Последовательный интерфейс

- Универсальный внешний последовательный интерфейс — *COM-порт* (Communications Port — коммуникационный порт) Этот порт обеспечивает обмен по стандарту RS-232C.
- Асинхронный, (синхронный)дуплексный (полудуплексный)интерфейс.
- Скорость передачи может выбираться из ряда 110, 150, 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 115200 бит/с
- Максимальное расстояние между передатчиком и приемником -15м.
- Компьютер может иметь до четырех последовательных портов COM1-COM4 .
-

COM - порт

- *Порты* занимают в пространстве ввода-вывода по 8 смежных 8-битных регистров каждый и могут располагаться по стандартным *базовым адресам* 3F8h (COM1), 2F8h (COM2), 3E8h (COM3), 2E8h (COM4). Порты могут вырабатывать *аппаратные прерывания* IRQ4 (обычно используются для COM1 и COM3) и IRQ3 (для COM2 и COM4).

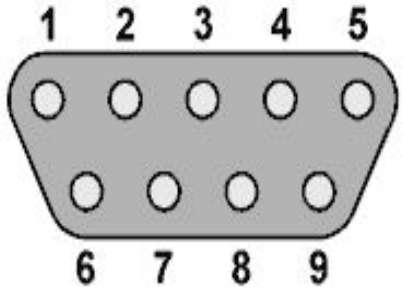
Интерфейс RS232c

Рис.2 Полная схема соединения по RS-232C

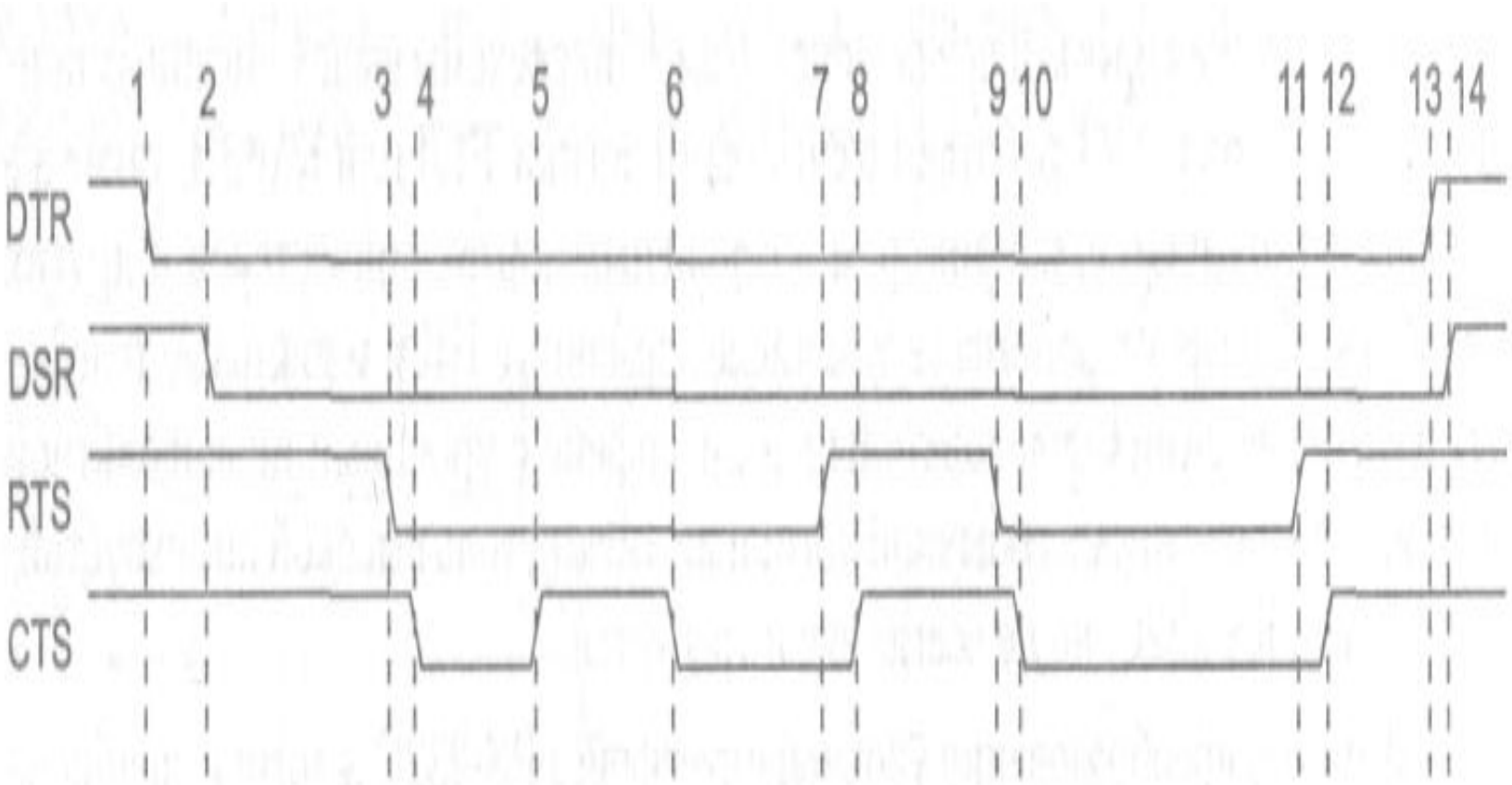
Рис. 3. Соединение по RS-232C нуль-модемным кабелем

Интерфейс RS-232C предназначен для подключения аппаратуры, передающей или принимающей данные *(АПД* — аппаратура передачи данных; *DTE — Data Terminal Equipment), к* оконечной аппаратуре каналов данных *(АКД; DCE—Data Communication Equipment например модем).*

Контакт	Сигнал	Направление	Описание
1	CD	Вход	Обнаружена несущая (линия активна)
2	RXD	Вход	Принимаемые данные
3	TXD	Выход	Передаваемые данные
4	DTR	Выход	Хост(Комп) 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(для модема если он получил вызов от удаленной системы)

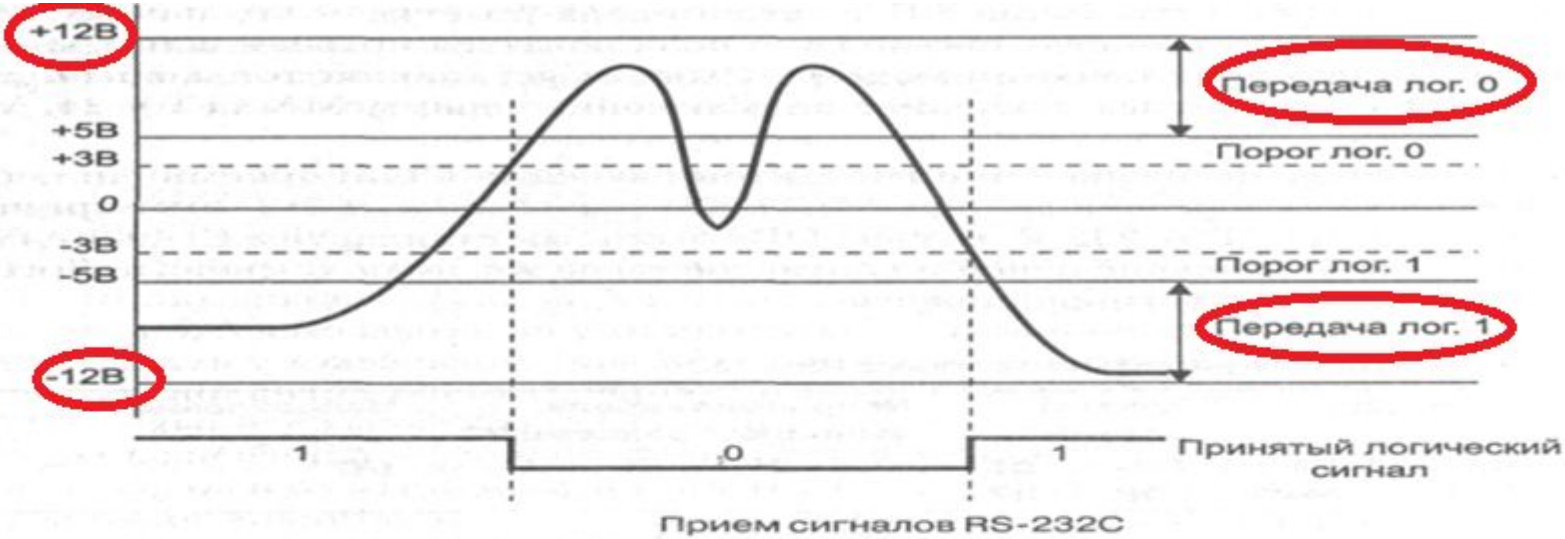


Последовательность управляющих сигналов интерфейса

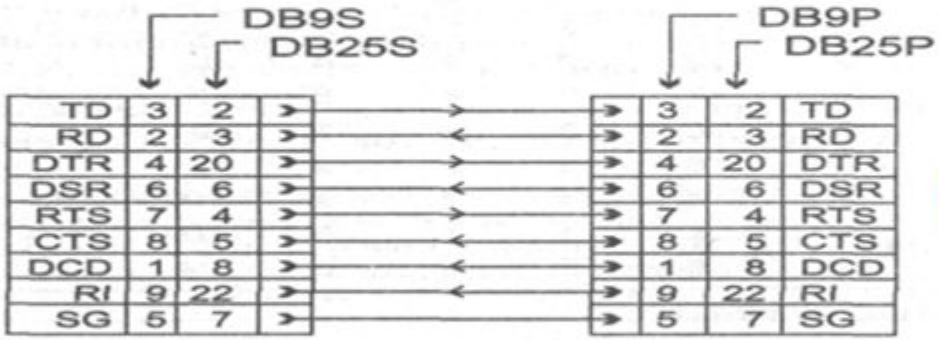


Последовательность управляющих сигналов интерфейса

1. Установкой DTR компьютер указывает на желание использовать модем.
2. Установкой DSR модем сигнализирует о своей готовности и установлении соединения.
3. Сигналом RTS компьютер запрашивает разрешение на передачу и заявляет о своей готовности принимать данные от модема.
4. Сигналом CTS модем уведомляет о своей готовности к приему данных от компьютера и передаче их в линию.
5. Снятием CTS модем сигнализирует о невозможности дальнейшего приема (например, буфер заполнен) — компьютер должен приостановить передачу данных.
6. Сигналом CTS модем разрешает компьютеру продолжить передачу (в буфере появилось место).
7. Снятие RTS может означать как заполнение буфера компьютера (модем должен приостановить передачу данных в компьютер), так и отсутствие данных для передачи в модем. Обычно в этом случае модем прекращает пересылку данных в компьютер.
8. Модем подтверждает снятие RTS сбросом CTS.
9. Компьютер повторно устанавливает RTS для возобновления передачи.
10. Модем подтверждает готовность к этим действиям.
11. Компьютер указывает на завершение обмена.
12. Модем отвечает подтверждением.
13. Компьютер снимает DTR, что обычно является сигналом на разрыв соединения («повесить трубку»).
14. Модем сбросом DSR сигнализирует о разрыве соединения.



Прием сигналов RS-232C



Кабель DTE - DTC

Кабели подключения модемов

Нуль-модемный кабель: *а* — минимальный, *б* — полный

Кабель DTE - DTE

Синхронизация передачи

Рис. 1. Формат асинхронной передачи.

В состав соединенных по RS232 устройств входят: тактовый генераторы , которые задают тактовую частоту приемопередатчика для данной скорости связи . Перед началом связи между двумя устройствами необходимо настроить их приемопередатчики на одинаковую скорость связи и формат кадра.

Приемник, поймав падающий фронт старт-бита, который передает передатчик отсчитывает несколько тактов и следующие три такта считывает (семплирует) порт RX. Это как раз середина старт-бита. Если большинство значений - "0", старт-бит считается состоявшимся, иначе приемник принимает его за шум и ждет следующего падающего фронта. После удачного определения старт-бита, приемник точно также семплирует серединки битов данных и по большинству семплов считает бит "0" или "1", записывая их в сдвиговый регистр. Стоп-биты тоже семплируются.

Появление USB постепенно приводит к вытеснению последовательного порта (RS-232). Так, например, последовательный порт уже давно отсутствует на всех современных ноутбуках. Тем не менее, многие приложения продолжают его использовать. Причин для этого много. Часто решающим фактором становится невозможность переработки программы, нежелание дополнительных трудозатрат или некоторая сложность работы с USB относительно привычного последовательного порта. В таких случаях бывает удобно использование преобразователя интерфейсов.

Основная задача преобразования интерфейса RS-232 в USB — получить со стороны программы обычный последовательный порт, позволяющий работать с интерфейсом "по старинке". Существует несколько путей решения этого вопроса.

Виртуальный COM – порт

COM – порт поверх USB

1. Вариант – приобретение специального кабеля переходника USB – COM с специальным драйвером. При установке драйвера в системе появляется виртуальный COM – порт, работающий поверх USB. Обращение к этому порту осуществляется с помощью функций доступа к COM – порту.

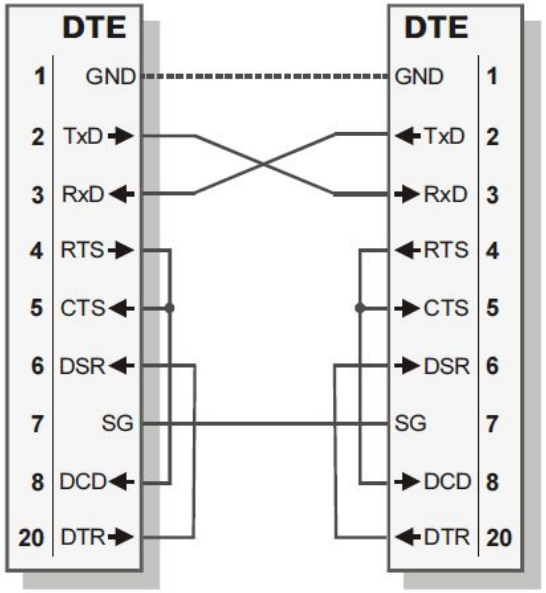
функций доступа к COM-порту.

2. Использование переходника и стандартного драйвера CDC (CommunicationDevice Class)

Работа с COM-портом

Два варианта управления обменом данными:

1. Программный метод не требует наличия дополнительных линий для синхрони

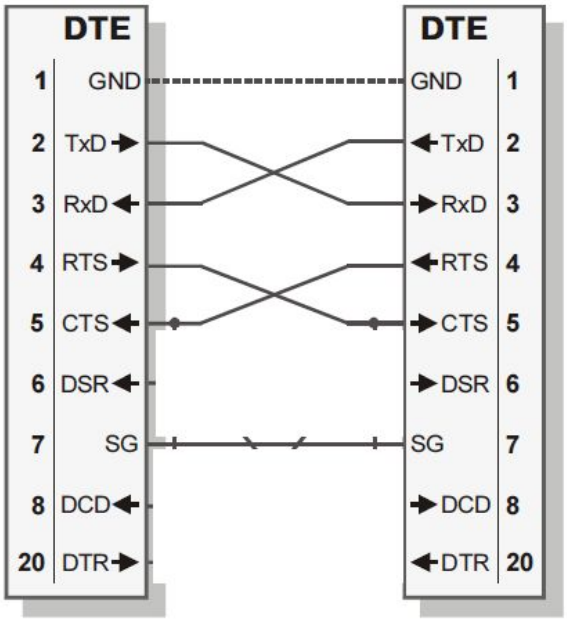


Управление с помощью посылки специальных пакетов(команд) старта и

Реализация программного метода управления (метод XON/XOFF) не требует наличия дополнительных сигнальных линий, поэтому обмен данными можно выполнять и в простейшей нуль-модемной конфигурации по двум линиям TxD и RxD. Для использования метода XON/XOFF протоколом предусмотрено два специальных байта, которые так и называются — байт XON и байт XOFF. Байт XON имеет десятичный код 17, а XOFF — 19. Байт XON используется для разрешения передачи данных, а XOFF — для остановки. На клавиатуре эти коды эмулируются комбинациями клавиш <Ctrl>+<Q> (XON) и <Ctrl>+<S> (XOFF). Применение метода XON/XOFF для управления потоком данных не вызывает затруднений. Для остановки передачи данных приемник посылает байт XOFF передатчику, а для возобновления передачи — байт XON

Аппаратный метод

- Управление передачей с помощью линий синхронизации
- **RTS, CTS**



Возможность управления отдельными линиями порта при работе с не стандартными устройствами (три сигнала на выход **DTR, RTS, Tx** и четыре на вход
DSR, CTR, DCD, RI)

Открытие порта

```
function CreateFile(
ipFileName: PChar;                  // имя файла (порта)
dwDesiredAccess,                    // способ доступа к файлу
dwShareMode: DWORD;                 // тип совместного доступа
lpSA : PSecurityAttributes;         // атрибуты защиты
dwCreationDisposition,              // параметры создания файла
dwFlagsAndAttributes: DWORD;        // атрибуты файла
hTemplateFile: Thandle              // дескриптор template-файла
) : THandle;
```

Закрытие порта

```
function CloseHandle(
hObject: THandle                    // дескриптор порта
): BOOL;
```

```
{Переменная для хранения дескриптора порта}
 var ComHandle   :   THandle;

{Открыть  порт}
ComHandle:= CreateFile ('\\.\COM1’,
GENERIC_READ  or  GENERIC_WRITE,
0,
nil,
OPEN_EXISTING,
FILE_ATTRIBUTE_NORMAL or FILE_FLAG_OVERLAPPED, 0 );
{Проверить результат}
if ComHandle = INVALID_HANDLE_VALUE then begin
{Ошибка открытия порта, функция GetLastError вернет код ошибки}
Exit; end;
{... порт открыт успешно ...}
{... использование порта через дескриптор ComHandle  ...}

{Закрытие порта}
 CloseHandle(ComHandle);
```

Чтение данных из порта

```
function ReadFile(
hFile: THandle;  // дескриптор, полученный от CreateFile
var Buffer;                                  // буфер для чтения
nNumberOfBytesToRead: DWORD;      // число байт для чтения
var lpNumberOfBytesRead: DWORD;// реально прочитанное число
                                                    //байт
ipOverlapped: Poverlapped         // параметры асинхронного чтения
): BOOL;
```

Запись данных в порт

```
function WriteFile(
hFile      : THandle;        // дескриптор, полученный от CreateFile
const Buffer;                // буфер данных
nNBTW      : DWORD;          // длина буфера
var lpNBW  : DWORD;          // реально отправленное число байт
lpOverlapped: POverlapped // параметры асинхронной записи ): BOOL;
```

Функции работы с нестандартными устройствами

Функция **EscapeCommFuntion** позволит управлять уровнем на выходных линиях **DTR** и **RTS**. Вызывается эта функция так: **EscapeCommFunction (Port, Command)**, где **Port** – дескриптор открытого файла, возвращенного функцией **CreateFile**,

.

Для контроля уровня на входных линиях используется функция **GetCommModemStatus (Port, Status)**, где **Port** – тот же самый дескриптор, а **Status** – переменная типа *длинное целое*, биты которой будут содержать информацию о различных параметрах COM-порта.

Все упомянутые функции Windows API **и константы описаны в файлах *windows.h* (для** C++**) и модуле *windows* (для** Delphi**), которые надо подключить к проекту.**