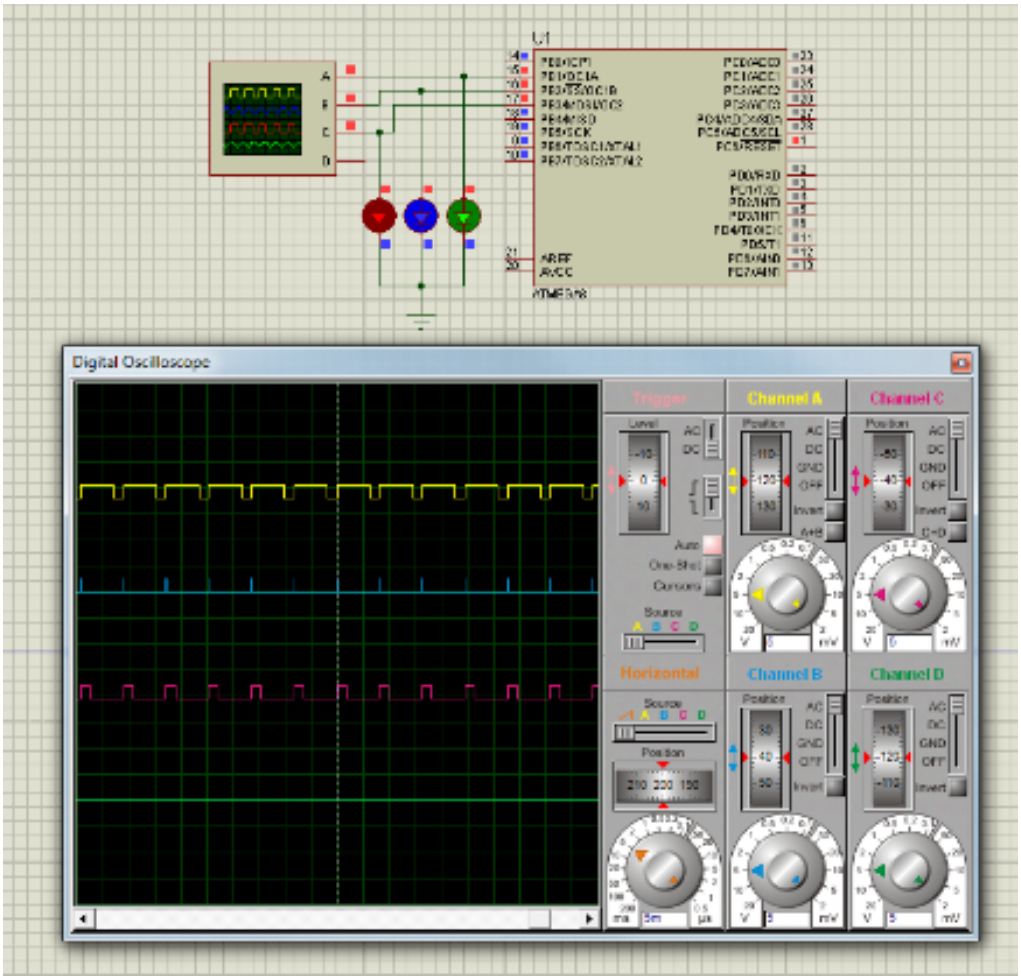
```
// Управление RGB светодиодом. Программный ШИМ
#include <avr/interrupt.h>
#include <avr/io.h>

volatile unsigned char pwm_r, pwm_g, pwm_b;

// Прерывание по переполнению T0
ISR(TIMER0_OVF_vect)
{
static unsigned char pwm_counter;
// Если значение яркости больше значения счетчика
// включаем соответствующий вывод
if(pwm_counter < pwm_r) PORTB |= (1 << PB1);
// Иначе вывод выключен
else PORTB &= ~(1 << PB1);
if(pwm_counter < pwm_g) PORTB |= (1 << PB2);
else PORTB &= ~(1 << PB2);
if(pwm_counter < pwm_b) PORTB |= (1 << PB3);
else PORTB &= ~(1 << PB3);
// Инкрементируем счетчик
pwm_counter++;
}

// Процедура задержки в микросекундах
void delay_us(unsigned char time_us)
{ register unsigned char i;

  for(i = 0; i < time_us; i++)   // 4 цикла
    {
```

```
      asm (" PUSH  R0 ");       // 2 цикла
      asm (" POP   R0 ");       // 2 цикла
        // 8 циклов = 1 us для 8MHz
    }
}
// Процедура задержки в миллисекундах
void delay_ms(unsigned int time_ms)
{ register unsigned int i;

  for(i = 0; i < time_ms; i++)
    { delay_us(250);
      delay_us(250);
      delay_us(250);
      delay_us(250);
    }
}

// Красный цвет
void red (unsigned char time)
{
for(unsigned char a = 0; a < 255; a++)
{
pwm_r = a; // Увеличение
pwm_g = 0;
pwm_b = 0;
delay_ms(time);
}
for(unsigned char a = 255; a > 0; a--)
{
pwm_r = a; // Уменьшение
pwm_g = 0;
pwm_b = 0;
delay_ms(time);
}
}

// Зеленый цвет
void green (unsigned char time)
{
for(unsigned char a = 0; a < 255; a++)
{
pwm_r = 0;
pwm_g = a;
pwm_b = 0;
delay_ms(time);
```

```
}
for(unsigned char a = 255; a > 0; a--)
{
pwm_r = 0;
pwm_g = a;
pwm_b = 0;
delay_ms(time);
}
}

// Синий цвет
void blue (unsigned char time)
{
for(unsigned char a = 0; a < 255; a++)
{
pwm_r = 0;
pwm_g = 0;
pwm_b = a;
delay_ms(time);
}
for(unsigned char a = 255; a > 0; a--)
{
pwm_r = 0;
pwm_g = 0;
pwm_b = a;
delay_ms(time);
}
}

// Белый цвет
void white (unsigned char time)
{
for(unsigned char a = 0; a < 255; a++)
{
pwm_r = a;
pwm_g = a;
pwm_b = a;
delay_ms(time);
}
for(unsigned char a = 255; a > 0; a--)
{
pwm_r = a;
pwm_g = a;
pwm_b = a;
delay_ms(time);
```

```
}
}

// Переход цветов
void rgb (unsigned char time)
{
for(unsigned char a = 0; a < 255; a++)
{
pwm_r = a;
pwm_g = 255 - a;
delay_ms(time);
}
for(unsigned char a = 0; a < 255; a++)
{
pwm_b = a;
pwm_r = 255 - a;
delay_ms(time);
}
for(unsigned char a = 0; a < 255; a++)
{
pwm_g = a;
pwm_b = 255 - a;
delay_ms(time);
}
}

int main (void)
{
// Порты ввода/вывода
DDRB = 0xFF; // Порт B выход
PORTB = 0x00;
// Таймер 0
TCCR0 |= (1 << CS00); // Тактирование T0 без предделителя
TIMSK |= (1 << TOIE0); // Разрешаем прерывание по переполнению T0

sei(); // Глобально разрешаем прерывания

while(1)
{
red(5); // Включаем эффекты
green(5);
blue(5);
white(10);
  for(;;)
  {
```

```
 rgb(100);
  }
}
}
```



Задания для выполнения:

1. Изучить принцип реализации программной ШИМ на основе микроконтроллеров.
2. Смоделировать работу программной ШИМ в среде Proteus.

Результат и архив выполнения отправить на e-mail: rasov@rambler.ru с темой ПрограммныйШИМ_ФИО