



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

О Т Ч Е Т

по лабораторной работе № 7

Название: Программирование и отладка программ на языке Си
для микроконтроллеров AVR.

Дисциплина: Микропроцессорные системы.

Студент

ИУ6-62Б

(Группа)

(Подпись, дата)

С.В. Астахов, Д.И. Вариханов

(И.О. Фамилия)

Преподаватель

(Подпись, дата)

(И.О. Фамилия)

Москва, 2022

Вариант 1.

Цели работы:

- изучение типовых инструкций Си 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);
}
```

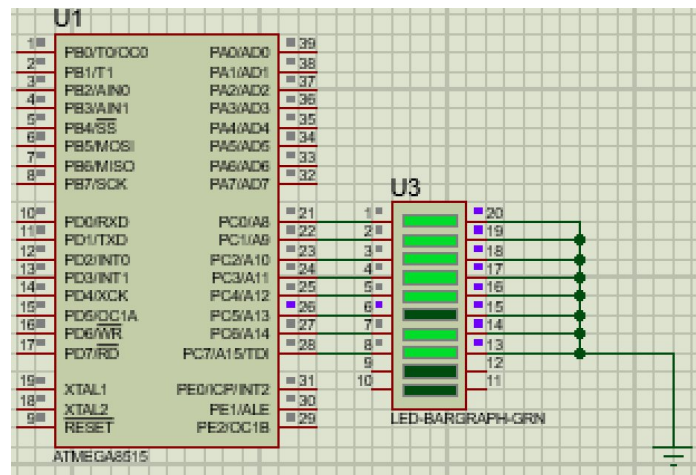


Рисунок 1 – схема в Proteus

Задание 2

Разработаем микроконтроллерное устройство, управляющее 2-я 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(300); // задержка 300 мс
     PORTB |= (1<<PB7); //PORTB=0b11000001 (выключаем светодиод LED7)
     _delay_ms(300);
   } 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 GICR (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).

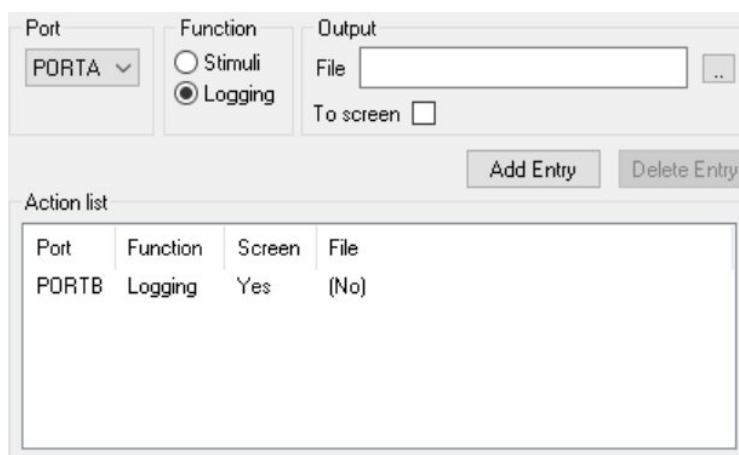


Рисунок 2 – настройка логирования порта B

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.

PORTB			
DDRB	0x17 (0x37)	0xC0	<div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div>
PINB	0x16 (0x36)	0x81	<div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div>
PORTB	0x18 (0x38)	0x81	<div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div>
PORTC			
PORTD			
PORTE			
SPI			
TIMER_COUNT			
OCR0	0x31 (0x51)	0x00	<div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div>
TCCR0	0x33 (0x53)	0x06	<div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div>
TCNT0	0x32 (0x52)	0x02	<div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div>
TIFR	0x38 (0x58)	0x00	<div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div>
TIMSK	0x39 (0x59)	0x00	<div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div>

Рисунок 3 – состояние портов и таймера

После этого посмотрим лог порта В (рисунок 4). Как можно заметить, в PORTB последовательно заносятся разные значения: C1, 41 и 81. При этом C1 – выключены оба светодиода, 41 – LED7 включен, а LED6 выключен, 81 – LED7 выключен, а LED6 включен.

Задержка занимает примерно 1212000 тактов. $T = N/f = 1212000 / 3686400 = 0.329$ с, что примерно соответствует заданной в программе задержке в 300 мс.

```
AVR Simulator: PORTB - 00000000:C1
AVR Simulator: PORTB - 00000015:81
AVR Simulator: PORTB - 00000179:C1
AVR Simulator: PORTB - 00000183:41
AVR Simulator: PORTB - 001212186:C1
AVR Simulator: PORTB - 002424192:41
AVR Simulator: PORTB - 003636195:C1
AVR Simulator: PORTB - 004848200:81
```

Рисунок 4 – лог порта В

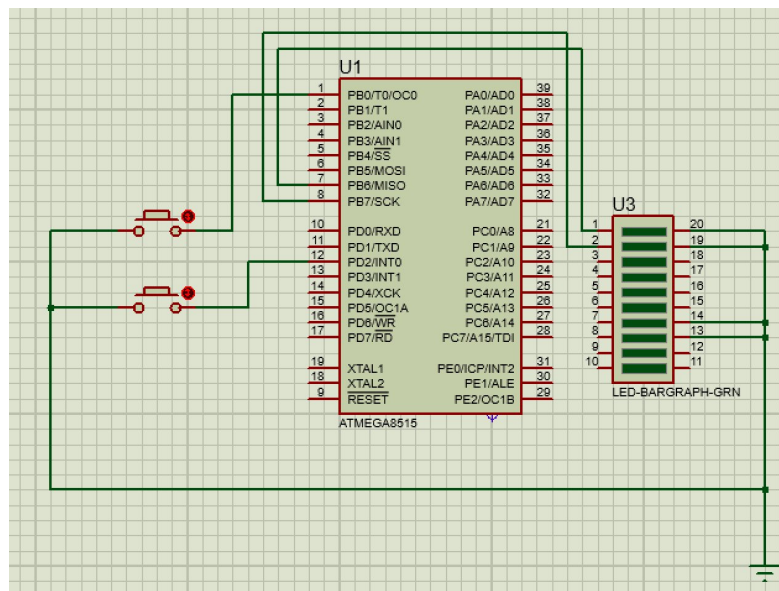


Рисунок 5 – схема в Proteus

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 (x) 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 8255A
  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();
  }
}

```




Рисунок 7 – инициализация микросхемы

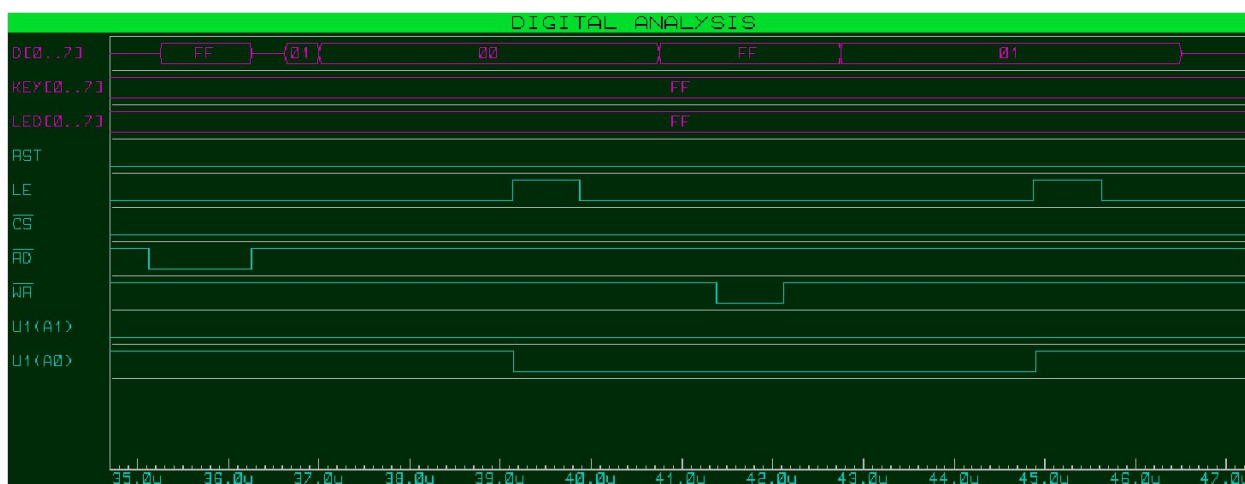


Рисунок 8 – временная диаграмма

На рисунке 7 адрес 0x03 указывает на регистр управления 8255A, 0x82 – управляющее слово.

$0x82 = 1000\ 0010_2$. Старший бит значит, что управляющее слово отвечает за выбор режима работы, 6^й и 5^й биты устанавливают режим стандартного ввода-вывода для PORTA. Ноль в 4^{ом} разряде – PORTA назначен на ввод. Ноль во 2^{ом} и единица в 1^{ом} - режим простого ввода-вывода и настройка PORTB на вывод соответственно.

0^й и 3^й биты отвечают за PORTC.

Формат управляющего слова представлен на рисунке 9.

D7	D6	D5	D4	D3	D2	D1	D0
Флаг управления 1 = выбор режима	Группа А				Группа В		
	Выбор режима: 00 = режим 0 01 = режим 1 1x = режим 2	Канал А: 1 = ввод 0 = вывод	Старшая тетрада канала С: 1 = ввод 0 = вывод		Выбор режима: 0 = режим 0 1 = режим 1	Канал В: 1 = ввод 0 = вывод	Младшая тетрада канала С: 1 = ввод 0 = вывод

Рисунок 9 – формат управляющего слова

Вывод: в результате выполнения данной лабораторной работы были получены навыки написания и отладки программ для микроконтроллеров AVR на языке Си. Кроме того, был изучен принцип работы самой схемы параллельного интерфейса 8255А.