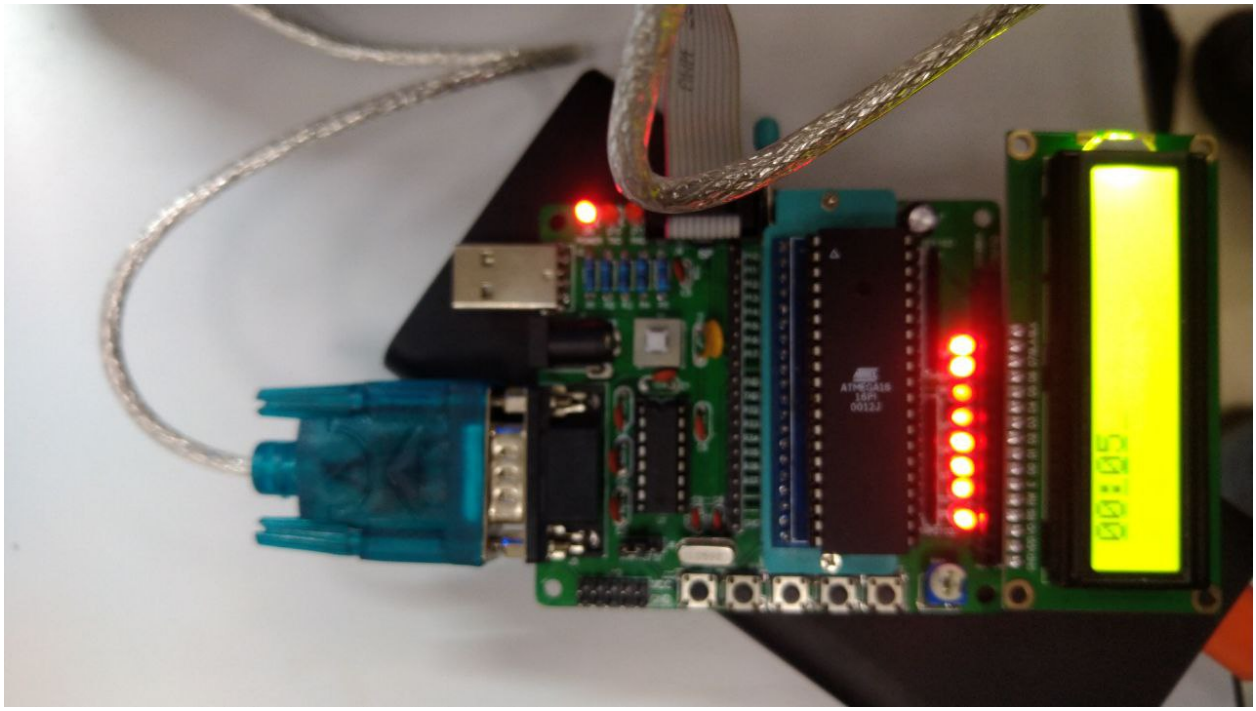
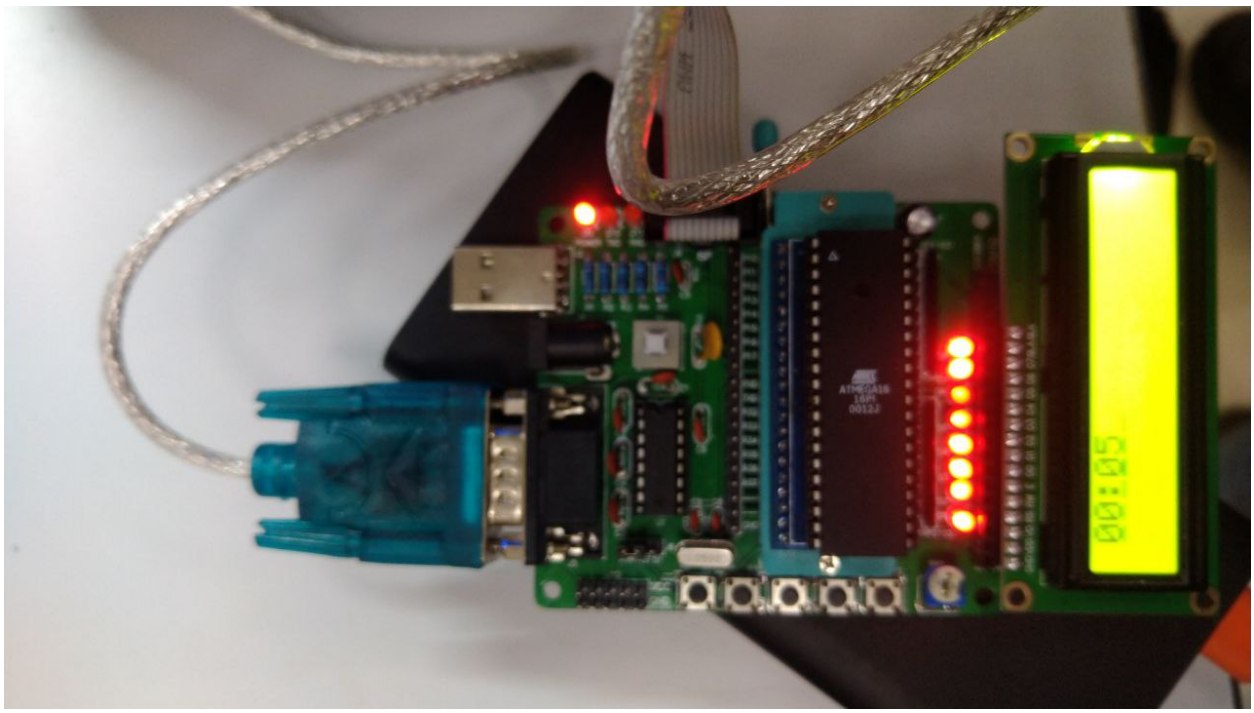
Ход работы:

1. **Инициализация оборудования:**
   - ЖК-дисплей подключен к портам PORTB и PORTC для отображения оставшегося времени.
   - UART настроен для работы на скорости передачи данных 9600 бод.
2. **Работа таймера:**
   - Используется таймер Timer1 в режиме CTC с предделителем 1024. Частота процессора — 11.0592 МГц.
   - Для отсчета одной секунды в регистр OCR1A записано значение 54736.
3. **Управление через UART:**
   - Прием строк осуществляется прерыванием USART_RXC_vect. Данные заносятся в буфер, после чего вызывается функция обработки строки ProcessString.
   - Реализован разбор строк формата:
     - U:mm:ss// — таймер считает вверх до указанного времени.
     - D:mm:ss// — таймер считает вниз от указанного времени.
4. **Вывод на ЖК-дисплей:**
   - Текущее время таймера обновляется каждую секунду и отображается в формате MM:SS.
5. **Завершение счета:**
   - Когда таймер достигает 0 в режиме убывания, таймер останавливается, и можно вывести сообщение об окончании работы.

Пример работы:





1. Передача команды U:02:30//:
    - Таймер запускается в режиме счета вверх до 2 минут 30 секунд.
    - На ЖК-дисплее последовательно отображаются значения: 00:01, 00:02 ... 02:30.
2. Передача команды D:00:45//:
    - Таймер начинает отсчет вниз от 45 секунд.
    - После достижения 0 можно вывести сообщение об окончании.

Выводы:

1. В ходе работы реализован таймер с настраиваемыми режимами счета вверх и вниз.
2. Обеспечена корректная работа с UART для приема команд и управления таймером.
3. ЖК-дисплей успешно отображает текущее состояние таймера, позволяя пользователю контролировать процесс в реальном времени.
4. Лабораторная работа помогла закрепить навыки работы с периферией микроконтроллера и интеграции различных интерфейсов.

```
#define F_CPU 11059200UL
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>
#include <stdio.h>

#define LEN 6
```

```
#define RS 2
#define RW 1
#define EN 0
#define TI 500

extern unsigned char rData;
extern unsigned char IDX;
extern unsigned char buffer [LEN];
extern int mode;

// Глобальные переменные
volatile uint8_t timer_mode = 0; // 1 - возрастающий, 2 -
убывающий
volatile uint16_t seconds = 0;
volatile uint16_t index = 0;
volatile int flag = 0;
unsigned char rData = 0;
unsigned char IDX = 0;
unsigned char buffer [LEN];
unsigned char datBuffer [LEN];

int mode = 0;
// Дисплей
void lcd_com(unsigned char p){
     PORTB &= ~(1<<RS);
     PORTB |= (1<<EN); // EN=1 начало записи команды
     PORTC = p; // Вывод команды на шину данных экрана
     _delay_us(TI);
     PORTB &=~(1<<EN); // EN=1 конец записи команды
     _delay_us(TI);
}

void lcd_dat(unsigned char p){
     PORTB|=(1<<RS)|(1<<EN); // Начало записи данных
     PORTC=p; // Установка значений
     _delay_us(TI);
     PORTB&=~(1<<EN); // Конец записи данных
     _delay_us(TI);
}

void lcd_init(void) {
     DDRB |= (1<<RS)|(1<<RW)|(1<<EN); // Управление на вывод

     // Обнуляем биты управления и шин данных
```

```
	PORTB=0x00;
	DDRC=0xFF;
	PORTC=0x00;

	_delay_us(TI);
	lcd_com(0x08); // Полное включение дисплея
	_delay_us(TI);
	lcd_com(0x3C); // 8 бит данных 2 строки
	_delay_us(TI);
	lcd_com(0x01); // очистка строки
	_delay_us(TI);
	lcd_com(0x06); // сдвиг гурсора вправо
	_delay_us(TI*6);
	lcd_com(0x0F); // курсор показан и мигает
}

void lcd_string(char *str){
	char data=0;
	while(*str){
		data=*str++;
		lcd_dat(data);
	}
}

// Uart
void UART_init(){
	UCSRB |=(1<<TXEN)|(1<<RXEN)|(1<<RXCIE);
	UCSRC |=(1<<URSEL)|(1<<UCSZ1)|(1<<UCSZ0);
	UBRRL=11;
}

void UART_Send_Char(unsigned char c){
	while(!(UCSRA&(1<<UDRE)));
	UDR=c;
}

void UART_Send_String(char *str){
	char data=0;
	while(*str){
		data=*str++;
		UART_Send_Char(data);
	}
}
```

```
void UART_Send_Arr(unsigned char mas[], char size){
     unsigned char i=0;
     for(i=0;i<size;i++){
          UART_Send_Char(mas[i]);
     }
}

// Функция приема символа через UART
char UART_Receive(void) {
     while (!(UCSRA & (1<<RXC))); // Ждем, пока данные будут
получены
     return UDR;
}


// Таймер
void Timer1_Init() {
     TCCR1B = (1 << WGM12) | (1 << CS12) | (1 << CS10); //
CTC, прескалер 1024
     OCR1A = 54736; // 1 секунда при 8 МГц и прескалере 1024
     TIMSK = (1 << OCIE1A); // Разрешаем прерывание по
совпадению
}

ISR(TIMER1_COMPA_vect) {
     if (timer_mode == 1) {
          seconds++;
          } else if (timer_mode == 2 && seconds > 0) {
          seconds--;
     }
}



// Обработка строки
void ProcessString(char *str) {
     if (str[0] == 'U') {
          timer_mode = 1; // Возрастающий таймер
          } else if (str[0] == 'D') {
          timer_mode = 2; // Убывающий таймер
          } else {
          timer_mode = 0;
          return;
     }
```

```
    char *token = strtok(str, ":");
    token = strtok(NULL, ":"); // Минуты
    uint8_t minutes = atoi(token);
    token = strtok(NULL, ":"); // Секунды
    uint8_t secs = atoi(token);
    seconds = minutes * 60 + secs;
    flag = 1;
}


ISR(USART_RXC_vect){
    mode=0;
    rData=UDR;
    buffer[index++]=rData;
    if (index== LEN){
        flag = 1;
        ProcessString(buffer);
    }
}


int main(void)
{
    mode=0;
    lcd_init();
    lcd_com(0x80);
    UART_init();
    Timer1_Init();
    sei();

    char display_buffer[16];
    while (1)
    {
        if (flag == 1){ // Вывод на дисплей
            uint8_t minutes = seconds / 60;
            uint8_t secs = seconds % 60;
            sprintf(display_buffer, "%02d:%02d", minutes,
secs);
            lcd_string(display_buffer);
            lcd_com(0x80);
        }
    }
    return 0;
```

```
}
```