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 2

Выполнил: ст. гр. ПС-14
Сайфутдинов Э.Р.
Проверил: доцент, доцент кафедры ИиСП Баев А.А.

г. Йошкар-Ола
2024

Цель работы:

Рассмотреть работу блоков АЦП и UART, приведены примеры работы с ними и исследованы возможные варианты их применения.

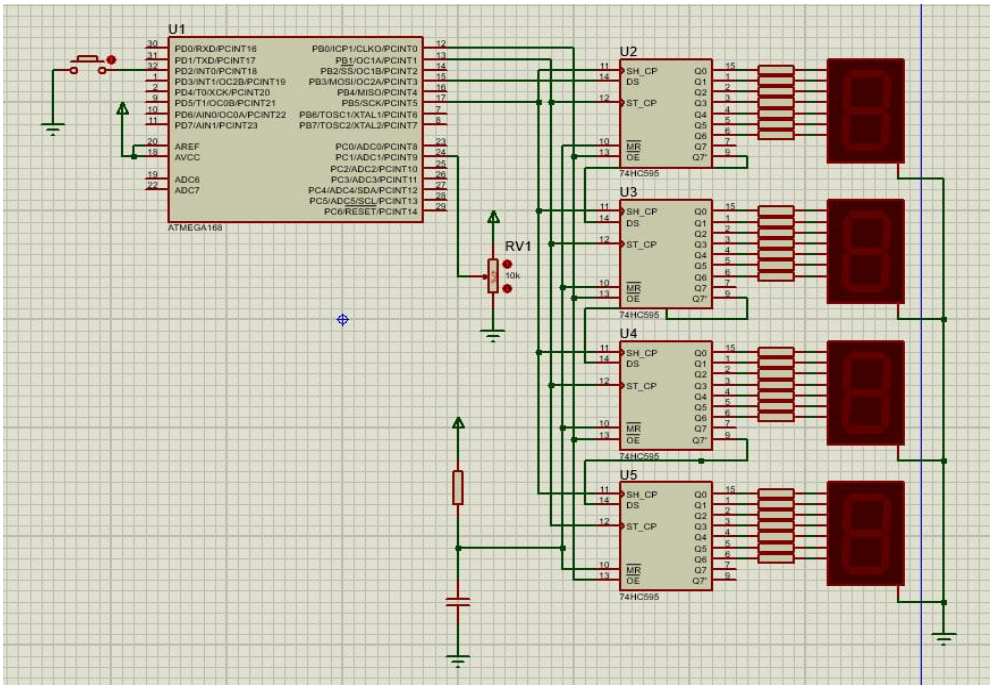
Задания на лабараторную работу:

1. **Теоретические сведения**

   Учебное пособие “Применение микроконтроллеров в радиотехнических и биомедицинских системах”.

2. Практическая часть

1) Работа с АЦП



Код:

```
#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);
```

```
}
```

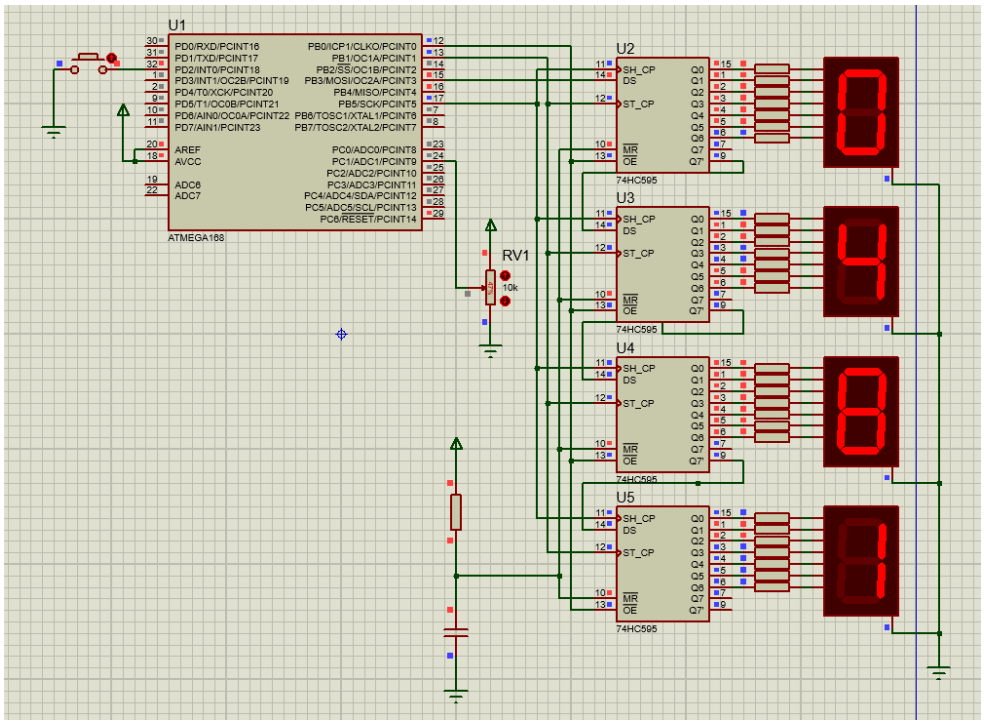
Оптимизированный код:

```
#define F_CPU 1000000UL

#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();
	InitTimer1();
	InitSPI();

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

	PORTB &= ~(1 << PINB0); // OE = low (active)
	DisplayData(0);

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

ISR(TIMER1_COMPB_vect) {}

ISR(INT0_vect) {}

ISR(ADC_vect) {
	display_val = ADC; // Считывание значения ADC
}

void InitPorts(void) {
```

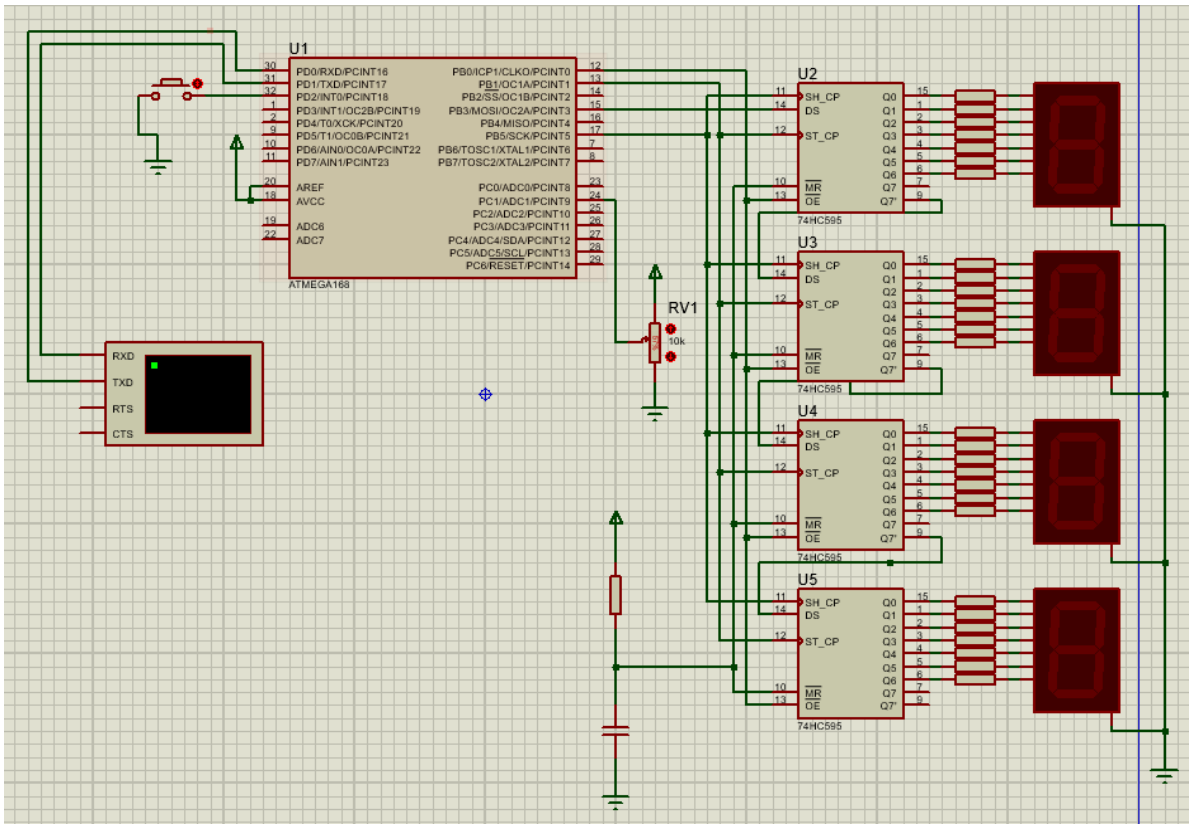
```
	DDRB = (1 << PINB0) | (1 << PINB1) | (1 << PINB3) | (1 << PINB5);
	DDRD &= ~(1 << PIND2); // PIND2 как вход
	PORTD |= (1 << PIND2); // Подтяжка к VCC
}

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; data %= 1000;
	bcd_buffer[2] = data / 100; data %= 100;
	bcd_buffer[1] = data / 10; data %= 10;
	bcd_buffer[0] = data;
}

void SendData(uint8_t data) {
	SPDR = data;
	while (!(SPSR & (1 << SPIF))); // Ждем завершения передачи
}

void DisplayData(uint16_t data) {
	Bin2Dec(data);
	PORTB &= ~(1 << PINB1); // 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 << PINB1); // clk_out = 1
}

void InitSPI(void) {
	DDRB |= (1 << PINB3) | (1 << PINB5); // MOSI и CLK как выход
	SPSR |= (1 << SPI2X); // Fclk = Fosc/2
	SPCR = (1 << SPE) | (1 << MSTR); // Включение SPI, мастер-режим
	PORTB &= ~(1 << PINB3) | (1 << PINB5); // Начальные значения - DAT низкий, CLK
низкий
}

void InitADC(void) {
	ADMUX = (1 << MUX0); // Выбор канала 1, выравнивание влево
	ADCSRB = (1 << ADTS2) | (1 << ADTS0); // Старт по COMPB таймера 1
	// Включение, автообновление, разрешение прерываний
	ADCSRA = (1 << ADEN) | (1 << ADATE) | (1 << ADIE);
}
```

Из программы удалены части кода, которые отвечали за работу секундомера и реакцию на нажатие кнопки. Теперь Таймер 1 настроен на срабатывание раз в 100 мс. Срабатывание таймера запускает преобразование АЦП (это настроено установкой бит ADTS2 и ADTS0 в регистре ADCSRB).

После окончания преобразования вызывается соответствующее прерывание АЦП, в котором выделенной переменной display_val присваивается значение, полученное при измерении. Стоит отметить, что АЦП используется в 10-битном формате, потому выделенная переменная 16-битная и в нее сохраняется содержимое обоих регистров данных АЦП – ADCH 92 и ADCL. Чтение данных двух регистров имеет свою особенность: при поочередном их чтении сначала должен быть прочитан ADCL, и только потом – ADCH. При записи вида display_val=ADC; за этим следит компилятор. В основном цикле программы происходит периодический вывод переменной на индикаторы. Таким образом, блок АЦП обладает широкими возможностями по настройке и позволяет получить необходимую логику работы.



2) Подключение виртуального прибора



Код:

```
#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]]);
```

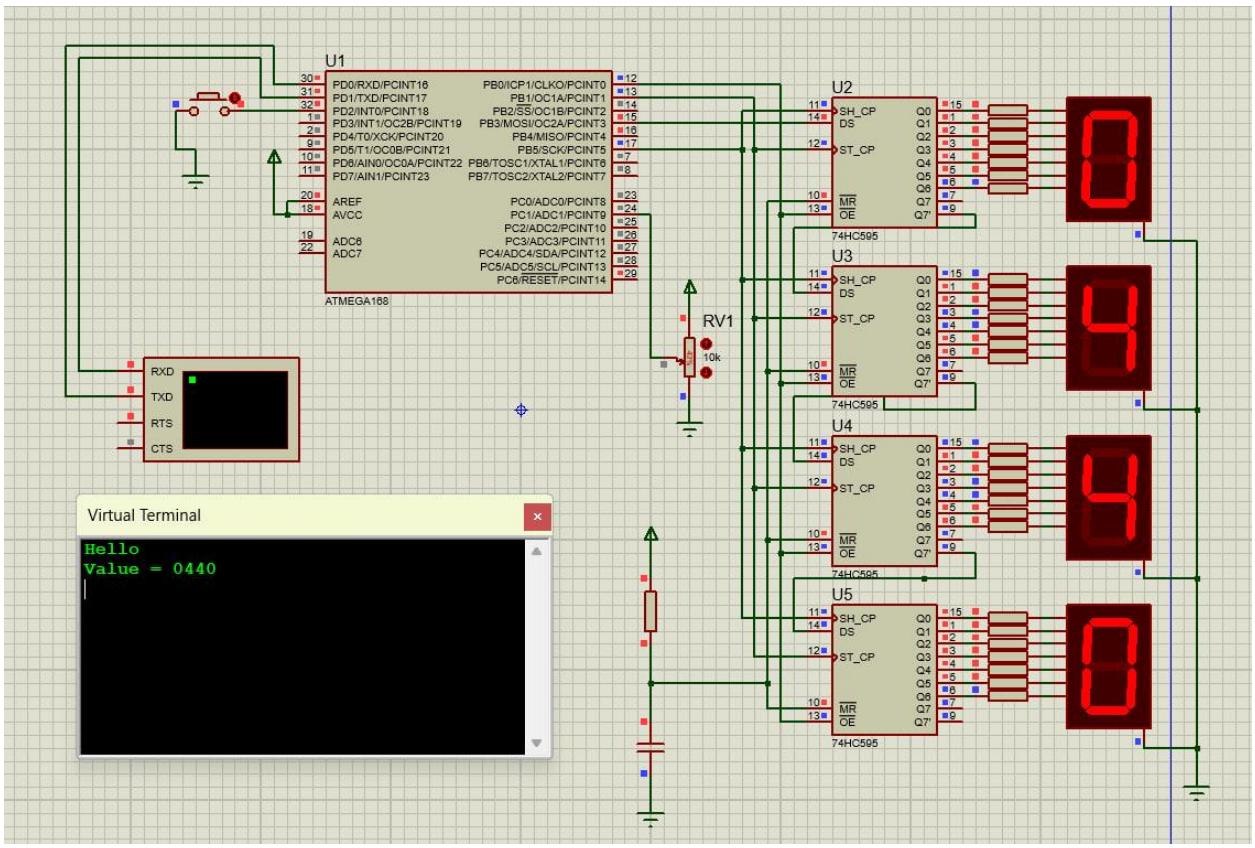
```
        SendData(segments[bcd_buffer[1]]);
        SendData(segments[bcd_buffer[2]]);
        SendData(segments[bcd_buffer[3]]);
        PORTB |= (1<<PB1); //clk_out = 1
}
void InitSPI( void){
        DDRB |= (1<<PB3 | 1<<PB5);//configure MOSI and CLK as out
        SPSR |= (1<<SPI2X); //Fclk = Fosc/2
        //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 1000000UL

#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);
```

```c
void InitADC(void);
void InitUSART(void);
void SendChar(char symbol);
void SendString(char * buffer);

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

	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);
	SendString("Hello\r\n");

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

//-----------------------------------------
ISR(TIMER1_COMPB_vect) {}

ISR(INT0_vect) {
	SendString("Value = ");
	for (int i = 3; i >= 0; i--) {
		SendChar(0x30 + bcd_buffer[i]); // Отправка BCD значений
	}
	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 &= ~(1 << PD2);       // PD2 как вход
	PORTD |= (1 << PD2);       // Подтяжка к VCC
}

void InitTimer1(void) {
	TCCR1A = 0;                // CTC mode - Clear Timer on Compare
	TCCR1B = (1 << CS11) | (1 << CS10) | (1 << WGM12); // Prescaler = sys_clk/64
	TCNT1 = 0;                 // Начальное значение счетчика
	TIMSK1 |= (1 << OCIE1B);
	OCR1A = 1562;              // Значение для сравнения
	OCR1B = 1562;
}

void Bin2Dec(uint16_t data) {
	bcd_buffer[3] = (uint8_t)(data / 1000);
	data %= 1000;
```

```
	bcd_buffer[2] = (uint8_t)(data / 100);
	data %= 100;
	bcd_buffer[1] = (uint8_t)(data / 10);
	data %= 10;
	bcd_buffer[0] = (uint8_t)(data);
}

void SendData(uint8_t data) {
	SPDR = data;
	while (!(SPSR & (1 << SPIF))); // Ждем завершения передачи
}

void DisplayData(uint16_t data) {
	Bin2Dec(data);
	PORTB &= ~(1 << PB1); // clk_out = 0
	for (int i = 0; i < 4; i++) {
		SendData(segments[bcd_buffer[i]]);
	}
	PORTB |= (1 << PB1); // clk_out = 1
}

void InitSPI(void) {
	DDRB |= (1 << PB3) | (1 << PB5); // MOSI и CLK как выход
	SPSR |= (1 << SPI2X); // Fclk = Fosc/2
	SPCR = (1 << SPE) | (1 << MSTR); // Включение SPI, мастер-режим
	PORTB &= ~(1 << PB3) | (1 << PB5); // Начальные значения - DAT низкий, CLK низкий
}

void InitADC(void) {
	ADMUX = (1 << MUX0); // Выбор канала 1, выравнивание влево
	ADCSRB = (1 << ADTS2) | (1 << ADTS0); // Старт по COMPB таймера 1
	ADCSRA = (1 << ADEN) | (1 << ADATE) | (1 << ADIE); // Включение, автообновление,
IRQ включение
}

void InitUSART(void) {
	UCSR0B = (1 << RXEN0) | (1 << TXEN0) | (1 << RXCIE0);
	UCSR0C = (1 << UCSZ01) | (1 << UCSZ00);
	UBRR0H = 0;
	UBRR0L = 0x0C; // Установка скорости в 9600
}

void SendChar(char symbol) {
	while (!(UCSR0A & (1 << UDRE0))); // Ожидание, пока передатчик готов
	UDR0 = symbol;
}

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


U1
ATMEGA168
PD0/RXD/PCINT16
PD1/TXD/PCINT17
PD2/INT0/PCINT18
PD3/INT1/OC2B/PCINT19
PD4/T0/XCK/PCINT20
PD5/T1/OC0B/PCINT21
PD6/AIN0/OC0A/PCINT22
PD7/AIN1/PCINT23
AREF
AVCC
ADC6
ADC7
PB0/ICP1/CLKO/PCINT0
PB1/OC1A/PCINT1
PB2/SS/OC1B/PCINT2
PB3/MOSI/OC2A/PCINT3
PB4/MISO/PCINT4
PB5/SCK/PCINT5
PB6/TOSC1/XTAL1/PCINT6
PB7/TOSC2/XTAL2/PCINT7
PC0/ADC0/PCINT8
PC1/ADC1/PCINT9
PC2/ADC2/PCINT10
PC3/ADC3/PCINT11
PC4/ADC4/SDA/PCINT12
PC5/ADC5/SCL/PCINT13
PC6/RESET/PCINT14
RXD
TXD
RTS
CTS
RV1
10k
U2
U3
U4
U5
74HC595
SH_CP
DS
ST_CP
MR
OE
Q0
Q1
Q2
Q3
Q4
Q5
Q6
Q7
Q7'
Virtual Terminal
Hello
Value = 0440

Вывод:

Таким образом, рассмотрена работа блоков АЦП и UART, приведены примеры работы с ними и исследованы возможные варианты их применения.