МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ

ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

кафедра «Вычислительная техника»

**Лабораторная работа №2 (вариант 5)**

**Тема: «Моделирование функций(алгоритмов) с применением микроконтроллера»**

по дисциплине: «Микропроцессорные системы»

Выполнил:

студент 4 курса, гр. ИВТВМбд-41

Захарычев Н.А

Проверил:

к.т.н, доцент кафедры ВТ

Игонин А.Г.

г. Ульяновск, 2017

**Содержание**

Содержание………………………………………………………………………..2

Задание на лабораторную работу…………………………………………….….3

Ход работы………………………………………………………………………..3

Тестирование……………………………………………………………………...6

Вывод……………………………………………………………………………...7

Список использованной литературы……………………………………………7

Приложение. Исходный код……………………………………………………..7

**Задание на лабораторную работу**

Необходимо реализовать лабораторную работу №1 с применением микроконтроллера вместо логических элементов.

Функции, заданные по варианту:

O1 = (!02 & !03 & !04) & T3

O2 = (I4 | O4 | I3) & T1

O3 = (I1 & I2) & T2

O4 = O3 & I4

Характеристики таймеров:

T1 = 5, T2 = 7, T3 = 9

Микроконтроллер по варианту:

Семейство микроконтроллеров AVR.

**Ход работы**

1. **Описание использованных программ**

Для выполнения данной лабораторной работы, было принято решение использовать систему автоматизированного проектирования **Proteus 8** и интегрированную среду разработки **Atmel Studio**.

**Proteus** — пакет программ для автоматизированного проектирования (САПР) электронных схем. Пакет представляет собой систему схемотехнического моделирования, базирующуюся на основе моделей электронных компонентов принятых в PSpice. Отличительной чертой пакета PROTEUS является возможность моделирования работы программируемых устройств: микроконтроллеров, микропроцессоров, DSP и проч. Библиотека компонентов содержит справочные данные.

**Atmel Studio** (ранее AVR Studio) — основанная на Visual Studio бесплатная проприетарная интегрированная среда разработки (IDE) для разработки приложений для 8- и 32-битных микроконтроллеров семейства AVR и 32-битных микроконтроллеров семейства ARM от компании Atmel, работающая в операционных системах WindowsNT/2000/XP/Vista/7/8/10. Atmel Studio содержит компилятор GNU C/C++ и эмулятор, позволяющий отладить выполнение программы без загрузки в микроконтроллер. Atmel Studio содержит в себе менеджер проектов, редактор исходного кода, инструменты виртуальной симуляции и внутрисхемной отладки, позволяет писать программы на ассемблере или на C/C++.

1. **Описание микроконтроллера.**

Микроконтроллер, используемый для моделирования – Atmega8 (рис.1).

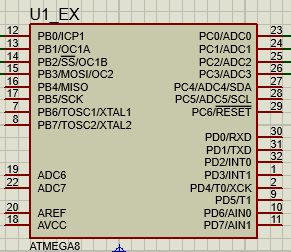
Микроконтроллер ATmega8 выполнен по технологии CMOS, 8-разрядный, микропотребляющий, основан на AVR-архитектуре RISC. Выполняя одну полноценную инструкцию за один такт, ATmega8 достигает производительности 1 MIPS на МГц, позволяя достигнуть оптимального соотношения производительности к потребляемой энергии.

Технические параметры:

* Память для программ составляет 8 Кб с возможностью перезаписать 10 000 раз
* 512 байт флеш-памяти для хранения переменных (100 000 циклов перезаписи)
* 1 Кб ОЗУ и 32 регистра общего назначения
* Два 8-разрядных Таймера/Счетчика с раздельным прескалером, режим сравнения
* 16-разрядный Таймер/Счетчик с раздельным прескалером, режим сравнения, режим захвата
* Таймер реального времени с независимым генератором
* 3 канала ШИМ
* 6 каналов 10-разрядного АЦП
* Двухпроводный последовательный интерфейс
* Программируемый последовательный USART
* Интерфейс SPI с режимами Master/Slave
* Программируемый сторожевой таймер с отдельным независимым генератором
* Встроенный аналоговый компаратор
* Сброс при включении питания, программируемая защита от провалов питания
* Встроенный калиброванный RC-генератор
* Обработка внутренних и внешних прерываний
* 5 режимов с пониженным энергопотреблением: Idle, ADC Noise Reduction, Power-save, Power-down, и Standby
* Напряжение питания 4.5 - 5.5В
* Тактовая частота 0-16 МГц

23 порта ввода/вывода, объединенных в 3 группы:

* Порт В (PB0 - РВ7)
* Порт С (PC0 - РС6 : 7 выводов)
* Порт D (PD0 - PD7 : 8 выводов)



**Рисунок 1. Atmega8 в Proteus**

Предварительная настройка направления и значений портов микроконтроллера:

// Устаналиваем PORTB на режим чтения

DDRB = 0x00;

PORTB = 0x00;

// Устаналиваем PORTС на режим записи

DDRC = 0xFF;

PORTC = 0x00;

// Создаем общий таймер CNT

int cnt = 0;

// Создаем I, заданные по варианту

int i1 = 0, i2 = 0, i3 = 0, i4 = 0;

Моделирование функций:

// подаем значения на "лапки" порта через битовый сдвиг влево

int inp = PINB;

i1 = inp & (1<<PB0);

i2 = inp & (1<<PB1);

i3 = inp & (1<<PB2);

i4 = inp & (1<<PB3);

// создаем таймеры, заданные по вараинту

int t1 = 0, t2 = 0, t3 = 0;

// получаем значения через остаток от деления

t1 = (cnt % 5 == 0 && cnt!= 0);

t2 = (cnt % 7 == 0 && cnt!= 0);

t3 = (cnt % 9 == 0 && cnt!= 0);

// значения функций, создание лампочек

int o1 = 0, o2 = 0, o3 = 0, o4 = 0;

if ((!o2 && !o3 && !o4) && t3) {

o1 = 1;

}

if ((i4 || o4 || i3) && t1) {

o2 = 1;

}

if ((i1 && i2) && t2) {

o3 = 1;

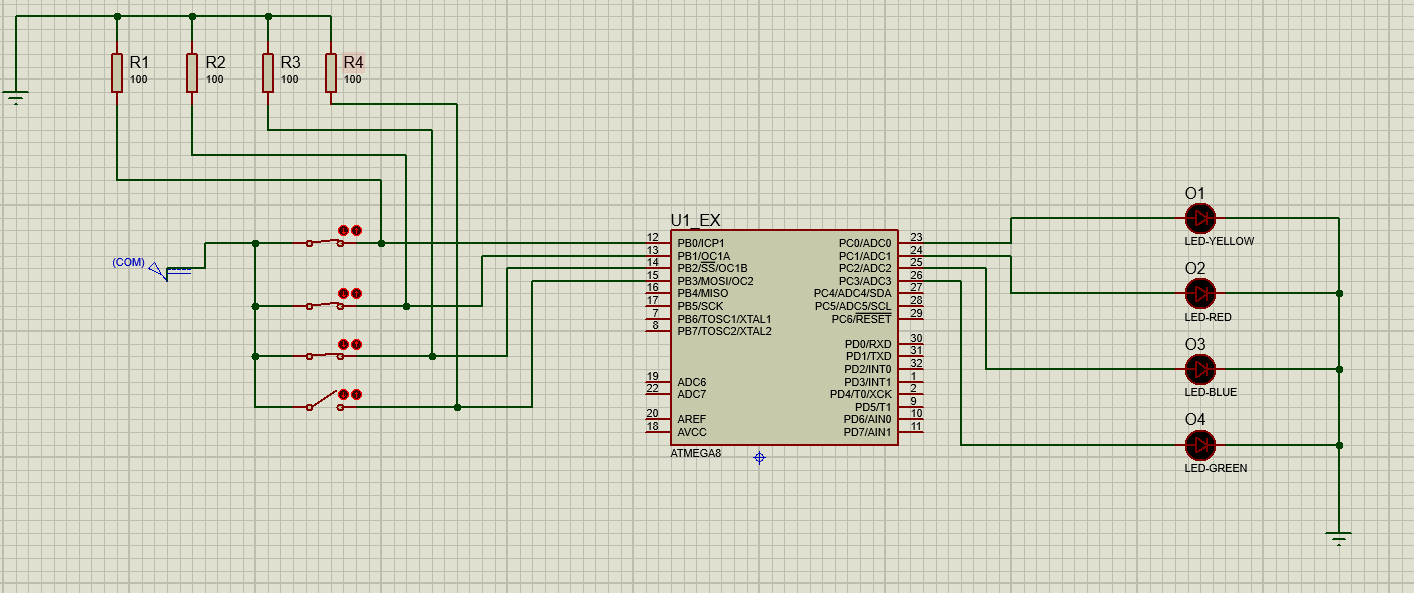
}

if (o3 && i3) {

o4 = 1;

}

Итоговая схема, реализующая функции, заданные вариантом (рис.2):



**Рисунок 2. Итоговая схема**

Входы:

|  |  |  |  |
| --- | --- | --- | --- |
| **I1** | **I2** | **I3** | **I4** |
| PBo | PB1 | PB2 | PB3 |

Выходы:

|  |  |  |  |
| --- | --- | --- | --- |
| **O1** | **O2** | **O3** | **O4** |
| O1 | O2 | O3 | O4 |

**Тестирование**

В результате тестирования было выявлено, что программа работает корректно.

**Вывод**

В ходе выполнения данной лабораторной работы, были получены базовые навыки в программировании микроконтроллеров семейства AVR.

**Список литературы**

1. Краткое описание микроконтроллера ATmega8 и назначение выводов [Электронный ресурс] http://iprg.ru/index.php?topic=129.0 (дата обращения: 12.11.2017)
2. Официальный сайт. [Электронный ресурс] <http://www.microchip.com> (дата обращения: 28.10.2017)

**Приложение. Исходный код**

#define *F\_CPU* 1000000

#include <avr/io.h>

#include <util/delay.h>

#include <stdlib.h>

#include <stdio.h>

#include <stdbool.h>

#include <time.h>

int main (void)

{

// Устаналиваем PORTB на режим чтения

DDRB = 0x00;

PORTB = 0x00;

// Устаналиваем PORTС на режим записи

DDRC = 0xFF;

PORTC = 0x00;

// Создаем общий таймер CNT

int cnt = 0;

// Создаем I, заданные по варианту

int i1 = 0, i2 = 0, i3 = 0, i4 = 0;

while(1) {

// каждый раз сбрасываем значение порта с выходами, иначе лампы будут гореть постоянно

PORTC = 0x00;

// подаем значения на "лапки" порта через битовый сдвиг влево

int inp = PINB;

i1 = inp & (1<<PB0);

i2 = inp & (1<<PB1);

i3 = inp & (1<<PB2);

i4 = inp & (1<<PB3);

// создаем таймеры, заданные по вараинту

int t1 = 0, t2 = 0, t3 = 0;

// получаем значения через остаток от деления

t1 = (cnt % 5 == 0 && cnt!= 0);

t2 = (cnt % 7 == 0 && cnt!= 0);

t3 = (cnt % 9 == 0 && cnt!= 0);

// значения функций, создание лампочек

int o1 = 0, o2 = 0, o3 = 0, o4 = 0;

if ((!o2 && !o3 && !o4) && t3) {

o1 = 1;

}

if ((i4 || o4 || i3) && t1) {

o2 = 1;

}

if ((i1 && i2) && t2) {

o3 = 1;

}

if (o3 && i3) {

o4 = 1;

}

// вывод значения на лампочки

if(o1)

PORTC = PORTC | (1<<PC0);

if(o2)

PORTC = PORTC | (1<<PC1);

if(o3)

PORTC = PORTC | (1<<PC2);

if(o4)

PORTC = PORTC | (1<<PC3);

// увеличение таймера и защита от переполнения

cnt++;

cnt %= 12;

*\_delay\_ms*(1000);

}

}