

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ПОВОЛЖСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ
УНИВЕРСИТЕТ»

Кафедра ИиСП

Отчет
по лабораторной работе № 7

по дисциплине «Машинно-зависимые языки программирования»
Вариант 1

Выполнил: ст. гр. ПС-11

Маркин И. А.

Проверил: доцент, доцент
кафедры ИиСП Баев А.А.

г. Йошкар-Ола

2025

Цель работы:

Научиться работать с АЦП и UART, прочитав документацию перед этим

Задания на лабораторную работу:

1. Написать код и собрать схему(Работа с АЦП, Работа с UART)
2. Оптимизация основного кода

Теоретические сведения

ПРИМЕНЕНИЕ МИКРОКОНТРОЛЛЕРОВ В РАДИОТЕХНИЧЕСКИХ И БИОМЕДИЦИНСКИХ СИСТЕМАХ

2. Практическая часть

1) Написать код и собрать схему (Работа с АЦП, Работа с UART)

1. Работа с АЦП

Код на C:

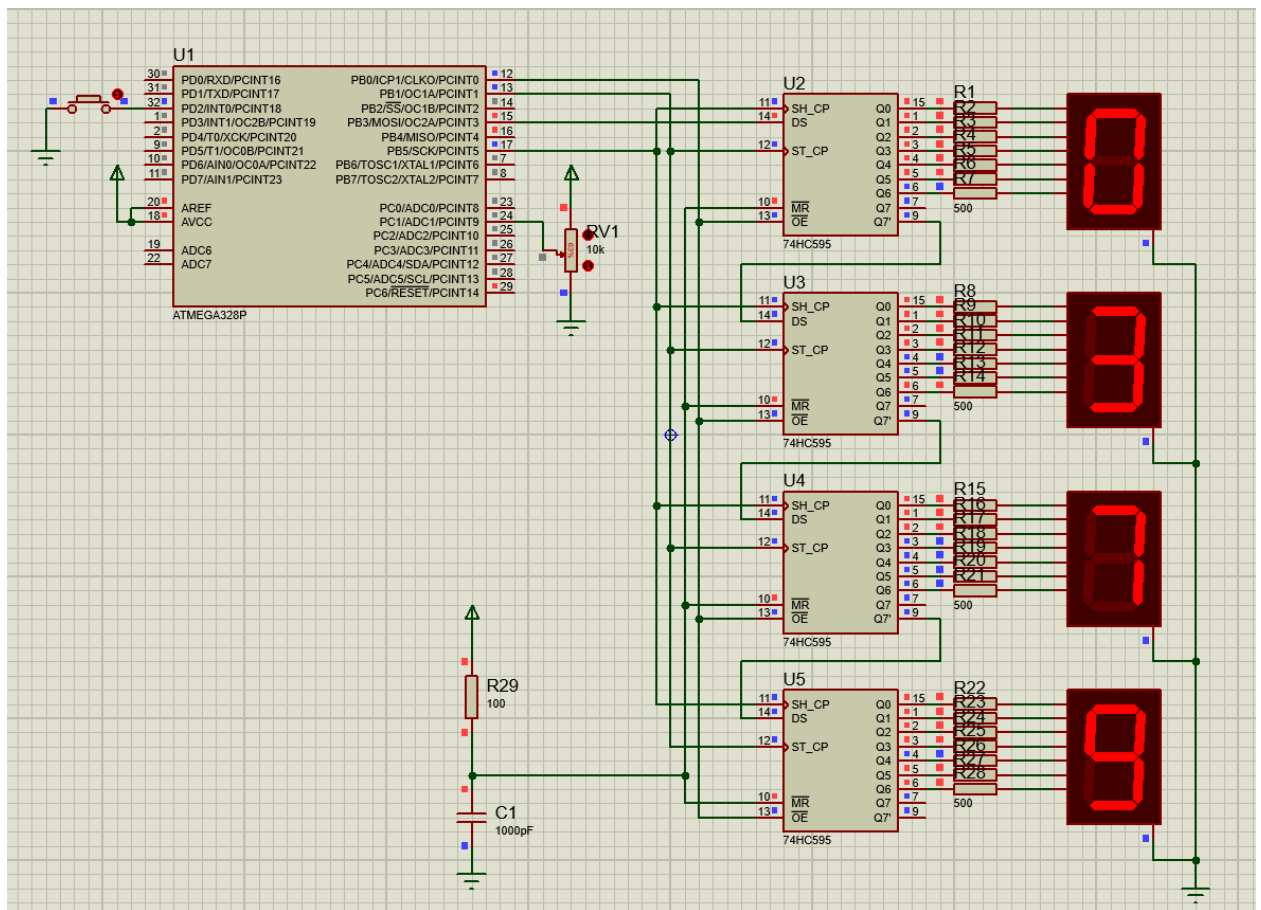
```
#define F_CPU 1000000UL
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>
uint8_t segments[]={
    // GFEDCBA
    0b00111111, // 0 - A, B, C, D, E, F
    0b00000110, // 1 - B, C
    0b01011011, // 2 - A, B, D, E, G
    0b01001111, // 3 - A, B, C, D, G
    0b01100110, // 4 - B, C, F, G
    0b01101101, // 5 - A, C, D, F, G
    0b01111101, // 6 - A, C, D, E, F, G
    0b00000111, // 7 - A, B, C
    0b01111111, // 8 - A, B, C, D, E, F, G
    0b01101111, // 9 - A, B, C, D, F, G
};
void InitPorts(void);
void InitTimer1(void);
void Bin2Dec(uint16_t data);
void SendData(uint8_t data);
void DisplayData(uint16_t data);
void InitSPI(void);
void InitADC(void);
volatile uint8_t bcd_buffer[] = {0,0,0,0};
volatile uint16_t display_val = 0;
int main(void){
    InitPorts();
    InitSPI();
    InitTimer1();
    EIMSK |= (1<<INT0); //Enable INT0
    EICRA |= (1<<ISC01); //Trigger on falling edge of INT0
    InitADC();
    sei(); //global interrupt enable
    PORTB &= ~(1<<PB0); //OE = low (active)
    DisplayData(0);
    while(1){
        DisplayData(display_val);
    }
}
//-----
ISR(TIMER1_COMPB_vect){
}
ISR(INT0_vect){
}
ISR(ADC_vect){
```

```

        display_val=ADC;
    }
    //-----
    void InitPorts(void){
        DDRB = (1<<PB0|1<<PB1|1<<PB3|1<<PB5);
        DDRD = (0<<PD2);
        PORTD |= (1<<PD2);
    }
    void InitTimer1(void){
        TCCR1A = 0;
        TCCR1B = (1<<CS11|1<<CS10|1<<WGM12);
        TCNT1 = 0;
        TIMSK1 |= (1<<OCIE1B);
        OCR1A = 1562;
        OCR1B = 1562;
    }
    void Bin2Dec(uint16_t data){
        bcd_buffer[3] = (uint8_t)(data/1000);
        data = data % 1000;
        bcd_buffer[2] = (uint8_t)(data/100);
        data = data % 100;
        bcd_buffer[1] = (uint8_t)(data/10);
        data = data % 10;
        bcd_buffer[0] = (uint8_t)(data);
    }
    void SendData(uint8_t data){
        SPDR = data;
        while(!(SPSR & (1<<SPIF)));
    }
    void DisplayData(uint16_t data){
        Bin2Dec(data);
        PORTB &= ~(1<<PB1); //clk_out = 0
        SendData(segments[bcd_buffer[0]]);
        SendData(segments[bcd_buffer[1]]);
        SendData(segments[bcd_buffer[2]]);
        SendData(segments[bcd_buffer[3]]);
        PORTB |= (1<<PB1); //clk_out = 1
    }
    void InitSPI(void){
        DDRB |= (1<<PB3|1<<PB5); //configure MOSI and CLK as out
        SPSR |= (1<<SPI2X); //Fclk = Fosc/2
        SPCR = (1<<SPE|1<<MSTR); //SPI enable, master mode,
        PORTB &= ~(1<<PB3|1<<PB5); //init values - DAT low, CLK low
    }
    void InitADC(void){
        ADMUX = (1<<MUX0); //Align left, ADC1
        ADCSRB = (1<<ADTS2|1<<ADTS0); //Start on Timer1 COMPB
        //Enable, auto update, IRQ enable
        ADCSRA = (1<<ADEN|1<<ADSC|1<<ADIFSC);
    }
}

```

Cxema:



2. Работа с UART

Код на C:

```
#define F_CPU 1000000UL
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>
uint8_t segments[] = {
    // GFEDCBA
    0b00111111, // 0 - A, B, C, D, E, F
    0b00000110, // 1 - B, C
    0b01011011, // 2 - A, B, D, E, G
    0b01001111, // 3 - A, B, C, D, G
    0b01100110, // 4 - B, C, F, G
    0b01101101, // 5 - A, C, D, F, G
    0b01111101, // 6 - A, C, D, E, F, G
    0b00000111, // 7 - A, B, C
    0b01111111, // 8 - A, B, C, D, E, F, G
    0b01101111, // 9 - A, B, C, D, F, G
};

void InitPorts(void);
void InitTimer1(void);
void Bin2Dec(uint16_t data);
void SendData(uint8_t data);
void DisplayData(uint16_t data);
void InitSPI(void);
void InitADC(void);
```

```

void InitUSART(void);
void SendChar(char symbol);
void SendString(char * buffer);
volatile uint8_t bcd_buffer[] = {0,0,0,0};
volatile uint16_t display_val = 0;
int main(void)
{
    InitPorts();
    InitSPI();
    InitTimer1();
    EIMSK |= (1<<INT0); //Enable INT0
    EICRA |= (1<<ISC01); //Trigger on falling edge of INT0
    InitADC();
    InitUSART();
    sei(); //global interrupt enable
    PORTB &= ~(1<<PB0); //OE = low (active)
    DisplayData(0);
    SendString("Hello\r\n");
    while(1)
    {
        DisplayData(display_val);
    }
}
//-----
ISR(TIMER1_COMPB_vect){}
ISR(INT0_vect){
    SendString("Value = ");
    SendChar(0x30 + bcd_buffer[3]);
    SendChar(0x30 + bcd_buffer[2]);
    SendChar(0x30 + bcd_buffer[1]);
    SendChar(0x30 + bcd_buffer[0]);
    SendString("\r\n");
}
ISR(ADC_vect){
    display_val = ADC;
}
ISR(USART_RX_vect){
    if(UDR0 == 0x20){
        SendString("Roger that\r\n");
    }
}
//-----
void InitPorts(void){
    DDRB = (1<<PB0 | 1<<PB1 | 1<<PB3 |
1<<PB5);
    DDRD = (0<<PD2);
    PORTD |= (1<<PD2);
}
void InitTimer1( void){
    TCCR1A = 0; //CTC mode - Clear Timer on Compare
    //prescaler = sys_clk/64

```

```

    TCCR1B = (1<<CS11 | 1<<CS10 | 1<<WGM12);
    TCNT1 = 0; //start value of counter
    TIMSK1 |= (1<<OCIE1B);
    OCR1A = 1562;
    OCR1B = 1562;
}
void Bin2Dec(uint16_t data){
    bcd_buffer[3] = (uint8_t)(data/1000);
    data = data % 1000;
    bcd_buffer[2] = (uint8_t)(data/100);
    data = data % 100;
    bcd_buffer[1] = (uint8_t)(data/10);
    data = data % 10;
    bcd_buffer[0] = (uint8_t)(data);
}
void SendData (uint8_t data){
    SPDR = data;
    while(!(SPSR & (1<<SPIF)));
}
void DisplayData (uint16_t data){
    Bin2Dec(data);
    PORTB &= ~(1<<PB1); //clk_out = 0
    SendData(segments[bcd_buffer[0]]);
    SendData(segments[bcd_buffer[1]]);
    SendData(segments[bcd_buffer[2]]);
    SendData(segments[bcd_buffer[3]]);
    PORTB |= (1<<PB1); //clk_out = 1
}
void InitSPI( void){
    DDRB |= (1<<PB3 | 1<<PB5); //configure MOSI
    and CLK as out
    SPSR |= (1<<SPI2X); //Fclk = Fosc/2
    //SPI enable, master mode, MSB first,
    CPOL=0, CPHA=0
    SPCR = (1<<SPE | 1<<MSTR);
    //init values - DAT low, CLK low
    PORTB &= ~(1<<PB3 | 1<<PB5); }
void InitADC( void){
    ADMUX = (1<<MUX0); //Align left, ADC1
    ADCSRB = (1<<ADTS2 | 1<<ADTS0); //Start on Timer1 COMPB
    //Enable, auto update, IRQ enable
    ADCSRA = (1<<ADEN | 1<<ADSC | 1<<ADIF);
}
void InitUSART(){
    UCSRB = (1<<RXEN0 | 1<<TXEN0 |
    1<<RXCIE0);
    UCSRC = (1<<UCSZ01 | 1<<UCSZ00);
    UBRRH = 0;
    UBRR0L = 0x0C;
}
void SendChar(char symbol){

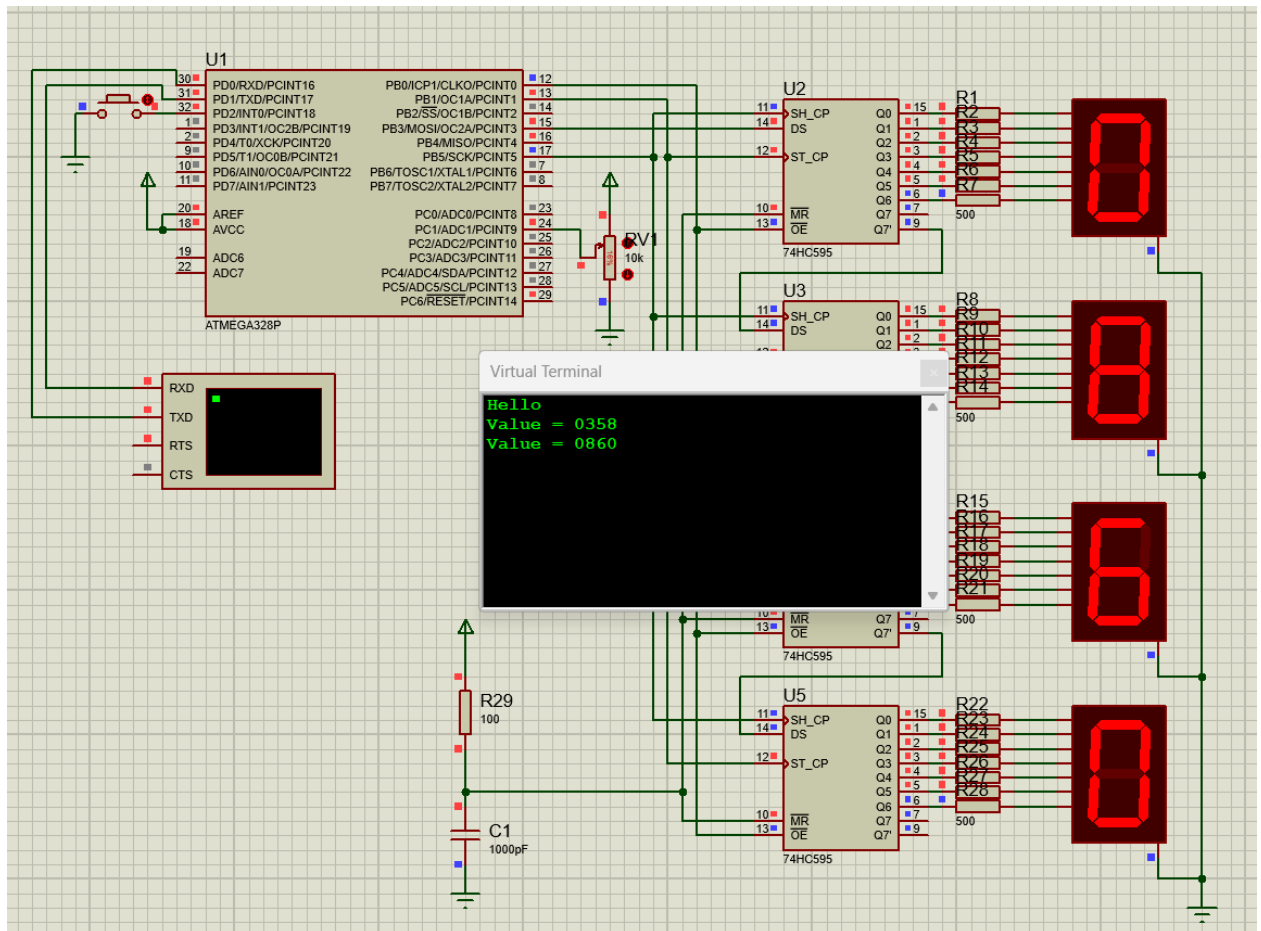
```



```

while (!(UCSR0A & (1<<UDRE0)));
UDR0 = symbol;
}
void SendString(char * buffer){
while(*buffer != 0){
SendChar(*buffer++);
}
}
}

```



2. Оптимизация основного кода

1. Реализация динамической индикации

Код на C:

```

#define F_CPU 16000000UL
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>
uint8_t segments[] =
{
    0b00111111,
    0b00000110,
    0b01011011,

```

```

        0b01001111,
        0b01100110,
        0b01101101,
        0b01111101,
        0b00000111,
        0b01111111,
        0b01101111,
    };
    volatile uint8_t bcd_buffer[4] = {0};
    volatile uint16_t display_val = 0;
    void InitPorts(void)
    {
        DDRB = (1 << PINB0 | 1 << PINB1 | 1 << PINB3 | 1 << PINB5);
        DDRD = (0 << PIND2);
        PORTD |= (1 << PIND2);
    }
    void InitTimer1(void)
    {
        TCCR1A = 0;
        TCCR1B = (1 << CS11 | 1 << CS10 | 1 << WGM12);
        TCNT1 = 0;
        TIMSK1 |= (1 << OCIE1B);
        OCR1A = 1562;
        OCR1B = 1562;
    }
    void Bin2Dec(uint16_t data)
    {
        bcd_buffer[3] = data / 1000;
        bcd_buffer[2] = (data / 100) % 10;
        bcd_buffer[1] = (data / 10) % 10;
        bcd_buffer[0] = data % 10;
    }
    void SendData(uint8_t data)
    {
        SPDR = data;
        while(!(SPSR & (1 << SPIF)));
    }
    void DisplayData(uint16_t data)
    {
        Bin2Dec(data);
        PORTB &= ~(1 << PINB1);
        SendData(segments[bcd_buffer[0]]);
        SendData(segments[bcd_buffer[1]]);
        SendData(segments[bcd_buffer[2]]);
        SendData(segments[bcd_buffer[3]]);
        PORTB |= (1 << PINB1);
    }
    void InitSPI(void)
    {
        DDRB |= (1 << PINB3 | 1 << PINB5);
        SPSR |= (1 << SPI2X);
    }

```

```

        SPCR = (1 << SPE | 1 << MSTR);
        PORTB &= ~(1 << PINB3 | 1 << PINB5);
    }
    void InitADC(void)
    {
        ADMUX = (1 << MUX0);
        ADCSRB = (1 << ADTS2 | 1 << ADTS0);
        ADCSRA = (1 << ADEN | 1 << ADSC | 1 << ADIF);
    }
    int main(void)
    {
        InitPorts();
        InitSPI();
        InitTimer1();
        EIMSK |= (1 << INT0);
        EICRA |= (1 << ISC01);
        InitADC();
        sei();
        PORTB &= ~(1 << PINB0);
        DisplayData(0);
        while(1)
        {
            DisplayData(display_val);
        }
    }
    ISR(ADC_vect)
    {
        display_val=ADC;
    }
}

```

2. Работа с UART

Код на C:

```

#define F_CPU 16000000UL
#include <avr/io.h>
#include <avr/interrupt.h>
uint8_t segments[] =
{
    0b00111111,
    0b00000110,
    0b01011011,
    0b01001111,
    0b01100110,
    0b01101101,
    0b01111011,
    0b0111101,
    0b00000111,
    0b01111111,
    0b01101111
};
volatile uint8_t bcd_buffer[4] = {0};
volatile uint16_t display_val = 0;
void InitPorts(void)

```

```

{
    DDRB = (1 << PINB0) | (1 << PINB1) | (1 << PINB3) | (1 << PINB5);
    DDRD &= ~(1 << PIND2);
    PORTD |= (1 << PIND2);
}
void InitTimer1(void)
{
    TCCR1A = 0;
    TCCR1B = (1 << WGM12) | (1 << CS11) | (1 << CS10);
    OCR1A = 1562;
    TIMSK1 |= (1 << OCIE1B);
}
void Bin2Dec(uint16_t data)
{
    bcd_buffer[3] = data / 1000;
    bcd_buffer[2] = (data / 100) % 10;
    bcd_buffer[1] = (data / 10) % 10;
    bcd_buffer[0] = data % 10;
}
void SendData(uint8_t data)
{
    SPDR = data;
    while(!(SPSR & (1 << SPIF)));
}
void DisplayData(uint16_t data)
{
    Bin2Dec(data);
    PORTB &= ~(1 << PINB1);
    SendData(segments[bcd_buffer[0]]);
    SendData(segments[bcd_buffer[1]]);
    SendData(segments[bcd_buffer[2]]);
    SendData(segments[bcd_buffer[3]]);
    PORTB |= (1 << PINB1);
}
void InitSPI(void)
{
    DDRB |= (1 << PINB3) | (1 << PINB5);
    SPSR |= (1 << SPI2X);
    SPCR = (1 << SPE) | (1 << MSTR);
}
void InitADC(void)
{
    ADMUX = (1 << MUX0);
    ADCSRB = (1 << ADTS2);
    ADCSRA = (1 << ADEN) | (1 << ADSC) | (1 << ADIF) | (1 << ADPS2)
| (1 << ADPS1);
}
void InitUSART()
{
    UCSR0B = (1 << RXEN0) | (1 << TXEN0) | (1 << RXCIE0);
    UCSR0C = (1 << UCSZ01) | (1 << UCSZ00);
}

```

```

        UBRR0 = 12;
    }
    void SendChar(char symbol)
    {
        while(!(UCSR0A & (1 << UDRE0)));
        UDR0 = symbol;
    }
    void SendString(char *buffer)
    {
        while(*buffer != 0) SendChar(*buffer++);
    }
    int main(void) {
        InitPorts();
        InitSPI();
        InitTimer1();
        EIMSK |= (1 << INT0);
        EICRA |= (1 << ISC01);
        InitADC();
        InitUSART();
        sei();
        PORTB &= ~(1 << PINB0);
        SendString("Hello\r\n");
        while(1)
        {
            DisplayData(display_val);
        }
    }
    ISR(TIMER1_COMPB_vect) {}
    ISR(INT0_vect)
    {
        SendString("Value = ");
        SendChar(0x30 + bcd_buffer[3]);
        SendChar(0x30 + bcd_buffer[2]);
        SendChar(0x30 + bcd_buffer[1]);
        SendChar(0x30 + bcd_buffer[0]);
        SendString("\r\n");
    }
    ISR(ADC_vect)
    {
        display_val = ADC;
    }
    ISR(USART_RX_vect)
    {
        if(UDR0 == 0x20) SendString("Roger that\r\n");
    }

```

Выводы: В данной лабораторной работе я узнал что такое АЦП и научился им пользоваться, а также выводить на UART сообщения.