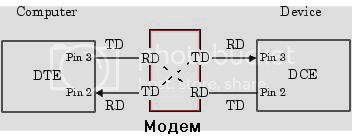
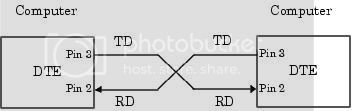
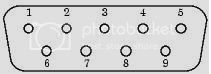
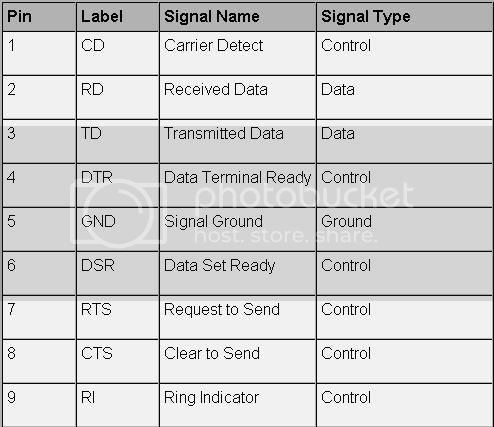
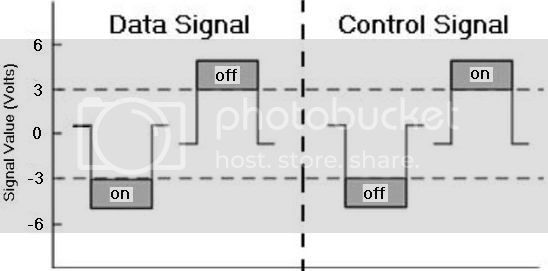
**Последовательный порт**

Последовательный порт (далее ПП) удобный инструмент для общения между разными периферийными устройствами (как собранные самостоятельно на основе какого-нибудь МК, так и заводские: принтеры, осциллографы и т.д.) с одной стороны, и ПК с другой. На сегодняшний день наиболее популярные из всех ПП являются **RS232 стандарт** (переводится как «Recommended Standard») за его простоту и **USB стандарт** («Universal Serial BUS») за его резвость.  
USB бесспорно вещь полезная, но сложная в управлении. Поскольку многим самодельным устройствам **бешенный** обмен данными с ПК не нужен, тогда на помощи приходит простой, надежный и многоопытный RS232 Интерфейс.  
  
  
По RS232 стандарту устройства участвующие в обмене данными бывают двух типов:  
*Data Terminal Equipment* (**DTE**) (устройство отдающее команды — ведущий) и  
*Data Circuit-Terminating Equipment* (**DCE**) (периферия, обслуживающая хозяина — ведомый). Нередко, некоторые периферийные устройства ведут себя как **DTE** (например осциллографы, или наши с вами девайсы).  
  
**Типы соединения**  
  
**Модемное соединение** — подразумевает наличие некой иерархии, то есть в случае когда в обмене данными участвуют больше чем два устройства им необходим некий арбитр (модем), разрешающий в определенный момент времени отсылать данные только одному устройству (в то время как читать могут хоть все остальные). Модемом может быть что угодно: отдельный девайс, или один из участников обмена данными, главное не допустить потери данных.  
  


**нуль-модемное соединение**

В случае когда устройств только два, или есть явный ведущий которого слушаются все остальные, никакого посредника им не нужно, а это означает что к их общению больше никто не подключится, и никакого арбитра в лице модема им не надо ( в отличие от предыдущего типа соединения, когда к одному принтеру можно подключить штук 10 ПК ). Опять-же главное не допустить одновременной отправки данных — в определенный момент времени, общается может только одна пара устройств. Такое соединение называется **нуль-модемное соединение**:  
  


**Типы передач данных**  
  
Минимальное количество проводков необходимое для обмена данными равно двум , если передача является односторонней (**[Tx, GND]**). В случае когда необходимо полноценное — двухстороннее общение число проводков возрастает до трех (**[Rx, Tx, GND]**). Большинство периферийных устройств поддерживают одновременную передачу и прием данных — **full-duplex**, но если один из собеседников на такое не способен, обмен переходит в разряд неполноценных — **half-duplex** (пока один не закончил передачу/прием другой ожидает).  
  
**Распиновка COM разъёма**  
  
  
  
  
  
В столбце **Signal Name**, *DATA Terminal* можно заменить на *ПК* (то есть ***Data Terminal Ready*** соответствует ***ПК готов к работе***), а *DATA Set* на *Периферия*.  
  
Как следует из предыдущей таблицы, все пины делятся на **управляющие (control pins)** и **транспортные (Data pins)**. Каждый пин в определенный момент времени может находится только в одном из двух состояний: **активном (on)** или **неактивном (off)**. Чтобы не запутаться, и как-то защитить данные от помех, разработчики решили что во время передачи данных они должны быть сначала усилены (**+5В –> +12В**, **0В –> -12В** ) а потом инвертированы, в то время как управляющие сигналы просто их усилили относительно общего провода).  
  
  
  
Назначение управляющих пинов (**[RTS, CTS]**, **[DTR, DSR]** и **[CD, RI]**) сводится к следующему:  
  
• Отслеживать состояние собеседника  
• Отслеживать поток данных  
  
Пара **[RTS, CTS]** — используется для обозначения готовности данной пары устройств к передачи/приему соответственно.  
  
Пример:  
  
1. *DTE* устройство устанавливает ***RTS = on***, сигнализируя о том что оно готово к приему данных. Если устройство получило достаточное количество данных то устанавливаем ***RTS =off***.  
2. *DCE* устройство устанавливает ***CTS =on***, сигнализируя о том что оно готово к приему данных. Если устройство получило достаточное количество данных то устанавливаем ***CTS =off***.  
  
Кто каким пином будет управлять (то есть кому быть *DTE* а кому *DCE*) решать вам. Соответственно программы управления этими устройствами должны выставить *RTS(выход)/CTS (вход)*, или наоборот, иначе могут быть глюки.  
  
Пара **[DTR, DSR]** — большинство устройств используют эти пины для сигнала что они подключены и готовы к работе.  
  
Пример:  
  
1. *DTE* устройство устанавливает ***DTR=on***, сообщая *DCE* устройству что оно готово к работе. Соответственно когда *DTE* устанавливает ***DTR=off***, то оно больше не желает (или не может) общатся (положило трубку :) )  
2. *DCE* устройство устанавливает ***DSR=on***, сообщая что оно подключено, а когда ***DSR=off*** – оно отключено.  
  
Такой метод контроля потока данных называется – ***hardware handshaking*** (чтото вроде ***аппаратное управление***). Пары **[DTR, DSR]** и **[RTS, CTS]** могут быть с легкостью взаимо-заменены без всякого ущерба.  
  
Пара **[CD, RI]** – используется для обозначения (в тот самом случае когда один принтер на много ПК) что в данный момент линии передачи данных кем-то заняты.  
Как правило этой парой управляет модем, но не обязательно.  
  
  
  
где:  
  
• St – Стартовый Бит (начало передачи данных) – логический ноль  
• 0..8 – позиция бита (данных) в пакете (позиция «0» – LSB)  
• P – бит парности (проверка успешной передачи данных)  
• Sp1,Sp2 – стоп биты (завершают передачу пакета) – логическая единица  
• [] – в скобках обозначены биты которые могут отсутствовать  
(биты данных с 5 по 8 так или иначе будут переданы, но не рассмотрены — мусор)  
• IDLE – ожидание (логическая единица)  
  
Во время передачи — данные инвертируются, так что если будете проверять осциллографом как отсылается пакет — не пугайтесь.  
  
Часто формат пакета обозначается следующим образом: **8-N-1** (8 бит данных, без бита проверки, один стоп бит) или **5-E-2** (5 бит данных (3 бита мусора), с проверкой на четность, два стоп бита).

**Интерфейс RS-232 (COM-порт)**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Разъем DB-9M  9-контактная вилка типа DB-9M Нумерация контактов со стороны штырьков Направление сигналов указано относительно хоста (компьютера) | |  |  |  |  | | --- | --- | --- | --- | | **Контакт** | **Сигнал** | **Направление** | **Описание** | | 1 | CD | Вход | Обнаружена несущая | | 2 | RXD | Вход | Принимаемые данные | | 3 | TXD | Выход | Передаваемые данные | | 4 | DTR | Выход | Хост готов | | 5 | GND | – | Общий провод | | 6 | DSR | Вход | Устройство готово | | 7 | RTS | Выход | Хост готов к передаче | | 8 | CTS | Вход | Устройство готово к приему | | 9 | RI | Вход | Обнаружен вызов | |

**Назначение выводов 25-контактного разъема**

|  |  |
| --- | --- |
| Разъем DB-25M | 25-контактная вилка типа DB-25M Нумерация контактов со стороны штырьков Направление сигналов указано относительно хоста (компьютера) |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  | | --- | --- | --- | --- | | **Контакт** | **Сигнал** | **Направление** | **Описание** | | 1 | SHIELD | – | Экран | | 2 | TXD | Выход | Передаваемые данные | | 3 | RXD | Вход | Принимаемые данные | | 4 | RTS | Выход | Хост готов к передаче | | 5 | CTS | Вход | Устройство готово к приему | | 6 | DSR | Вход | Устройство готово | | 7 | GND | – | Общий провод | | 8 | CD | Вход | Обнаружена несущая | | 9 | – | – | Резерв | | 10 | – | – | Резерв | | 11 | – | – | Не используется | | 12 | SCD | Вход | Обнаружена несущая #2 | | 13 | SCTS | Вход | Устройство готово к приему #2 | | |  |  |  |  | | --- | --- | --- | --- | | **Контакт** | **Сигнал** | **Направление** | **Описание** | | 14 | STXD | Выход | Передаваемые данные #2 | | 15 | TRC | Вход | Тактирование передатчика | | 16 | SRXD | Вход | Принимаемые данные #2 | | 17 | RCC | Вход | Тактирование приемника | | 18 | LLOOP | Выход | Локальная петля | | 19 | SRTS | Выход | Хост готов к передаче #2 | | 20 | DTR | Выход | Хост готов | | 21 | RLOOP | Выход | Внешняя петля | | 22 | RI | Вход | Обнаружен вызов | | 23 | DRD | Вход | Определена скорость данных | | 24 | TRCO | Выход | Тактирование внешнего передатчика | | 25 | TEST | Вход | Тестовый режим | |

Из таблиц видно, что 25-контактный интерфейс отличается наличием полноценного второго канала приема-передачи (сигналы, обозначенные "#2"), а также многочисленных дополнительных управляющих и контрольных сигналов. Однако, часто, несмотря на наличие в компьютере "широкого" разъема, дополнительные сигналы на нем просто не подключены.

**Электрические характеристики**

**Логические уровни передатчика:** "0" – от +5 до +15 Вольт, "1" – от -5 до -15 Вольт.

**Логические уровни приемника:** "0" – выше +3 Вольт, "1" – ниже -3 Вольт.

**Максимальная нагрузка на передатчик:** входное сопротивление приемника не менее 3 кОм.

Данные характеристики определены стандартом как минимальные, гарантирующие совместимость устройств, однако реальные характеристики обычно существенно лучше, что позволяет, с одной стороны, питать маломощные устройства от порта (например, так спроектированы многочисленные самодельные data-кабели для сотовых телефонов), а с другой – подавать на вход порта *инвертированный* TTL-уровень вместо двуполярного сигнала.

**Описание основных сигналов интерфейса**

**CD** – Устройство устанавливает этот сигнал, когда обнаруживает несущую в принимаемом сигнале. Обычно этот сигнал используется модемами, которые таким образом сообщают хосту о обнаружении работающего модема на другом конце линии.

**RXD** – Линия приема хостом данных от устройства. Подробно описана в разделе "Протокол обмена данными".

**TXD** – Линия передачи хостом данных к устройству. Подробно описана в разделе "Протокол обмена данными".

**DTR** – Хост устанавливает этот сигнал, когда готов к обмену данными. Фактически сигнал устанавливается при открытии порта коммуникационной программой и остается в этом состоянии все время, пока порт открыт.

**DSR** – Устройство устанавливает этот сигнал, когда включено и готово к обмену данными с хостом. Этот и предыдущий (DTR) сигналы должны быть установлены для обмена данными.

**RTS** – Хост устанавливает этот сигнал перед тем, как начать передачу данных устройству, а также сигнализирует о готовности к приему данных от устройства. Используется при аппаратном управлении обменом данными.

**CTS** – Устройство устанавливает этот сигнал в ответ на установку хостом предыдущего (RTS), когда готово принять данные (например, когда предыдущие присланные хостом данные переданы модемом в линию или есть свободное место в промежуточном буфере).

**RI** – Устройство (обычно модем) устанавливает этот сигнал при получении вызова от удаленной системы, например при приеме телефонного звонка, если модем настроен на прием звонков.

**Протокол обмена данными**

В протоколе RS-232 существуют два метода управления обменом данных: аппаратный и программный, а также два режима передачи: синхронный и асинхронный. Протокол позволяет использовать любой из методов управления совместно с любым режимом передачи. Также допускается работа без управления потоком, что подразумевает постоянную готовность хоста и устройства к приему данных, когда связь установлена (сигналы DTR и DSR установлены).

**Аппаратный метод управления** реализуется с помощью сигналов RTS и CTS. Для передачи данных хост (компьютер) устанавливает сигнал RTS и ждет установки устройством сигнала CTS, после чего начинает передачу данных до тех пор, пока сигнал CTS установлен. Сигнал CTS проверяется хостом непосредственно перед началом передачи очередного байта, поэтому байт, который уже начал передаваться, будет передан полностью независимо от значения CTS. В полудуплексном режиме обмена данными (устройство и хост передают данные по очереди, в полнодуплексном режиме они могут делать это одновременно) снятие сигнала RTS хостом означает его переход в режим приема.

**Программный метод управления** заключается в передаче принимающей стороной специальных символов остановки (символ с кодом 0x13, называемый XOFF) и возобновления (символ с кодом 0x11, называемый XON) передачи. При получении данных символов передающая сторона должна соответственно остановить передачу или возобновить ее (при наличии данных, ожидающих передачи). Этот метод проще с точки зрения реализации аппаратуры, однако обеспечивает более медленную реакцию и соответственно требует заблаговременного извещения передатчика при уменьшении свободного места в приемном буфере до определенного предела.

**Синхронный режим передачи** подразумевает непрерывный обмен данными, когда биты следуют один за другим без дополнительных пауз с заданной скоростью. Этот режим COM-портом **не поддерживается**.

**Асинхронный режим передачи** состоит в том, что каждый байт данных (и бит контроля четности, в случае его наличия) "оборачивается" синхронизирующей последовательностью из одного нулевого старт-бита и одного или нескольких единичных стоп-битов. Схема потока данных в асинхронном режиме представлена на рисунке.

https://denvo.ru/r/image/hard/interfaces/rs232-bits.png

**Один из возможных алгоритмов работы приемника** следующий:

1. Ожидать уровня "0" сигнала приема (RXD в случае хоста, TXD в случае устройства).
2. Отсчитать половину длительности бита и проверить, что уровень сигнала все еще "0"
3. Отсчитать полную длительность бита и текущий уровень сигнала записать в младший бит данных (бит 0)
4. Повторить предыдущий пункт для всех остальных битов данных
5. Отсчитать полную длительность бита и текущий уровень сигнала использовать для проверки правильности приема с помощью контроля четности (см. далее)
6. Отсчитать полную длительность бита и убедиться, что текущий уровень сигнала "1".
7. Вернуться к ожиданию начала следующего байта данных (шаг 1)

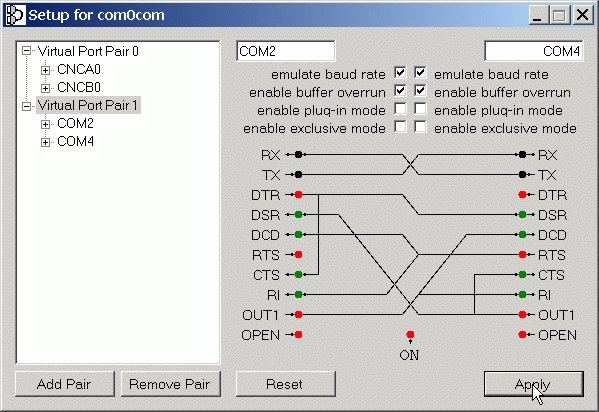
**Протокол имеет ряд переменных параметров**, которые должны быть приняты одинаковыми на стороне приемника и на стороне передатчика для успешного обмена данными:

* **Скорость обмена данными** задается в битах в секунду, определяя длительность одного бита, выбирается из ряда стандартных значений (300, 1200, 2400, 4800, 9600, 19200, 38400, 57600), но могут быть и нестандартными, если поддерживаются обеими сторонами;
* **Количество бит данных** может быть от 4 до 8;
* **Контроль четности** может быть четным ("even", когда общее число единичных битов в принятых данных, включая сам бит четности, должно быть четным), нечетным ("odd", когда общее число единичных битов в принятых данных, включая сам бит четности, должно быть четным) или вообще отсутствовать;
* **Длина стоп-бита** может составлять одну, полторы или две длительности бита.

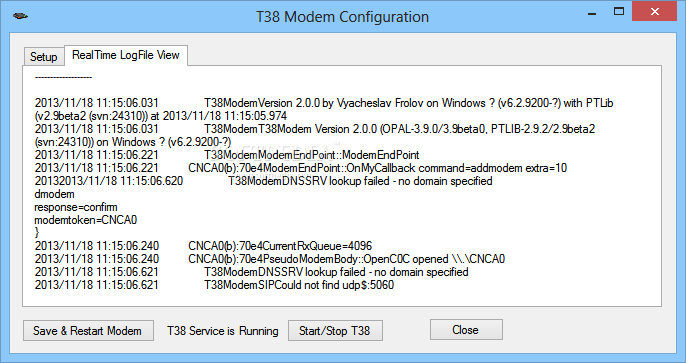
**ЭМУЛЯТОРЫ**

**com0com**]

**Null-modem emulator** [1] это драйвер виртуального последовательного порта Windows, работающий в режиме ядра (kernel-mode driver), с открытым исходным кодом, доступный под защитой лицензии GPL. Null-modem emulator позволяет Вам создавать неограниченное количество пар виртуальных COM-портов. Каждая пара COM-портов состоит из 2 COM-портов, виртуально соединенных друг с другом каналом данных. Выход одного COM-порта пары соединяется со входом другого COM-порта пары, и наоборот. Следовательно, можно использовать любую пару для соединения любого одного приложения, рассчитанного на работу с COM-портом, с другим приложением, также работающего с COM-портом, и они будут передавать данные друг другу.



Null-modem emulator может использоваться для предоставления последовательного интерфейса к эмуляторам устройств. В этом случае программа эмуляции устройства использует один порт пары, и другой порт пары может использоваться операционной системой Windows или приложением DOS, которое требует COM-порта для обмена с устройством. Например, прием/отправка факсов через сетевой канал IP: Вы можете соединить приложение Windows Fax с приложением **t38modem** [2] (T38FAX Pseudo Modem, это часть проекта t38modem) через пару виртуальных COM-портов.



[**com2tcp**]

Также Null-modem можно использовать для последовательного интерфейса к редиректорам COM-порта. Например, с приложением com2tcp [3] (редиректор COM-порта в сокет TCP, это часть проекта com0com [1]) Вы можете обмениваться через последовательный интерфейс с серверами TCP/IP (очень полезно для работы с модулями наподобие ESP8266 [4]). Если Вы чувствуете, что com2tcp это примерно то, что искали, но все-таки в нем нет каких-то нужных возможностей, то присмотритесь к проекту hub4com вместо него (например, для поддержки RFC 2217).



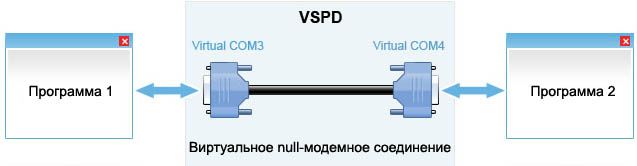
[**hub4com**]

Вместе с приложением hub4com [5] (HUB for communications, хаб для обмена данными, это часть проекта com0com [1]) можно  несколькими приложениями обрабатывать данные и сигналы одного устройства с последовательным интерфейсом (например, несколько приложений могут использовать одни и те же данные от одного устройства GPS). Также можно использовать реальные последовательные порты находящегося в отдалении компьютера (подключенного через сеть), как если бы эти интерфейсы существовали на локальном компьютере.

Вы можете найти больше информации по использованию com0com и com2tcp в файле ReadMe.txt для com0com, ReadMe.txt для com2tcp, ReadMe.txt для hub4com и так далее на страничке проекта сайта SourceForge [1].

[**Virtual Serial Port Driver**]

Альтернативно можно для тех же целей использовать Virtual Serial Port Driver [6] компании Eltima Software. Это программное обеспечение позволяет создавать большое количество виртуальных COM-портов и дает Вам большие возможности для полной эмуляции последовательного порта. Virtual Serial Port Driver предоставляет гибкое создание порта, управление портами, удаление портов, что позволяет через виртуальные порты тестировать программы, поддерживать управляющие сигналы и высокие скорости передачи.



Основные возможности Virtual Serial Port Driver:

• Создание любого количество виртуальных COM-портов.  
• Полная эмуляция настроек реальных портов.  
• Управление всеми сигнальными линиями.  
• Немедленное создание портов позволяет работать с портами, даже до того, как PnP manager определил только что созданные порты.  
• Полная поддержка Windows 10.

[**VIRTUAL NULL MODEM**]

Назначение нуль-модемного кабеля - позволить двум RS-232 устройствам типа DTE (Data Terminal Equipment, под этим обычно подразумеваются модемы и любые устройства, которыми можно управлять через текстовую консоль терминала) обмениваться данными между собой без использования дополнительного оборудования (т.е. устройств типа DCE, расшифровывается как Data Circuit-terminating Equipment, под этим обычно подразумеваются компьютеры или другие устройства, к которым подключены устройства DTE).

Virtual Null Modem - специальная программа, которая эмулирует одну или несколько пар портов RS232, соединенных между собой нуль-модемным кабелем. Функционал абсолютно тот же, что и у com0com. Вы можете создать множество виртуальных последовательных портов, которые практически будут неотличимы от настоящих портов (например, COM10, COM11, COM127 и т.д.), соединенных попарно виртуальным нуль-модемным кабелем.

Для чего нужна утилита Virtual Null Modem? Допустим, что у вас есть две программы, которые вы хотите соединить между собой. Обычно вам необходимо два настоящих последовательных порта, соединенных нуль-модемным кабелем. Современные компьютеры обычно имеют в наличии только один порт, а ноутбуки не имеют и одного. При использовании программы Virtual Null Modem вы можете эмулировать нуль-модемное соединение, так что две программы на одном компьютере могут общаться между собой без дополнительного оборудования и железа.

