БГУИР

Кафедра ЭВМ

Лабораторная работа №1

Асинхронная двунаправленная

побайтная передача данных

Выполнил Проверил:

Студент группы 750502 Глоба А.А

Никанов И.В.

Минск 2019

**Цель работы:**

Разработать модуль асинхронной побайтной передачи данных, соответствующий физическому уровню модели OSI, на основе последовательных интерфейсов RS-232 и RS-485.

**Основные теоретические сведения:**

Протокол UART (Universal asynchronous receiver/transmitter) или, по-русски, УАПП (универсальный асинхронный приемопередатчик) — старейший и самый распространенный на сегодняшний день физический протокол передачи данных. Наиболее известен из семейства UART протокол RS-232.

Надо сказать, что изначально интерфейс УАПП появился в США как средство для передачи телеграфных сообщений, и рабочих бит там было пять (как в азбуке Морзе). Для передачи использовались механические устройства. Потом появились компьютеры, и коды ASCII, которые потребовали семь бит. В начале 60-х на смену пришла всем известная 8-битная таблица ASCII, и тогда формат передачи стал занимать полноценный байт, плюс управляющие три бита.

Основные рабочие линии у нас – RXD и TXD, или просто RX и TX. Передающая линия – TXD (Transmitted Data), а порт RXD (Received Data) – принимающая.  
Эти линии СОМ-порта задействованы при передаче без аппаратного управления потоком данных. При аппаратном потоке задействованы еще дополнительные интерфейсные линии (DTS, RTS и пр.). Выход передатчика TX соединен с входом приемника RX и наоборот. Электрический принцип работы RS-232 отличается от стандартной 5-вольтовой TTL логики. В этом протоколе логический нуль лежит от +3 до +12 вольт, а единица от -3 до -12, соответственно. Промежуток от -3 до +3 вольт считается зоной неопределенности. Учти, что все напряжения указаны относительно корпуса компьютера, или земли. Теперь, я думаю, ты понимаешь, зачем в компьютерном блоке питания существует сразу два напряжения: -12 и +12 вольт. Они были введены специально для работы СОМ-порта.

Стандартная посылка занимает 10 бит. Но правило это распространяется только на стандартные настройки СОМ-порта. В принципе, его можно перенастроить так, чтобы он даже интерфейс One-Wire понимал. В режиме простоя, когда по линии ничего не передается, она находится в состоянии логической единицы, или -12 вольт. Начало передачи обозначают передачей стартового бита, который всегда равен нулю. Затем идет передача восьми бит данных. Завершает посылку бит четности и стоповый бит. Бит четности осуществляет проверку переданных данных. Стартовый бит говорит нам, что пересылка данных завершена. Надо отметить, что STOP-бит может занимать 1, 1.5, и 2 бита. Не стоит думать, что это дробные биты, это число говорит только о его длительности. Стоповый бит, как и стартовый, равен нулю. Линейка скоростей СОМ-порта стандартизирована. Как правило, все устройства работают на трех стандартных скоростях: 9600, 19200, 115200. Но возможны другие варианты, даже использование нестандартных скоростей или скорости, меняющейся во времени.

Традиционное назначение цифровых цепей RS-232:

– SOUT (Serial Output) – выход передатчика;

– SIN (Serial Input) – вход приемника;

– RTS (Request to Send) – сигнал-запрос от UART к модему о передаче байта;

– CTS (Clear to Send) – сигнал-подтверждение от модема к UART о готовности принять байт для передачи;

– DSR (Data Set Ready) – сигнал от модема к UART о готовности к взаимодействию;

– DTR (Data Terminal Ready) – сигнал от UART к модему о готовности к взаимодействию;

– DCD (Data Carrier Detect) – сигнал от модема к UART об обнаружении данных;

– RI (Ring Indicator) – сигнал от модема к UART об обнаружении входящего телефонного звонка.

**Код лабораторной работы:**

Файл serial.java:

**public class** serial **implements** serialinter {  
 **private** SerialPort **serialPort**;  
 **private boolean opened**;  
 *//создание переменной ком-порта* **public** serial(String portName) {  
 **this**.**serialPort** = **new** SerialPort(portName);  
 **this**.**opened** = **false**;   
 }  
 *//открытие ком-порта* **public boolean** open() {  
 **try** {  
 **this**.**opened** = **serialPort**.openPort();  
 System.***out***.println(**"Port is opened."**);  
 **return this**.**opened**;  
 } **catch** (SerialPortException ex) {  
 ex.printStackTrace();  
 **return false**;  
 }  
 }  
 *//освобождение ком-порта* **public boolean** close() {  
 **try** {  
 **if** (**this**.**opened**) {  
 **this**.**opened** = **false**;  
 System.***out***.println(**"Port is closed."**);  
 **return serialPort**.closePort();  
 }  
 **return false**;  
 } **catch** (SerialPortException ex) {  
 ex.printStackTrace();  
 **return false**;  
 }  
 }  
 *//отправка данных* **public boolean** write(**byte**[] bytes, **boolean** flagCRC) {  
 **if**(){  
 **try** {  
 **return serialPort**.writeBytes(bytes);  
 } **catch** (SerialPortException ex) {  
 ex.printStackTrace();  
 **return false**;  
 }  
 } **else** {  
 **try** {  
 **return serialPort**.writeBytes(bytes);  
 } **catch** (SerialPortException ex) {  
 ex.printStackTrace();  
 **return false**;  
 }  
 }  
  
 }  
 *//прием данных* **public byte**[] read(**int** byteCount) {  
 **if**() {  
 **try** {  
 **byte**[] in = **serialPort**.readBytes(byteCount);  
 **return**(in);  
 } **catch** (SerialPortException ex) {  
 ex.printStackTrace();  
 **return null**;  
 }  
 } **else** {  
 **try** {  
 **byte**[] in = **serialPort**.readBytes(byteCount);  
 **return** in;  
 } **catch** (SerialPortException ex) {  
 ex.printStackTrace();  
 **return null**;  
 }  
 }  
 }  
 *//выставляем параметры: скорость передачи, размер данных, стоп-биты и проверка на четность* **public void** setParams(**int** baudRate, **int** dataBits, **int** stopBits, **int** parity) {  
 **try** {  
 **serialPort**.setParams(baudRate, dataBits, stopBits, parity);  
 } **catch** (SerialPortException ex) {  
 ex.printStackTrace();  
 }  
 }  
 *//обработчик событий* **public void** addListener(SerialPortEventListener listener) {  
 **try** {  
 **serialPort**.setEventsMask(SerialPort.***MASK\_RXCHAR***);  
 **serialPort**.addEventListener(listener);  
 } **catch** (SerialPortException ex) {  
 ex.printStackTrace();  
 }  
 }