Университет ИТМО
Факультет программной инженерии и компьютерной техники

Лабораторная работа № 4

По дисциплине "Системы ввода\вывода"

Вариант: **2**

Выполнили:
1.4
Баянов Р. Д.
Андреева А. Р.
Лянгузов Д. М.
Кузнецов Д.А.
1.5
Юхно Т. И.

Содержание

Цель

Изучить основные принципы работы с контроллерами ввода-вывода на примере UART.

Задачи

1. Разработать программу для микроконтроллера Atmega328, которая будет принимать и отправлять пакеты через интерфейс UART в соответствии с заданным форматом. Драйвер UART необходимо реализовать с использованием операций ввода/вывода в регистры аппаратного контроллера UART.
2. Микроконтроллер должен принимать данные от ПК, проверять их на корректность и отправлять обратно правильные пакеты. Пакеты с ошибками должны отбрасываться.
3. Микроконтроллер должен ежесекундно передавать данные с указанного в задании датчика.
4. Создать клиентскую программу на ПК для отправки и получения пакетов к микроконтроллеру через интерфейс UART, которая будет моделировать как корректные отправки, так и ситуации с ошибками: неверная длина, отсутствие синхробайта, недостаточное количество данных.
5. Подключить микроконтроллер к ПК и протестировать работоспособность разработанных программ.
6. Записать осциллограмму передачи любого пакета через интерфейс UART.
7. Подготовить отчет о выполненной работе в электронном виде.

№ варианта	Датчик	Скорость UART	Четность	Кол-во стоповых бит
2	BMP280, SPI температура и давление	38400	odd parity	2

Исходный код разработанной программы

```
#include <Adafruit_BMP280.h>
#include <HardwareSerial.h>

#define BMP_SCK  (13)
#define BMP_MISO (12)
#define BMP_MOSI (11)
#define BMP_CS   (10)

Adafruit_BMP280 bmp(BMP_CS); // hardware SPI

#define SYNC_BYTE 0x5A
#define MAX_PACKET_DATA 64

typedef enum {
      WAIT_START = 0,
      WAIT_LEN,
      WAIT_DATA,
      WAIT_CRC
} PacketState;

PacketState packet_state = WAIT_START;
uint8_t packet_length = 0;
uint8_t data_index = 0;
uint8_t packet_data[MAX_PACKET_DATA];
uint8_t received_crc = 0;
uint8_t packet_ready = 0;

String custom_message = ""; // Переменная, которая хранит
пользовательское сообщение

uint8_t crc8(String data) {
      uint8_t crc = 0x00;
      const uint8_t polynomial = 0x07;
      for (size_t i = 0; i < data.length(); i++) {
      crc ^= data[i];
      for (uint8_t j = 0; j < 8; j++) {
            if (crc & 0x80) {
                  crc = (crc << 1) ^ polynomial;
            } else {
                  crc <<= 1;
            }
      }
```

```
      }
      return crc;
}

void serialWriteByte(uint8_t byte) {
  while (!(UCSR0A & (1 << UDRE0))); // Ждём, пока буфер пуст
  UDR0 = byte;
}

void serialWriteString(String data) {
  for (size_t i = 0; i < data.length(); i++) {
      serialWriteByte(data[i]);
  }
}

void serial_send_packet(String data) {
  if (data.length() > 0xFF) return;

  serialWriteByte(SYNC_BYTE);
  serialWriteByte(data.length());
  serialWriteString(data);
  serialWriteByte(crc8(data));
}

String dataToString(uint8_t* data, uint8_t length) {
      String result = "";
      for (uint8_t i = 0; i < length; i++) {
      result += (char)data[i];
      }
      return result;
}

void processSerial() {
      while (Serial.available()) {
      uint8_t b = Serial.read();
      switch (packet_state) {
            case WAIT_START:
                  if (b == SYNC_BYTE) {
                  packet_state = WAIT_LEN;
                  }
                  break;

            case WAIT_LEN:
                  packet_length = b;
                  data_index = 0;
                  if (packet_length == 0)
```

```
                  packet_state = WAIT_CRC;
                  else
                  packet_state = WAIT_DATA;
                  break;

            case WAIT_DATA:
                  if (data_index < MAX_PACKET_DATA) {
                  packet_data[data_index++] = b;
                  if (data_index >= packet_length) {
                        packet_state = WAIT_CRC;
                  }
                  } else {
                  packet_state = WAIT_START; // too much data
                  }
                  break;

            case WAIT_CRC:
                  received_crc = b;
                  if (crc8(dataToString(packet_data, packet_length)) ==
received_crc) {
                  custom_message = dataToString(packet_data,
packet_length);
                  packet_ready = 1;
                  }
                  packet_state = WAIT_START;
                  break;

            default:
                  packet_state = WAIT_START;
                  break;
      }
      }
}

void serialSetup() {
  uint16_t baudRate = 38400;
  uint16_t ubrr = 16000000 / 16 / baudRate - 1;

  UBRR0H = (unsigned char)(ubrr >> 8);
  UBRR0L = (unsigned char)ubrr;

  SetBit(UCSR0B, TXEN0);
  SetBit(UCSR0B, RXEN0);

  UCSR0C = (1 << UPM01) | (0 << UPM00) |
           (1 << USBS0) |
```

```
            (1 << UCSZ01) | (1 << UCSZ00);
}

void setup() {
      serialSetup();
      bmp.begin();

      bmp.setSampling(Adafruit_BMP280::MODE_NORMAL,
                  Adafruit_BMP280::SAMPLING_X2,
                  Adafruit_BMP280::SAMPLING_X16,
                  Adafruit_BMP280::FILTER_X16,
                  Adafruit_BMP280::STANDBY_MS_500);
}

void loop() {
      processSerial();

      if (custom_message != "") {
      serial_send_packet(custom_message);
      // custom_message = ""; // Clear the message after sending
      } else {
      serial_send_packet("Temperature = " +
String(bmp.readTemperature()) + " *C\n");
      serial_send_packet("Pressure = " + String(bmp.readPressure()) + "
Pa\n");
      }

      delay(1000);
}
```

Микроконтроллер принимает, проверяет и возвращает данные, а также периодически отправляет показания датчика.

Код клиентской программы

```
import serial
import serial.tools.list_ports

def calculate_crc8(data):
      crc = 0
      for byte in data:
      crc ^= byte
      for _ in range(8):
            if crc & 0x80:
                  crc = (crc << 1) ^ 0x07
            else:
                  crc = crc << 1
      crc &= 0xFF
      return crc

def write_packet(ser, data):
      # S | L | data | CRC8
      # Prepare the packet
      sync_byte = bytes([0x5A])
      length_byte = bytes([len(data)])
      crc = calculate_crc8(data)
      crc_byte = bytes([crc])

      # Assemble the full packet
      packet = sync_byte + length_byte + data + crc_byte

      # Send the packet
      bytes_written = ser.write(packet)
      return bytes_written

def read_packet(ser):
      # S | L | data | CRC8
      # Read synchrobyte
      sync_byte = ser.read(1)
      if not sync_byte or sync_byte[0] != 0x5A:
      raise ValueError(f"Invalid synchrobyte: {sync_byte.hex() if
sync_byte else 'None'}, expected: 0x5A")

      # Read length byte
      length_byte = ser.read(1)
      if not length_byte:
      raise ValueError("Failed to read length byte")

      data_length = length_byte[0]
```

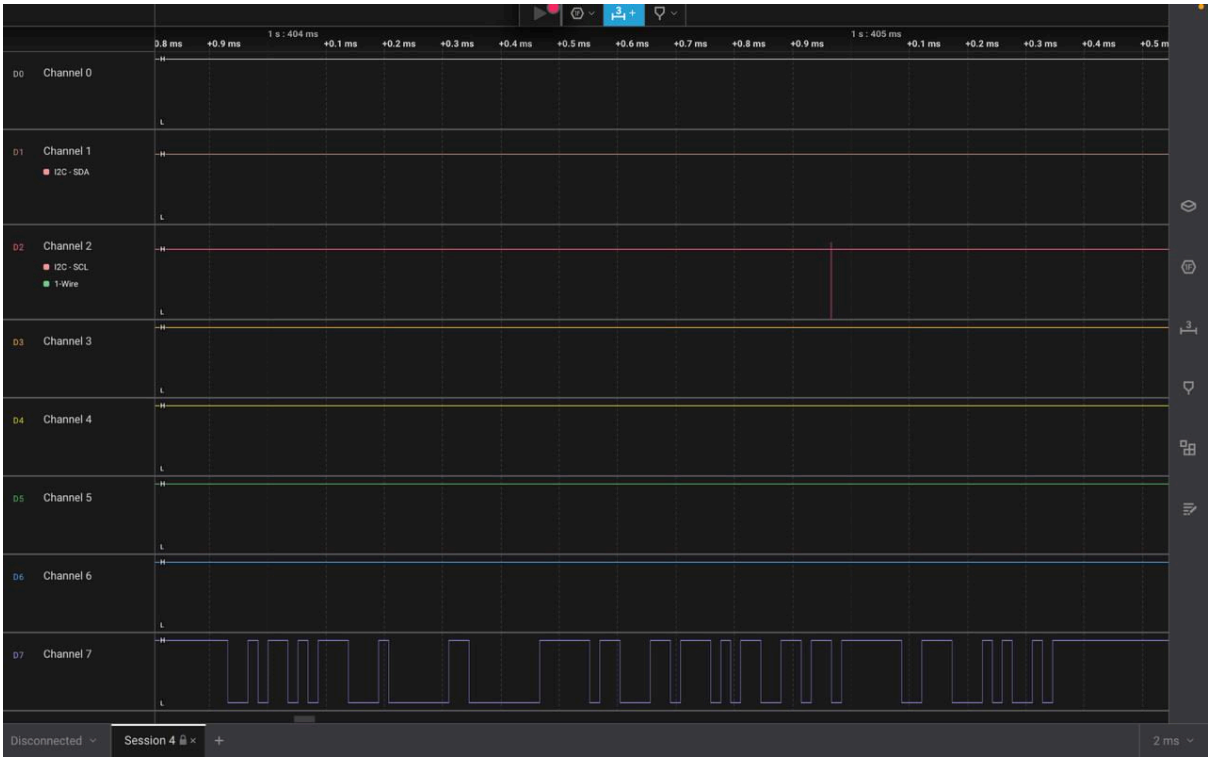
```
      # Read data
      data = ser.read(data_length)
      if len(data) != data_length:
      raise ValueError(f"Incomplete data: read {len(data)} bytes,
expected {data_length} bytes")

      # Read CRC8
      crc_byte = ser.read(1)
      if not crc_byte:
      raise ValueError("Failed to read CRC byte")

      calculated_crc = calculate_crc8(data)

      if calculated_crc != crc_byte[0]:
      raise ValueError(f"CRC check failed: received {crc_byte[0]},
calculated {calculated_crc}")

      return data

def read_single_packet(ser):
      try:
      data = read_packet(ser)
      #print(f"Received data (hex): {data.hex()}")
      try:
            print(f"Received data (string): {data.decode('utf-8')}")
      except UnicodeDecodeError:
            print("Received data is not a valid UTF-8 string")
      return True
      except ValueError as ve:
      print(f"Packet error: {ve}")
      return False

def read_continuous(ser):
      print("Waiting for data. Press Ctrl+C to exit...")
      try:
      while True:
            read_single_packet(ser)
      except KeyboardInterrupt:
      print("\nReceiving stopped by user")

def write_message(ser):
      message = input("Enter message to send: ")
      if not message:
      print("Empty message, not sending")
      return
```

```
        bytes_written = write_packet(ser, message.encode('utf-8'))
        print(f"Sent {bytes_written} bytes")

def display_menu():
        print("\n--- Serial Communication Menu ---")
        print("1. Write a message")
        print("2. Read a single packet")
        print("3. Read continuously")
        print("4. Exit")
        return input("Select option (1-4): ")

def main():
        baudrate = 9600
        ser = None

        try:
        # Find and open the COM port
        ports = serial.tools.list_ports.comports()
        port = next((p.device for p in ports), None)
        if port is None:
                raise ValueError("No COM port found.")

        # ser = serial.Serial(port, baudrate=baudrate)
        ser = serial.Serial(
                port=port,
                baudrate=baudrate,
                bytesize=serial.EIGHTBITS,
                parity=serial.PARITY_ODD,
                stopbits=serial.STOPBITS_TWO,
                timeout=1
        )
        print(f"Connected to {port} at {baudrate} baud")

        while True:
                choice = display_menu()

                if choice == '1':
                        write_message(ser)
                elif choice == '2':
                        print("Reading a single packet...")
                        read_single_packet(ser)
                elif choice == '3':
                        read_continuous(ser)
                elif choice == '4':
                        print("Exiting...")
```

```
                  break
            else:
                  print("Invalid option, please try again.")



      except ValueError as ve:
      print("Error:", str(ve))
      except serial.SerialException as se:
      print("Serial port error:", str(se))
      except Exception as e:
      print("An error occurred:", str(e))
      finally:
      # Ensure serial port is closed
      if ser and ser.is_open:
            ser.close()
            print("Serial port closed")

if __name__ == "__main__":
      main()
```

Клиентская программа эмулирует работу устройства, включая ошибочные сценарии для проверки устойчивости системы.

Временная диаграмма передачи одного пакета



Вывод

В ходе выполнения лабораторной работы были изучены принципы работы с контроллером UART на примере микроконтроллера ATmega328.

Реализована программа,
обеспечивающая обмен данными между микроконтроллером и ПК с проверкой целостности пакетов

Результаты работы демонстрируют эффективность аппаратного управления UART и важность проверки целостности данных в последовательных интерфейсах.