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

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К КУРСОВОЙ РАБОТЕ

НА ТЕМУ:

Локальная безадаптерная сеть

Студенты ИУ5-63Б _____ Назаров М.М.
(Группа) (Подпись, дата) (Фамилия И.О.)

_____ Терентьев В.О.
(Подпись, дата) (Фамилия И.О.)

_____ Халимонов А.М.
(Подпись, дата) (Фамилия И.О.)

Руководитель курсовой работы _____ _____
(Подпись, дата) (Фамилия И.О.)

Консультант _____ _____
(Подпись, дата) (Фамилия И.О.)

2021 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 _____
(Индекс)
_____ _____
(И.О.Фамилия)
« _____ » _____ 20 ____ г.

З А Д А Н И Е
на выполнение курсовой работы

по дисциплине ___Сетевые технологии в АСОИУ_____

Студенты группы _ИУ5-63Б_____

Назаров Максим Михайлович, Терентьев Владислав Олегович, Халимонов Антон Михайлович
(Фамилия, имя, отчество)

Тема курсовой работы _Локальная безадаптерная сеть_____
_____

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

Источник тематики (кафедра, предприятие, НИР) _____

График выполнения работы: 25% к ___ нед., 50% к ___ нед., 75% к __ нед., 100% к ___ нед.

Задание __Разработать протоколы взаимодействия объектов до прикладного уровня_____ локальной сети, состоящей из N ПК, соединенных нульмодемно в направленное маркерное_ кольцо через интерфейсы RS-232-C (COM1 и COM2), и реализующей функции передачи___ текстов диалога любой пары абонентов. Параметры обмена устанавливает ведущая станция. Для контроля ошибки использовать циклический [7,4]-код._____

Оформление курсовой работы:

Расчетно-пояснительная записка на _24_ листах формата А4.

_____
_____
_____

Дата выдачи задания « ___ » _____ 2021 г.

Руководитель курсовой работы _____ _____
(Подпись, дата) (Фамилия И.О.)

Студенты _____ _____Назаров Н.Н.___
(Подпись, дата) (Фамилия И.О.)
_____ _____Терентьев В.О._
(Подпись, дата) (Фамилия И.О.)
_____ _____Халимонов А.М._
(Подпись, дата) (Фамилия И.О.)

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

Оглавление

1. Введение

Данная программа, выполненная в рамках курсовой работы по предмету «Сетевые технологии в АСОИУ», предназначена для организации обмена текстовыми сообщениями между соединенными с помощью интерфейса RS-232-C компьютерами. Программа позволяет обмениваться текстовыми сообщениями любым компьютерам из N компьютеров в кольце, которые соединены между собой через COM-порты, при условии запуска этой программы на всех N компьютерах.

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 сообщений.
4. программа должна выполняться под управлением OS Windows 7 или выше.

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 предшествует специальный стартовый бит, после группы битов следуют бит проверки на четность и два стоповых бита (см. Рисунок 1).

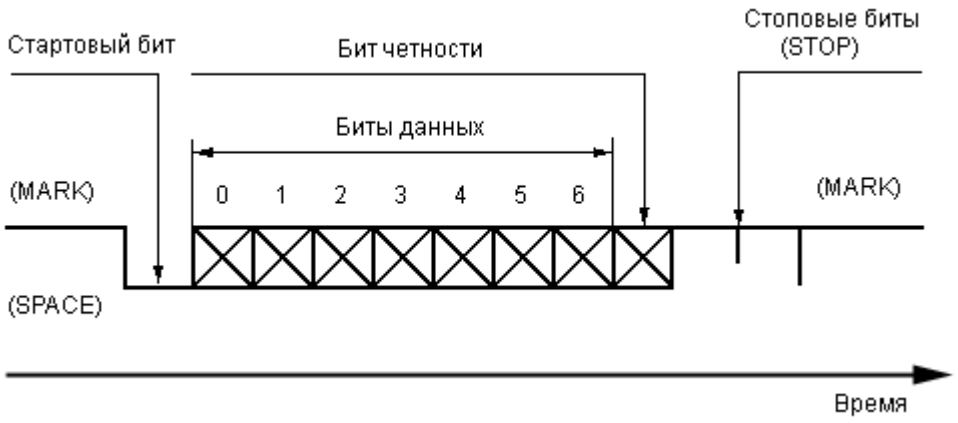

Рис. 1.

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. В нашем случае контрольный бит установлен с контролем на четность.

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 для обнаружения ошибок также используется кодирование циклическим кодом.

В самом конце передаются 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.

Интерфейс RS-232-C 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 антисимметричным, если для всех цепей обмена интерфейса используется один общий возвратный провод – сигнальная «земля».

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	CTS	Готовность к передаче	125
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-232-C) 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-232-C** 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-232-C** 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 «+» означает 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.

Методы класса:

Имя	Описание
Close	Закрывает соединение порта, присваивает свойству IsOpen значение false и уничтожает внутренний объект Stream.
CreateObjRef	Создает объект, который содержит всю необходимую информацию для создания прокси-сервера, используемого для взаимодействия с удаленным объектом. (Унаследовано от MarshalByRefObject.)
DiscardInBuffer	Удаляет данные из буфера приема последовательного драйвера.

DiscardOutBuffer	Удаляет данные из буфера передачи последовательного драйвера.
Dispose()	Освобождает все ресурсы, используемые объектом Component. (Унаследовано от Component.)
Dispose(Boolean)	Освобождает неуправляемые ресурсы, используемые объектом SerialPort (при необходимости освобождает и управляемые ресурсы). (Переопределяет Component.Dispose(Boolean).)
Equals(Object)	Определяет, равен ли заданный объект Object текущему объекту Object. (Унаследовано от Object.)
Finalize	Освобождает неуправляемые ресурсы и выполняет другие операции очистки перед тем, как объект Component будет освобожден при сборке мусора. (Унаследовано от Component.)
GetHashCode	Играет роль хэш-функции для определённого типа. (Унаследовано от Object.)
GetLifetimeService	Извлекает объект обслуживания во время существования, который управляет политикой времени существования данного экземпляра. (Унаследовано от MarshalByRefObject.)
GetPortNames	Получает массив имен последовательных портов для текущего компьютера.
GetService	Возвращает объект, представляющий службу, обеспечиваемую компонентом Component или его контейнером Container. (Унаследовано от Component.)
GetType	Возвращает объект Type для текущего экземпляра. (Унаследовано от Object.)
InitializeLifetimeService	Возвращает объект обслуживания во время существования для управления политикой времени существования данного экземпляра. (Унаследовано от MarshalByRefObject.)
MemberwiseClone	Создает неполную копию текущего объекта Object. (Унаследовано от Object.)

MemberwiseClone(Boolean)	Создает неполную копию текущего объекта MarshalByRefObject. (Унаследовано от MarshalByRefObject.)
Open	Открывает новое соединение последовательного порта.
Read(Byte[], Int32, Int32)	Считывает из входного буфера SerialPort определенное число байтов и записывает их в байтовый массив, начиная с указанной позиции.
Read(Char[], Int32, Int32)	Считывает из входного буфера SerialPort определенное число символов и записывает их в символьный 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.
ReadLine	Считывает данные из входного буфера до значения NewLine.
ReadTo	Считывает из входного буфера строку до указанного значения value.
ToString	Возвращает строку String, содержащую имя компонента Component, если таковое имеется.Этот метод не следует переопределять. (Унаследовано от Component.)
Write(String)	Записывает указанную строку в последовательный порт.
Write(Byte[], Int32, Int32)	Записывает указанное число байтов в последовательный порт, используя данные из буфера.
Write(Char[], Int32, Int32)	Записывает указанное число символов в последовательный порт, используя данные из буфера.
WriteLine	Записывает указанную строку и значение NewLine в выходной буфер.

Свойства класса:

Имя	Описание
BaseStream	Получает базовый объект Stream для объекта SerialPort.
BaudRate	Получает или задает скорость передачи для последовательного порта (в бодах).
BreakState	Получает или задает состояние сигнала разрыва.
BytesToRead	Получает число байтов данных, находящихся в буфере приема.
BytesToWrite	Получает число байтов данных, находящихся в буфере отправки.
CanRaiseEvents	Возвращает значение, показывающее, может ли компонент вызывать событие. (Унаследовано от Component.)
CDHolding	Получает состояние линии обнаружения несущей для порта.
Container	Возвращает контейнер IContainer, содержащий компонент Component. (Унаследовано от Component.)
CtsHolding	Получает состояние линии готовности к приему.
DataBits	Получает или задает стандартное число битов данных в байте.
DesignMode	Возвращает значение, указывающее, находится ли данный компонент Component в режиме конструктора в настоящее время. (Унаследовано от Component.)
DiscardNull	Получает или задает значение, показывающее, игнорируются ли пустые байты (NULL), передаваемые между портом и буфером приема.
DsrHolding	Получает или задает состояние сигнала готовности данных (DSR).
DtrEnable	Получает или задает значение, включающее поддержку сигнала готовности терминала (DTR) в сеансе последовательной связи.

Encoding	Получает или задает кодировку байтов для преобразования текста до и после передачи.
Events	Возвращает список обработчиков событий, которые прикреплены к этому объекту Component. (Унаследовано от Component.)
Handshake	Получает или задает протокол установления связи для передачи данных через последовательный порт.
IsOpen	Получает значение, указывающее состояние объекта SerialPort — открыт или закрыт.
NewLine	Получает или задает значение, используемое для интерпретации окончания вызова методов ReadLine и WriteLine.
Parity	Получает или задает протокол контроля четности.
ParityReplace	Получает или задает байт, которым заменяются недопустимые байты потока данных при обнаружении ошибок четности.
PortName	Получает или задает последовательный порт, в частности, любой из доступных портов COM.
ReadBufferSize	Получает или задает размер входного буфера SerialPort.
ReadTimeout	Получает или задает срок ожидания в миллисекундах для завершения операции чтения.
ReceivedBytesThreshold	Получает или задает число байтов, содержащихся во внутреннем входном буфере перед наступлением события DataReceived.
RtsEnable	Получает или задает значение, показывающее, включен ли сигнал запроса передачи (RTS) в сеансе последовательной связи.
Site	Получает или задает экземпляр ISite для компонента Component. (Унаследовано от Component.)
StopBits	Получает или задает стандартное число стоповых битов в байте.
WriteBufferSize	Получает или задает размер выходного буфера последовательного порта.
WriteTimeout	Получает или задает срок ожидания в миллисекундах для завершения операции записи.

События класса:

Имя	Описание
DataReceived	Представляет метод обработки события получения данных для объекта SerialPort.
Disposed	Происходит при удалении компонента вызовом метода Dispose. (Унаследовано от Component.)
ErrorReceived	Представляет метод обработки события ошибки объекта SerialPort.
PinChanged	Представляет метод для обработки события изменения последовательной линии объекта SerialPort.

5.2. Описание класса Connection

Поля класса:

Имя	Описание
_baudRate	скорость передачи для последовательного порта (в бодах)
_parity	протокол контроля четности
_stopBits	число стоповых битов, используемых в объекте SerialPort
_dataBits	число битов данных в байте
_portName	имя последовательный порт (любой из доступных портов COM)
incomePort	экземпляр класса SerialPort для входящего порта
outcomePort	экземпляр класса SerialPort для исходящего порта

Методы класса:

Имя	Описание
GetPortNames	Возвращает список доступных на данный момент портов
OpenPorts	Создает новое соединение для входящего и исходящего порта
ClosePort	Закрывает соединение для входящего и исходящего порта
SendBytes	Записывает массив байтов в исходящий порт
ReceiveBytes	Вызывается, когда есть данные, ожидающие в буфере
ReadBytes	Считывает массив байтов из входного порта и отправляет его на пользовательский уровень

5.3. Описание класса CyclicCode

Методы класса:

Имя	Описание
Coding	Получает на вход незакодированный массив байтов и возвращает закодированный с помощью циклического кода массив байтов
Decoding	Получает на вход закодированный массив байтов и возвращает кортеж, содержащий декодированный массив байтов и информацию о наличии найденных ошибок
CyclicCoding	Кодирует кодовую комбинацию v(x)=x^3+x^2+x+1 циклическим кодом с образующим полиномом g(x)=x^3+x+1
ErrorCheck	Проверяет закодированный вектор на наличие в нём ошибок передачи
division	Возвращает остаток от деления кодового вектора на образующий полином

6. Канальный уровень

6.1. Функции канального уровня

На канальном уровне выполняются следующие функции:

1. Запрос логического соединения;
2. Формирование кадров;
3. Управление передачей кадров;
4. Обеспечение необходимой последовательности блоков данных, передаваемых через межуровневый интерфейс;
5. Контроль и обработка ошибок;
6. Проверка целостности и поддержание логического соединения;
7. Посылка подтверждения безошибочного приема кадра;
8. Посылка запроса на повторную отправку кадра;
9. Запрос на разъединение логического соединения.

6.2. Протокол связи

Протокол связи использует топологию типа «кольцо» и маркерный доступ. Станции в локальной вычислительной сети организованы в кольцевую топологию, с данными, передаваемыми последовательно от одной станции в кольце к другой. Протокол использует специальный блок данных, называемый маркером, который перемещается по кольцу. Владение маркером

предоставляет его обладателю право передавать данные, обеспечивая справедливый доступ для всех станций и отсутствие коллизий. Все передаваемые данные закодированы циклическим [7,4]-кодом для обнаружения ошибок. Защита передаваемой информации рассмотрена более подробно ниже в пункте 6.3. Все передаваемые данные имеют формат кадра с одинаковой структурой, которая рассмотрена ниже в пункте 6.4.

Перед началом передачи данных требуется установить соединение между всеми сторонами, тем самым проверяется доступность приемных устройств и их готовность воспринимать данные. Для этого передающее устройство посылает специальный кадр: кадр установки соединения, содержащий адрес и текстовый псевдоним пользователя передающей станции. При получении данного кадра на принимающей станции, она посылает кадр подтверждения безошибочного приема кадра на передающую станцию, добавляет новый адрес и текстовый псевдоним своего пользователя к кадру установки соединения и отправляет этот дополненный кадр следующему устройству, подключенному к кольцу. Если передающая станция не получила кадр подтверждения безошибочного приема кадра по истечению определенного времени (тайм-аута), это означает, что приемное устройство не было готово воспринимать данные, и станция повторяет отправку кадра установки соединения.

После успешной установки соединения ведущая станция отправляет специальный кадр, называемый маркером, который циркулирует по логическому кольцу из рабочих станций. При получении станцией маркера, она отправляет кадр подтверждения безошибочного приема кадра на станцию, с которой был получен этот маркер, и изменяет адрес отправителя в маркере на свой. Если станция, передающая маркер, не получила кадра безошибочного приема кадра по истечении тайм-аута, это сигнализирует о неисправностях в физическом канале, и станция информирует об этом пользователя и пытается восстановить соединения, повторно отправив маркер.

Если станция, принявшая маркер, не имеет информационных кадров для отправки на другую станцию, она просто перенаправляет маркер следующей станции. Но если у этой станции есть информационные кадры для отправки, она захватывает маркер и начинает последовательную отправку этих информационных кадров, дожидаясь перед каждой отправкой следующего информационного кадра получения кадра безошибочного приема от станции назначения. Если по истечении тайм-аута, кадр безошибочного приема на информационный кадр не пришел, это сигнализирует о неисправностях в

физическом канале, и станция информирует об этом пользователя и пытается восстановить соединения, повторно отправив текущий информационный кадр. После отправки всех информационных кадров, станция освобождает маркер и отправляет его на следующую станцию.

При обнаружении ошибки циклическим [7,4]-кодом или ошибки в кадре, принимающая станция вместо кадра подтверждения безошибочного приема посылает кадр запроса повторения последнего отправленного кадра. При получении кадра запроса повторения последнего отправленного кадра, передающая станция повторяет отправку последнего отправленного кадра и снова дожидается прихода кадра подтверждения безошибочного приема кадра от станции назначения.

Если станция, успешно отправившая маркер, через определенное установленное время не получила обратно маркер, это сигнализирует о неисправностях в соединении или в физическом канале, и станция информирует об этом пользователя.

Если пользователь какой-либо станции захотел разорвать соединение, эта станция посылает широковещательный кадр разрыва соединения на следующую станцию и, после отправки, закрывает соединение. Станция, принявшая кадр разрыва соединения, отсылает этот кадр следующей станции и также закрывает соединение.

6.3. 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 не обеспечивают надежного обнаружения множественных ошибок, поэтому контроль ошибок совершается применением циклического [7,4]-кода.

В ходе кодирования 4-х разрядного информационного вектора циклическим [7,4]-кодом к нему присоединяется один-единственный набор из 3-х контрольных разрядов, значения которых зависят от значений разрядов информационного вектора. Приемное устройство получает 7-ми разрядный вектор и декодирует его. Если при передаче ошибки не возникли, то в результате декодирования должен быть получен заранее известный ответ. Если этот ответ не совпадает с ожидаемым, то это указывает на наличие ошибок.

Опишем более подробно алгоритмы кодирования, декодирования и обнаружения ошибки циклическим [7,4]-кодом.

Алгоритм кодирования:

Пусть задан информационный вектор 1101 с количеством информационных разрядов $k = 4$. В качестве порождающего полинома выбран примитивный полином $g(x) = x^3 + x + 1$ степени $r = 3$. Представим информационный вектор в виде полинома степени $(k - 1) = 4 - 1 = 3$:

$$m(x) = x^3 + x^2 + 1.$$

Умножим полином $m(x)$ на $x^{(n-k)}$:

$$m(x) * x^3 = (x^3 + x^2 + 1) * x^3 = x^{14} + x^6 + x^5 + x^3,$$

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

$$1101000,$$

где $n = r + k$, r – степень образующего полинома, k – число информационных разрядов кодового вектора.

Получим остаток $p(x)$ от деления полинома $x^{(n-k)} * m(x)$ на порождающий полином $g(x)$. Степень остатка $\leq n - k - 1$.

$$p(x) = x^{(n-k)} * m(x) \bmod g(x) = 1101000 \bmod 1001 = 001$$

Таким образом, остаток $p(x) = 1$.

Выполним операцию конкатенации полученного кодового вектора остатка $p(x)$ и исходного кодового вектора полинома $m(x)$:

$$1101 \text{ @ } 001 = 1101001,$$

где @ – конкатенация. В результате получили циклический [7,4]-код.

Алгоритм декодирования:

Пусть $v(x)$ – передаваемый кодовый полином, $r(x)$ – принятый кодовый полином.

Пусть вектор ошибки равен $e(x) = x^4$, тогда принятый полином будет иметь вид:

$$r(x) = v(x) + e(x) = x^6 + x^5 + x^4 + x^3 + 1,$$

или:

$$1101001 \oplus 0010000 = 1111001.$$

Для обнаружения ошибки необходимо разделить принятый полином на порождающий:

$$s(x) = r(x) \bmod g(x) = 1111001 \bmod 1001 = 110.$$

Разделив $r(x)$ на порождающий полином $g(x)$, получим:

$$r(x) = g(x) * q(x) + s(x)$$

где $q(x)$ – частное, $s(x)$ – остаток.

Если остаток равен нулю, т. е. принятый кодовый вектор кратен порождающему полиному, то, следовательно, ошибки нет или она не обнаружена. Если остаток не равен нулю, то принятый кодовый вектор не является кодовым полиномом, т. е. содержит ошибку.

Эффективность циклического кода:

Так как порождающий полином $g(x)$ имеет степень $(n-k)$, то существует кодовый вектор, являющийся пакетом длиной $(n-k+1)$, т. е. содержащий $(n-k+1)$ единиц подряд. Если полином вектора ошибки $e(x)$ представляет собой пакет длиной $(n-k)$ или меньшей, то согласно выражению $e(x) = [m(x) + q(x)] * g(x) + s(x)$ синдром никогда не будет равен нулю. Это означает, что циклический [7, 4]-код пригоден для обнаружения любого пакета ошибок длиной 3 или меньшей, а также отдельных пакетов и большей кратности. Приведенный анализ показывает, что циклические коды весьма эффективны для обнаружения ошибок, поэтому их широко применяют в системах телекоммуникации.

Алгоритм применения циклического [7,4]-кода состоит в том, что каждый байт, подлежащий кодированию, разбивается на две части по 4 бита, после чего каждая часть кодируется циклическим [7,4]-кодом и 1 байт передается на принимающее устройство, т.е. в итоге из каждого байта получается два. На принимающей стороне производится декодирование двух пришедших байтов и объединение в один байт информационных битов. Если в ходе операций декодирования ошибки не были обнаружены, то данные пришли безошибочно, иначе посылается кадр запроса повторения последнего отправленного кадра.

6.4. Формат кадров

Все кадры имеют одинаковую структуру, представленную ниже:

Стартовый байт	Адрес получателя	Адрес отправителя	Тип кадра	Длина поля данных*	Данные*	Стоповый байт
1 байт	1 байт	1 байт	1 байт	1 байт	N байт	1 байт

Примечание: поля, отмеченные «*» – не обязательны и зависят от типа кадра.

Стартовый и стоповый байт – служат для определения начала и конца кадра. Имеют значение 0xFF.

Адрес получателя – байт, содержащий адрес получателя кадра. Существует широковещательный адрес (0x7F).

Адрес отправителя – байт, содержащий адрес отправителя кадра. Адреса в системе назначаются динамически и лежат в промежутке от 0x01 до 0x7E.

Тип кадра – байт, содержащий код типа кадра.

Длина поля данных – не обязательное поле, содержит длину поля Данные.

Данные – не обязательное поле, содержит какие-либо данные, передаваемые в кадре.

6.4.1. Супервизорный Token - Кадр

Кадр-маркер. Код типа кадра: 0x00. Служит маркером в направленном маркерном кольце. В поле Адрес получателя – обязательно широковещательный адрес. Поля Длина поля данных и Данные – отсутствуют.

6.4.2. Информационный I - Кадр

Информационный кадр. Код типа кадра: 0x01. Служит для передачи коротких сообщений между компьютерами, включенными в кольцо. Поля

Длина поля данных и Данные – присутствуют. Поле Данные содержит текст передаваемого короткого сообщения.

6.4.3. Супервизорный Link - Кадр

Кадр установки соединения. Код типа кадра: 0x02. Служит для установки логического соединение типа «кольцо». В поле Адрес получателя – обязательно широковещательный адрес, т.к. используется в системе в начальный момент, когда адреса всех компьютеров сети не определены. Поля Длина поля данных и Данные – присутствуют. Поле Данные содержит адреса и текстовые псевдонимы всех пользователей сети.

6.4.4. Супервизорный Dis - Кадр

Кадр разрыва соединения. Код типа кадра: 0x03. Служит для разрыва логического соединение типа «кольцо». В поле Адрес получателя – обязательно широковещательный адрес. Поля Длина поля данных и Данные – отсутствуют.

6.4.5. Супервизорный ACK - Кадр

Кадр подтверждения безошибочного приема кадра. Код типа кадра: 0x04. Служит для контроля передачи кадров. В поле Адрес получателя – не может быть широковещательного адреса. Поля Длина поля данных и Данные – отсутствуют.

6.4.6. Супервизорный Ret - Кадр

Кадр запроса повторения последнего отправленного кадра. Код типа кадра: 0x05. Служит для исправления возникших ошибок в ходе передачи кадров. В поле Адрес получателя – не может быть широковещательного адреса. Поля Длина поля данных и Данные – отсутствуют.

7. Прикладной уровень.

Функции прикладного уровня обеспечивают интерфейс программы с пользователем через систему окон и меню. Прикладной уровень предоставляет канальному уровню информацию о выбранных COM-портах, об имени пользователя, о том является ли данная станция ведущей и сообщения, отправленные пользователем.

Пользовательский интерфейс разработан с помощью Windows Presentation Foundation (WPF) в среде разработки Visual Studio 2019.

При запуске программы появляется главное окно «MainWindow». В данном окне есть следующие возможности:

1. Выбрать доступные COM-порты компьютера
2. Указать имя пользователя
3. Выбрать является ли данный компьютер ведущей станцией или нет

Открывающиеся списки «Исходящий COM-порт» и «Входящий COM-порт» позволяют выбрать один COM-порт из списка доступных для исходящих и входящих данных соответственно.

Текстовое поле «Имя пользователя» позволяет задать имя пользователя. При нажатии на текстовое поле текстовая подсказка «Введите имя пользователя» пропадает, и пользователь может указать имя пользователя. Если пользователь не указывает своё имя пользователя и переходит настраивать другой элемент, то текстовая подсказка вернётся и снова пропадёт при новом взаимодействие пользователя.

После того, как пользователь указал правильно всю информацию и нажал кнопку «Установить соединение», устанавливается соединение и открывается окно «Чат».

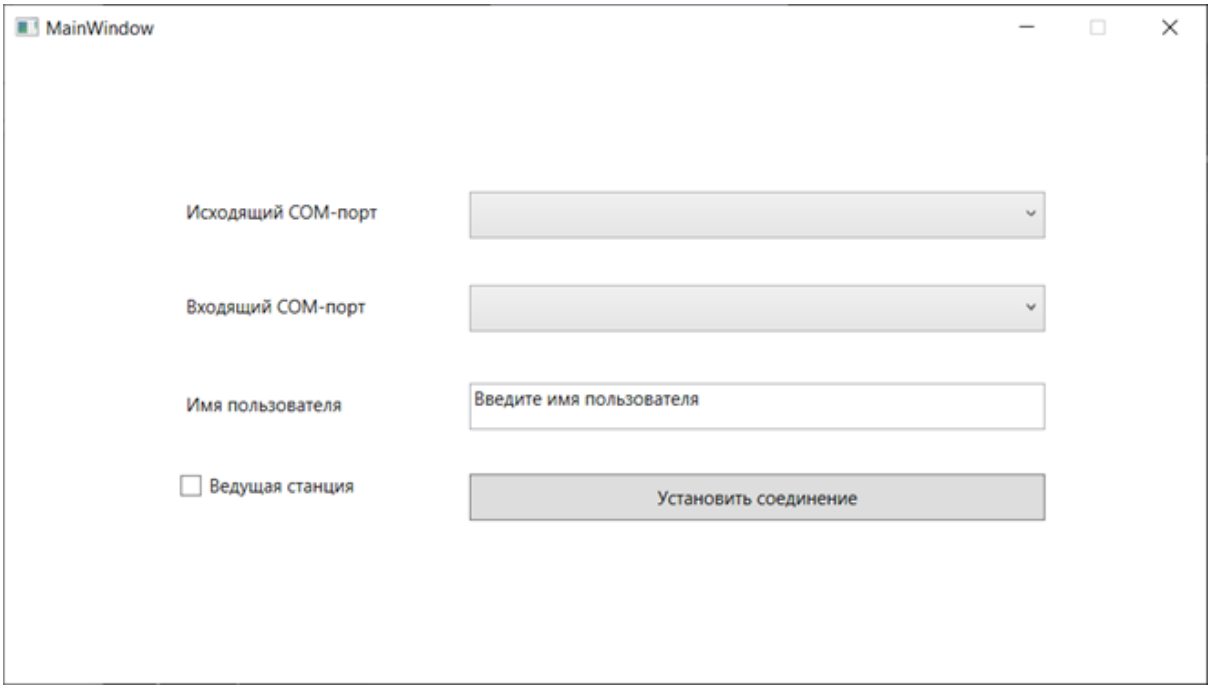

Рис. 2. Главное окно после запуска программы

Основным окном программы является окно «Чат». В данном окне есть следующие возможности:

1. Просмотр сообщений
2. Отправка сообщений
3. Список пользователей
4. Личные чаты
5. Общие чаты
6. Проверка статуса соединения
7. Уведомление о новом сообщение в неактивном чате с помощью списка пользователей

Название окна служит подсказкой для пользователя, указывая введенное им имя пользователя в главном окне, а также показывает, является данная станция ведущий или нет.

Список со всеми чатами позволяет пользователю выбрать нужный ему чат: общий для всех пользователей или чат с конкретным пользователем. Выбранный чат выделяется. При получении нового сообщения из неактивного чата пользователь будет извещен об этом с помощью списка чатов, чат с новыми сообщениями будет помещен в начало списка и в название будет указываться символ «*». После перехода в этот чат символ «*» пропадает.

Текстовое поле с подсказкой «Введите сообщение» аналогично текстовому полю «Имя пользователя» из главной формы.

Кнопка «Отправить» служит для отправки введённого пользователем сообщения в чат.

Кружок состояния соединения служит для отображения статуса соединения. Если соединения прерывается, то кружок состояния соединения становится красным до момента восстановления соединения.

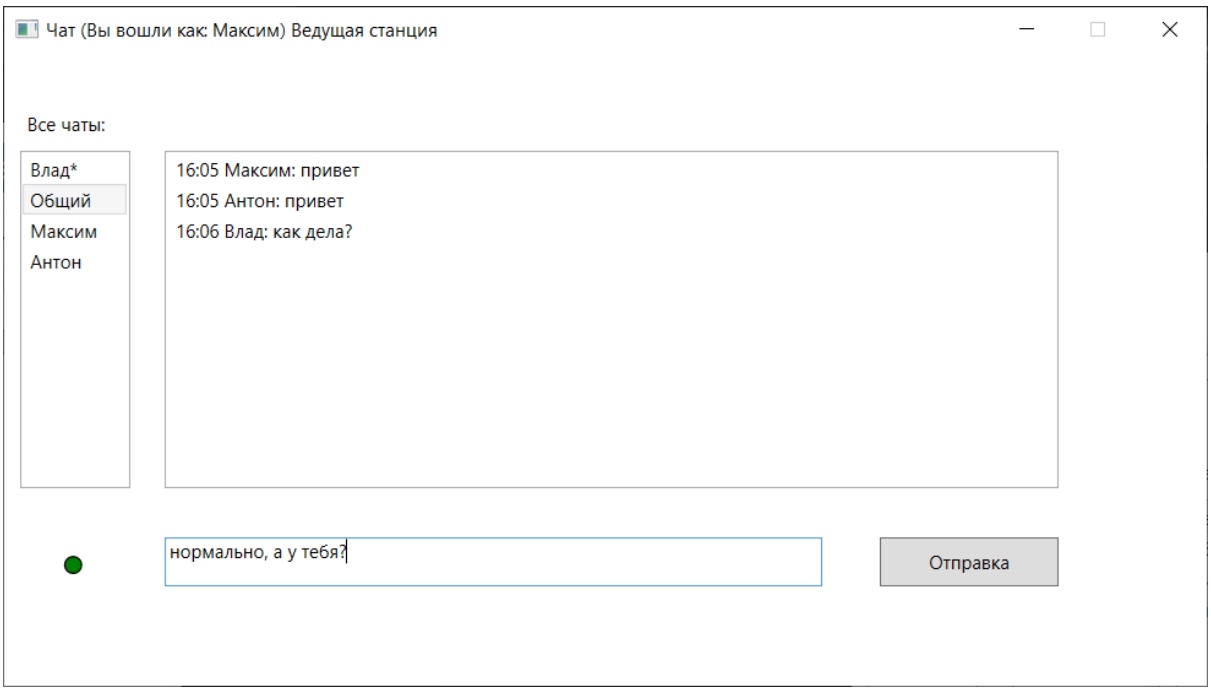

Рис. 3. Окно «Чат (Вы вошли как: …) Ведущая станция» с выбранным чатом «Общий»

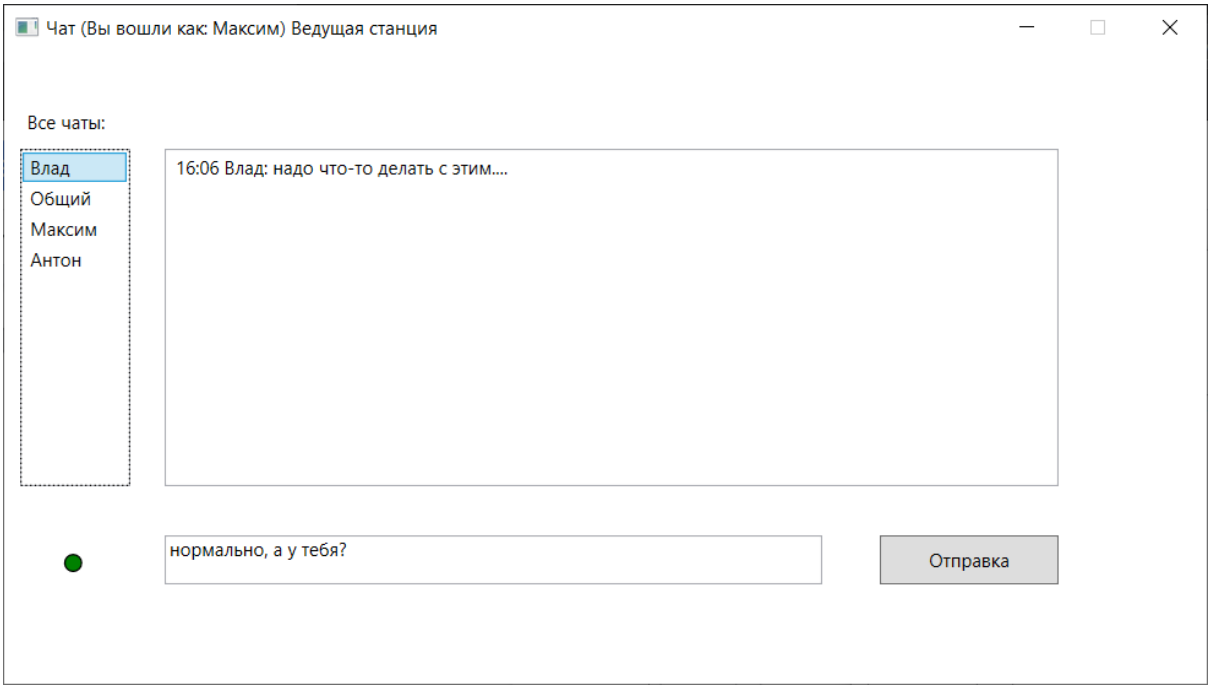

Рис. 4. Окно «Чат (Вы вошли как: …) Ведущая станция» с открытым личным чатом