|  |  |
| --- | --- |
|  | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ **Информатика и системы управления**

КАФЕДРА **Компьютерные системы и сети (ИУ6)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.03.01 Информатика и вычислительная техника**

**Отчет**

|  |  |
| --- | --- |
| **по лабораторной работе №** | 7 |

**Название:** Программирование и отладка программ на языке Си

для микроконтроллеров AVR.

**Дисциплина:** Микропроцессорные системы.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ИУ6-62Б |  |  | Ашуров Д. Н. Марчук И. С. |
|  | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |
| Преподаватель |  |  |  |  |
|  |  |  | (Подпись, дата) | (И.О. Фамилия) |

Москва, 2022

**Вариант 2.**

**Цели работы:**

* изучение типовых инструкций Cи для настройки ресурсов микроконтроллеров AVR;
* знакомство с встроенным в AVR Studio 4 компилятором AVR GCC;
* отладка, модификация и прогон тестовых программ.

**Ход работы.**

**Задание 1**

Запустив AVR Studio 4, создать в рабочей папке проект. В окно программы ввести программу на языке Си для последовательного переключения светодиодов STK500.

Изменить последовательность переключения светодиодов, от старшего разряда к младшему с временем включения каждого светодиода, равным 2 с.

Код измененной программы приведен в листинге 1. Схема для проверки корректности программы приведена на рисунке 1.

Листинг 1 – программа переключения светодиодов

#include <avr/interrupt.h>

#include <avr/io.h>

#define xtal 3686400

#define fled 0.5

unsigned char led\_status=0x7f;

ISR(TIMER1\_OVF\_vect)

{

TCNT1=0x10000-(xtal/1024/fled);

led\_status>>=1;

led\_status|=0x80;

if (led\_status==0xff) led\_status=0x7f;

PORTC=led\_status;

}

int main(void)

{

DDRC=0xff;

PORTC=led\_status;

TCCR1A=0;

TCCR1B=5;

TCNT1=0x10000-(xtal/1024/fled);

TIFR=0;

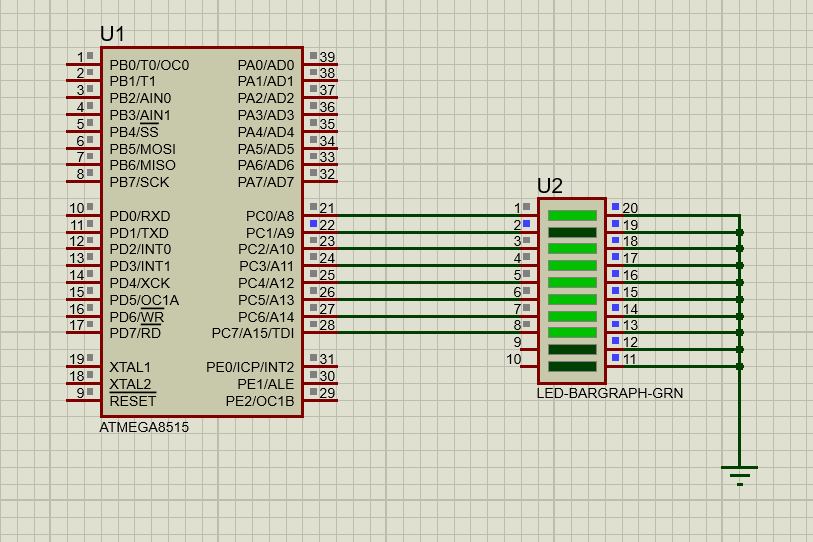
TIMSK=0x80;

GICR=0;

sei();

while (1);

}



**Рисунок 1 – схема в Proteus**

**Задание 2**

Разработаем микроконтроллерное устройство, управляющее двумя светодиодами, один из которых показывает готовность к работе, второй переключается по числу нажатий кнопки управления.

Исходный код заданной программы приведен в листинге 2.

Листинг 2 – исходный код программы с обработкой прерываний

#include <avr/interrupt.h>

#include <avr/io.h>

#include <util/delay.h>

// Обработки внешнего прерывания INT0

ISR(INT0\_vect)

{ char timer; // локальная переменная

timer = TCNT0;

if (timer != 0)

{TCNT0 = 0; // сброс таймера/счётчика

PORTB |= (1<<PB6); //PORTB=0b11000001 (выключаем светодиод LED6)

do {

PORTB &= ~(1<<PB7);//PORTB=0b01000001 (включаем светодиод LED7)

\_delay\_ms(750); // задержка 750 мс

PORTB |= (1<<PB7); //PORTB=0b11000001 (выключаем светодиод LED7)

\_delay\_ms(750);

} while (--timer != 0);

PORTB &= ~(1<<PB6); //PORTB=0b10000001 (включаем светодиод LED6)

}

}

int main(void)

{

// Инициализация портов

DDRB=0xC0; // PB7,PB6 для вывода на LED7,LED6 PB0- для ввода

PORTB=0b10000001; // выключаем LED7, PB0-подтягивающий резистор кнопки

DDRD=0;

PORTD=(1<<PD2); // PD2-подтягивающий резистор

// Инициализация таймера 0

TCCR0=0x06;

TCNT0=0x00;

GICR=(1<<INT0); // инициализация прерывания INT0 в GIСR (или GIMSK)

MCUCR=(1<<SE); // разрешение перехода в режим Idle

sei(); // глобальное разрешение прерываний

for (;;) {

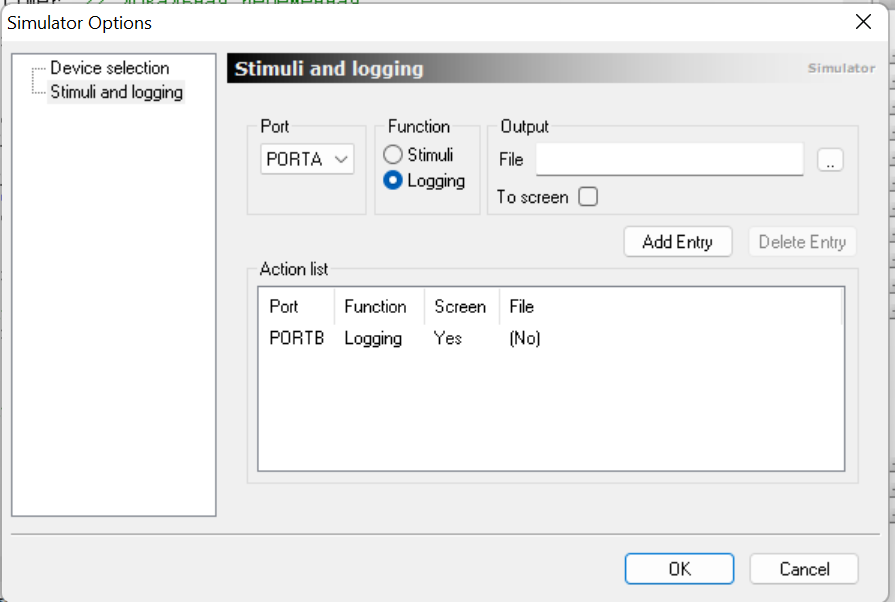
asm("sleep"); // переход в режим Idle

asm("nop");

}

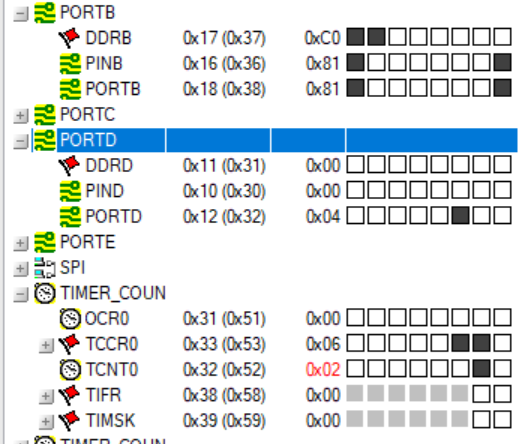
}

Для отладки программы добавили логирование значения PORTB на экран (рисунок 2).



**Рисунок 2 – настройка логирования порта В**

После этого занесли в TCNT0 значение 2, таким образом смоделировав 2 нажатия на SW0, занесли в PIND.2 значение 0 (кнопка нажата) и в PINB.0 значение 1 (кнопка не нажата). Состояние портов и таймера приведено на рисунке 3.



**Рисунок 3 – состояние портов и таймера**

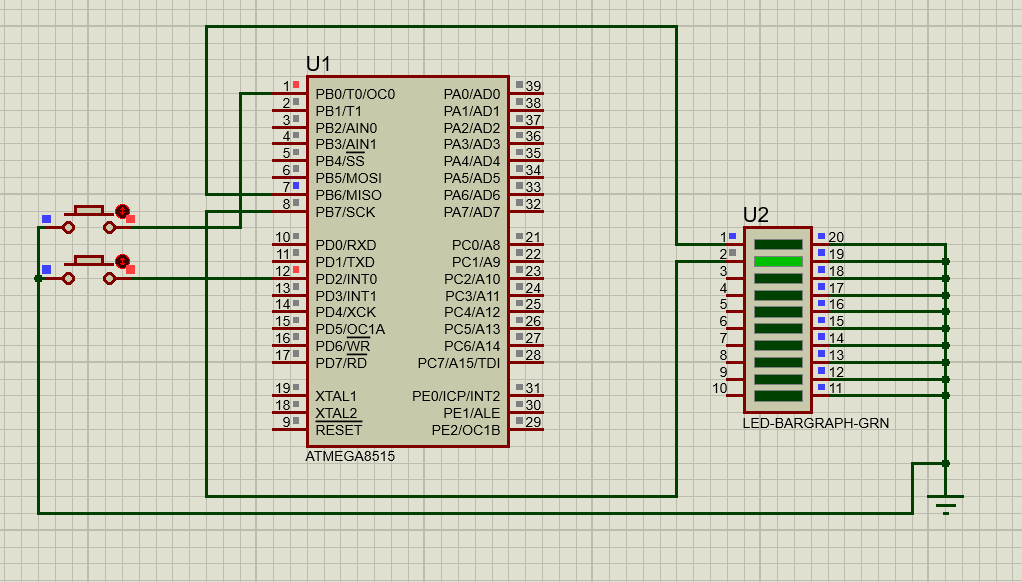
После этого просмотрим лог порта В (рисунок 4). Как можно заметить, в PORTB последовательно заносились разные значения: С1, 41 и 81. При этом С1 – выключены оба светодиода, 41 – LED7 включен, а LED6 выключен, 81 – LED7 выключен, а LED6 включен.

Задержка занимает примерно 3000060 тактов. T = N/f = 3000060 / 3686400 = 813 мc, что примерно соответствует заданной в программе задержке в 750 мс.

Изображение выглядит как текст

Автоматически созданное описание

**Рисунок 4 – лог порта В**



**Рисунок 5 – схема в Proteus**

**Задание 3.**

Провести исследование программы настройки микросхемы параллельного интерфейса 8255A.

Исходный код программы приведен в листинге 3.

Листинг 3 – программа настройки микросхемы параллельного интерфейса 8255A

#include <avr/io.h>

#define uchar unsigned char

// Определения уровней сигналов бита (х) порта

#define sbit(x,PORT) ((PORT) |= (1<<x))

#define cbit(x,PORT) ((PORT) &= ~(1<<x))

// определения интерфейсных сигналов

// RST,LE,CS,RD,WR

#define srst sbit(0,PORTD)

#define crst cbit(0,PORTD)

#define sle sbit(1,PORTD)

#define cle cbit(1,PORTD)

#define scs sbit(2,PORTD)

#define ccs cbit(2,PORTD)

#define srd sbit(3,PORTD)

#define crd cbit(3,PORTD)

#define swr sbit(4,PORTD)

#define cwr cbit(4,PORTD)

// Вывод адресов/данных, ввод данных

#define out PORTC

#define in PINC

// 'Защёлкивание' адреса в регистре

void latch\_it(void)

{ cle;

asm("nop");

sle;

asm("nop");

cle;

}

int main()

{ uchar temp;

SPL = 0x54;

SPH = 0x04;

// Инициализация порта PD

DDRD = 0xff;

// Неактивные входы 8255A

srd;

swr;

scs;

// Разрешение 8255A и сброс

ccs;

srst;

asm("nop"); asm("nop"); asm("nop");

crst;

//

DDRC=0xff;

out = 0x03; //адрес регистра управления 8255А

latch\_it();

out = 0x82;

cwr;

asm("nop");

swr;

while(1)

{ //

DDRC=0xff;

out = 0x01; //адрес на вывод

latch\_it();

DDRC = 0; //KEY на ввод

crd;

asm("nop");

temp = in; //данные KEY

srd;

//

DDRC = 0xff;

out = 0x00; //адрес на вывод

latch\_it();

out = temp; //данные на LED

cwr;

asm("nop");

swr;

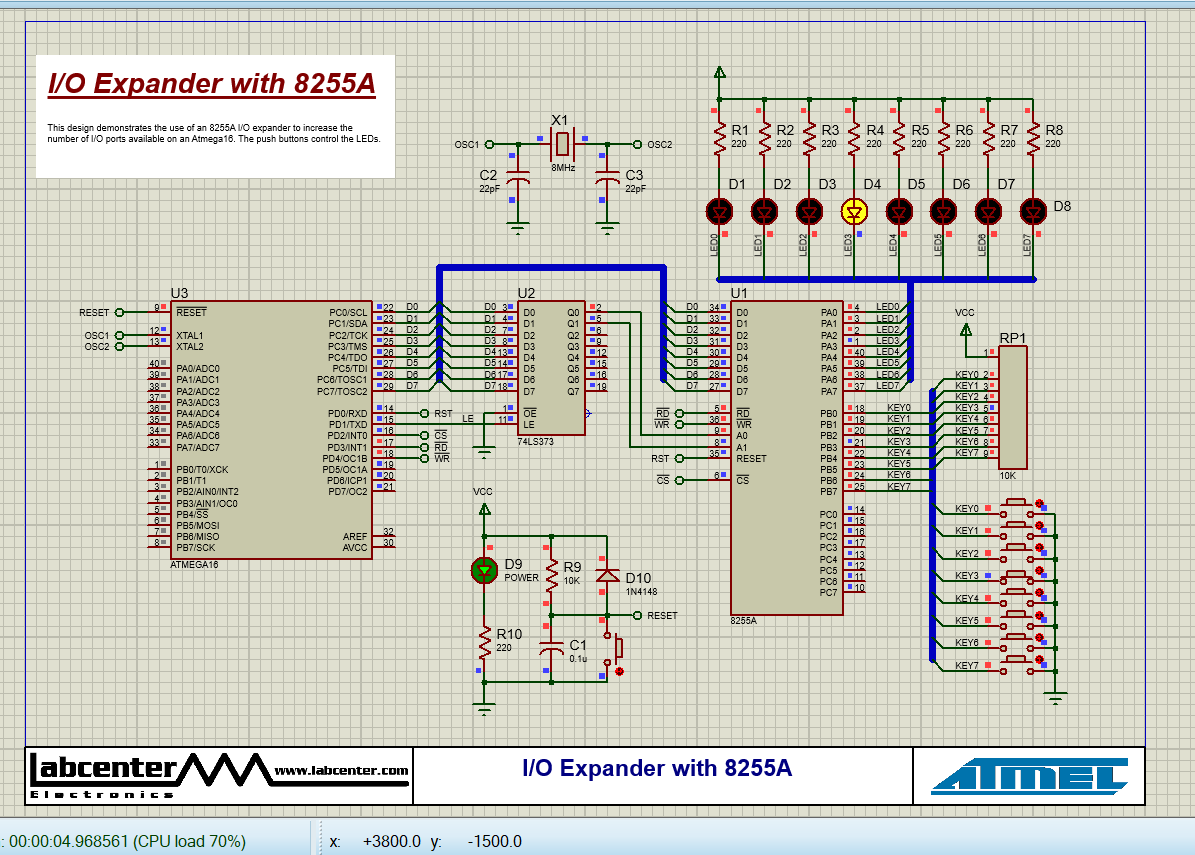
}

}

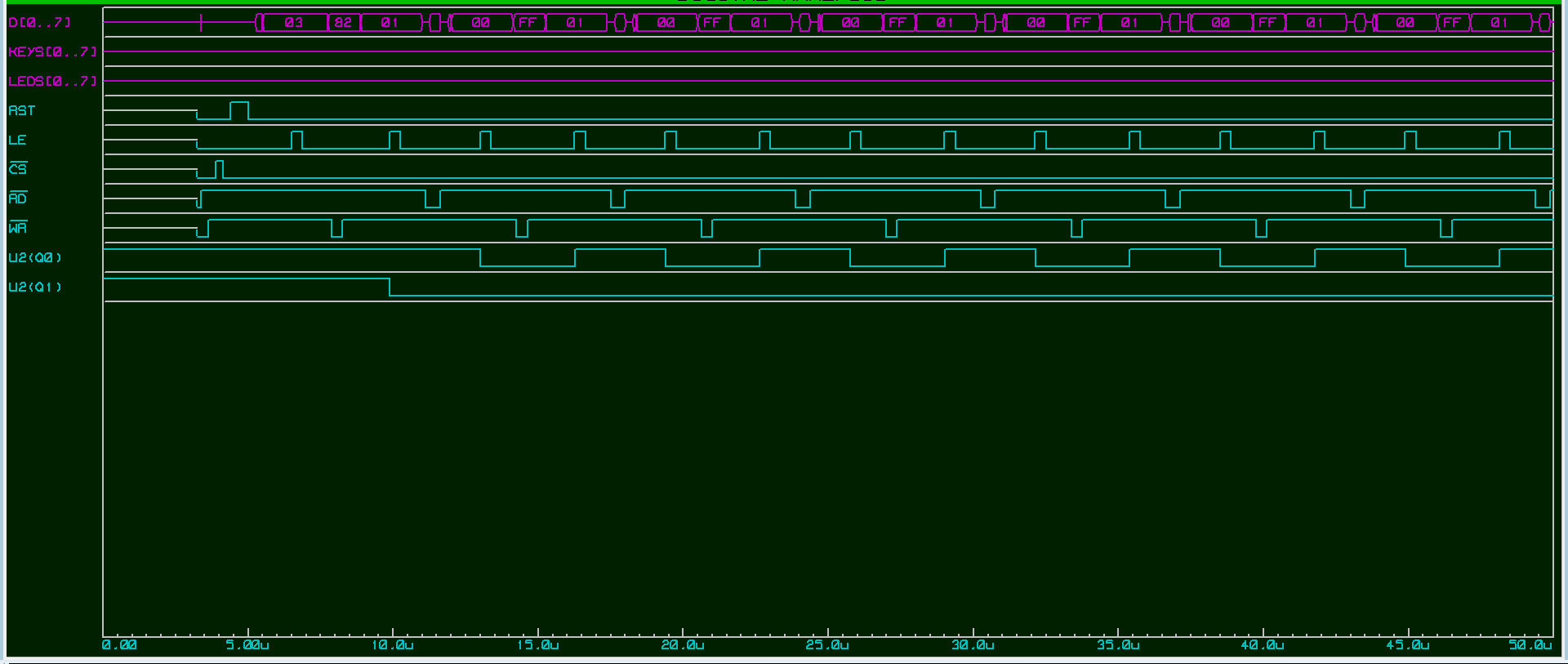
Программа работает по следующему циклу:

* защелкивание адреса 0x01 (отвечает за выбор порта В на 8255А);
* установка порта С микроконтроллера в режим считывания;
* cчитывание значений с кнопок;
* установка порта С микроконтроллера в режим вывода;
* защелкивание адреса 0x00 (отвечает за выбор порта А на 8255А);
* вывод значений на светодиоды.

Схема в Proteus и временная диаграмма работы интерфейса приведены на рисунках 6 и 7 соответственно.



**Рисунок 6 – схема в Proteus**



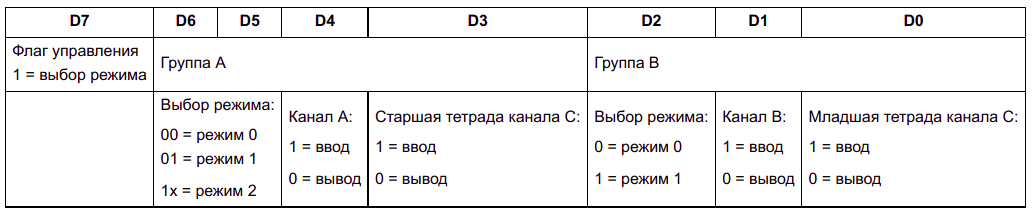
**Рисунок 7 – временная диаграмма**

На рисунке 7 адрес 0х03 указывает на регистр управления 8255А, 0х82 – управляющее слово.

0х82 = 1000 00102. Старший бит значит, что управляющее слово отвечает за выбор режима работы, 6й и 5й биты устанавливают режим стандартного ввода-вывода для PORTA. Ноль в 4ом разряде – PORTA назначен на вывод. Ноль во 2ом и единица в 1ом - режим простого ввода-вывода и настройка PORTB на ввод соответственно.

0й и 3й биты отвечают за PORTC.

Формат управляющего слова представлен на рисунке 9.



**Рисунок 9 – формат управляющего слова**

**Вывод:** в результате выполнения данной лабораторной работы были изучены типовые инструкций Cи для настройки ресурсов микроконтроллеров AVR, способы отладки. Также изучено взаимодействие микроконтроллера AVR со схемой параллельного интерфейса 8255A, а именно ввод данных с кнопок и вывод данных на светодиоды путем загрузки управляющего слова в регистр управления 8255A.