Московский государственный технический университет им. Н.Э. Баумана

Утверждаю:
Галкин В.А. "__"_____2017 г.

Курсовая работа по дисциплине «Сетевые технологии в АСОИУ» «Локальная безадаптерная сеть»

(вариант №19)

Пояснительная записка
(вид документа)

писчая бумага
(вид носителя)

22
(количество листов)

ИСПОЛНИТЕЛИ:

студенты группы ИУ5-63

Лузин Д.С. _____

Ореликов М.Г. _____

Бодунов А.Г. _____

Москва - 2017

1. Введение.

Данная программа, шифр «SendMessage», выполненная в рамках курсовой работы по предмету «Сетевые технологии в АСОИУ», предназначена для организации обмена текстовыми сообщениями между соединёнными с помощью интерфейса RS-232C компьютерами. Программа позволяет обмениваться двум или более машинам (количество машин задается при создании кольца), для передачи коротких сообщений(до 255 символов), в режиме диалога между любой парой пользователей при условии, запуска этой программы на всех компьютерах и логическом соединении с кольцом обоих пользователей. Но по техническим причинам в силу договоренности данная программа использует установку нульмодемного соединиения с помощью SerialMon. Для работы программы требуется 1 компьютер.

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 98 или выше. Поэтому было решено выполнить реализацию программы с помощью среды разработки Qt4 и Python версии 2.7.

3. Определение структуры программного продукта.

При взаимодействии компьютеров между собой выделяются 3 уровня: нижний уровень должен обеспечивать физическое соединение, средний – логическое соединение, верхний – обеспечить интерфейс пользователя. Уровни называются, соответственно: физический, канальный и прикладной (см. Приложение «Структурная схема программы»).

Уровни выполняют следующие функции:

- Физический уровень предназначен для сопряжения компьютера со средой передачи (нуль-модемный кабель).

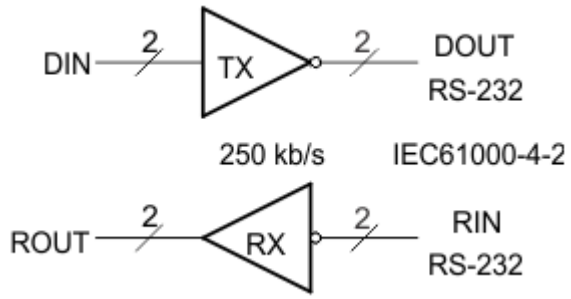
- Канальный уровень предназначен для установления и поддержания соединения, формирования и проверка пакетов обмена для протоколов верхних уровней.

- Прикладной уровень предназначен для выполнения прикладных задач программы.

4. Физический уровень.

4.1. Функции физического уровня.

Вначале определимся с терминологией. В передаче данных по последовательному порту участвуют два блока:
1. UART (Universal Asynchronous Reciver-Transmitter), который соединен с процессором по параллельной шине и находится в пространстве ввода-вывода процессора по фиксированным базовым адресам (2F8 для COM1, 3F8 для COM2 и т.д). Он преобразует информацию от/к процессора в параллельном коде (8, 16 бит) в последовательный код, который может быть передан по единственному проводнику (для каждого направления: приема и передачи). Пример UART – микросхема 8250, или ее более поздняя версия 16450, которая имеет 16-байтный буфер FIFO и более высокие скорости обмена.
2. Драйвер, обеспечивающий преобразование логических уровней микросхемы UART (0-5 Вольт или 0-3 Вольт в более современных машинах) в уровни напряжений COM-порта. Диапазон напряжений COM-порта составляет от -15 до -3В для передачи логической 1 и +3…+15В для передачи логического 0. Обычно микросхема драйвера содержит 2 преобразователя уровня, поскольку аппаратное управление потоком используется не во всех случаях, только для TX и RX. Пример драйвера – микросхема MAX232:



Если же необходимо поддерживать аппаратное управление потоком, устанавливается несколько драйверов MAX232, или устанавливается драйвер, поддерживающий все 8 сигналов интерфейса RS-232.
Иногда одна микросхема объединяет в себе функции UART и драйвера, но, поскольку в последнее время UART принято встраивать даже в дешевые микроконтроллеры стоимостью до 2$, микросхемы драйверов обычно выпускаются отдельно.

Основными функциями физического уровня являются:
1. Задание параметров COM-порта.
2. Установление физического канала.
3. Разъединение физического канала.
4. Передача информации из буфера памяти в микросхему интерфейса UART.
5. Прием информации от другого компьютера и ее накопление в буфере.
6. Межуровневый интерфейс.

4.2. Описание физического уровня.

Последовательная передача данных означает, что данные передаются по единственной линии в каждом направлении. 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). Иногда бит проверки на четность может отсутствовать (обозначение в схеме - N).

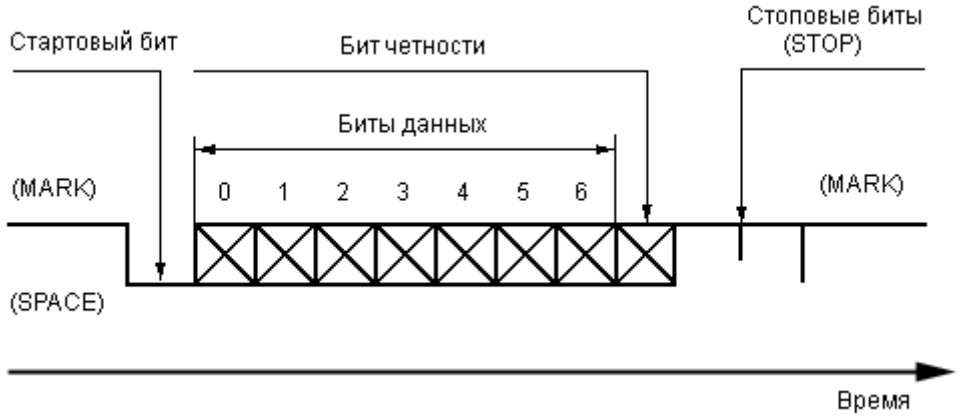


Рисунок 1.

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

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

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

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

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

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

Другая важная характеристика — скорость передачи данных. Она также должна быть одинаковой для передатчика и приемника в пределах одного соединения 2-х COM-портов, но может быть различной для различных соединений в кольце.

Скорость передачи данных обычно измеряется в бодах.

Иногда используется другой термин — биты в секунду (bits per second - 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».

Интерфейс 9-ти контактный разъем.

Номер контакта	Обозначение	Назначение
1	DCD	Обнаружение несущей
2	RD	Принимаемые данные
3	TD	Отправляемые данные
4	DTR	Готовность терминала к работе
5	SG	Земля сигнала (схемная)
6	DSR	Готовность DCE
7	RTS	Запрос передачи
8	CTS	Готовность к передаче
9	RI	Индикатор вызова

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) 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 справен, 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.

4.4. Работа с COM-портом средствами pySerial

4.4.1 Начальная настройка структуры, определяющей параметры COM-порта, перед его открытием

Драйвер операционной системы использует эту структуру для работы с COM-портом. Драйвер ОС может получить эти параметры и после открытия порта, но проще настроить перед порт открытием.

```
ser = serial.Serial(
 port='/dev/ttyUSB1',#для Linux и преобразователя COM-USB, или просто "COM1",
                     #если мы используем COM-порт на материнской плате с Windows
    baudrate=9600,   #определяем скорость работы порта
    parity=serial.PARITY_ODD,     #определяем четность
    stopbits=serial.STOPBITS_TWO, #определяем количество стоповых бит
    bytesize=serial.SEVENBITS     #определяем количество бит в посылке
)
```

Можно заполнять структуру параметров COM-порта и таким образом, иногда это удобнее:

```
ser = serial.Serial()
ser.port = "/dev/ttyUSB1"
ser.baudrate = 9600
ser.bytesize = serial.SEVENBITS     #определяем количество бит в посылке
ser.parity = serial.PARITY_ODD,     #определяем четность
ser.stopbits = serial.STOPBITS_TWO, #определяем количество стоповых бит
#ser.timeout = None                 #блочное чтение
ser.timeout = 1                     #посимвольное чтение
#ser.timeout = 2                    #тайм-аут для блочного чтения
ser.xonxoff = False                 #отключили программное управление потоком
ser.rtscts = True       #включили аппаратное (RTS/CTS) управление потоком
ser.dsrdtr = True       #включили аппаратное (DSR/DTR) управление потоком
ser.writeTimeout = 2    #таймаут для записи
```

4.4.2 Открытие порта

```
ser.isOpen() # Открываем порт
```

4.4.3 Запись данных в порт

Здесь используется свойство динамического преобразования типов в Python, когда мы просто создаем тестовую строку и передаем указатель на буфер, содержащий эту строку, функции write.

```
input = "Hello World!"
ser.write(input + '\r\n')# Посылаем созданную строку в последовательный порт
```

4.4.4 Чтение данных из порта

```
  out = '' # создаем пустой строковый массив
 while ser.inWaiting() > 0: #ждем
     out += ser.read(1)     #читаем очередной символ из порта и добавляем его
                            #к созданному строковому массиву
```

4.4.5 Сброс входного буфера

Все принятые по COM интерфейсу, но не прочитанные процессором из UART данные уничтожаются

```
  ser.flushInput() #очищаем входной буфер
```

4.4.6 Сброс выходного буфера

Все записанные процессором в UART, но еще не переданные по COM интерфейсу данные уничтожаются

```
ser.flushOutput()#очищаем выходной буфер
```

4.4.7 Закрытие порта

```
ser.close() #закрываем порт
```

5. Канальный уровень.

5.4 Функции канального уровня.

На канальном уровне выполняются следующие функции:

1. Запрос логического соединения;
2. Разбивка данных на блоки (кадры);
3. Извлечение данных из информационных кадров
4. Управление передачей кадров;
5. Обеспечение необходимой последовательности блоков данных, передаваемых через межуровневый интерфейс;
6. Контроль и обработка ошибок;
7. Проверка поддержания соединения;
8. Запрос на разъединение логического соединения;
9. Межуровневый интерфейс.

5.5 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 чтобы 2 стороны были логически соединены с кольцом, тем самым проверяется готовность сторон принимать и отправлять данные.

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.(кадр Link).Также необходимо предусмотреть кадр, который посылается пользователем и уведомляет всех других пользователей о его логическом присоединении(если он отсоединен) или отсоединении от кольца(если он присоединен)(кадр Off)

5.6 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 (контрольная сумма), значения которых зависят от фактического содержания кадра и присоединяются 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 (та же контрольная сумма). 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 над кодовыми комбинациями к действием над 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), степень 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

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

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, наибольшее число разрядов остатка R(x) не превышает числа r.

Умножая обе части равенства (1) на P(x) и произведя некоторые перестановки, получаем: $F(x) = C(x)\,P(x) = Q(x)\,x^r + R(x)$

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

1) умножение 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);

2) умножения 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 4 бита, после чего делится на полином и результат деления, 1 байт, передаётся по сети, т.е. 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.

5.7 Формат кадров.

Кадры, передаваемые с помощью функций канального уровня, имеют различное назначение. Выделены супервизорные и информационные кадры.

5.7.1 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. Формат эти кадров:

StartByte	Type	StopByte
Флаг начала кадра	Тип супервизорного кадра	Флаг конца кадра

5.7.2 Супервизорные кадры передачи параметров.

Супервизорные кадры передачи параметров используются для синхронизации параметров COM-портов, как принимающего, так и отправляющего. Кадр данного типа формируется когда на одном из компьютеров изменяются параметры. Формат эти кадров:

StartByte	Type	Data	StopByte
Флаг начала кадра	Тип супервизорного кадра	Параметры COM-порта	Флаг конца кадра

5.7.3 Информационные кадры.

Информационные кадры применяются для передачи закодированных циклическим кодом пользовательских сообщений. Формат эти кадров:

StartByte	Type	Data	StopByte
Флаг начала кадра	Тип супервизорного кадра	Закодированные данные (текстовая строка)	Флаг конца кадра

Кадр можно разделить на несколько блоков – флаг начала кадра, тип кадра, данные и флаг конца кадра.

Флаги начала и конца кадра представляют собой байты, с помощью которых программа выделяет кадр, определяя соответственно начало и конец кадра.

Поле типа кадра обеспечивает правильное определение и распознавание разновидностей кадров и обработки их соответствующими процедурами.

Данные представляют собой либо закодированную строку в информационном кадре или параметры порта в супервизорном кадре передачи параметров.

5.8 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 выполнен в среде Qt4. При его разработке принимались во внимание соображения по простоте, удобству и функциональности интерфейса.

В общем виде интерфейс программы организован в виде окна станции, отвечающей за логическое соединение, окон меню пользователей и окон отображения диалогов между двумя пользователями.

5.8.1 Главное окно

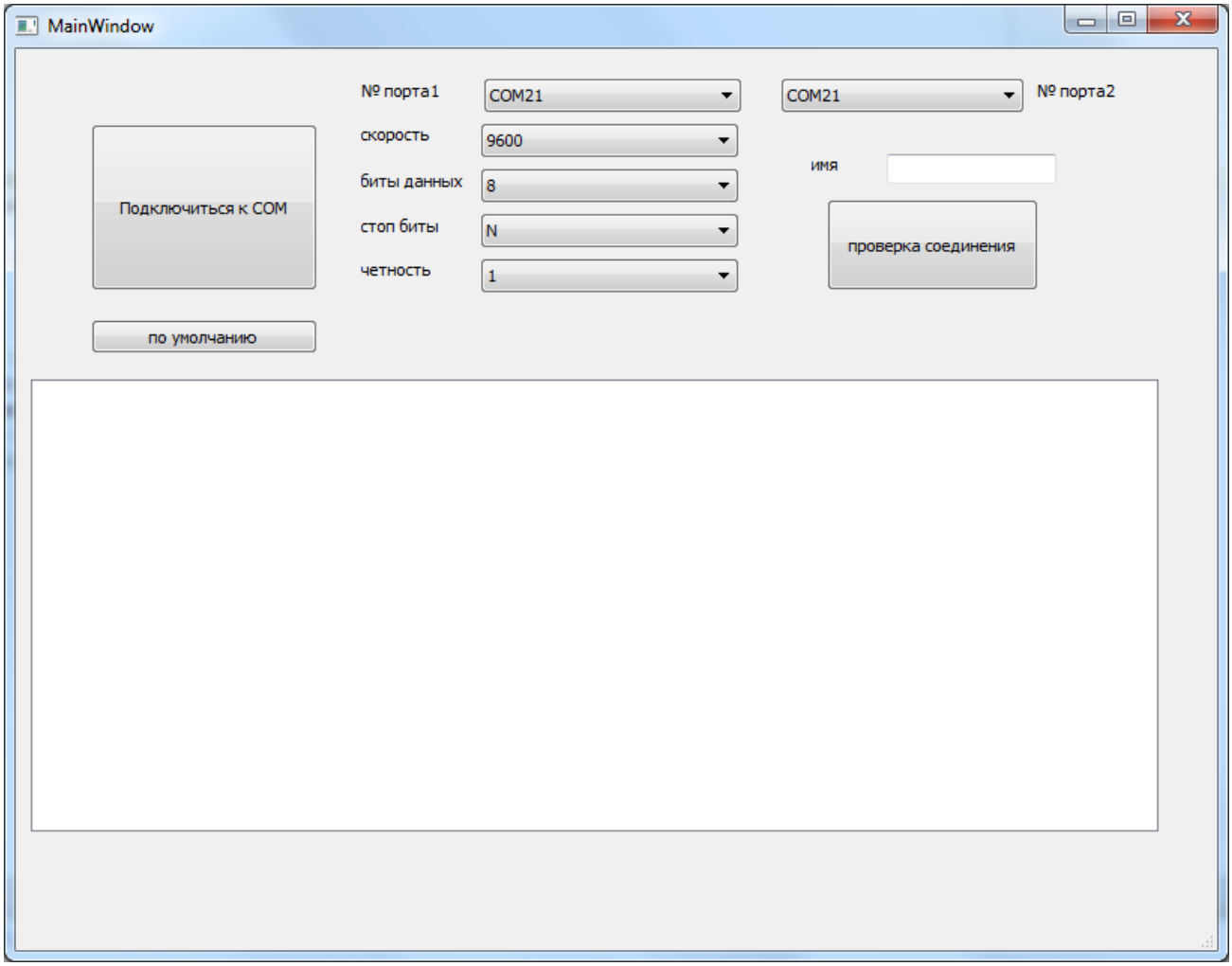
Главным окном программы является окно станции , отвечающей за логическое соединение ComSend.

Контекстное меню и кнопки на главном окне позволяют администратору кольца выполнять следующие операции:

1. Задать структуру блока данных физического уровня путем указания скорости передачи информации, количества информационных, стоповых бит и типа контроля четности («Настройки порта»)
2. Выбрать номера COM портов (входного и выходного) для организации физического и логического соединения
3. Ввести имя пользователя

4. Зарегистрировать пользователя кольца в соответствии с параметрами из предыдущих 3-х пунктов (первый зарегистрированный пользователь считается администратором кольца и работает не только как администратор, но и как пользователь)
5. Выйти из программы нажатием стандартной кнопки закрытия окна в правом верхнем углу
6. Инициализировать проверку соединения с отображением результата в поле

С учетом этих требований разработан следующий интерфейс главного окна:

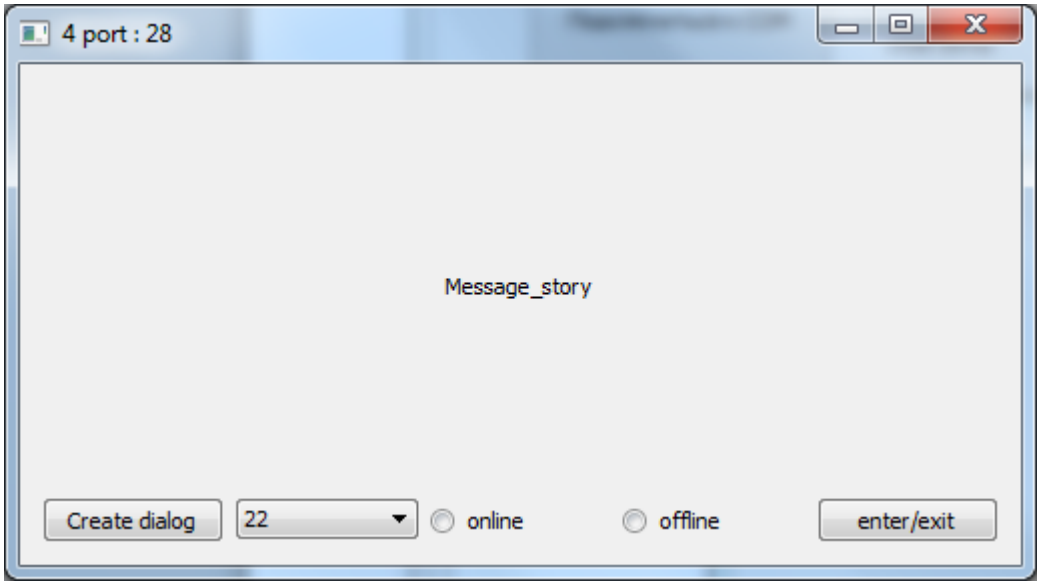


5.8.2 Окно меню пользователя.

Меню пользователя и кнопки на главном окне позволяют пользователю программы выполнить следующие операции:

1. Выбрать имя пользователя, с которым он хочет создать диалог.
2. Создать диалог с этим пользователем
3. Установить/разорвать логическое соединение

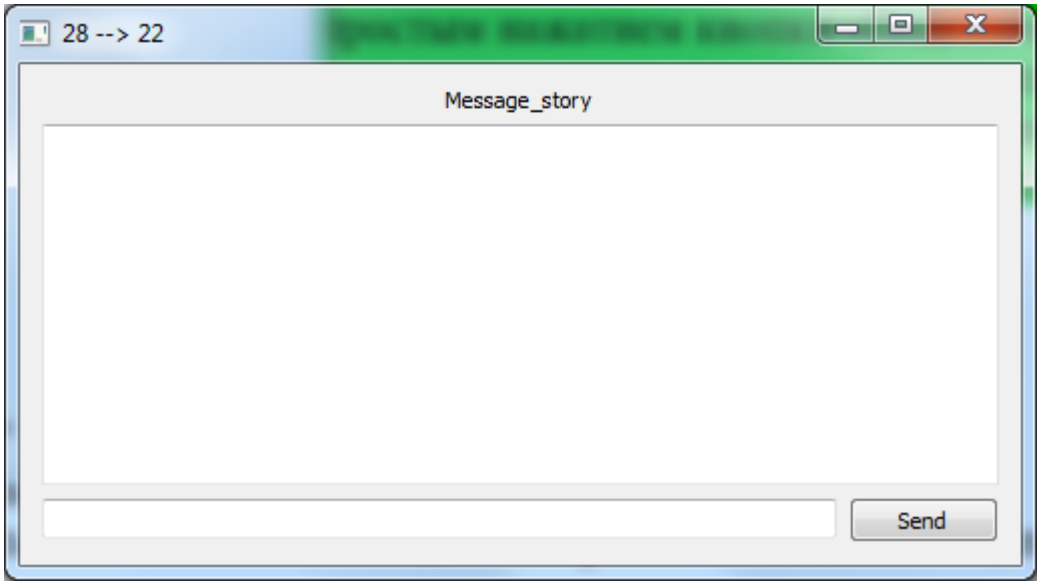
С учетом этих возможностей разработан следующий вариант окна меню пользователя:



5.8.3 Окно отображения диалога между двумя пользователями

Для простоты набора сообщения имеются следующие возможности:

1. Набор и редактирование сообщения в окне “Message_story”;
2. Отправка его кнопкой “Отправить”.
3. Просмотр диалога в окне с возможностью наблюдать направление сообщения (входящее/исходящее).



6. Список использованной литературы.

1. В.А. Галкин, Ю.А. Григорьев «Телекоммуникации и сети»
2. Прохоренок Н.А. Python 3 и PyQt. Разработка приложений. СПб. БХВ-Петербург, 2012. 704 с.: ил.
3. PySerial API. https://pythonhosted.org/pyserial/pyserial_api.html#module-functions-and-attributes
4. UART 16550 Transceiver Datasheet. http://www.ti.com/lit/ds/symlink/pc16550d.pdf