|  |  |
| --- | --- |
|  | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ \_\_\_\_\_\_\_\_ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

КАФЕДРА \_\_\_\_\_\_\_СИСТЕМЫ ОБРАБОТКИ ИНФОРМАЦИИ И УПРАВЛЕНИЯ (ИУ5)\_\_\_\_

**РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

***К КУРСОВОЙ РАБОТЕ***

***НА ТЕМУ:***

***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

***\_\_\_Программа\_пересылки\_диалога\_абонентов\_\_\_\_\_\_***

***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

Студент \_\_\_\_ИУ5Ц-82Б\_\_\_\_ **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_А.Е.\_Чиварзин\_\_\_**

(Группа) (Подпись, дата) (И.О.Фамилия)

Руководитель курсовой работы **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_В.А.\_Галкин\_\_\_\_\_\_**

(Подпись, дата) (И.О.Фамилия)

Консультант **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_В.А.\_Галкин\_\_\_\_\_\_**

(Подпись, дата) (И.О.Фамилия)

*2022 г.*

**Министерство науки и высшего образования Российской Федерации**

**Федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«Московский государственный технический университет имени Н.Э. Баумана**

**(национальный исследовательский университет)»**

**(МГТУ им. Н.Э. Баумана)**

УТВЕРЖДАЮ

Заведующий кафедрой \_\_ИУ5\_\_\_\_\_

(Индекс)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(И.О.Фамилия)

« \_\_\_\_\_ » \_\_февраля\_\_\_ 20 \_22\_ г.

**ЗАДАНИЕ**

**на выполнение курсовой работы**

по дисциплине \_\_\_\_\_\_\_\_Сетевые\_технологии\_в\_АСОИУ\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Студент группы \_\_\_ИУ5Ц-82Б\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Чиварзин\_Александр\_Евгеньевич\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Фамилия, имя, отчество)

Тема курсовой работы \_\_\_\_Программа\_пересылки\_диалога\_абонентов\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Направленность КР (учебная, исследовательская, практическая, производственная, др.)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Источник тематики (кафедра, предприятие, НИР) \_\_\_кафедра\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

График выполнения работы: 25% к \_3\_ нед., 50% к \_9\_ нед., 75% к 12 нед., 100% к 15\_ нед.

***Задание*** \_*\_Разработать\_протоколы\_взаимодействия\_объектов\_до\_прикладного\_уровня\_\_\_\_\_\_\_\_\_*

*\_локальной\_сети,\_состоящей\_из\_2-х\_ПК,\_соединённых\_нульмодемно\_через\_интерфейс\_RS232C,\_и\_*

*\_реалезующий\_функцию\_передачи\_текста\_диалога\_абонентов.\_Принемаемый\_и\_передаваемый\_\_\_*

*\_тексты\_отображать\_в\_разных\_окнах.\_Скорость\_обмена\_и\_параметры\_COM-порта\_выбирает\_*

*\_пользователь\_одного\_из\_ПК.\_Передаваемую\_информацию\_защитить\_[7,4]\_кодом\_Хэминга.\_\_\_\_\_*

***Оформление курсовой работы:***

Расчетно-пояснительная записка на \_17\_ листах формата А4.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата выдачи задания « \_\_\_ » \_\_\_февраля\_\_ 2022 г.

**Руководитель курсовой работы**  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Подпись, дата) (И.О.Фамилия)

**Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

(Подпись, дата) (И.О.Фамилия)

Примечание: Задание оформляется в двух экземплярах: один выдается студенту, второй хранится на кафедре.

**Оглавление**

[1. Введение 4](#_Toc97923134)

[2. Требования к программе 4](#_Toc97923135)

[3. Требования к программе 4](#_Toc97923136)

[4. Физический уровень 5](#_Toc97923137)

[4.1 Сигналы интерфейса RS-232-C. 5](#_Toc97923138)

[4.2 Нуль-модемный интерфейс 8](#_Toc97923139)

[4.2.1. Настройка COM-порта средствами библиотеки для C# 10](#_Toc97923140)

[4.2.2. Описание класса SerialPort 10](#_Toc97923141)

[4.3. Функции физического уровня. 13](#_Toc97923142)

[4.4.1. Открытие порта 13](#_Toc97923143)

[4.4.2. Закрытие порта 14](#_Toc97923144)

[4.4.3. Передача данных 15](#_Toc97923145)

[4.4.4. Прием данных 15](#_Toc97923146)

[5. Канальный уровень 15](#_Toc97923147)

[5.1. Функции канального уровня 15](#_Toc97923148)

[5.2. Протокол связи 16](#_Toc97923149)

[5.3. Защита передаваемой информации 16](#_Toc97923150)

[5.4. Кодирование кодом Хемминга 17](#_Toc97923151)

1. Введение

Данная программа, выполненная в рамках курсовой работы по предмету «Сетевые технологии в автоматизированных системах обработки информации и управления», предназначена для пересылки текста диалога абонентов между двумя соединёнными с помощью интерфейса RS232C компьютерами.

1. Требования к программе

Программное изделие выполняется на С# под управлением MS Windows.

Для работы программы требуются 2 ПК типа IBM PC AT (/XT), соединенные нульмодемным кабелем через интерфейс RS-232C.

Важно, чтобы пользователь обладал правами администратора в ОС Windows на ПК.

1. Требования к программе

См. Схему «Структурная схема программы»

1. Физический уровень
   1. Сигналы интерфейса RS-232-C.

Последовательная передача данных означает, что данные передаются по единственной линии. При этом биты байта данных передаются по очереди с использованием одного провода. Для синхронизации группе битов данных обычно предшествует специальный *стартовый бит*, после группы битов следуют *бит проверки на четность* и один или два *стоповых бита,* как показано на рисунке 1. Иногда бит проверки на четность может отсутствовать.

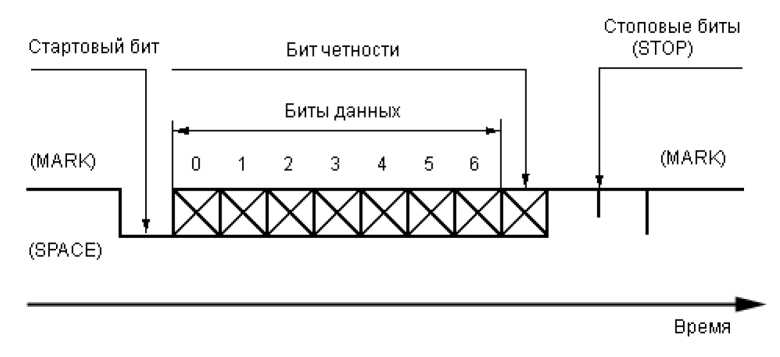


Рисунок 1.

Из рисунка видно, что исходное состояние линии последовательной передачи данных - уровень логической 1. Это состояние линии называют отмеченным — **MARK**. Когда начинается передача данных, уровень линии переходит в 0. Это состояние линии называют пустым — **SPACE**. Если линия находится в таком состоянии больше определенного времени, считается, что линия перешла в состояние разрыва связи — **BREAK**.

Стартовый бит **START** сигнализирует о начале передачи данных. Далее передаются биты данных, вначале младшие, затем старшие.

Контрольный бит формируется на основе правила, которое создается при настройке передающего и принимающего устройства. Контрольный бит может быть установлен с контролем на четность, нечетность, иметь постоянное значение 1 либо отсутствовать совсем.

Если используется бит четности **P**, то передается и он. Бит четности имеет такое значение, чтобы в пакете битов общее количество единиц (или нулей) было четно или нечетно, в зависимости от установки регистров порта. Этот бит служит для обнаружения ошибок, которые могут возникнуть при передаче данных из-за помех на линии. Приемное устройство заново вычисляет четность данных и сравнивает результат с принятым битом четности. Если четность не совпала, то считается, что данные переданы с ошибкой. Конечно, такой алгоритм не дает стопроцентной гарантии обнаружения ошибок. Так, если при передаче данных изменилось четное число битов, то четность сохраняется, и ошибка не будет обнаружена. Поэтому на практике применяют более сложные методы обнаружения ошибок.

В самом конце передаются один или два стоповых бита **STOP**, завершающих передачу байта. Затем до прихода следующего стартового бита линия снова переходит в состояние **MARK**.

Использование бита четности, стартовых и стоповых битов определяют формат передачи данных. Очевидно, что передатчик и приемник должны использовать один и тот же формат данных, иначе обмен будет невозможен.

Другая важная характеристика — скорость передачи. Она также должна быть одинаковой для передатчика и приемника.

Скорость изменения информативного параметра сигнала обычно измеряется в бодах.

Иногда используется другой термин — биты в секунду (bps). Здесь имеется в виду эффективная скорость передачи данных, без учета служебных битов.

Интерфейс RS232C описывает несимметричный интерфейс, работающий в режиме последовательного обмена двоичными данными. Интерфейс поддерживает как асинхронный, так и синхронный режимы работы.

Интерфейс называется несимметричным, если для всех цепей обмена интерфейса используется один общий возвратный провод — сигнальная «земля».

Интерфейсы 25-ти (DB25) или 9-ти (DB9) контактный разъем.

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование сигнала | Цепь | Номер контакта | |
| DB25P | DB9S |
| DCD (Data Carrier Detect) | 109 | 8 | 1 |
| RD (Receive Data) | 104 | 3 | 2 |
| TD (Transmit Data) | 103 | 2 | 3 |
| DTR (Data Terminal Ready) | 108 | 20 | 4 |
| GND (Signal Ground) | 102 | 7 | 5 |
| DSR (Data Set Ready) | 107 | 6 | 6 |
| RTS (Reguest To Send) | 105 | 4 | 7 |
| CTS (Clear To Send) | 106 | 5 | 8 |
| RI (Ring Indicator) | 125 | 22 | 9 |

Таблица 1.

В интерфейсе реализован биполярный потенциальный код на линиях между DTE и DCE. Напряжения сигналов в цепях обмена симметричны по отношению к уровню сигнальной «земли» и составляют не менее +3В для двоичного нуля и не более -3В для двоичной единицы.

Входы TD и RD используются устройствами DTE и DCE по-разному. DTE использует вход TD для передачи данных, а вход RD для приема данных. И наоборот, устройство DCE использует вход TD для приема, а вход RD для передачи данных. Поэтому для соединения двух DTE необходимо перекрестное соединение линий TD и RD в нуль-модемном кабеле.

Рассмотрим самый низкий уровень управления связью - подтверждение связи.

В начале сеанса связи компьютер (DTE) должен удостовериться, что модем (DCE) находится в рабочем состоянии. Для этой цели компьютер подает сигнал по линии DTR. В ответ модем подает сигнал по линии DSR. Затем, после вызова абонента, модем подает сигнал по линии DCD, чтобы сообщить компьютеру, что он произвел соединение с удаленной системой.

Более высокий уровень используется для управления потоком (скоростью обмена данными) и также реализуется аппаратно. Этот уровень необходим для того, чтобы предотвратить передачу большего числа данных, чем то, которое может быть обработано принимающей системой.

В полудуплексных соединениях DTE подает сигнал RTS, когда оно желает передать данные. DCE отвечает сигналом по линии CTS, когда оно готово, и DTE начинает передачу данных. До тех пор, пока оба сигнала RTS и CTS не примут активное состояние, только DCE может передавать данные. Иногда для сое­динения двух устройств DTE эти линии (RTS и CTS) соединяются вместе на каждом конце кабеля. В результате получаем то, что другое устройство всегда готово для получения данных (если при большой скорости передачи принимающее устройство не успевает принимать и обрабатывать данные, возможна потеря данных).

Для решения всех этих проблем для соединения двух устройств типа DTE используется специальный нуль-модемный кабель.

* 1. Нуль-модемный интерфейс

Обмен сигналами между адаптером компьютера (DTE) и модемом (DCE) (или 2-м компьютером, присоединенным к исходному посредством кабеля стандарта RS-232C) строится по стандартному сценарию, в котором каждый сигнал генерируется сторонами лишь после наступления определенных условий. Такая процедура обмена информацией называется запрос/ответным режимом, или “**рукопожатием**” (**handshaking**). Большинство из приведенных в таблице сигналов как раз и нужны для аппаратной реализации “рукопожатия” между адаптером и модемом.

Обмен сигналами между сторонами интерфейса **RS-232C** выглядит так:

1. компьютер после включения питания и открытия СОМ-порта выставляет сигнал **DTR**, который удерживается активным. Если модем включен в электросеть и исправен, он отвечает компьютеру сигналом **DSR**. Этот сигнал служит подтверждением того, что **DTR** принят, и информирует компьютер о готовности модема к приему информации;
2. если компьютер получил сигнал **DSR** и хочет передать данные, он выставляет сигнал **RTS**;
3. если модем готов принимать данные, он отвечает сигналом **CTS**. Он служит для компьютера подтверждением того, что **RTS** получен модемом и модем готов принять данные от компьютера. С этого момента адаптер может бит за битом передавать информацию по линии **TD**;
4. получив байт данных, модем может сбросить свой сигнал **CTS**, информируя компьютер о необходимости “притормозить” передачу следующего байта, например, из-за переполнения внутреннего буфера; программа компьютера, обнаружив сброс **CTS**, прекращает передачу данных, ожидая повторного появления **CTS**.

Модем может передать данные в компьютер, когда он обнаружит несущую в линии и выставит сигнал — **DCD**. Программа компьютера, принимающая данные, обнаружив этот сигнал, читает приемный регистр, в который сдвиговый регистр “собрал” биты, принятые по линии приема данных **RD**. Когда для связи используются только приведенные в таблице данные, компьютер не может попросить модем “повременить” с передачей следующего байта. Как следствие, существует опасность переопределения помещенного ранее в приемном регистре байта данных вновь “собранным” байтом. Поэтому при приеме информации компьютер должен очень быстро освобождать приемный регистр адаптера. В полном наборе сигналов **RS-232C** есть линии, которые могут аппаратно “приостановить” модем.

Нуль-модемный интерфейс характерен для прямой связи компьютеров на небольшом расстоянии (длина кабеля до 15 метров). Для нормальной работы двух непосредственно соединенных компьютеров нуль-модемный кабель должен выполнять следующие соединения:

1. RI-1 + DSR-1 — DTR-2;
2. DTR-1 — RI-2 + DSR-2;
3. CD-1 — CTS-2 + RTS-2;
4. CTS-1 + RTS-1 — CD-2;
5. RD-1 — TD-2;
6. TD-1 — RD-2;
7. SG-1 — SG-2;

Знак «+» обозначает соединение соответствующих контактов на одной стороне кабеля.

* + 1. Настройка COM-порта средствами библиотеки для C#

Прикладная библиотека Serial COM для C# предлагает широкие возможности по настройке COM-порта. Подробное описание библиотеки

[https://docs.microsoft.com/ru-](https://docs.microsoft.com/ru-ru/dotnet/api/system.io.ports.serialport?view=netframework-4.8)

[ru/dotnet/api/system.io.ports.serialport?view=netframework-4.8](https://docs.microsoft.com/ru-ru/dotnet/api/system.io.ports.serialport?view=netframework-4.8)

* + 1. Описание класса SerialPort

Класс SerialPort дает возможность управления последовательными портами компьютера. Он определяет минимальную функциональность для работы с ними.

* + 1. **Конструкторы класса**

|  |  |
| --- | --- |
| [**SerialPortQ**](https://docs.microsoft.com/ru-ru/dotnet/api/system.io.ports.serialport.-ctor?view=netframework-4.8%23System_IO_Ports_SerialPort__ctor) | Инициализация нового экземпляра  класса [SerialPort.](https://docs.microsoft.com/ru-ru/dotnet/api/system.io.ports.serialport?view=netframework-4.8) |

|  |  |
| --- | --- |
| [**SerialPort(IContainer)**](https://docs.microsoft.com/ru-ru/dotnet/api/system.io.ports.serialport.-ctor?view=netframework-4.8%23System_IO_Ports_SerialPort__ctor_System_ComponentModel_IContainer_) | Инициализирует новый экземпляр  класса [SerialPort,](https://docs.microsoft.com/ru-ru/dotnet/api/system.io.ports.serialport?view=netframework-4.8) используя указанный  объект [IContainer.](https://docs.microsoft.com/ru-ru/dotnet/api/system.componentmodel.icontainer?view=netframework-4.8) |
| [**SerialPort(String)**](https://docs.microsoft.com/ru-ru/dotnet/api/system.io.ports.serialport.-ctor?view=netframework-4.8%23System_IO_Ports_SerialPort__ctor_System_String_) | Инициализирует новый экземпляр  класса [SerialPort,](https://docs.microsoft.com/ru-ru/dotnet/api/system.io.ports.serialport?view=netframework-4.8) используя указанное имя порта. |
| [**SerialPort(String, Int32)**](https://docs.microsoft.com/ru-ru/dotnet/api/system.io.ports.serialport.-ctor?view=netframework-4.8%23System_IO_Ports_SerialPort__ctor_System_String_System_Int32_) | Инициализирует новый экземпляр  класса [SerialPort,](https://docs.microsoft.com/ru-ru/dotnet/api/system.io.ports.serialport?view=netframework-4.8) используя указанное имя порта и скорость передачи в бодах. |
| [**SerialPort(String, Int32,**](https://docs.microsoft.com/ru-ru/dotnet/api/system.io.ports.serialport.-ctor?view=netframework-4.8%23System_IO_Ports_SerialPort__ctor_System_String_System_Int32_System_IO_Ports_Parity_)[**Parity)**](https://docs.microsoft.com/ru-ru/dotnet/api/system.io.ports.serialport.-ctor?view=netframework-4.8%23System_IO_Ports_SerialPort__ctor_System_String_System_Int32_System_IO_Ports_Parity_) | Инициализирует новый экземпляр  класса [SerialPort,](https://docs.microsoft.com/ru-ru/dotnet/api/system.io.ports.serialport?view=netframework-4.8) используя указанное имя порта, скорость передачи в бодах и бит четности. |
| [**SerialPort(String, Int32,**](https://docs.microsoft.com/ru-ru/dotnet/api/system.io.ports.serialport.-ctor?view=netframework-4.8%23System_IO_Ports_SerialPort__ctor_System_String_System_Int32_System_IO_Ports_Parity_System_Int32_)[**Parity, Int32)**](https://docs.microsoft.com/ru-ru/dotnet/api/system.io.ports.serialport.-ctor?view=netframework-4.8%23System_IO_Ports_SerialPort__ctor_System_String_System_Int32_System_IO_Ports_Parity_System_Int32_) | Инициализирует новый экземпляр  класса [SerialPort,](https://docs.microsoft.com/ru-ru/dotnet/api/system.io.ports.serialport?view=netframework-4.8) используя указанное имя порта, скорость передачи в бодах, бит четности и биты данных. |
| [**SerialPort(String, Int32,**](https://docs.microsoft.com/ru-ru/dotnet/api/system.io.ports.serialport.-ctor?view=netframework-4.8%23System_IO_Ports_SerialPort__ctor_System_String_System_Int32_System_IO_Ports_Parity_System_Int32_System_IO_Ports_StopBits_)[**Parity, Int32, StopBits)**](https://docs.microsoft.com/ru-ru/dotnet/api/system.io.ports.serialport.-ctor?view=netframework-4.8%23System_IO_Ports_SerialPort__ctor_System_String_System_Int32_System_IO_Ports_Parity_System_Int32_System_IO_Ports_StopBits_) | Инициализирует новый экземпляр  класса [SerialPort,](https://docs.microsoft.com/ru-ru/dotnet/api/system.io.ports.serialport?view=netframework-4.8) используя указанное имя порта, скорость передачи в бодах, бит четности, биты данных и стоп-бит. |

Таблица 2.

**4.2.4. Методы**

|  |  |
| --- | --- |
| [**Close()**](https://docs.microsoft.com/ru-ru/dotnet/api/system.io.ports.serialport.close?view=netframework-4.8%23System_IO_Ports_SerialPort_Close) | Закрывает соединение порта, присваивает свойству [IsOpen](https://docs.microsoft.com/ru-ru/dotnet/api/system.io.ports.serialport.isopen?view=netframework-4.8%23System_IO_Ports_SerialPort_IsOpen) значение false и уничтожает внутренний объект [Stream.](https://docs.microsoft.com/ru-ru/dotnet/api/system.io.stream?view=netframework-4.8) |
| [**CreateObjRef(Type)**](https://docs.microsoft.com/ru-ru/dotnet/api/system.marshalbyrefobject.createobjref?view=netframework-4.8%23System_MarshalByRefObject_CreateObjRef_System_Type_) | Creates an object that contains all the relevant information required to generate a proxy used to communicate with a remote object.  (Унаследовано от [MarshalByRefObject)](https://docs.microsoft.com/ru-ru/dotnet/api/system.marshalbyrefobject?view=netframework-4.8) |
| [**DiscardInBuffer()**](https://docs.microsoft.com/ru-ru/dotnet/api/system.io.ports.serialport.discardinbuffer?view=netframework-4.8%23System_IO_Ports_SerialPort_DiscardInBuffer) | Удаляет данные из буфера приема последовательного драйвера. |
| [**DiscardOutBuffer()**](https://docs.microsoft.com/ru-ru/dotnet/api/system.io.ports.serialport.discardoutbuffer?view=netframework-4.8%23System_IO_Ports_SerialPort_DiscardOutBuffer) | Удаляет данные из буфера передачи последовательного драйвера. |
| [**Dispose()**](https://docs.microsoft.com/ru-ru/dotnet/api/system.componentmodel.component.dispose?view=netframework-4.8%23System_ComponentModel_Component_Dispose) | Освобождает все ресурсы, занятые [Component.](https://docs.microsoft.com/ru-ru/dotnet/api/system.componentmodel.component?view=netframework-4.8) (Унаследовано от [Component)](https://docs.microsoft.com/ru-ru/dotnet/api/system.componentmodel.component?view=netframework-4.8) |
| [**Dispose(Boolean)**](https://docs.microsoft.com/ru-ru/dotnet/api/system.io.ports.serialport.dispose?view=netframework-4.8%23System_IO_Ports_SerialPort_Dispose_System_Boolean_) | Освобождает неуправляемые ресурсы, используемые [SerialPort,](https://docs.microsoft.com/ru-ru/dotnet/api/system.io.ports.serialport?view=netframework-4.8) и дополнительно освобождает управляемые ресурсы. |
| [**Equals(Object)**](https://docs.microsoft.com/ru-ru/dotnet/api/system.object.equals?view=netframework-4.8%23System_Object_Equals_System_Object_) | Определяет, равен ли указанный объект текущему объекту.  (Унаследовано от [Object)](https://docs.microsoft.com/ru-ru/dotnet/api/system.object?view=netframework-4.8) |
| [**GetHashCode()**](https://docs.microsoft.com/ru-ru/dotnet/api/system.object.gethashcode?view=netframework-4.8%23System_Object_GetHashCode) | Служит в качестве хэш-функции по умолчанию. (Унаследовано от [Object)](https://docs.microsoft.com/ru-ru/dotnet/api/system.object?view=netframework-4.8) |

|  |  |
| --- | --- |
| [**GetLifetimeService()**](https://docs.microsoft.com/ru-ru/dotnet/api/system.marshalbyrefobject.getlifetimeservice?view=netframework-4.8%23System_MarshalByRefObject_GetLifetimeService) | Retrieves the current lifetime service object that controls the lifetime policy for this instance. (Унаследовано от [MarshalByRefObject)](https://docs.microsoft.com/ru-ru/dotnet/api/system.marshalbyrefobject?view=netframework-4.8) |
| [**GetPortNames()**](https://docs.microsoft.com/ru-ru/dotnet/api/system.io.ports.serialport.getportnames?view=netframework-4.8%23System_IO_Ports_SerialPort_GetPortNames) | Получает массив имен последовательных портов для текущего компьютера. |
| [**GetService(Type)**](https://docs.microsoft.com/ru-ru/dotnet/api/system.componentmodel.component.getservice?view=netframework-4.8%23System_ComponentModel_Component_GetService_System_Type_) | Возвращает объект, представляющий службу, предоставляемую классом [Component](https://docs.microsoft.com/ru-ru/dotnet/api/system.componentmodel.component?view=netframework-4.8) или классом [Container.](https://docs.microsoft.com/ru-ru/dotnet/api/system.componentmodel.container?view=netframework-4.8)  (Унаследовано от [Component)](https://docs.microsoft.com/ru-ru/dotnet/api/system.componentmodel.component?view=netframework-4.8) |
| [**GetType()**](https://docs.microsoft.com/ru-ru/dotnet/api/system.object.gettype?view=netframework-4.8%23System_Object_GetType) | Возвращает объект [Type](https://docs.microsoft.com/ru-ru/dotnet/api/system.type?view=netframework-4.8) для текущего экземпляра.  (Унаследовано от [Object)](https://docs.microsoft.com/ru-ru/dotnet/api/system.object?view=netframework-4.8) |
| [**InitializeLifetimeService()**](https://docs.microsoft.com/ru-ru/dotnet/api/system.marshalbyrefobject.initializelifetimeservice?view=netframework-4.8%23System_MarshalByRefObject_InitializeLifetimeService) | Obtains a lifetime service object to control the lifetime policy for this instance.  (Унаследовано от [MarshalByRefObject)](https://docs.microsoft.com/ru-ru/dotnet/api/system.marshalbyrefobject?view=netframework-4.8) |
| [**MemberwiseClone()**](https://docs.microsoft.com/ru-ru/dotnet/api/system.object.memberwiseclone?view=netframework-4.8%23System_Object_MemberwiseClone) | Создает неполную копию текущего объекта [Object.](https://docs.microsoft.com/ru-ru/dotnet/api/system.object?view=netframework-4.8)  (Унаследовано от [Object)](https://docs.microsoft.com/ru-ru/dotnet/api/system.object?view=netframework-4.8) |
| [**MemberwiseClone(Boolean)**](https://docs.microsoft.com/ru-ru/dotnet/api/system.marshalbyrefobject.memberwiseclone?view=netframework-4.8%23System_MarshalByRefObject_MemberwiseClone_System_Boolean_) | Creates a shallow copy of the current [MarshalByRefObj ect](https://docs.microsoft.com/ru-ru/dotnet/api/system.marshalbyrefobject?view=netframework-4.8) object. (Унаследовано от [MarshalByRefObject)](https://docs.microsoft.com/ru-ru/dotnet/api/system.marshalbyrefobject?view=netframework-4.8) |
| [**Open()**](https://docs.microsoft.com/ru-ru/dotnet/api/system.io.ports.serialport.open?view=netframework-4.8%23System_IO_Ports_SerialPort_Open) | Открывает новое соединение последовательного порта. |
| [**Read(Byte[], Int32, Int32)**](https://docs.microsoft.com/ru-ru/dotnet/api/system.io.ports.serialport.read?view=netframework-4.8%23System_IO_Ports_SerialPort_Read_System_Byte___System_Int32_System_Int32_) | Считывает из входного  буфера [SerialPort](https://docs.microsoft.com/ru-ru/dotnet/api/system.io.ports.serialport?view=netframework-4.8) определенное число байтов и записывает их в байтовый массив, начиная с указанной позиции. |
| [**Read(Char[], Int32, Int32)**](https://docs.microsoft.com/ru-ru/dotnet/api/system.io.ports.serialport.read?view=netframework-4.8%23System_IO_Ports_SerialPort_Read_System_Char___System_Int32_System_Int32_) | Считывает из входного  буфера [SerialPort](https://docs.microsoft.com/ru-ru/dotnet/api/system.io.ports.serialport?view=netframework-4.8) определенное число символов и записывает их в символьный массив, начиная с указанной позиции. |
| [**ReadByte()**](https://docs.microsoft.com/ru-ru/dotnet/api/system.io.ports.serialport.readbyte?view=netframework-4.8%23System_IO_Ports_SerialPort_ReadByte) | Считывает из входного буфера [SerialPort](https://docs.microsoft.com/ru-ru/dotnet/api/system.io.ports.serialport?view=netframework-4.8) один байт в синхронном режиме. |
| [**ReadChar()**](https://docs.microsoft.com/ru-ru/dotnet/api/system.io.ports.serialport.readchar?view=netframework-4.8%23System_IO_Ports_SerialPort_ReadChar) | Считывает из входного буфера [SerialPort](https://docs.microsoft.com/ru-ru/dotnet/api/system.io.ports.serialport?view=netframework-4.8) один символ в синхронном режиме. |
| [**ReadExisting()**](https://docs.microsoft.com/ru-ru/dotnet/api/system.io.ports.serialport.readexisting?view=netframework-4.8%23System_IO_Ports_SerialPort_ReadExisting) | Считывает все непосредственно доступные байты в соответствии с кодировкой из потока и из входного буфера объекта [SerialPort.](https://docs.microsoft.com/ru-ru/dotnet/api/system.io.ports.serialport?view=netframework-4.8) |
| [**ReadLine()**](https://docs.microsoft.com/ru-ru/dotnet/api/system.io.ports.serialport.readline?view=netframework-4.8%23System_IO_Ports_SerialPort_ReadLine) | Считывает данные из входного буфера до значения [NewLine.](https://docs.microsoft.com/ru-ru/dotnet/api/system.io.ports.serialport.newline?view=netframework-4.8%23System_IO_Ports_SerialPort_NewLine) |
| [**ReadTo(String)**](https://docs.microsoft.com/ru-ru/dotnet/api/system.io.ports.serialport.readto?view=netframework-4.8%23System_IO_Ports_SerialPort_ReadTo_System_String_) | Считывает из входного буфера строку до указанного значения value. |
| [**ToString()**](https://docs.microsoft.com/ru-ru/dotnet/api/system.componentmodel.component.tostring?view=netframework-4.8%23System_ComponentModel_Component_ToString) | Возвращает объект [String,](https://docs.microsoft.com/ru-ru/dotnet/api/system.string?view=netframework-4.8) содержащий имя [Component,](https://docs.microsoft.com/ru-ru/dotnet/api/system.componentmodel.component?view=netframework-4.8) если оно есть. Этот метод не следует переопределять.  (Унаследовано от [Component)](https://docs.microsoft.com/ru-ru/dotnet/api/system.componentmodel.component?view=netframework-4.8) |
| [**Write(Byte[], Int32, Int32)**](https://docs.microsoft.com/ru-ru/dotnet/api/system.io.ports.serialport.write?view=netframework-4.8%23System_IO_Ports_SerialPort_Write_System_Byte___System_Int32_System_Int32_) | Записывает указанное число байтов в последовательный порт, используя данные из буфера. |
| [**Write(Char[], Int32, Int32)**](https://docs.microsoft.com/ru-ru/dotnet/api/system.io.ports.serialport.write?view=netframework-4.8%23System_IO_Ports_SerialPort_Write_System_Char___System_Int32_System_Int32_) | Записывает указанное число символов в последовательный порт, используя данные из буфера. |
| [**Write(String)**](https://docs.microsoft.com/ru-ru/dotnet/api/system.io.ports.serialport.write?view=netframework-4.8%23System_IO_Ports_SerialPort_Write_System_String_) | Записывает указанную строку в последовательный порт. |
| [**WriteLine(String)**](https://docs.microsoft.com/ru-ru/dotnet/api/system.io.ports.serialport.writeline?view=netframework-4.8%23System_IO_Ports_SerialPort_WriteLine_System_String_) | Записывает указанную строку и значение [NewLine](https://docs.microsoft.com/ru-ru/dotnet/api/system.io.ports.serialport.newline?view=netframework-4.8%23System_IO_Ports_SerialPort_NewLine) в выходной буфер. |

Таблица 3.

4.3. Функции физического уровня.

4.4.1. Открытие порта

В ОС Windows доступ к COM-портам предоставляется посредством файловых интерфейсов. В данной работе доступ предоставляется на основе класса SerialPort. Этот класс используется для управления файловым ресурсом последовательного порта. Данный класс предоставляет возможности управления вводом-выводом в синхронном режиме или на основе событий, доступа к состоянию линии и состоянию разрыва, а также доступа к свойствам последовательного драйвера.

Соответственно, для открытия порта в языке C# используется функция класса SerialPort Open(). В качестве параметров этому методу передаются параметры, настраивающие передачу данных на физическом уровне, упомянутые выше, а также имя доступного COM-порта для подключения. При успешном выполнении функции свойство IsOpen экземпляра класса SerialPort становится равным true, в противном случае генерируется одно из следующих исключений:

* [UnauthorizedAccessException](http://msdn.microsoft.com/ru-ru/library/system.unauthorizedaccessexception.aspx)
* [ArgumentOutOfRangeException](http://msdn.microsoft.com/ru-ru/library/system.argumentoutofrangeexception.aspx)
* [ArgumentException](http://msdn.microsoft.com/ru-ru/library/system.argumentexception.aspx)
* [lOException](http://msdn.microsoft.com/ru-ru/library/system.io.ioexception.aspx)
* [InvalidOperationException](http://msdn.microsoft.com/ru-ru/library/system.invalidoperationexception.aspx)

Тайм-ауты необходимы для правильной работы функций чтения из порта и записи в порт. В программе определяются следующие значения:

* ReadTimeout определяет срок ожидания в миллисекундах для завершения операции чтения из порта.
* WriteTimeout определяет срок ожидания в миллисекундах для завершения операции записи в порт.

По истечении данных таймаутов в случае неуспешной записи/чтения генерируется исключение TimeoutException - задает сообщение с описанием источника исключения. Если метод вызывает это исключение, обычно выдается сообщение "Операция не завершена в связи с истечением тайм-аута".

Также при выполнении метода экземпляру SerialPort предоставляется базовый объект Stream для работы, - в целом, класс Stream является абстрактным базовым классом всех потоков. Поток — это абстракция последовательности байтов, например файл, устройство ввода-вывода, канал взаимодействия процессов или сокет TCP/IP. В данном случае Steam представляет собой абстракцию для канала, осуществляющего запись данных в порт.

* + 1. Закрытие порта

Закрытие порта осуществляется путем вызова метода Close() на экземпляре класса SerialPort. Данный метод закрывает соединение порта, присваивает свойству [IsOpen](http://msdn.microsoft.com/ru-ru/library/system.io.ports.serialport.isopen.aspx) значение false и уничтожает внутренний объект [Stream.](http://msdn.microsoft.com/ru-ru/library/system.io.stream.aspx)

* + 1. Передача данных

В данной работе передача данных между двумя компьютерами осуществляется вызова метода [Write(Byte[], Int32, Int32)](http://msdn.microsoft.com/ru-ru/library/ms143551.aspx) класса SerialPort.

Для передачи данных в классе SerialPort существует набор методов Write, отличающихся параметрами:

* Записывает указанную строку в последовательный порт.
* Записывает указанное число байтов в последовательный порт, используя данные из буфера.

• Записывает указанное число символов в последовательный порт, используя данные из буфера.

4.4.4. Прием данных

Прием данных реализуется как обработчик события DataReceived класса SerialPort, - данное событие активизируется при наличии данных во входном буфере порта. В рамках физического уровня обработчика данные считываются с помощью набора методов для считывания данных Read, отличающихся принимаемыми параметрами:

* Считывает из входного буфера SerialPort определенное число байтов и записывает их в байтовый массив, начиная с указанной позиции.
* ReadByte - Считывает из входного буфера SerialPort один байт в синхронном режиме.
* ReadChar - Считывает из входного буфера SerialPort один символ в синхронном режиме.
* ReadExisting - Считывает все непосредственно доступные байты в соответствии с кодировкой из потока и из входного буфера объекта SerialPort.

Для анализа числа поступивших батов используется свойство [BytesToRead,](http://msdn.microsoft.com/ru-ru/library/system.io.ports.serialport.bytestoread.aspx) получающее число байтов данных, находящихся в буфере приема.

1. Канальный уровень
   1. Функции канального уровня

На канальном уровне должны выполняться следующие функции:

1. Запрос логического соединения;
2. Управление передачей кадров;
3. Обеспечение необходимой последовательности блоков данных, передаваемых через межуровневый интерфейс;
4. Контроль и обработка ошибок;
5. Проверка целостности логического соединения;
6. Посылка подтверждения.
7. Запрос на разъединение логического соединения.
   1. Протокол связи

Важнейшими понятиями протокола связи являются управление ошибками и управление потоком. В основном протокол содержит набор соглашений или правил, которого должны придерживаться обе стороны связи для обеспечения получения и корректной интерпретации информации, передаваемой между двумя сторонами. Таким образом, помимо управления ошибками и потоком протокол связи регулирует также следующие вопросы

* формат передаваемых данных (число битов на каждый элемент и тип используемой схемы кодирования);
* тип и порядок сообщений, подлежащих обмену для обеспечения (свободной от ошибок и дубликатов) передачи информации между двумя взаимодействующими сторонами.

Например, перед началом передачи данных требуется установить соединение между двумя сторонами, тем самым проверяется доступность приемного устройства и его готовность воспринимать данные. Для этого передающее устройство посылает специальную команду: запрос на соединение и ожидает ее приема с другого COM-порта.

* 1. Защита передаваемой информации

При передаче данных по линиям могут возникать ошибки, вызванные электрическими помехами, связанными, например, с шумами, порожденными коммутирующими элементами сети. Эти помехи могут вызвать множество ошибок в цепочке последовательных битов.

Метод четности/нечетности или полученная в развитие этого метода контрольная сумма блока не обеспечивают надежного обнаружения пачки ошибок. Для этих случаев чаще всего применяется альтернативный метод, основанный на полиномиальных кодах. Полиномиальные коды используются в схемах покадровой (или поблочной) передачи. Это означает, что для каждого передаваемого кадра формируется один-единственный набор контрольных разрядов, значения которых зависят от фактического содержания кадра. Приемник выполняет те же вычисления с полным содержимым кадра; если при передаче ошибки не возникли, то в результате вычислений должен быть получен заранее известный ответ. Если этот ответ не совпадает с ожидаемым, то это указывает на наличие ошибок.

В данной программе передаваемая информация защищена кодом Хемминга [7,4].

* 1. Кодирование кодом Хемминга

Алгоритм кодирования состоит в том, что …