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

**Московский государственный технический университет им. Н.Э.Баумана**

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

Утверждаю:

Галкин В.А.

"\_\_"\_\_\_\_\_\_\_\_\_\_\_\_\_ 2019 г.

Курсовая работа по курсу «Сетевые технологии в АСОИУ»

**«Локальная безадаптерная сеть»**

Пояснительная записка

(вид документа)

писчая бумага

(вид носителя)

19

(количество листов)

Исполнители: Абросимова Н.Г. ИУ5-61

Гладова А.Г. ИУ5-61

Корнеева А.П. ИУ5-64

Москва 2019 г

**Оглавление**

[1. Введение 3](#_Toc8864031)

[2. Требования к программе 3](#_Toc8864032)

[3. Определение структуры программного продукта 3](#_Toc8864033)

[4. Физический уровень 3](#_Toc8864034)

[4.1. Функции физического уровня 3](#_Toc8864035)

[4.2. Описание физического уровня. 4](#_Toc8864036)

[4.3. Нуль-модемный интерфейс. 6](#_Toc8864037)

[5. Настройка COM-порта средствами C# 7](#_Toc8864038)

[5.1. Описание класса SerialPort 7](#_Toc8864039)

[5.2. Описание класса CommunicationManager 11](#_Toc8864040)

[6. Канальный уровень 12](#_Toc8864041)

[6.1. Функции канального уровня 12](#_Toc8864042)

[6.2. Протокол связи 12](#_Toc8864043)

[6.3. Защита передаваемой информации 13](#_Toc8864044)

[6.4. Передача данных. 15](#_Toc8864045)

[6.5. Функции кодирования/декодирования. 15](#_Toc8864046)

[6.6. Форматы кадров. 16](#_Toc8864047)

[6.7. Информационный и управляющий кадры. 16](#_Toc8864048)

[7. Прикладной уровень 16](#_Toc8864049)

1. Введение

Данная программа, выполненная в рамках курсовой работы по предмету «Сетевые технологии в АСОИУ», предназначена для организации передачи текстовых файлов между соединёнными с помощью интерфейса RS232C компьютерами. Программа позволяет обмениваться текстовыми файлами между 2 ПК, соединёнными между собой через интерфейс RS232C нульмодемным кабелем.

2. Требования к программе

К программе предъявляются следующие требования. Программа должна:

1. Устанавливать соединение между компьютерами и контролировать его целостность;
2. Обеспечивать правильность передачи и приема данных с помощью алгоритма циклического кодирования пакета;
3. Обеспечивать функцию передачи текстовых файлов;
4. Программа должна выполняться под управлением OS Windows 7 или выше. Поэтому было решено выполнить реализацию программы с помощью среды разработки MS Visual Studio 2015.

3. Определение структуры программного продукта

При взаимодействии компьютеров между собой выделяются несколько уровней: нижний уровень должен обеспечивать соединение компьютера со средой передачи, а верхний – обеспечить интерфейс пользователя. Программа разбивается на три уровня: физический, канальный и прикладной (см. Приложение «Структурная схема программы»).

- Физический уровень предназначен для сопряжения компьютера со средой передачи.

- Канальный уровень занимается установлением и поддержанием соединения, формированием и проверкой пакетов обмена протоколов верхних модулей.

- Прикладной уровень занимается выполнением задач программы.

4. Физический уровень

4.1. Функции физического уровня

Основными функциями физического уровня являются:

1. Задание параметров COM-порта.

2. Установление физического канала.

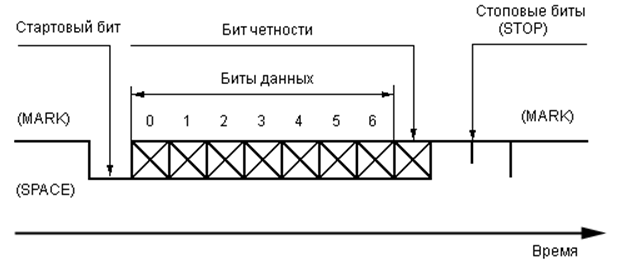
3. Разъединение физического канала.

4. Передача информации из буфера в интерфейс.

5. Прием информации и ее накопление в буфере.

4.2. Описание физического уровня.

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* (см. Рис. 1). Иногда бит проверки на четность может отсутствовать.



*Рис. 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 данных. Она также должна быть одинаковой для передатчика и приемника.

Скорость передачи данных 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.

Последовательная передача данных означает, что данные передаются по единственной линии. При этом биты байта данных передаются по очереди с использованием одного провода. Интерфейс называется несимметричным, если для всех цепей обмена интерфейса используется один общий возвратный провод – сигнальная «земля».

*Табл. 1. Интерфейс 9-ти контактный разъем.*

|  |  |  |  |
| --- | --- | --- | --- |
| **Номер контакта** | **Обозначение** | **Назначение** | **Обозначение CCITT** |
| 1 | DCD | Обнаружение несущей | 109 |
| 2 | RD | Принимаемые данные | 104 |
| 3 | TD | Отправляемые данные | 103 |
| 4 | DTR | Готовность терминала к работе | 108/2 |
| 5 | SG | Земля сигнала (схемная) | 102 |
| 6 | DSR | Готовность DCE | 107 |
| 7 | RTS | Запрос передачи | 105 |
| 8 | TD | Отправляемые данные | 103 |
| 9 | RI | Индикатор вызова | 125 |

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.

Каждый байт данных сопровождается специальными сигналами «старт» – стартовый бит и «стоп» – стоповый бит. Сигнал «старт» имеет продолжительность в один тактовый интервал, а сигнал «стоп» может длиться один, полтора или два такта.

При синхронной передаче данных через интерфейс передаются сигналы синхронизации, без которых компьютер не может правильно интерпретировать потенциальный код, поступающий по линии RD.

4.3. Нуль-модемный интерфейс.

Обмен сигналами между адаптером компьютера и модемом (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 выставляет сигнал **DTR**, который постоянно 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**.

Когда модему необходимо передать данные в компьютер, он (модем) выставляет сигнал на разъеме 8 – **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-1;
6. TD-1 — RD-1;
7. SG-1 — SG-2;

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

5. Настройка COM-порта средствами C#

Пространство имен System.IO.Ports предлагает широкие возможности по настройке COM-порта.

## **5.1. Описание 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.

*Табл. 2. Методы класса.*

|  |  |
| --- | --- |
| **Имя** | **Описание** |
| [Close](http://msdn.microsoft.com/ru-ru/library/system.io.ports.serialport.close.aspx) | 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). |
| [CreateObjRef](http://msdn.microsoft.com/ru-ru/library/system.marshalbyrefobject.createobjref.aspx) | Создает объект, который содержит всю необходимую информацию для создания прокси-сервера, используемого для взаимодействия с удаленным объектом. (Унаследовано от [MarshalByRefObject](http://msdn.microsoft.com/ru-ru/library/system.marshalbyrefobject.aspx).) |
| [DiscardInBuffer](http://msdn.microsoft.com/ru-ru/library/system.io.ports.serialport.discardinbuffer.aspx) | Удаляет данные из буфера приема последовательного драйвера. |
| [DiscardOutBuffer](http://msdn.microsoft.com/ru-ru/library/system.io.ports.serialport.discardoutbuffer.aspx) | Удаляет данные из буфера передачи последовательного драйвера. |
| [Dispose()](http://msdn.microsoft.com/ru-ru/library/3cc9y48w.aspx) | Освобождает все ресурсы, используемые объектом [Component](http://msdn.microsoft.com/ru-ru/library/system.componentmodel.component.aspx). (Унаследовано от [Component](http://msdn.microsoft.com/ru-ru/library/system.componentmodel.component.aspx).) |
| [Dispose(Boolean)](http://msdn.microsoft.com/ru-ru/library/cekaawx6.aspx) | Освобождает неуправляемые ресурсы, используемые объектом [SerialPort](http://msdn.microsoft.com/ru-ru/library/system.io.ports.serialport.aspx) (при необходимости освобождает и управляемые ресурсы). (Переопределяет [Component.Dispose(Boolean)](http://msdn.microsoft.com/ru-ru/library/d9yzd5cx.aspx).) |
| [Equals(Object)](http://msdn.microsoft.com/ru-ru/library/bsc2ak47.aspx) | Определяет, равен ли заданный объект [Object](http://msdn.microsoft.com/ru-ru/library/system.object.aspx) текущему объекту [Object](http://msdn.microsoft.com/ru-ru/library/system.object.aspx). (Унаследовано от [Object](http://msdn.microsoft.com/ru-ru/library/system.object.aspx).) |
| [Finalize](http://msdn.microsoft.com/ru-ru/library/az5741fh.aspx) | Освобождает неуправляемые ресурсы и выполняет другие операции очистки перед тем, как объект [Component](http://msdn.microsoft.com/ru-ru/library/system.componentmodel.component.aspx) будет освобожден при сборке мусора. (Унаследовано от [Component](http://msdn.microsoft.com/ru-ru/library/system.componentmodel.component.aspx).) |
| [GetHashCode](http://msdn.microsoft.com/ru-ru/library/system.object.gethashcode.aspx) | Играет роль хэш-функции для определенного типа. (Унаследовано от [Object](http://msdn.microsoft.com/ru-ru/library/system.object.aspx).) |
| [GetLifetimeService](http://msdn.microsoft.com/ru-ru/library/system.marshalbyrefobject.getlifetimeservice.aspx) | Извлекает объект обслуживания во время существования, который управляет политикой времени существования данного экземпляра. (Унаследовано от [MarshalByRefObject](http://msdn.microsoft.com/ru-ru/library/system.marshalbyrefobject.aspx).) |
| [GetPortNames](http://msdn.microsoft.com/ru-ru/library/system.io.ports.serialport.getportnames.aspx) | Получает массив имен последовательных портов для текущего компьютера. |
| [GetService](http://msdn.microsoft.com/ru-ru/library/system.componentmodel.component.getservice.aspx) | Возвращает объект, представляющий службу, обеспечиваемую компонентом [Component](http://msdn.microsoft.com/ru-ru/library/system.componentmodel.component.aspx) или его контейнером [Container](http://msdn.microsoft.com/ru-ru/library/system.componentmodel.container.aspx). (Унаследовано от [Component](http://msdn.microsoft.com/ru-ru/library/system.componentmodel.component.aspx).) |
| [GetType](http://msdn.microsoft.com/ru-ru/library/system.object.gettype.aspx) | Возвращает объект [Type](http://msdn.microsoft.com/ru-ru/library/system.type.aspx) для текущего экземпляра. (Унаследовано от [Object](http://msdn.microsoft.com/ru-ru/library/system.object.aspx).) |
| [InitializeLifetimeService](http://msdn.microsoft.com/ru-ru/library/system.marshalbyrefobject.initializelifetimeservice.aspx) | Возвращает объект обслуживания во время существования для управления политикой времени существования данного экземпляра. (Унаследовано от [MarshalByRefObject](http://msdn.microsoft.com/ru-ru/library/system.marshalbyrefobject.aspx).) |
| [MemberwiseClone](http://msdn.microsoft.com/ru-ru/library/system.object.memberwiseclone.aspx) | Создает неполную копию текущего объекта [Object](http://msdn.microsoft.com/ru-ru/library/system.object.aspx). (Унаследовано от [Object](http://msdn.microsoft.com/ru-ru/library/system.object.aspx).) |
| [MemberwiseClone(Boolean)](http://msdn.microsoft.com/ru-ru/library/ms131262.aspx) | Создает неполную копию текущего объекта [MarshalByRefObject](http://msdn.microsoft.com/ru-ru/library/system.marshalbyrefobject.aspx). (Унаследовано от [MarshalByRefObject](http://msdn.microsoft.com/ru-ru/library/system.marshalbyrefobject.aspx).) |
| [Open](http://msdn.microsoft.com/ru-ru/library/system.io.ports.serialport.open.aspx) | Открывает новое соединение последовательного порта. |
| [Read(Byte[], Int32, Int32)](http://msdn.microsoft.com/ru-ru/library/ms143549.aspx) | Считывает из входного буфера [SerialPort](http://msdn.microsoft.com/ru-ru/library/system.io.ports.serialport.aspx) определенное число байтов и записывает их в байтовый массив, начиная с указанной позиции. |
| [Read(Char[], Int32, Int32)](http://msdn.microsoft.com/ru-ru/library/34t733fh.aspx) | Считывает из входного буфера [SerialPort](http://msdn.microsoft.com/ru-ru/library/system.io.ports.serialport.aspx) определенное число символов и записывает их в символьный массив, начиная с указанной позиции. |
| [ReadByte](http://msdn.microsoft.com/ru-ru/library/system.io.ports.serialport.readbyte.aspx) | Считывает из входного буфера [SerialPort](http://msdn.microsoft.com/ru-ru/library/system.io.ports.serialport.aspx) один байт в синхронном режиме. |
| [ReadChar](http://msdn.microsoft.com/ru-ru/library/system.io.ports.serialport.readchar.aspx) | Считывает из входного буфера [SerialPort](http://msdn.microsoft.com/ru-ru/library/system.io.ports.serialport.aspx) один символ в синхронном режиме. |
| [ReadExisting](http://msdn.microsoft.com/ru-ru/library/system.io.ports.serialport.readexisting.aspx) | Считывает все непосредственно доступные байты в соответствии с кодировкой из потока и из входного буфера объекта [SerialPort](http://msdn.microsoft.com/ru-ru/library/system.io.ports.serialport.aspx). |
| [ReadLine](http://msdn.microsoft.com/ru-ru/library/system.io.ports.serialport.readline.aspx) | Считывает данные из входного буфера до значения [NewLine](http://msdn.microsoft.com/ru-ru/library/system.io.ports.serialport.newline.aspx). |
| [ReadTo](http://msdn.microsoft.com/ru-ru/library/system.io.ports.serialport.readto.aspx) | Считывает из входного буфера строку до указанного значения value. |
| [ToString](http://msdn.microsoft.com/ru-ru/library/z819d1t5.aspx) | Возвращает строку [String](http://msdn.microsoft.com/ru-ru/library/system.string.aspx), содержащую имя компонента [Component](http://msdn.microsoft.com/ru-ru/library/system.componentmodel.component.aspx), если таковое имеется.Этот метод не следует переопределять. (Унаследовано от [Component](http://msdn.microsoft.com/ru-ru/library/system.componentmodel.component.aspx).) |
| [Write(String)](http://msdn.microsoft.com/ru-ru/library/y2sxhat8.aspx) | Записывает указанную строку в последовательный порт. |
| [Write(Byte[], Int32, Int32)](http://msdn.microsoft.com/ru-ru/library/ms143551.aspx) | Записывает указанное число байтов в последовательный порт, используя данные из буфера. |
| [Write(Char[], Int32, Int32)](http://msdn.microsoft.com/ru-ru/library/1050fs1h.aspx) | Записывает указанное число символов в последовательный порт, используя данные из буфера. |
| [WriteLine](http://msdn.microsoft.com/ru-ru/library/system.io.ports.serialport.writeline.aspx) | Записывает указанную строку и значение [NewLine](http://msdn.microsoft.com/ru-ru/library/system.io.ports.serialport.newline.aspx) в выходной буфер. |

*Табл. 3. Свойства класса.*

|  |  |
| --- | --- |
| **Имя** | **Описание** |
| [BaseStream](http://msdn.microsoft.com/ru-ru/library/system.io.ports.serialport.basestream.aspx) | Получает базовый объект [Stream](http://msdn.microsoft.com/ru-ru/library/system.io.stream.aspx) для объекта [SerialPort](http://msdn.microsoft.com/ru-ru/library/system.io.ports.serialport.aspx). |
| [BaudRate](http://msdn.microsoft.com/ru-ru/library/system.io.ports.serialport.baudrate.aspx) | Получает или задает скорость передачи для последовательного порта (в бодах). |
| [BreakState](http://msdn.microsoft.com/ru-ru/library/system.io.ports.serialport.breakstate.aspx) | Получает или задает состояние сигнала разрыва. |
| [BytesToRead](http://msdn.microsoft.com/ru-ru/library/system.io.ports.serialport.bytestoread.aspx) | Получает число байтов данных, находящихся в буфере приема. |
| [BytesToWrite](http://msdn.microsoft.com/ru-ru/library/system.io.ports.serialport.bytestowrite.aspx) | Получает число байтов данных, находящихся в буфере отправки. |
| [CanRaiseEvents](http://msdn.microsoft.com/ru-ru/library/system.componentmodel.component.canraiseevents.aspx) | Возвращает значение, показывающее, может ли компонент вызывать событие. (Унаследовано от [Component](http://msdn.microsoft.com/ru-ru/library/system.componentmodel.component.aspx).) |
| [CDHolding](http://msdn.microsoft.com/ru-ru/library/system.io.ports.serialport.cdholding.aspx) | Получает состояние линии обнаружения несущей для порта. |
| [Container](http://msdn.microsoft.com/ru-ru/library/system.componentmodel.component.container.aspx) | Возвращает контейнер [IContainer](http://msdn.microsoft.com/ru-ru/library/system.componentmodel.icontainer.aspx), содержащий компонент [Component](http://msdn.microsoft.com/ru-ru/library/system.componentmodel.component.aspx). (Унаследовано от [Component](http://msdn.microsoft.com/ru-ru/library/system.componentmodel.component.aspx).) |
| [CtsHolding](http://msdn.microsoft.com/ru-ru/library/system.io.ports.serialport.ctsholding.aspx) | Получает состояние линии готовности к приему. |
| [DataBits](http://msdn.microsoft.com/ru-ru/library/system.io.ports.serialport.databits.aspx) | Получает или задает стандартное число битов данных в байте. |
| [DesignMode](http://msdn.microsoft.com/ru-ru/library/system.componentmodel.component.designmode.aspx) | Возвращает значение, указывающее, находится ли данный компонент [Component](http://msdn.microsoft.com/ru-ru/library/system.componentmodel.component.aspx) в режиме конструктора в настоящее время. (Унаследовано от [Component](http://msdn.microsoft.com/ru-ru/library/system.componentmodel.component.aspx).) |
| [DiscardNull](http://msdn.microsoft.com/ru-ru/library/system.io.ports.serialport.discardnull.aspx) | Получает или задает значение, показывающее, игнорируются ли пустые байты (NULL), передаваемые между портом и буфером приема. |
| [DsrHolding](http://msdn.microsoft.com/ru-ru/library/system.io.ports.serialport.dsrholding.aspx) | Получает или задает состояние сигнала готовности данных (DSR). |
| [DtrEnable](http://msdn.microsoft.com/ru-ru/library/system.io.ports.serialport.dtrenable.aspx) | Получает или задает значение, включающее поддержку сигнала готовности терминала (DTR) в сеансе последовательной связи. |
| [Encoding](http://msdn.microsoft.com/ru-ru/library/system.io.ports.serialport.encoding.aspx) | Получает или задает кодировку байтов для преобразования текста до и после передачи. |
| [Events](http://msdn.microsoft.com/ru-ru/library/system.componentmodel.component.events.aspx) | Возвращает список обработчиков событий, которые прикреплены к этому объекту [Component](http://msdn.microsoft.com/ru-ru/library/system.componentmodel.component.aspx). (Унаследовано от [Component](http://msdn.microsoft.com/ru-ru/library/system.componentmodel.component.aspx).) |
| [Handshake](http://msdn.microsoft.com/ru-ru/library/system.io.ports.serialport.handshake.aspx) | Получает или задает протокол установления связи для передачи данных через последовательный порт. |
| [IsOpen](http://msdn.microsoft.com/ru-ru/library/system.io.ports.serialport.isopen.aspx) | Получает значение, указывающее состояние объекта [SerialPort](http://msdn.microsoft.com/ru-ru/library/system.io.ports.serialport.aspx) — открыт или закрыт. |
| [NewLine](http://msdn.microsoft.com/ru-ru/library/system.io.ports.serialport.newline.aspx) | Получает или задает значение, используемое для интерпретации окончания вызова методов [ReadLine](http://msdn.microsoft.com/ru-ru/library/system.io.ports.serialport.readline.aspx) и [WriteLine](http://msdn.microsoft.com/ru-ru/library/system.io.ports.serialport.writeline.aspx). |
| [Parity](http://msdn.microsoft.com/ru-ru/library/system.io.ports.serialport.parity.aspx) | Получает или задает протокол контроля четности. |
| [ParityReplace](http://msdn.microsoft.com/ru-ru/library/system.io.ports.serialport.parityreplace.aspx) | Получает или задает байт, которым заменяются недопустимые байты потока данных при обнаружении ошибок четности. |
| [PortName](http://msdn.microsoft.com/ru-ru/library/system.io.ports.serialport.portname.aspx) | Получает или задает последовательный порт, в частности, любой из доступных портов COM. |
| [ReadBufferSize](http://msdn.microsoft.com/ru-ru/library/system.io.ports.serialport.readbuffersize.aspx) | Получает или задает размер входного буфера [SerialPort](http://msdn.microsoft.com/ru-ru/library/system.io.ports.serialport.aspx). |
| [ReadTimeout](http://msdn.microsoft.com/ru-ru/library/system.io.ports.serialport.readtimeout.aspx) | Получает или задает срок ожидания в миллисекундах для завершения операции чтения. |
| [ReceivedBytesThreshold](http://msdn.microsoft.com/ru-ru/library/system.io.ports.serialport.receivedbytesthreshold.aspx) | Получает или задает число байтов, содержащихся во внутреннем входном буфере перед наступлением события [DataReceived](http://msdn.microsoft.com/ru-ru/library/system.io.ports.serialport.datareceived.aspx). |
| [RtsEnable](http://msdn.microsoft.com/ru-ru/library/system.io.ports.serialport.rtsenable.aspx) | Получает или задает значение, показывающее, включен ли сигнал запроса передачи (RTS) в сеансе последовательной связи. |
| [Site](http://msdn.microsoft.com/ru-ru/library/system.componentmodel.component.site.aspx) | Получает или задает экземпляр [ISite](http://msdn.microsoft.com/ru-ru/library/system.componentmodel.isite.aspx) для компонента [Component](http://msdn.microsoft.com/ru-ru/library/system.componentmodel.component.aspx). (Унаследовано от [Component](http://msdn.microsoft.com/ru-ru/library/system.componentmodel.component.aspx).) |
| [StopBits](http://msdn.microsoft.com/ru-ru/library/system.io.ports.serialport.stopbits.aspx) | Получает или задает стандартное число стоповых битов в байте. |
| [WriteBufferSize](http://msdn.microsoft.com/ru-ru/library/system.io.ports.serialport.writebuffersize.aspx) | Получает или задает размер выходного буфера последовательного порта. |
| [WriteTimeout](http://msdn.microsoft.com/ru-ru/library/system.io.ports.serialport.writetimeout.aspx) | Получает или задает срок ожидания в миллисекундах для завершения операции записи. |

*Табл. 4. События класса.*

|  |  |
| --- | --- |
| **Имя** | **Описание** |
| [DataReceived](http://msdn.microsoft.com/ru-ru/library/system.io.ports.serialport.datareceived.aspx) | Представляет метод обработки события получения данных для объекта [SerialPort](http://msdn.microsoft.com/ru-ru/library/system.io.ports.serialport.aspx). |
| [Disposed](http://msdn.microsoft.com/ru-ru/library/system.componentmodel.component.disposed.aspx) | Происходит при удалении компонента вызовом метода [Dispose](http://msdn.microsoft.com/ru-ru/library/system.componentmodel.component.dispose.aspx). (Унаследовано от [Component](http://msdn.microsoft.com/ru-ru/library/system.componentmodel.component.aspx).) |
| [ErrorReceived](http://msdn.microsoft.com/ru-ru/library/system.io.ports.serialport.errorreceived.aspx) | Представляет метод обработки события ошибки объекта [SerialPort](http://msdn.microsoft.com/ru-ru/library/system.io.ports.serialport.aspx). |
| [PinChanged](http://msdn.microsoft.com/ru-ru/library/system.io.ports.serialport.pinchanged.aspx) | Представляет метод для обработки события изменения последовательной линии объекта [SerialPort](http://msdn.microsoft.com/ru-ru/library/system.io.ports.serialport.aspx). |

## **5.2. Описание класса CommunicationManager**

*Табл. 5. Поля класса.*

|  |  |
| --- | --- |
| **Имя** | **Описание** |
| [\_baudRate](http://msdn.microsoft.com/ru-ru/library/system.io.ports.serialport.close.aspx) | скорость передачи для последовательного порта (в бодах) |
| [\_parity](http://msdn.microsoft.com/ru-ru/library/system.marshalbyrefobject.createobjref.aspx) | протокол контроля четности |
| [\_stopBits](http://msdn.microsoft.com/ru-ru/library/system.io.ports.serialport.discardinbuffer.aspx) | число стоповых битов, используемых в объекте [SerialPort](http://msdn.microsoft.com/ru-ru/library/system.io.ports.serialport%28v=VS.90%29.aspx) |
| \_dataBits | число битов данных в байте |
| \_portName | имя последовательный порт (любой из доступных портов COM) |
| \_displayWindow | текстовое поле |
| MessageColor | цвет сообщения |
| comPort | экземляр класса SerialPort |
| \_linkActive | активность соединения |
| \_currentMsg | последнее напечатанное сообщение |

*Табл. 6. Перечисления класса.*

|  |  |
| --- | --- |
| **Имя** | **Описание** |
| MessageType | Содержит типы сообщений. |
| FrameType | Содержит типы кадров |

*Табл. 7. Свойства класса.*

|  |  |
| --- | --- |
| **Имя** | **Описание** |
| [BaudRate](http://msdn.microsoft.com/ru-ru/library/system.io.ports.serialport.baudrate.aspx) | Получает или задает скорость передачи для последовательного порта (в бодах). |
| [Parity](http://msdn.microsoft.com/ru-ru/library/system.io.ports.serialport.parity.aspx) | Получает или задает протокол контроля четности. |
| [StopBits](http://msdn.microsoft.com/ru-ru/library/system.io.ports.serialport.stopbits.aspx) | Получает или задает стандартное число стоповых битов в байте. |
| [DataBits](http://msdn.microsoft.com/ru-ru/library/system.io.ports.serialport.databits.aspx) | Получает или задает стандартное число битов данных в байте. |
| [PortName](http://msdn.microsoft.com/ru-ru/library/system.io.ports.serialport.portname.aspx) | Получает или задает последовательный порт, в частности, любой из доступных портов COM. |
| [CurrentMsg](http://msdn.microsoft.com/ru-ru/library/system.io.ports.serialport.datareceived.aspx) | Получает или задает последнее напечатанное сообщение |
| LinkActive | Получает или задает активность соединения |
| DisplayWindow | Получает или задает текстовое поле |

*Табл. 8. Методы класса.*

|  |  |
| --- | --- |
| **Имя** | **Описание** |
| FrameAnalysis | Анализирует кадры |
| WriteData | Записывает указанную строку и тип кадра в последовательный порт. |
| DisplayData | Отображает данные с последовательного порта на экране |
| OpenPort | Открывает новое соединение последовательного порта. |
| [PortName](http://msdn.microsoft.com/ru-ru/library/system.io.ports.serialport.portname.aspx) | Получает или задает последовательный порт, в частности, любой из доступных портов COM. |
| [CurrentMsg](http://msdn.microsoft.com/ru-ru/library/system.io.ports.serialport.datareceived.aspx) | Получает или задает последнее напечатанное сообщение |
| comPort\_DataReceived | Вызывается, когда есть данные, ожидающие в буфере |
| SetParityValues | Заполняет ComboBox протоколами контроля четности |
| SetStopBitValues | Заполняет ComboBox количеством стоповых битов в байте |
| SetPortNameValues | Заполняет ComboBox именами доступных портов |
| ClosePort | Закрывает соединение порта |

# **6. Канальный уровень**

## **6.1. Функции канального уровня**

На канальном уровне выполняются следующие функции:

1. Запрос логического соединения.

2. Разбивка данных на блоки (кадры).

3. Управление передачей кадров.

4. Обеспечение необходимой последовательности блоков данных, передаваемых через межуровневый интерфейс.

5. Контроль и обработка ошибок.

6. Проверка поддержания соединения.

7. Запрос на разъединение логического соединения.

## **6.2. Протокол связи**

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 такие вопросы, как 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.

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, сопровождаемую ответом приемного устройства, например, о приеме или отклонении вызова.

Также необходимо информировать пользователя о неисправностях в физическом канале, поэтому для поддержания логического соединения необходимо предусмотреть специальный кадр, который непрерывно будет посылаться с одного компьютера на другой, сигнализируя тем самым, что логическое соединение активно.

## **6.3. Защита передаваемой информации**

Во время передачи данных по линиям, входящим в коммутируемую сеть, чаще всего возникают ошибки из-за электрических помех. Данные помехи могут вызвать ошибки в цепочке или пакете последовательных битов.

Для обнаружения ошибок применяют корректирующие коды, такие как линейный код, код Хемминга, циклический код, логический код 4B/5B.

Для практической реализации циклических кодов могут быть использованы схемы с регистрами сдвига и обратными связями, или таблицы (в программе используется полином X3+X+1).

Т.к. по заданию курсовой работы необходима реализация алгоритма кодирования циклическим кодом, рассмотрим данный алгоритм.

Этапы кодирования:

* + - 1. Задана информационная последовательность m(x). Умножить заданный полином степени (k – 1) на х(n-k), т.е. сдвинуть в сторону старших разрядов на (n – k);

где n = r+k, r — степень образующего полинома, k — число информационных разрядов данной последовательности.

2. Получить остаток от деления полинома x(n-k)\*m(x) на g(x) — образующий полином. Степень остатка <= n – k – 1.

3. Объединить остаток р(х) и исходный полином x(n-k)\*m(x) для получения кодового слова; x(n-k)\*m(x) @ p(x), где @ — конкатенация;

Декодирование циклического кода:

V(x) — передаваемый кодовый полином, r(x) — принятый.

r(x) = g(x)\*q(x)+S(x),

где q(x) — частное, S(x) — остаток от деления принятого полинома на порождающий полином (если S(x) = 0, ошибки нет или она не обнаружена).

r(x) = V(x)+e(x), где e(x) — вектор ошибки;

e(x) = V(x)+q(x)\*g(x)+S(x) или e(x) = [ m(x)+q(x)]\*g(x)+s(x),

т.е. синдром ошибки s(x) есть остаток от деления вектора ошибки на порождающий полином.

Функция декодирующего устройства заключается в оценке полинома вектора ошибки e(x) по синдрому s(x).

Для различных сочетаний одиночных ошибок в кодовой комбинации двоичного циклического [7,4] —кода соответствующие им синдромы представлены в таблице 1. Рассмотрим пример двоичного циклического [7,4] — кода (n=7, k=4). Порождающий полином такого кода является примитивный полином степени (n-k): g(x) = x3 + x + 1.

*Табл. 9.*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Ошибка e (x) | Синдром s(x) | Вектор синдрома | | |
| s3 | s2 | s1 |
| x0 | x0 | 0 | 0 | 1 |
| x1 | x1 | 0 | 1 | 0 |
| x2 | x2 | 1 | 0 | 0 |
| x3 | x + 1 | 0 | 1 | 1 |
| x4 | x2 + x | 1 | 1 | 0 |
| x5 | x2 + x + 1 | 1 | 1 | 1 |
| x6 | x2 + 1 | 1 | 0 | 1 |

Рассмотрим пример:

Пусть нам необходимо закодировать кодовый вектор с k = 4 1101. Представим его в виде полинома степени (k-1): m(x) = x3 + x2 + 1.

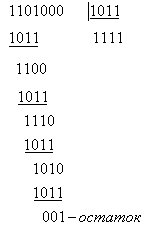
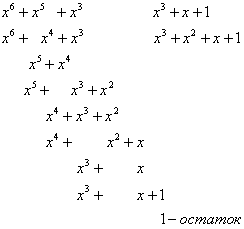
Сделаем кодирование:   
1. Умножаем m(x) на (xn-k):

m(x)•x3 = (x3 + x2 + 1)•x3 = x6 + x5 + x3 ,

что соответствует сдвигу кодового вектора в сторону старших разрядов на (n-k) разряда и добавлению в освободившиеся разряды нулей: 1101000.

* + - 1. Делим m(x)•x3 на g(x):

    или



Таким образом, остаток: p(x) = p0.

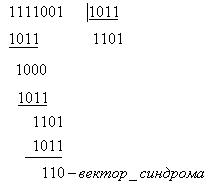
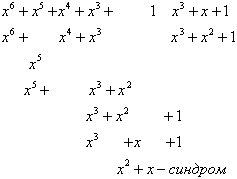
* + - 1. Приписываем остаток к информационным разрядам: v(x) = m(x)•xn-k + p(x) = x6 + x5 + x3 + 1, или выполняем операцию конкатенации исходного кодового вектора и вектора остатка: 1101.001, в результате получаем циклический (7,4) —код.

Сделаем декодирование:

Пусть вектор ошибки равен e(x) = x4, тогда принятый полином будет иметь вид:   
r(x) = v(x) + e(x) = x6 + x5 + x4 + x3 + 1 или 1101001+0010000=1111001

Для обнаружения ошибки необходимо разделить принятый полином на порождающий:

    или



По виду синдрома из таблицы 1 определяем место ошибки — разряд с весом 4.

## **6.4. Передача данных.**

В программе ведущей станцией автоматически назначается станция, инициирующая передачу. Компьютер-передатчик после выбора пользователем файла для передачи передаёт компьютеру-приёмнику информационные кадры, содержащие имя файла, информационные байты файла (структуру кадра рассмотрена на листе «Структура протокольных блоков данных»). Компьютер — приёмник принимает эти кадры и компьютеру — передатчику отправляет положительную квитанцию на каждый из этих кадров. В случае ошибки передачи компьютер – приёмник передаёт компьютеру — передатчику отрицательную квитанцию и передатчик повторяет передачу.

В случае прихода положительной квитанций на получение заголовка файла и согласия на прием, компьютер-передатчик начинает поблочную передачу файлов, дожидаясь прихода положительной квитанции после отсылки каждого блока. В случае ошибки передачи компьютер – приёмник передаёт компьютеру — передатчику отрицательную квитанцию и передатчик повторяет передачу.

После успешной передачи файлов программа возвращается в исходное состояние.

## **6.5. Функции кодирования/декодирования.**

Кодирование и декодирование данных в программе осуществляется циклическим кодом с помощью функций:

*Табл. 10. Функции.*

|  |  |
| --- | --- |
| Название функции | Функция |
| Кодирование | byte[] Kodir(byte[] Ish, int k, string PorPolinom) |
| Декодирование | byte[] Dekodir(string Ish, int k, string PorPolinom) |
| Проверка правильности полученного кода | bool Proverka(string Ish, int k, string PorPolinom) |

В рассматриваемых функциях int k — число кодируемых за раз битов.

## **6.6. Форматы кадров.**

*Табл. 11. Форматы кадров*

|  |  |
| --- | --- |
| Название кадра | Назначение кадра |
| I | информационный |
| A | квитанция с подтверждением об успешной передаче |
| R | отрицательная квитанция |
| Y | пользователь согласен и готов принимать файл |
| N | пользователь отклонил прием файла |
| E | конец передачи |

## **6.7.** **Информационный и управляющий кадры.**

*Табл. 12. Информационный и управляющий кадры.*

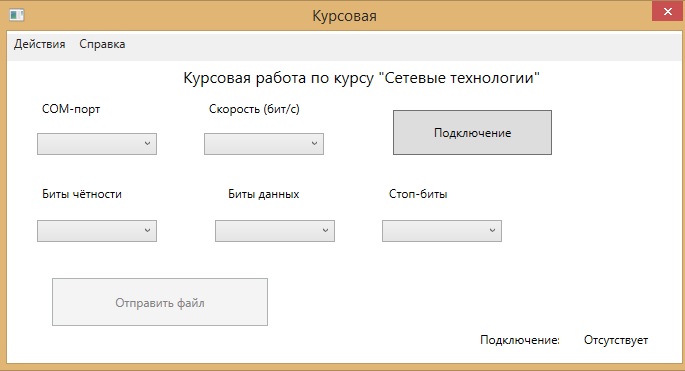
|  |  |  |
| --- | --- | --- |
| Название кадра | Структура кадра | Примечание |
| Информационный кадр | [стартовый байт][тип кадра][длина информационной части (в байтах)][Информационные байты][стоповый байт] | Все поля, кроме информационного — по одному байту. |
| Управляющий кадр | [стартовый байт][тип кадра][стоповый байт]. | Передаем заголовок, затем ждем: если получаем Y-кадр, то передаем сам файл, если получаем N-кадр — ничего не делаем. E — кадр указывает на то, что передан весь файл. |

# **7. Прикладной уровень**

Функции прикладного уровня обеспечивают интерфейс программы с пользователем через систему форм и меню. Основными функциями являются:

* интерфейс с пользователем через систему меню,
* установка режима работы,
* установка номера COM-порта для канала,
* установка скорости передачи и/или всех параметров СОМ-порта,
* имя передаваемого файла указывается на передающей ПЭВМ, а имя подкаталога для размещения полученного файла указывается на ПЭВМ-получателе.

При запуске программы появляется главная форма:



*Рис. 2. Главная форма.*

Надпись в правом нижнем углу отображает текущее состояние подключения («Подключение: Отсутствует», «Подключение: Активно»).

Кнопка “Отправить файл” служит для выбора и отправки файла.

В меню присутствуют следующие элементы:

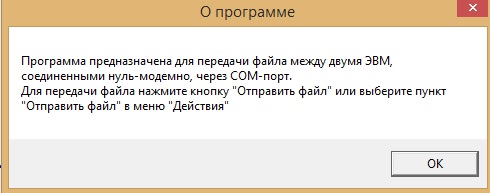
1. Действия.

1.1. Отправить файл – открывает диалог для выбора отправляемого файла

1.2. Закрыть – закрывает окно программы

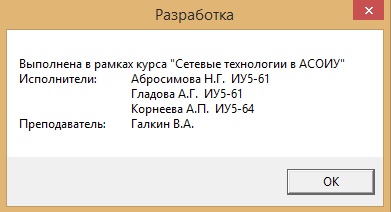
2. Справка

2.1. О программе – выдает диалоговое сообщение с назначением программы и кратким описанием ее применения:



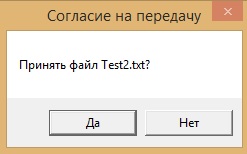
*Рис. 3. Форма «О программе».*

2.2. Разработка – выдает диалоговое сообщение с информацией о разработке программы:



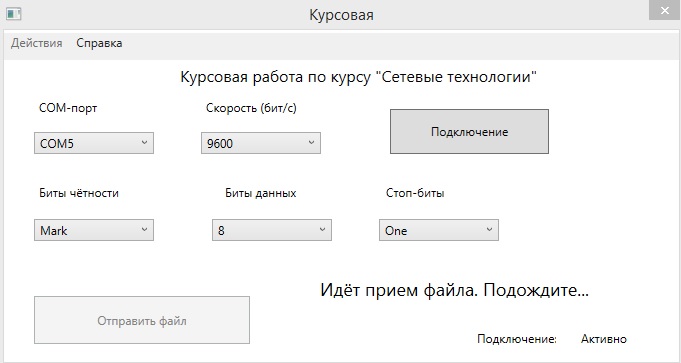
*Рис. 4. Форма «Разработка».*

При отправке файла на принимающей машине появляется окно для разрешения на передачу:



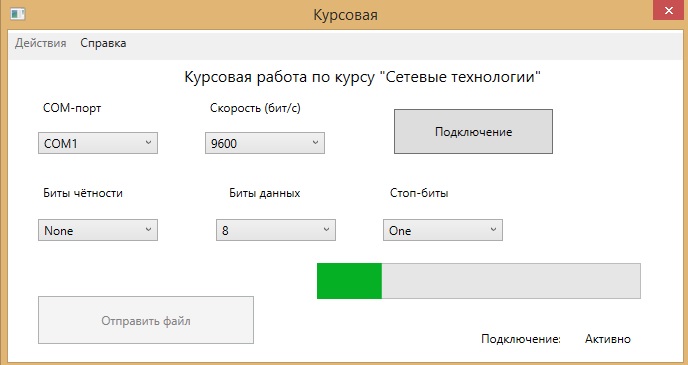
*Рис. 5. Форма «Согласие на передачу».*

Во время передачи файла на принимающей машине появляется надпись «Идет прием файла. Подождите…»:



*Рис. 6. Передача файла.*

Во время передачи файла на передающей машине появляется полоса заполнения, показывающая ход выполнения передачи:



*Рис. 7. Ход выполнения передачи.*