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

Факультет информатики и
вычислительной техники

Кафедра ИиСП

Отчет
по лабораторной работе № 7

по дисциплине «Машинно-зависимые языки программирования»

Выполнил: студент группы ПС-11
Щеглов Г.С
Проверил: Баев А.А.

г. Йошкар-Ола
2024

Цель работы: научиться работать с USART и АЦП

Задания на лабораторную работу:

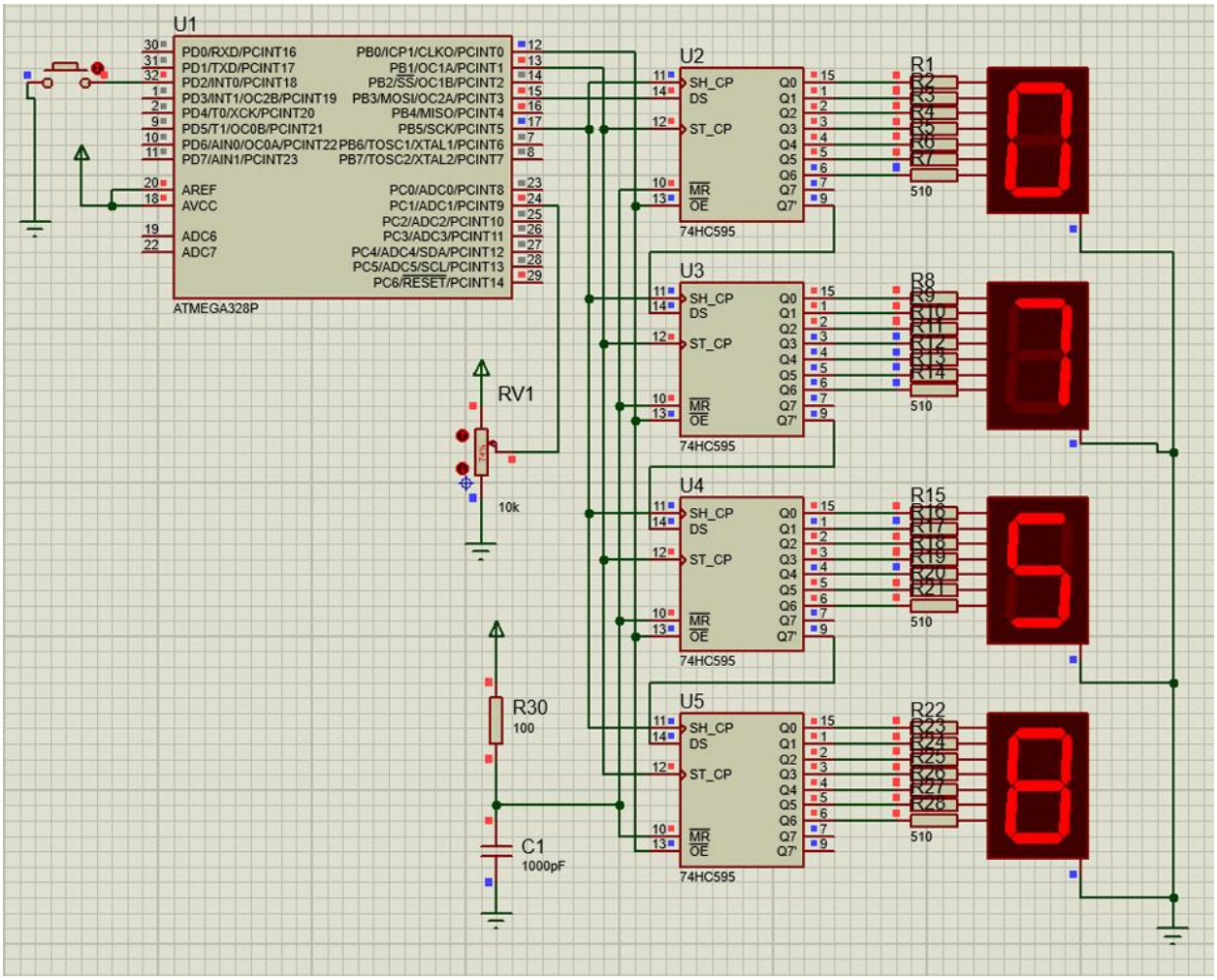
1. Схема с АЦП
2. Схема с USART
3. Оптимизация кода
4. Задание из методички

1. Теоретические сведения

Учебное пособие - ПРИМЕНЕНИЕ МИКРОКОНТРОЛЛЕРОВ В РАДИОТЕХНИЧЕСКИХ И БИОМЕДИЦИНСКИХ СИСТЕМАХ

2. Практическая часть

Схема с АЦП



Код на С:

```
#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<<ADATE|1<<ADIE);
}
```

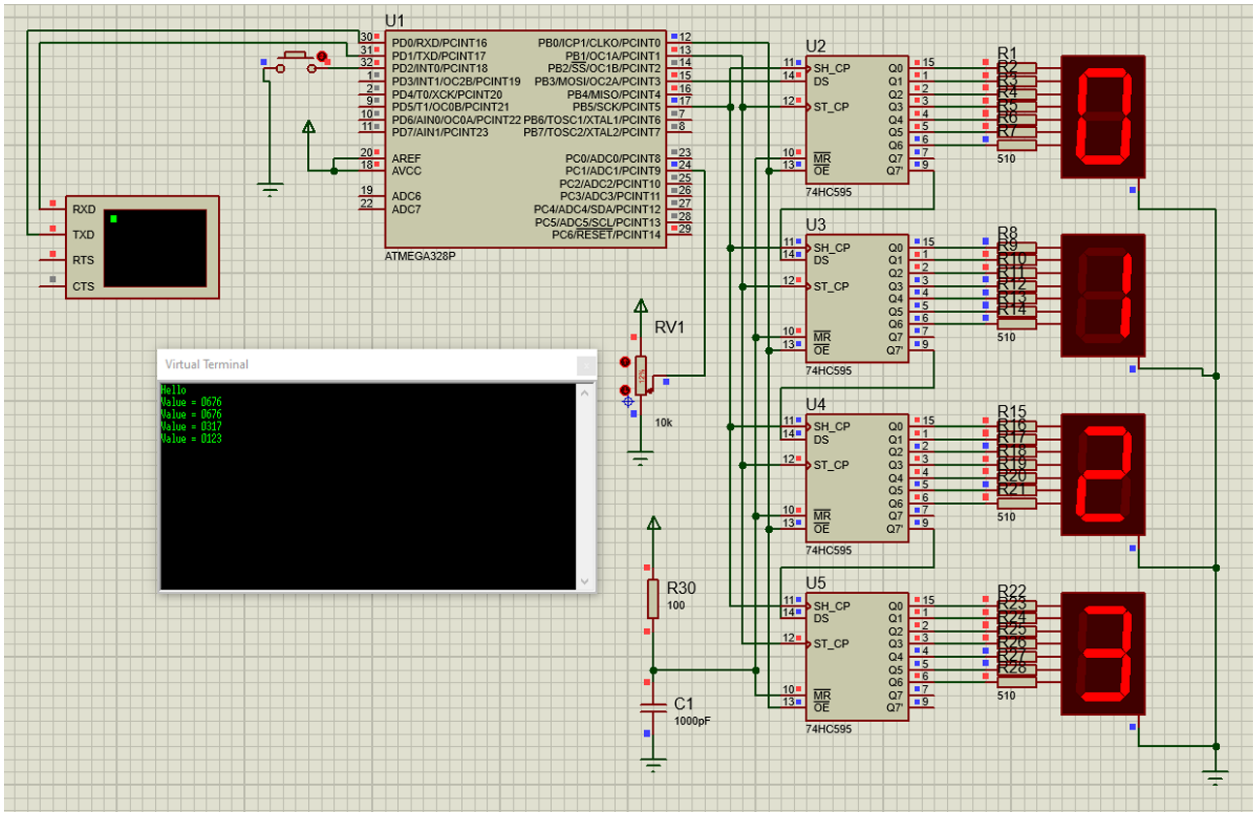
Оптимизированный код:

```
#define F_CPU 1600000UL
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>
```

```
uint8_t segments[] = {
  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
};


volatile uint8_t bcd_buffer[4] = {0};
volatile uint16_t display_val = 0;

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);

int main(void) {
  InitPorts();
  InitSPI();
  InitTimer1();
  InitADC();

  EIMSK |= (1 << INT0);         // Enable INT0
  EICRA |= (1 << ISC01);        // Trigger on falling edge of INT0
  sei();                        // Global interrupt enable

  PORTB &= ~(1 << PB0);         // OE = low (active)
  DisplayData(0);

  while (1) {
    DisplayData(display_val);
  }
}

//-------------------------------------------
ISR(ADC_vect) {
  display_val = ADC;
}

//-------------------------------------------
```

```
void InitPorts(void) {
  DDRB = (1 << PB0) | (1 << PB1) | (1 << PB3) | (1 << PB5);
  DDRD &= ~(1 << PD2);
  PORTD |= (1 << PD2);
}

void InitTimer1(void) {
  TCCR1B = (1 << CS11) | (1 << CS10) | (1 << WGM12); // Prescaler: 64,
CTC mode
  OCR1A = 1562;
  OCR1B = 1562;
  TIMSK1 |= (1 << OCIE1B);
}

void Bin2Dec(uint16_t data) {
  bcd_buffer[3] = (data / 1000) % 10;
  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 << PB1);
  for (uint8_t i = 0; i < 4; i++) {
    SendData(segments[bcd_buffer[i]]);
  }
  PORTB |= (1 << PB1);
}

void InitSPI(void) {
  DDRB |= (1 << PB3) | (1 << PB5); // Configure MOSI and CLK as output
  SPCR = (1 << SPE) | (1 << MSTR); // SPI enable, master mode
  SPSR |= (1 << SPI2X);            // Fclk = Fosc / 2
}

void InitADC(void) {
  ADMUX = (1 << MUX0);             // ADC1, left-aligned
  ADCSRB = (1 << ADTS2) | (1 << ADTS0); // Start on Timer1 COMPB
  ADCSRA = (1 << ADEN) | (1 << ADATE) | (1 << ADIE); // Enable ADC,
auto-trigger, interrupt
}
```

Схема с USART:



Код на C:

```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<<ADATE | 1<<ADIE);
}
void InitUSART(){
  UCSR0B = (1<<RXEN0 | 1<<TXEN0 |
```

```
  1<<RXCIE0);
  UCSR0C = (1<<UCSZ01 | 1<<UCSZ00);
  UBRR0H = 0;
  UBRR0L = 0x0C;
}
void SendChar(char symbol){
  while (!(UCSR0A & (1<<UDRE0)));
  UDR0 = symbol;
}
void SendString(char * buffer){
  while(*buffer != 0){
    SendChar(*buffer++);
  }
}
```

Оптимизированный код:

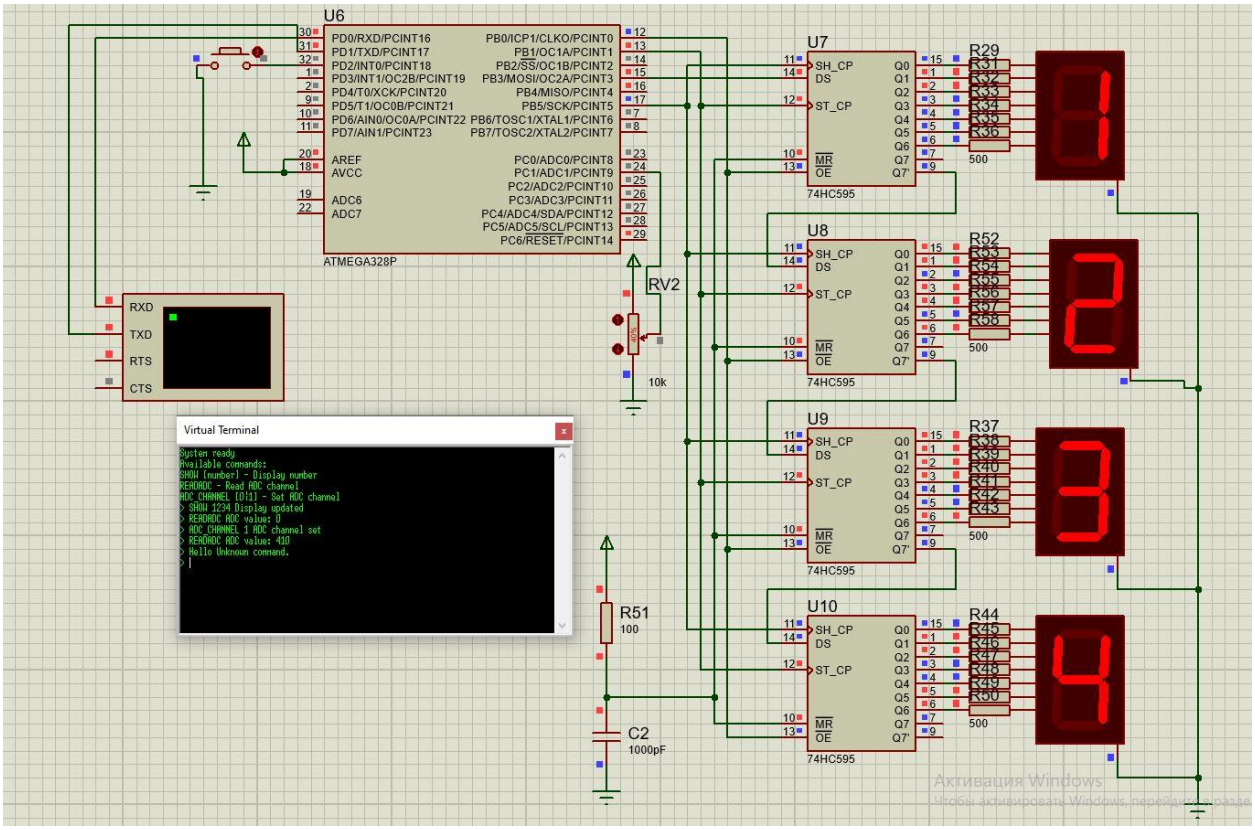
```
#define F_CPU 16000000UL
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>

uint8_t segments[] = {
  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
};


volatile uint8_t bcd_buffer[4] = {0};
volatile uint16_t display_val = 0;

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(const char *buffer);

int main(void)
{
  InitPorts();
  InitSPI();
  InitTimer1();
  EIMSK |= (1<<INT0);
  EICRA |= (1<<ISC01);
  InitADC();
  InitUSART();
  sei();

  PORTB &= ~(1<<PB0);
  DisplayData(0);
  SendString("Hello\r\n");

  while(1) {
    DisplayData(display_val);
  }
}

ISR(TIMER1_COMPB_vect) {}

ISR(INT0_vect) {
  SendString("Value = ");
  for(uint8_t i = 4; i > 0; ) {
    SendChar('0' + bcd_buffer[--i]);
  }
  SendString("\r\n");
}

ISR(ADC_vect) {
  display_val = ADC;
}

ISR(USART_RX_vect) {
  if(UDR0 == ' ') {
    SendString("Roger that\r\n");
  }
}

void InitPorts(void) {
  DDRB = (1<<PB0)|(1<<PB1)|(1<<PB3)|(1<<PB5);
  PORTD |= (1<<PD2);
}

void InitTimer1(void) {
  TCCR1A = 0;
```

```
  TCCR1B = (1<<CS11)|(1<<CS10)|(1<<WGM12);
  TCNT1 = 0;
  TIMSK1 |= (1<<OCIE1B);
  OCR1A = OCR1B = 1562;
}

void Bin2Dec(uint16_t data) {
  bcd_buffer[3] = (data / 1000) % 10;
  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<<PB1);
  for(uint8_t i = 0; i < 4; i++) {
    SendData(segments[bcd_buffer[i]]);
  }
  PORTB |= (1<<PB1);
}

void InitSPI(void) {
  DDRB |= (1<<PB3)|(1<<PB5);
  SPSR |= (1<<SPI2X);
  SPCR = (1<<SPE)|(1<<MSTR);
  PORTB &= ~((1<<PB3)|(1<<PB5));
}

void InitADC(void) {
  ADMUX = (1<<MUX0);
  ADCSRB = (1<<ADTS2)|(1<<ADTS0);
  ADCSRA = (1<<ADEN)|(1<<ADATE)|(1<<ADIE);
}

void InitUSART(void) {
  UCSR0B = (1<<RXEN0)|(1<<TXEN0)|(1<<RXCIE0);
  UCSR0C = (1<<UCSZ01)|(1<<UCSZ00);
  UBRR0L = 0x0C;
}

void SendChar(char symbol) {
  while (!(UCSR0A & (1<<UDRE0)));
  UDR0 = symbol;
}
```

```
void SendString(const char *buffer) {
  while(*buffer) {
    SendChar(*buffer++);
  }
}
```

Задание из методички:



Код на C:

```
#define F_CPU 16000000UL
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>
#include <string.h>
#include <stdlib.h>

uint8_t segments[] = {
  0b00111111,
  0b00000110,
  0b01011011,
  0b01001111,
  0b01100110,
```

```
  0b01101101,
  0b01111101,
  0b00000111,
  0b01111111,
  0b01101111,
};

volatile uint8_t bcd_buffer[] = {10, 10, 10, 10};
volatile uint16_t display_val = 0;
volatile uint8_t uart_buffer[32];
volatile uint8_t uart_index = 0;
volatile uint8_t command_ready = 0;
volatile uint8_t adc_channel = 0;

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 InitUART(void);
void ProcessCommand(char* command);
void UART_SendString(const char* str);
void UART_SendNumber(uint16_t num);
uint16_t ReadADC(void);

int main(void) {
  InitPorts();
  InitSPI();
  InitTimer1();
  InitUART();
  InitADC();

  sei();

  PORTB &= ~(1 << PB0);

  UART_SendString("System ready\r\n");
  UART_SendString("Available commands:\r\n");
  UART_SendString("SHOW [number] - Display number\r\n");
  UART_SendString("READADC - Read ADC channel\r\n");
  UART_SendString("ADC_CHANNEL [0|1] - Set ADC channel\r\n");
  UART_SendString("> ");

  while (1) {
    if (command_ready) {
      ProcessCommand((char*)uart_buffer);
      command_ready = 0;
      uart_index = 0;
```

```
      memset((void*)uart_buffer, 0, sizeof(uart_buffer));
      UART_SendString("> ");
    }
    DisplayData(display_val);
  }
}

void InitTimer1(void) {
  TCCR1A = 0;
  TCCR1B = (1 << CS12);
  TIMSK1 = (1 << TOIE1);
}

void ProcessCommand(char* command) {
  char* cmd = strtok(command, " ");

  if (strcmp(cmd, "SHOW") == 0) {
    char* num_str = strtok(NULL, " ");
    if (num_str) {
      display_val = atoi(num_str);
      UART_SendString(" Display updated\r\n");
    }
    } else if (strcmp(cmd, "READADC") == 0) {
    uint16_t value = ReadADC();
    UART_SendString(" ADC value: ");
    UART_SendNumber(value);
    UART_SendString("\r\n");
    } else if (strcmp(cmd, "ADC_CHANNEL") == 0) {
    adc_channel = atoi(strtok(NULL, " "));
    if (adc_channel == 0 || adc_channel == 1) {
      ADMUX = (ADMUX & 0xF0) | (adc_channel & 0x0F);
      UART_SendString(" ADC channel set\r\n");
      } else {
      UART_SendString(" Invalid channel value\r\n");
    }
    } else {
    UART_SendString(" Unknown command.");
    UART_SendString("\r\n");
  }
}

void InitUART(void) {
  UBRR0H = 0;
  UBRR0L = 12;
  UCSR0B = (1 << RXEN0) | (1 << TXEN0) | (1 << RXCIE0);
  UCSR0C = (1 << UCSZ01) | (1 << UCSZ00);
}

void InitPorts(void) {
  DDRB = (1 << PB0) | (1 << PB1) | (1 << PB3) | (1 << PB5);
```

```
  DDRD = (1 << PD1);
}

void InitSPI(void) {
  DDRB |= (1 << PB3) | (1 << PB5);
  SPSR |= (1 << SPI2X);
  SPCR = (1 << SPE) | (1 << MSTR);
}

void InitADC(void) {
  ADMUX = (1 << REFS0);
  ADCSRA = (1 << ADEN) | (1 << ADPS1) | (1 << ADPS0);
}

uint16_t ReadADC(void) {
  ADCSRA |= (1 << ADSC);
  while (ADCSRA & (1 << ADSC));
  return ADC;
}

void UART_SendString(const char* str) {
  while (*str) {
    while (!(UCSR0A & (1 << UDRE0)));
    UDR0 = *str++;
  }
}

void UART_SendNumber(uint16_t num) {
  char buffer[6];
  itoa(num, buffer, 10);
  UART_SendString(buffer);
}

void Bin2Dec(uint16_t data) {
  bcd_buffer[3] = (data / 1000) % 10;
  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 << PB1);

  SendData(segments[bcd_buffer[0]]);
```

```
  SendData(segments[bcd_buffer[1]]);
  SendData(segments[bcd_buffer[2]]);
  SendData(segments[bcd_buffer[3]]);

  PORTB |= (1 << PB1);
}

ISR(USART_RX_vect) {
  uint8_t received = UDR0;

  if (received == '\r' || received == '\n') {
    if (uart_index > 0) {
      uart_buffer[uart_index] = '\0';
      command_ready = 1;
    }
    } else if (received == 8 || received == 127) {
    if (uart_index > 0) {
      uart_index--;
      UART_SendString("\b \b");
    }
    } else if (uart_index < sizeof(uart_buffer) - 1) {
    uart_buffer[uart_index++] = received;
    UDR0 = received;
  }
}

ISR(TIMER1_OVF_vect) {
}

ISR(ADC_vect) {
  display_val = ADC;
     }
```

Выводы: мы научились работать с АЦП и USART