***ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ РФ***
***МГТУ им. Н.Э.Баумана***

*23 0102*

***Согласовано***

*_____ Галкин В. А*
*“____” _____ _____*

***Утверждаю***

*_____ Галкин В. А*
*“____” _____ _____*

# *Курсовая работа по дисциплине «Сетевые технологии»*

*«Локальная безадаптерная сеть»*
*Расчетно-пояснительная записка*


*Студент 4 курса группы ИУ5-72*
*_____ Гуща А. В*
*“____” _____ _____*

*Студент 4 курса группы ИУ5-72*
*_____ Нардид А. Н*
*“____” _____ _____*

*Студент 4 курса группы ИУ5-72*
*_____ Оганян Л. П*
*“____” _____ _____*


*Москва*
*2013 г.*

# Содержание

# 1 Введение

Данная программа, «PowerCom», выполненная в рамках курсовой работы по предмету «Сетевые технологии», предназначена для организации обмена текстовыми сообщениями мужду соединенными с помощью интерфейса RS-232C компьютерами. Программа позволяет обмениваться с двум компьютерам, соединенный через COM-порты, текстовыми сообщениями, при условии запуска этой программы на обоих компьютерах.

# 2 Требования к программе

К программе предъявляются следующие требования. Программа должна:

- Устанавливать соединение между компьютерами и контролировать его целостность;
- Обеспечивать правильность передачи и приема данных с помощью алгоритма циклического кодирования пакета;
- Обеспечивать функцию передачи сообщений;
- Программа должна выполняться под управлением операционной системы OS Windows XP/7, GNU/Linux.

# 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 верхний модулей;
- Прикладной уровень занимается выполнением задач программы.

# 4 Физический уровень

## 4.1 Функции физического уровня

Основными функциями физического уровня являются:

а) Задание параметров COM-порта;

б) Установление физического канала;

в) Разъединение физического канала;

г) Передача информации из буфера в интерфейс;

д) 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. рисунок 1. Иногда бит проверки на четность может отсутствовать.

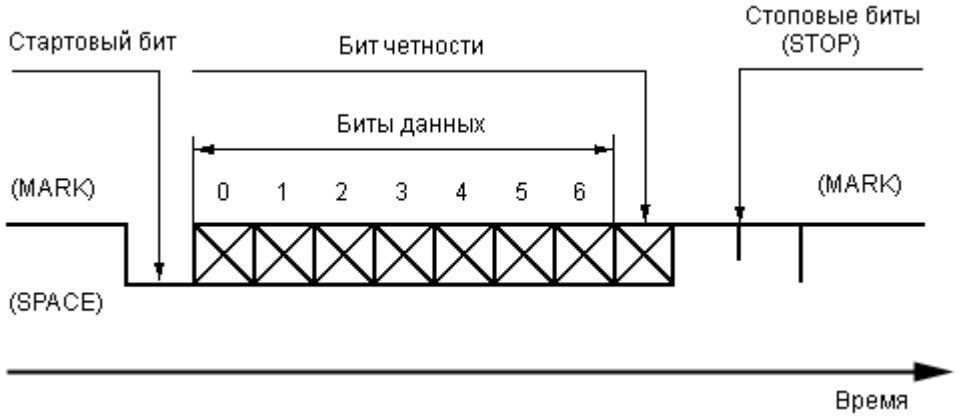


Рисунок 1 – Временная диаграмма передачи кадра.

Из рисунка видно, что исходное состояние линии последовательной передачи данных - уровень логической единицы. Это состояние линии называют отмеченным - **MARK**. Когда начинается передача данных, уровень линни переходит в логический нуль. 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 постоянное значение логической единицы, 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 линнии. Приемное устройство заново вычисляет четность данных и сравнивает результат с принятым битом четности. Если четность не совпала, то считается, что данные переданы с ошибокй. Конечно, такой алгоритм не дает стопроцентной гарантии обнаружения ошибок. Так, если при передачи данных изменилось четное число битов, то четность сохраняется, и ошибка не будет обнаружена. Поэтому на практике применяются 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 RS-232C 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».

В интерфейсе реализован биполярный потенциальный код на линиях между DTE и DCE. Напряжения сигналов в цепях обмена симметричны по

| Номер контакта | Обозначение | Назначение | Цепь |
|---|---|---|---|
| 1 | DCD (Data Carrier Detect) | Обнаружение несущей | 109 |
| 2 | RD (Receive Data) | Принимаемые данные | 104 |
| 3 | TD (Transmit Data) | Отправляемые данные | 103 |
| 4 | DTR (Data Terminal Ready) | Готовность терминала к работе | 108/2 |
| 5 | SG (Signal Ground) | Земля сигнала (схемная) | 102 |
| 6 | DSR (Data Set Ready) | Готовность DCE | 107 |
| 7 | RTS (Request To Send) | Запрос передачи | 105 |
| 8 | CTS (Clear To Send) | Готовность DCE к приему | 106 |
| 9 | RI (Ring Indicator) | Индикатор вызова | 125 |

Таблица 1 – Интерфейс девяти контактного разъема.

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 сигнальными 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 вторым компьютером присоединенном 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 информацие 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:

а) 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 тог, что **DTR** принят, и информирует компьютер о готовности модема к приему информации;

б) Если компьютер получил сигнал **DSR** и хочет передать данные, он выставляет сигнал **RTS**;

в) 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**;

г) 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, модем выставляет 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 опсность переполнения 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:

а) RI+DSR-1 $\rightarrow$ DTR-2;

б) DTR-1 $\rightarrow$ RI-2+DSR-2;

в) CD-1 $\rightarrow$ CTS-2 + RTS-2;

г) CTS-1+RTS-1 $\rightarrow$ CD-2;

д) RD-1 $\rightarrow$ TD-1;

е) TD-1 $\rightarrow$ RD-1;

ж) SG-1 $\rightarrow$ SG-2;

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

# 5 Настройка COM-порта средствами HASKELL

Язык Haskell имеет все нобходимые средства для работы с COM портами. Для этого необходимо установить пакет **serialport** через пакетный менеджер **cabal**.

## 5.1 Описание типов данных

– **CommSpeed** - скорость передачи данных в бодах в секунду.

```
data CommSpeed =
  CS110    |
  CS300    |
  CS600    |
  CS1200   |
  CS2400   |
  CS4800   |
  CS9600   |
  CS19200  |
  CS38400  |
  CS57600  |
  CS57600
```

Определенные instances:

```
Show CommSpeed
```

– **StopBits** - количество используемых стоп битов.

```
data StopBits =
  One |
  Two
```

– **Parity** - тип бита четности, проверка на четность/нечетность или его отсутствие.

```
data Parity =
  Even      |
  Odd       |
  NoParity
```

– **FlowControl** - флаг включающий, выключающий использование возможности «притормозить» удаленное передающее устройство.

```
data FlowControl =
  Software      |
  NoFlowControl
```

– **SerialPort** - тип, инкапсулирующий открытый COM-порт.

  Определенные instances:

```
Typeable    SerialPort
BufferedIO  SerialPort
RawIO       SerialPort
IODevice    SerialPort
```

– **SerialPortSettings** - настройки COM-порта.

```
data SerialPortSettings =
  commSpeed    :: CommSpeed
  bitsPerWord  :: Word8
  stopb        :: StopBits
  parity       :: Parity
  flowControl  :: FlowControl
  timeout      :: Int
```

  Разъяснение параметров конструктора:

  – **commSpeed** - скорость порта в бодах в секунду;
  – **bitsPerWord** - количество битов в передаваемых словах (для асихронного режима);
  – **stopb** - количество стоп битов;
  – **parity** - тип бита четности;
  – **flowControl** - наличие контроля потока, возможность «приостановить» передающее устройство.
  – **timeout** - время в десятых долях секунды, после которого подвисшее соединение считается разорванным.

## 5.2 Описание функций

```
defaultSerialSettings :: SerialPortSettings
```

Наиболее часто используемы настройки COM-порта:

- скорость 9600 бод в секунду;
- в байте 8 бит;
- 1 стоп бит;
- без бита четности;
- без контроля потока;
- 0.1 секунда для timeout.

```
setSerialSettings :: SerialPort -> SerialPortSettings -> IO SerialPort
```

Устанавливает настройки COM-порта:

- **SerialPort** - текущий открытый COM-порт;
- **SerialPortSettings** - новые настройки COM-порта;
- **IO SerialPort** - новое состояние COM-порта.

```
getSerialSettings :: SerialPort -> SerialPortSettings
```

Получение текущих настроек COM-порта:

- **SerialPort** - текущий открытый COM-порт;
- **SerialPortSettings** - настройки COM-порта;

```
hOpenSerial :: FilePath -> SerialPortSettings -> IO Handle
```

Открывает и настраивает COM-порт, возвращает стандартный тип ссылки на устройство ввода/вывода:

- **FilePath** - название COM-порта, например «COM1» или «COM2»;
- **SerialPortSettings** - первичные настройки COM-порта;
- **Handle** - ссылка на открытый COM-порт;

```
openSerial :: FilePath -> SerialPortSettings -> IO SerialPort
```

Открывает и настраивает COM-порт:

- **FilePath** - название COM-порта, например «COM1» или «COM2»;
- **SerialPortSettings** - первичные настройки COM-порта;
- **IO SerialPort** - новый открытый и настроенный COM-порт.

```
1 closeSerial :: SerialPort -> IO ()
```

Закрывает открытый COM-порт:

- **SerialPort** - открытый COM-порт.

```
1 withSerial :: String -> SerialPortSettings -> (SerialPort -> IO a) ->
    IO a
```

Безопасная функция для работы с COM-портами, закрывает порт после выполенения операции с ним:

- **String** - название COM-порта, например «COM1» или «COM2»;
- **SerialPortSettings** - первичные настройки COM-порта;
- **SerialPort -> IO a** - функция, которая будет выполнена после открытия порта;
- **IO a** - результат выполнения функции над COM-портом.

```
1 send :: SerialPort -> ByteString -> IO Int
```

Отсылает байты через COM-порт:

- **SerialPort** - текущий открытый COM-порт;
- **ByteString** - буфер для отсылки;
- **Int** - количество байтов, которое было отослано.

```
1 recv :: SerialPort -> Int -> IO ByteString
```

Получение байтов из COM-порта с ограничением по количеству байтов сверху:

- **SerialPort** - текущий открытый COM-порт;
- **Int** - максимальное количество байтов для приема;
- **IO ByteString** - принятый буфер.

```
flush :: SerialPort -> IO ()
```

Принудительно отсылает ждущие отправки данные во временных буфера COM-порта:

- **SerialPort** - текущий открытый COM-порт.

```
setDTR :: SerialPort -> Bool -> IO ()
```

Устанавливает сигнал **DTR** (Data Terminal Ready) в заданное значение:

- **SerialPort** - текущий открытый COM-порт;
- **Bool** - значение, которое должен принять **DTR** провод.

```
setRTS :: SerialPort -> Bool -> IO ()
```

Устанавливает сигнал **RTS** (Ready To Send) в заданное значение:

- **SerialPort** - текущий открытый COM-порт;
- **Bool** - значение, которое должен принять **RTS** провод.

# 6 Канальный уровень

## 6.1 Функции канального уровня

Канальный уровень выполняет следующие функции:

а) Запрос логического соединения;

б) Разбивка данных на блоки (кадры);

в) Управление передачей кадров;

г) Обеспечение необходимой последовательности блоков данных, передаваемых через межуровневый интерфейс;

д) Контроль и обработка ошибок, включая транзит сообщений об ошибках от физического уровня к прикладному;

е) Проверка поддержания соединения;

ж) 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 потокм протокол связи регулирует также такие вопрсы, 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

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 контрольная сумма блока не обеспечивают надежного обнаружения нескольких (например, двух) 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 (вырабатывается) один-единственный набор контрольных разрядов, значения которых зависят от фактического содержания кадра и присоединяются передатчиком к «хвосту» 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.

Опишем кратко математический аппарат циклического кодирования. Код, в котором кодовая комбинация, полученная путем циклического сдвига разрешенной кодовой комбинации является также разрешенной кодовой комбинацией называется циклическим (полиномиальным, кодом с циклическими избыточными проверками-ЦИП). Сдвиг осуществляется справа налево, при этом крайний левый символ переносится в конец комбинации.

Циклический код относится к линейным, блочным, корректирующим, равномерным кодам. В циклических кодах кодовые комбинации представляются в виде многочленов, что позволяет свести действия надо кодовыми комбинациями к действием надо многочленами (используя аппарат полиномиальной алгебры). Циклические коды являются разновидностью систематических кодов и поэтому обладают всеми их свойствами. Первоначально они были созданы для упрощения схем кодирования и декодирования. Их эффективность при обнаружении и исправлении ошибок обеспечила им широкое применение на практике.

Циклические коды используются в ЭВМ при последовательной передаче данных. Сдвиг справа налево осуществляется путем умножения полинома на $x$. Операции сложения и вычитания выполняются по модулю 2. Они являются эквивалентными и ассоциативными. Операция деления является обычным делением многочленов, только вместо вычитания используется сложение по модулю 2.

Идея построения циклических кодов базируется на использовании неприводимых многочленов. Неприводимым называется многочлен, который не может быть представлен в виде произведения многочленов низших степеней, т.е. такой многочлен делиться только на самого себя или на единицу и не делиться ни на какой другой многочлен. На такой многочлен делиться без остатка двучлен $x^n+1$. Неприводимые многочлены в теории циклических кодов играют роль образующих полиномов.

Чтобы понять принцип построения циклического кода, умножаем комбинацию простого k-значного кода $Q(x)$ на одночлен $x^r$, а затем делим на образующий полином $P(x)$, степеь которого равна $r$. В результате умножения $Q(x)$ на $x^r$ степень каждого одночлена, входящего в $Q(x)$, повышается на $r$. ПРи делении произведения $x^rQ(x)$ на образующий полином получается частное $C(x)$ такой же степени, как и $Q(x)$. Результат можно представить в виде:

$$\frac{Q(x)x^r}{P(x)} = C(x) + \frac{R(x)}{P(x)} \qquad (1)$$

где$R(x)$ - остаток от деления $Q(x)x^r$ на $P(x)$.

Частное $C(x)$ имеет такую же степень, как и кодовая комбинация $Q(x)$ простого кода, поэтому $C(x)$ является кодовой комбинацией этого же простого k-значного кода. Следует заметить, что степень остатка не может быть больше степени образующего полинома, т.е. его наивысшая степень может быть равна $(r-1)$. Следовательно, наибольшие число разрядов остатка $R(x)$ не превышает числа $r$.

Умножая обе части равенства (1) на $P(x)$ и произведя некоторые перестановки, получаем:

$$F(x) = C(x)P(x) = Q(x)x^r + R(x) \quad (2)$$

Таким образом, кодовая комбинация циклического n-значного кода может быть получена двумя способами:

а) умножением кодовой комбинации $Q(x)$ простого кода на одночлен $x^r$ и добавление к этому произведению остатка $R(x)$, полученного в результате деления произведения $Q(x)x^r$ на образующий полином $P(x)$;

б) умножением кодовой комбинации $C(x)$ простого k-значного на образующий полином $P(x)$.

При построении циклических кодов первым способом расположение информационных символов во всех комбинациях строго упорядочено - они занимают k старших разрядов комбинации, а остальные $(n - k)$ разрядов отводятся под контрольные.

При втором способе образования циклических кодов информационные и контрольные символы в комбинациях циклического кода не отделены друг от друга, что затрудняет процесс декодирования.

Как было указано выше, циклическое кодирование обладает свойством избыточности (буквально информация удваивается), и так же данный алгоритм кодирования обладает свойствами исправления ошибки в одном бите.

Алгоритм кодирования состоит в том, что каждый байт, подлежащий кодированию, разбивается на части по четыре бита, после чего делится на полином и результат деления, один байт, передается по сети, то есть, в итоге из каждого байта получается два. На принимающей стороне производится обратные операции, определяем частное и остаток. По остатку определяем вектор ошибки, если остаток нулевой, то данные дошли безошибочно, если же ненулевой, то отсылаем отрицательную квитанцию - просьбу повторить посылку пакета.

## 6.4 Формат кадров

Кадры, передаваемые с помощью функций канального уровня, имеют различное назначение. Выделены супервизорные и информационные кадры, но все кадры имеют одинаковую структуру:

| Номер поля | Имя поля | Размер в байтах | Пояснение |
|---|---|---|---|
| 1 | Type | 1 | Тип кадра. |
| 2 | Length | 1 | Длина поля данных, может отсутствовать для супервизорных кадров. |
| 3 | Data | Length | Данные кадра, может отсутствовать для супервизорных кадров. |

Таблица 2 – Формат кадров.

### 6.4.1 Служебные супервизорные кадры

Эти кадры используются для передачи служебной информации и реализуют следующие функции канального уровня: установление и разъединение логического канала, подтверждение приема информационного кадра без ошибок, запрос на повторную передачу принятого с ошибкой кадра. Типы этих кадров отображены в таблице 3.

### 6.4.2 Формат информационных кадров

Существует два типа информационных кадров, первый тип информирует получателя о последующих кадрах с частями сообщения (разбиение на множество информационных кадров используется при больших сообщениях). Такой начальный кадр **InformationFrame** (поле Type = 0) содержит имя пользователя и число-количество последующих частей информации. После **InformationFrame** кадра следует последовательность кадров **DataPartFrame** (поле Type = 1), содержащих непосредственное тело сообщения. Приход меньшего числа кадров **DataPartFrame** приводит к отбрасыванию всего накопленного сообщения. Лишние кадры **DataPartFrame** отбрасываются.

| Поле Type | Название кадра | Описание | Данные |
|---|---|---|---|
| 2 | Link | Установление соединения | Текстовый псевдоним пользователя, в ответ удаленная сторона также присылает кадр Link с именем пользователя. |
| 3 | Unlink | Разрыв соединения | Текстовый псевдоним пользователя. |
| 4 | Ack | Подтверждение безошибочного приема кадра | Отсутствуют |
| 5 | Ret | Запрос повторения последнего отправленного кадра | Отсутствуют |
| 6 | Option | Изменение параметров соединения, посылается, когда пользователь с одной из сторон меняет параметры | Набор изменившихся параметров и их значения, включая псевдоним пользователя. |
| 7 | Upcheck | Проверка наличия соединения между пользователями производится с помощью периодической посылки данного кадра. | Отсутствуют |

Таблица 3 – Формат супервизорных кадров.

# 7 Прикладной уровень

Функции прикладного уровня обеспечивают интерфейс программы с пользователем через систему форм и меню. Прикладной уровень предоставляет нижнему уровню текстовое сообщение. На данном уровне обеспечивается вывод принятых и отправленных сообщений в окно диалога пользователей.

Пользовательский интерфейс реализован на основе библиотеки **Gtk2Hs**. При его разработке в первую очередь учитывались соображения простоты,

удобности и эргономичности пользовательского интерфейса.

После запуска программы пользователь видит главное окно программы, показанное на рис. 2

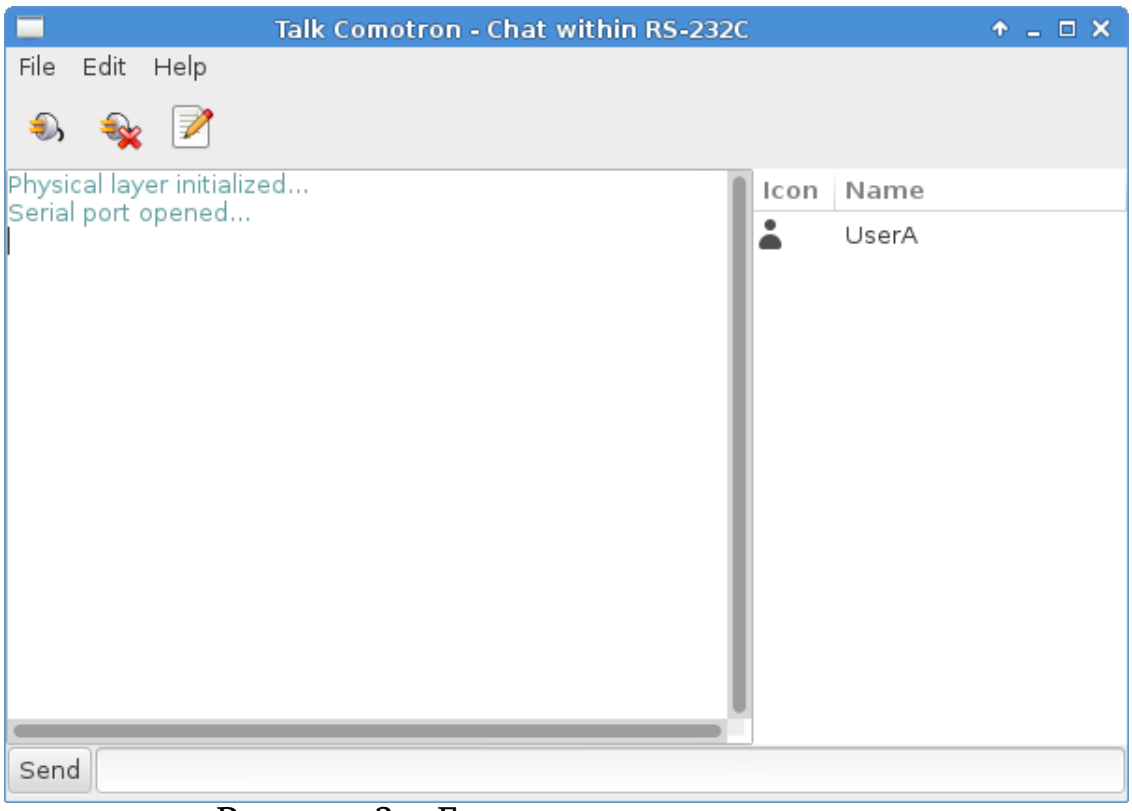


Рисунок 2 – Главное окно приложения.

Для настройки параметров COM-порта и своего псевдонима, пользователь может перейти к окну настроек, нажав на кнопку "Options". Окно настроек показано на рис. 3.

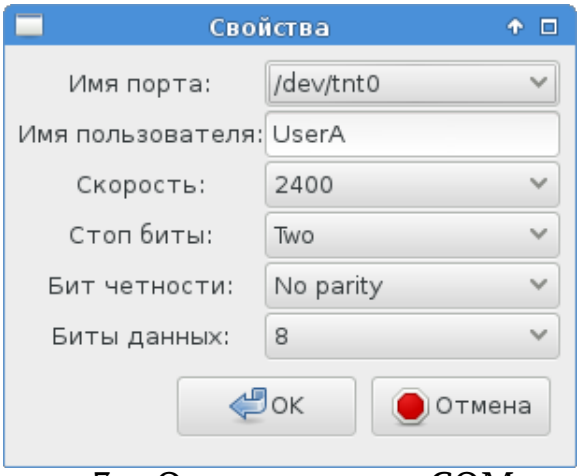


Рисунок 3 – Окно настроек COM-порта.

После настройки своего последовательного порта пользователь может нажать на кнопку "Connect находящуюся на панели инструментов (или во вкладке главного меню "Edit"). Если соединение произойдет успешно, то в правом списке отобразится псевдоним собеседника. После успешного со-

единения пользователи могут обмениваться сообщениями, набирая сообщения в нижней строке ввода и нажимая кнопку "Send"или клавишу "Enter"на клавиатуре. Процесс обмена сообщениями представлен на рис.4.

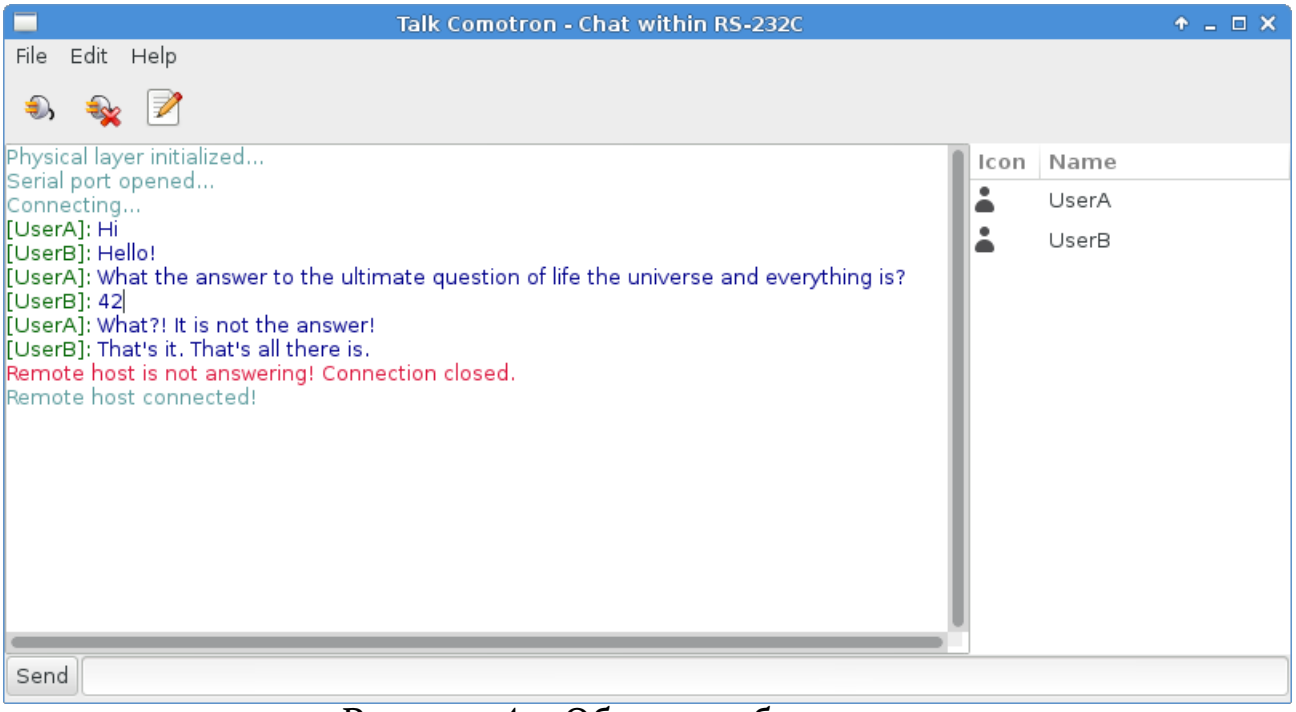


Рисунок 4 – Обмен сообщениями.

Для удобства пользователей есть возможность сохранять и загружать файлы истории, диалоги сохранения и загрузки представлены на рисунках 5 и 6. История разговора загружается в текстовую область чата и подсвечивается специальным цветом, состояние главного окна с загруженной историей представлено на рисунке рис. 7. Также пользователь может получить подробную информацию о программе из диалога «О программе», представленного на рис. 8.

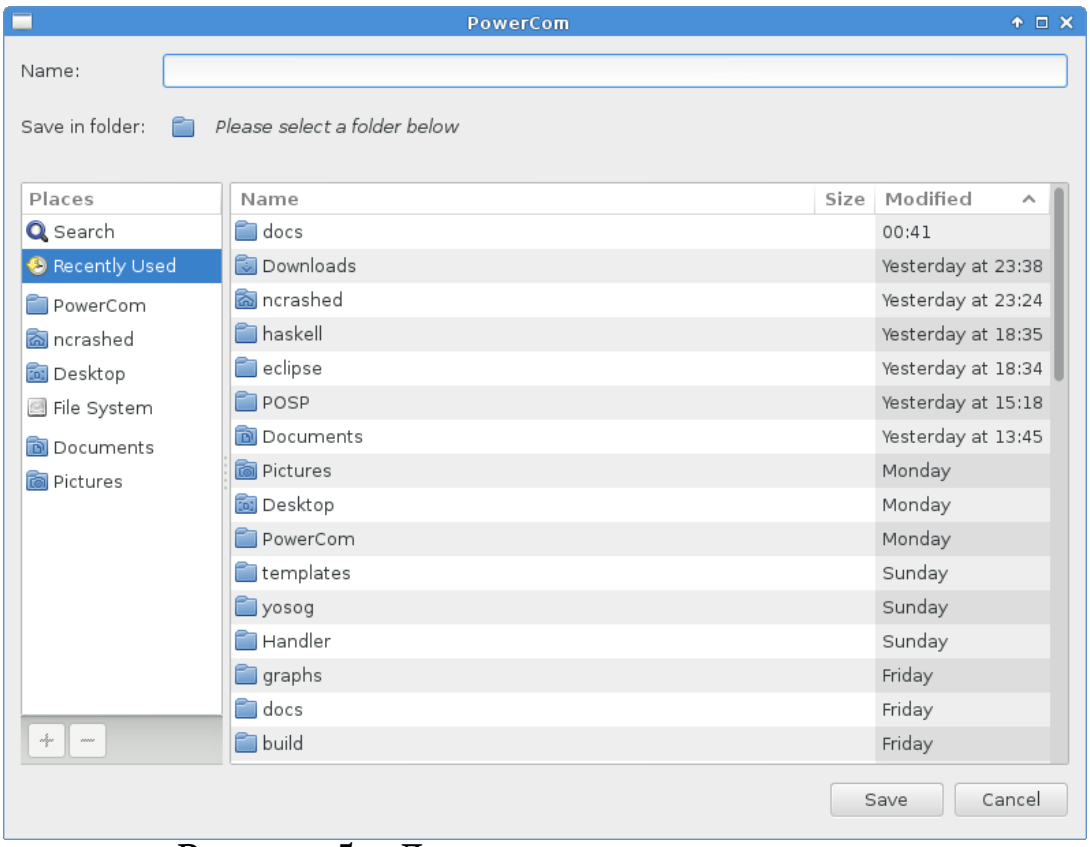


Рисунок 5 – Диалог сохранения истории.

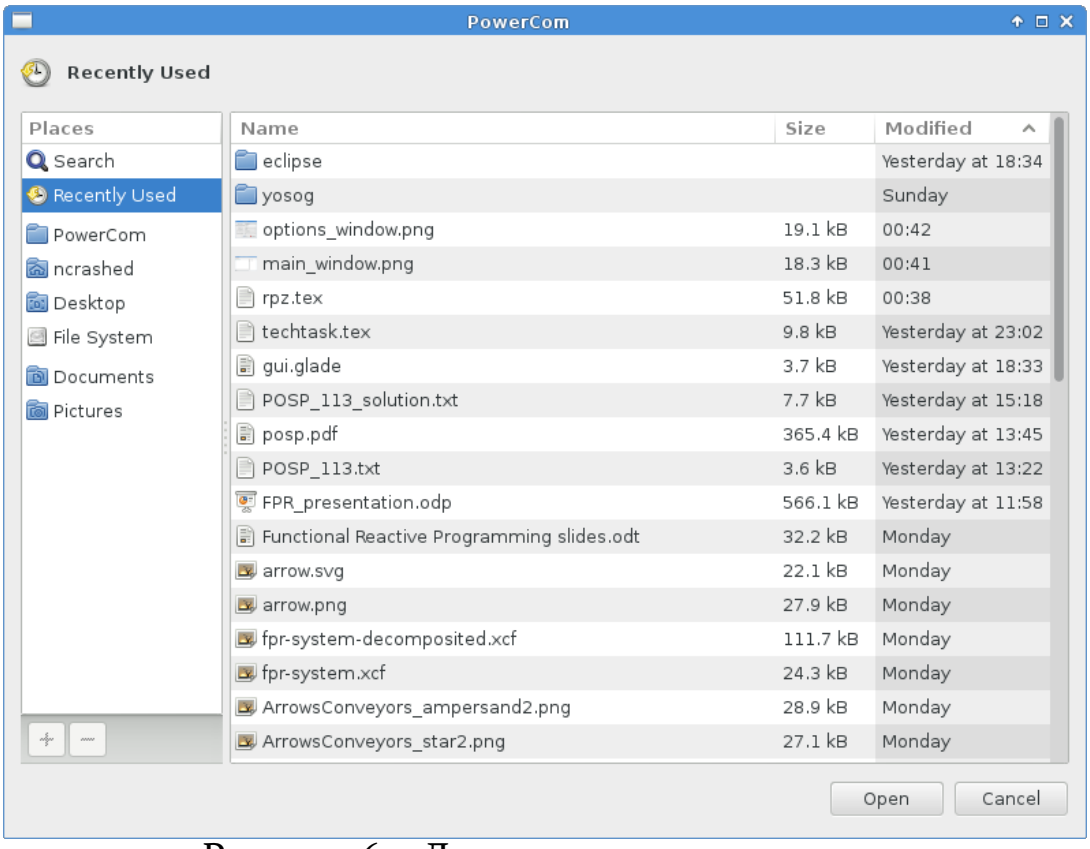


Рисунок 6 – Диалог загрузки истории.

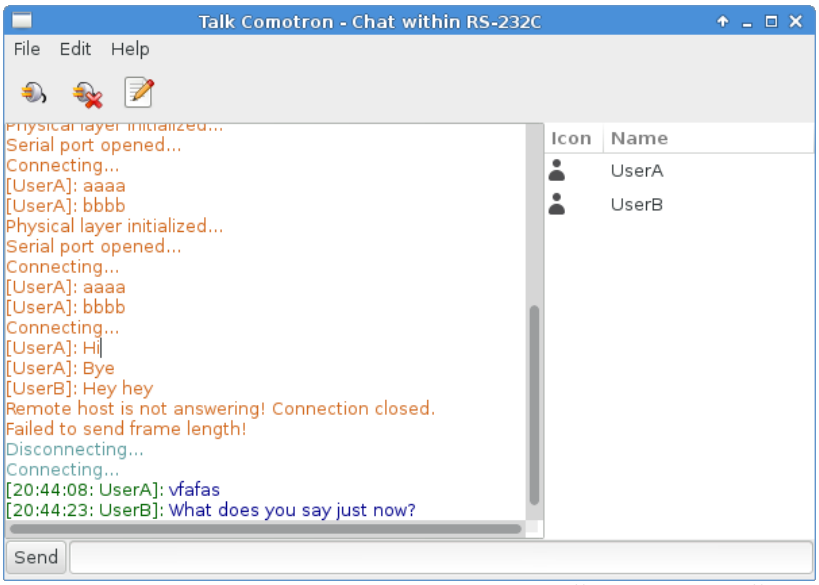


Рисунок 7 – Главное меню с загруженной историей разговора.

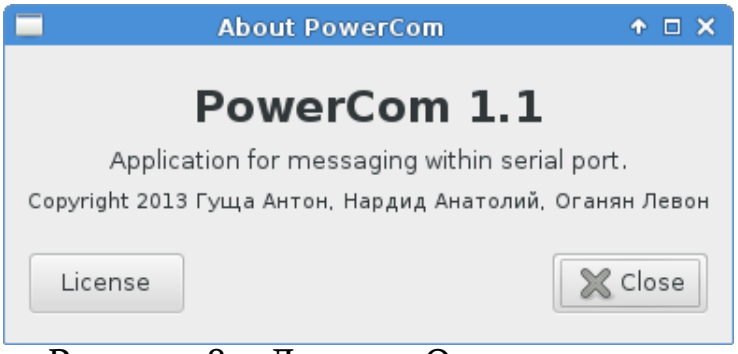


Рисунок 8 – Диалог «О программе».