Федеральное государственное автономное образовательное
учреждение высшего образования
"СЕВАСТОПОЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ"

Институт радиоэлектроники и информационной безопасности

И Н Т Е Р Ф Е Й С Ы ПЕРСОНАЛЬНОГО КОМПЬЮТЕРА. Лабораторный практикум

Учебно-методическое пособие
для студентов очной и заочной форм обучения
направления подготовки
11.03.04 «Электроника и наноэлектроника»

Севастополь
СевГУ
2018

УДК 621.382

Р е ц е н з е н т:
Д.Г. Мурзин - канд. техн. наук, доцент кафедры «Электронная техника»

Составители: С.В. Деордица, Д.В. Начаров

Интерфейсы персонального компьютера. Лабораторный практикум: учеб.-метод. пособие для студентов очной и заочной форм обучения направления подготовки 11.03.04 «Электроника и наноэлектроника» / СевГУ; сост. С.В. Деордица, Д.В. Начаров. — Севастополь: СевГУ, 2018. — 40 с.

Целью учебно-методического пособия является изучение способов подключения внешнего устройства к персональному компьютеру, освоение навыков по проектированию устройств сопряжения персонального компьютера с внешним устройством, организации обмена данными между внешним устройством и персональным компьютером.

УДК 621.382

Методические указания рассмотрены и утверждены на заседании кафедры «Электронная техника», протокол № 5 от 19 апреля 2018 г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Дисциплина «Интерфейсы персонального компьютера» входит в раздел «Б1.В.ДВ.4.1.» вариативной части ФГОС по направлению подготовки ВПО 11.03.04 — Электроника и наноэлектроника. Дисциплина является вариативной в обучении бакалавра, ей предшествуют курсы: «Схемотехника», «Цифровая электроника» предполагающие проведение лекционных и семинарских занятий с обязательными итоговыми контролями.

Дисциплина является предшествующей написанию выпускной квалификационной работы бакалавра.

Целью методических указаний является оказание помощи студентам в подготовке и выполнении лабораторных работ по дисциплине «Интерфейсы персонального компьютера», а также в получении ими практических навыков по проектированию устройств сопряжения и организации обмена данными между персональным компьютером и внешним устройством.

В данную методическую разработку включены пять лабораторных работ.

Лабораторные работы 1-2 посвящены изучению протоколов передачи данных Centronics и Nibble Mode и их реализации с помощью LPT-порта персонального компьютера.

Лабораторная работа 3 посвящена изучению протокола передачи данных RS-232 и его реализаций в виде COM-порта и универсального асинхронного приемопередатчика UART.

Лабораторные работы 4-5 посвящены изучению способов обмена данными по стандарту RS-232 с использованием приемопередатчика UART и физического интерфейса USB. В работе рассматриваются схемотехнические решения по преобразованию уровней напряжения с помощью дискретных электронных компонентов и интегральных схем, а также рассмотрены способы преобразования интерфейсов USB - UART.

Отчеты по лабораторным работам должны иметь титульный лист и содержать следующие разделы:

— цель работы;

— скриншоты графических интерфейсов и листинги разработанных программ передачи данных по интерфейсам;

— результаты выполнения индивидуальных заданий;

— выводы.

Лабораторная работа № 1.
LPT-ПОРТ

Цель работы — изучить основные свойства LPT-порта; научиться определять базовые адреса портов; изучить метод прямого доступа к LPT-портам.

1.1. Теоретические сведения

1.1.1. Описание LPT порта

Порт параллельного интерфейса был введен в персональный компьютер (ПК) для подключения принтера — отсюда и пошло его название LPT-порт (Line PrinTer — построчный принтер). Традиционный, он же стандартный, LPT порт (так называемый SPP порт) ориентирован на вывод данных, хотя с некоторыми ограничениями позволяет и вводить данные. Существуют различные модификации LPT-порта — двунаправленный, EPP, ECP и другие, расширяющие его функциональные возможности, повышающие производительность и снижающие нагрузку на процессор. Поначалу они являлись фирменными решениями отдельных производителей, позднее был принят стандарт IEEE 1284.

С внешней стороны порт имеет 8-битную шину данных, 5-битную шину сигналов состояния и 4-битную шину управляющих сигналов, выведенные на разъем-розетку DB-25S. В LPT порту используются логические уровни ТТЛ, что ограничивает допустимую длину кабеля из-за невысокой помехозащищенности ТТЛ-интерфейса. Гальваническая развязка отсутствует — схемная земля подключаемого устройства соединяется со схемной землей компьютера. Из-за этого порт является уязвимым местом компьютера, страдающим при нарушении правил подключения и заземления устройств. Поскольку порт обычно располагается на системной плате, в случае его «выжигания» зачастую выходит из строя и его ближайшее окружение, вплоть до выгорания всей системной платы.

С программной стороны LPT-порт представляет собой набор регистров, расположенных в пространстве ввода-вывода. Регистры порта адресуются относительно базового адреса порта, стандартными значениями которого являются 3BCh, 378h и 278h. Порт может использовать линию запроса аппаратного прерывания, обычно IRQ7 или IRQ5. В расширенных режимах может использоваться и канал DMA.

Порт имеет поддержку на уровне BIOS — поиск установленных портов во время теста POST и сервисы печати Int 17h обеспечивают вывод символа (по опросу готовности, не используя аппаратных прерываний), инициализацию интерфейса и принтера, а также опрос состояния принтера.

К LPT портам подключают принтеры, плоттеры, сканеры, коммуникационные устройства и устройства хранения данных, а также электронные ключи, программаторы и прочие устройства.

1.1.2. Системная поддержка LPT-порта

Системная поддержка LPT-порта включает поиск установленных портов и сервисы печати. В процессе начального тестирования POST BIOS проверяет наличие параллельных портов по адресам 3BCh, 378h и 278h и помещает базовые адреса обнаруженных портов в ячейки BIOS Data Area 0:0408h, 0:040Ah, 0:040Ch, 0:040Eh. Эти ячейки хранят адреса портов LPT1…LPT4, нулевое значение адреса является признаком отсутствия порта с данным номером. В ячейки 0:0478h, 0:0479h, 0:047Ah, 0:047Bh заносятся константы, задающие тайм-аут для этих портов.

Для обращения к памяти Bios Data Area используются специальные команды; так в среде Турбо Паскаль предусмотрена команда MEMW [Seg : Ofs], где сегмент памяти (Seg) и смещение (Ofs) формируют адрес памяти (PhA), записываемый как PhA = Seg * 16 +Ofs.

Пример 1.1. Получение базового адреса порта по адресу Bios Data Area 0408h.

```
Base := MEMW [$40 : $08];
```

1.1.3. Традиционный LPT-порт

Традиционный, он же стандартный, LPT-порт называется стандартным параллельным портом (Standard Parallel Port — SPP), или SPP-портом, и является однонаправленным портом, через который программно реализуется протокол обмена Centronics. Название и назначение сигналов разъема порта (табл. 1.1) соответствуют интерфейсу Centronics.

Таблица 1.1 — Разъем стандартного LPT порта

Контакт DB-25S	Направление[1]	Бит регистра[2]	Название сигнала
1	O	CR.0	Strobe
2	O(I)	DR.0	Data 0
3	O(I)	DR.1	Data 1
4	O(I)	DR.2	Data 2
5	O(I)	DR.3	Data 3
6	O(I)	DR.4	Data 4
7	O(I)	DR.5	Data 5
8	O(I)	DR.6	Data 6
9	O(I)	DR.7	Data 7
10	I[3]	SR.6	Ack#
11	I	SR.7	Busy
12	I	SR.5	PaperEnd(PE)
13	I	SR.4	Select
14	O/I	CR.1\	Auto LF# (AutoFeedW)
15	I	SR.3	Error

Контакт DB-25S	Направление[1]	Бит регистра[2]	Название сигнала
16	O/I	CR.2	lnit#
17	O/I	CR.3\	Select ln#
18-25	—	—	GND

Примечания к табл.1.1:

1. I/O (Input/Output — вход/выход) задает направление передачи сигнала порта. O/I обозначает выходные линии, состояние которых считывается при чтении из портов вывода; O(I) — выходные линии, состояние которых может быть считано только при особых условиях (см. ниже).

2. Символом «\» отмечены инвертированные сигналы (1 в регистре соответствует низкому уровню линии).

3. Вход Ack# соединен резистором (10 кОм) с питанием +5 В.

1.1.4. Регистры LPT-порта

Адаптер SPP-порта содержит три 8-битных регистра, расположенных по соседним адресам в пространстве ввода-вывода, начиная с базового адреса порта BASE (3BCh, 378h или 278h).

Data Register (DR) — регистр данных, адрес = BASE. Данные, записанные в этот регистр, выводятся на выходные линии Data [7:0]. Данные, считанные из этого регистра, в зависимости от схемотехники адаптера соответствуют либо ранее записанным данным, либо сигналам на тех же линиях, что не всегда одно и тоже.

Status Register (SR) — регистр состояния (только чтение), адрес = BASE+1. Регистр отображает 5-битный порт ввода сигналов состояния принтера (биты SR.4…SR.7) и флаг прерывания. Бит SR.7 инвертируется — низкому уровню сигнала соответствует единичное значению бита в регистре, и наоборот. В табл. 1.2 приведено назначение бит регистра состояния.

Запрос аппаратного прерывания (обычно IRQ7 или IRQ5) вырабатывается по отрицательному перепаду сигнала на выводе 10 разъема интерфейса (Ack#) при установке CR.4 = 1. Во избежание ложных прерываний контакт 10 соединен резистором с шиной +5 В. Прерывание вырабатывается, когда принтер подтверждает прием предыдущего байта. Как уже было сказано, BIOS это прерывание не использует и не обслуживает.

Control Register (CR) — регистр управления, адрес = BASE+2, допускает запись и чтение. Регистр связан с 4-битным портом вывода управляющих сигналов (биты 0-3) для которых возможно и чтение; выходной буфер обычно имеет тип «открытый коллектор». Это позволяет корректно использовать линии данного регистра как входные при программировании их в высокий уровень. Биты 0,1,3 инвертируются. В табл. 1.3 приведено назначение бит регистра состояния.

Таблица 1.2 — Назначение битов регистра состояния LPT порта

Бит	Название	Назначение
SR.7	Busy	Инверсное отображение состояния линии Busy (11): при низком уровне на линии устанавливается единичное значения бита — разрешение на вывод очередного байта
SR.6	Ack (Acknowledge)	Отображение состояния линии Ack# (10)
SR.5	PE (Paper End)	Отображение состояния линии Paper End (12). Единичное значение соответствует высокому уровню линии — сигналу о конце бумаги в принтере
SR.4	Select	Отображение состояния линии Select (13). Единичное значение соответствует высокому уровню линии — сигналу о включении принтера
SR.3	Error	Отображение состояния линии Error (15). Нулевое значение соответствует низкому уровню линии — сигналу о любой ошибке принтера
SR.2	PIRQ	Флаг прерывания по сигналу Ack# (только для порта PS/2). Бит обнуляется, если сигнал Ack# вызвал аппаратное прерывание. Единичное значение устанавливается по аппаратному сбросу и после чтения регистра состояния
SR[1:0]	—	Зарезервированы

Таблица 1.3 — Назначение бит регистра управления LPT порта

Бит	Название	Назначение
CR [7:6]	—	Зарезервированы
CR.5	Direction	Бит управления направлением передачи (только для портов PS/2). Запись единицы переводит порт данных в режим ввода. При чтении состояние бита не определено
CR. 4	AcklNTEN (Ack Interrupt Enable)	Единичное значение разрешает прерывание по спаду сигнала на линии Ack# — сигнал запроса следующего байта
CR.3	Select In	Единичное значение бита соответствует низкому уровню на выходе Select ln# (17) — сигналу, разрешающему работу принтера по интерфейсу Centronics
CR.2	Init	Нулевое значение бита соответствует низкому уровню на выходе Init# (16) — сигнал аппаратного сброса принтера
CR. 1	Auto LF	Единичное значение бита соответствует низкому уровню на выходе Auto LF# (14) — сигналу на автоматический перевод строки (LF — Line Feed) по приему байта возврата каретки (CR). Иногда сигнал и бит называют AutoFD или AutoFDXT
CR.0	Strobe	Единичное значение бита соответствует низкому уровню на выходе Strobe (1) — сигналу стробирования выходных данных

Пример 1.2. Запись данных в порт (прямое обращение к регистру данных в среде Турбо Паскаль):

```
PORT [Base] := Data; {где Base — базовый адрес, data — данные}
```

Пример 1.3. Чтение данных из порта (прямое обращение к регистру состояния в среде Турбо Паскаль):

SR := PORT[Base + 1]; {где SR — переменная типа байт}

1.2. Практическая часть

1.2.1. LPT-порт в операционной системе Windows

Операционные системы NT-платформы (2000, NT, XP и выше) не позволяют напрямую работать с LPT-портами. Поэтому в качестве решения можно либо пользоваться API функциями Windows, что не всегда неудобно, либо пользоваться драйвером, который обеспечит прямой доступ к регистрам порта. В качестве примера рассмотрим использование готового драйвера «inpout32.dll», который является пассивным, т.е. не требует регистрации (установки) в реестре операционной системы. Это обычный файл библиотеки, который должен находиться в одной директории с проектом.

Библиотека inpout32.dll работает следующим образом: сначала проводится проверка установленной ОС; если установлена 9x, то при обращении к библиотеке используются функции Out32(адрес порта, байт данных) — для записи в порт и Inp32(адрес порта) — для чтения из порта. Если же установлена ОС NT, то при обращении к ней драйвер конвертирует запросы к стандартному драйверу ОС, через который и идет обмен информацией с портом. Таким образом, библиотека содержит две подпрограммы:

— function Inp32(PortAdr: word): byte;

— procedure Out32(PortAdr: word; Data: byte);

где PortAdr — адрес порта, Data — данные для записи в порт.

Драйвер inpout32.dll подключается к существующему проекту следующим образом[1]:

```
function Inp32(PortAdr: word): byte; stdcall; external 'inpout32.dll';

procedure Out32(PortAdr: word; Data: byte); stdcall; external 'inpout32.dll';
```

Пример 1.4. Процедура записи данных в регистр данных (Data Register) по нажатию кнопки Write (Button1) из поля Edit1. Базовый адрес порта выбирается из выпадающего меню ComboBox1.

```
procedure TForm1.Button1Click(Sender: TObject);
  var
    Base: Word;  {Base – базовый адрес}
    Data: Byte;  {Data – данные для записи в порт}
```

[1] Здесь и далее рассматриваются примеры в среде Delphi

```
begin
   Data:=StrToInt(Edit1.Text);
   Case ComboBox1.ItemIndex of {Выбор адреса}
     0: Base:=$378;
     1: Base:=$278;
     2: Base:=$3BC;
   end;
   Out32(Base,Data);     {Запись данных в порт}
   end;
```

Пример 1.5. Процедура чтения данных из регистра состояния (Status Register) по нажатию кнопки Write (Button2) в поле Edit2. Базовый адрес порта выбирается из выпадающего меню ComboBox2.

```
procedure TForm1.Button4Click(Sender: TObject);
var Base : Word;     {Base — базовый адрес}
    Data : Byte;     {Data — данные для записи в порт}
  begin
 Case ComboBox1.ItemIndex of  {Выбор адреса}
    0 : Base:=$378 + 1;
     1 : Base:=$278 + 1;
     2 : Base:=$3BC + 1;
 end;
   Data:=Inp32(Base);      {Чтение регистра состояния}
   Edit2.Text:=IntToStr(Data);
   end;
```

1.2.2. Порядок выполнения работы

1) В среде Турбо Паскаль написать программу, которая определяет и выводит на экран базовый адрес LPT порта с номером, который вводит пользователь (LPT1…LPT3). Если порт не найден, то выдать соответствующее сообщение.

2) Реализовать программу, имеющую внешний вид программы рис.1.1 и позволяющую:

а) определять адрес порта по значению номера порта из выпадающего меню ComboBox1.

б) выводить данные в регистр данных LPT порта из поля Edit1 по нажатию кнопки Write (Button1).

в) читать данные из регистра состояния LPT порта в поле Edit2 по нажатию кнопки Read (Button2).

3) Используя кабель LPT удлинителя, подключить к LPT-порту ПК лабораторный макет. Используя программу п.2 осуществить чтение регистра состояния (переключатели макета) и запись в регистр данных (светодиоды макета).

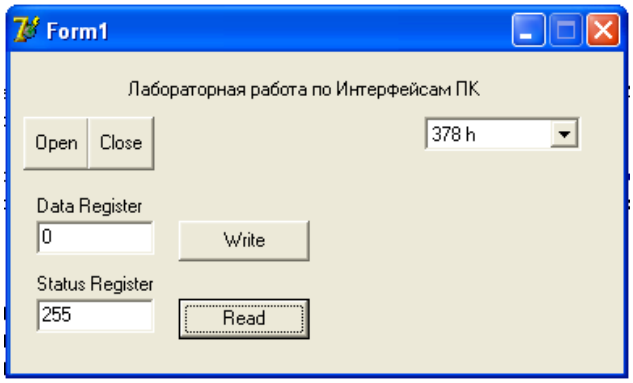

Рисунок 1.1 — Внешний вид программы работы с LPT портом

4) Реализовать программу п.2, но вывод и чтение данных осуществить согласно индивидуальным заданиям. Подключить к LPT-порту ПК лабораторный макет и проверить правильность выполнения работы.

5) Оформить отчет по лабораторной работе.

1.2.3. Индивидуальные задания

Варианты индивидуального задания выбираются по последней цифре номера зачетной книжки студента.

1) Прочитать состояние трех переключателей, вывести его на экран в бинарном коде (напр., 101) и продублировать свечением светодиодов, т.е. задействовано три светодиода (если переключатель нажат — соответствующий светодиод светится).

2) Прочитать состояние трех переключателей, вывести на экран код нажатия, переведенный из бинарного кода в десятичный вид (напр., 5, что соответствует нажатию 101), и продублировать свечением соответствующего светодиода (для данного примера 5-ый светодиод).

3) Прочитать состояние трех переключателей, осуществить управление светодиодами следующим образом: нажат переключатель 1 — горят четные светодиоды, нажат 2 — горят нечетные, нажат 3 — все потушены, приоритетность переключателей с 1-го по 3-ий. Вывести на экран номер нажатого переключателя.

4) Прочитать состояние трех переключателей, осуществить управление светодиодами следующим образом: нажат переключатель 1 — горят все светодиоды, нажат 2 — не горят, нажат 3 — светятся четные, приоритетность переключателей с 1-го по 3-ий. Вывести на экран номер нажатого переключателя.

5) Прочитать состояние трех переключателей, вывести на экран десятичный код нажатия. Осуществить «бегущий огонек» светодиодов, скорость смены состояния светится / не светится зависит от полученного десятичного кода состояний переключателей. Чем выше значение кода, тем выше скорость свечения. Скорость смены состояний подобрать самостоятельно.

6) Аналогично задания 1, но выводить код на экран и зажигать светодиоды в инверсии

7) Аналогично задания 2, но выводить код на экран и зажигать светодиоды в инверсии

8) Аналогично задания 3, но выводить код на экран в инверсии, приоритетность в инверсии

9) Аналогично задания 4, но выводить код на экран в инверсии, приоритетность в инверсии

10) Аналогично задания 5, но с увеличением кода — скорость падает.

1.3. Контрольные вопросы

1) Назовите основные отличительные особенности LPT-порта.

2) Поясните назначение регистров LPT-порта.

3) Приведите примеры применения LPT-порта.

4) Что такое базовый адрес порта?

5) Где располагаются базовые адреса и как определить наличие LPT-портов?

6) Какие существуют возможности доступа к портам в среде Windows NT?

7) Приведите пример записи данных в порт, используя драйвер inpout32.dll.

8) Приведите пример чтения данных из регистра состояния порта, используя драйвер inpout32.dll.

Лабораторная работа № 2.
РАБОТА С ПРОТОКОЛАМИ ПАРАЛЛЕЛЬНОГО LPT ПОРТА

Цель работы — изучить протоколы Centronics и NibbleMode, а также освоить работу с LPT-портом, используя API функции Windows.

2.1. Теоретические сведения

2.1.1. Протокол обмена Centronics

Основным назначением интерфейса Centronics является подключение к ПК принтеров различных типов. Поэтому распределение контактов разъема, назначение сигналов, программные средства управления интерфейсом ориентированы именно на это использование. В то же время с помощью данного интерфейса можно подключать к компьютеру и другие внешние устройства, имеющие разъем Centronics, а также специально разработанные пользовательские схемы.

Основным достоинством использования Centronics для подключения УС по сравнению с ISA является значительно меньший риск вывести компьютер из строя. Главный недостаток этого подхода — значительно меньшая скорость обмена. На рис. 2.1 представлена диаграмма передачи данных по протоколу Centronics.

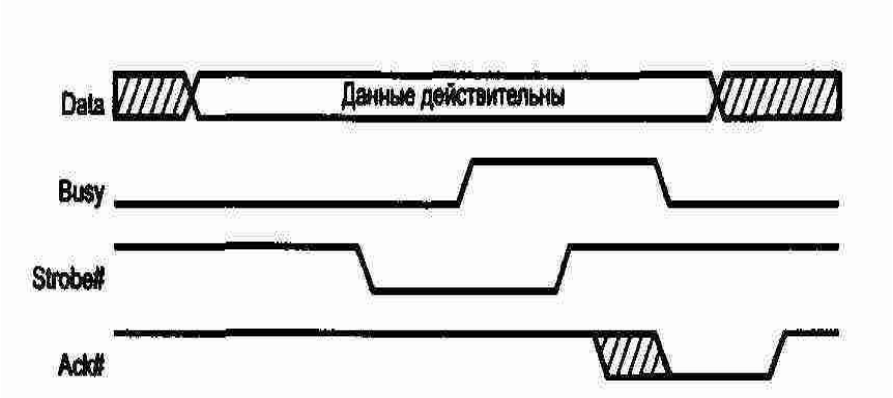

Рисунок 2.1 — Диаграмма работы по протоколу Centronics

Перечислим шаги процедуры вывода байта по интерфейсу Centronics.

1) Выставить бит “Select In” в 1 (CR.3 = 1). Единичное значение бита соответствует низкому уровню на выходе Select ln# (17) — сигналу, разрешающему работу принтера по интерфейсу Centronics.

2) Убедиться, что сигналы линии Busy = 0 и линии Ack = 1 (SR.7 = 1, SR.6 = 1). Т.е. периферийное устройство готово к передаче данных. Этот шаг зацикливается до получения готовности или до срабатывания программного тайм-аута.

3) Вывод данных в регистр данных.

4) Формирование строба (CR.0 = 1). Strobe - единичное значение бита соответствует низкому уровню на выходе Strobe (сигналу стробирования выходных данных).

5) Выдержать паузу некоторое время (>500 мкс), т.к. периферийное устройство должно успеть считать данные. Снять сигнал строб (CR.0 = 0).

6) При получении строба периферийное устройство формирует сигнал на линии Busy = 1 (SR.7 = 0), а после обработки данных выставляет сигнал на линии Ack = 0 (SR.6 = 0), снимает Busy = 0 и выставляет Ack = 1.

7) Пункты 1-6 повторяются.

Видно, что для вывода одного байта требуется 4-5 операций ввода-вывода с регистрами порта (в лучшем случае, когда готовность обнаружена по первому чтению регистра состояния). Отсюда вытекает главный недостаток вывода через стандартный порт — невысокая скорость обмена при значительной загрузке процессора. Порт удается разогнать до скоростей 150 Кбайт/с при полной загрузке процессора, что недостаточно для печати на лазерном принтере. Другой недостаток функциональный — сложность использования в качестве порта ввода.

2.1.2. Протокол полубайтного обмена Nibble Mode

Стандартный порт асимметричен — при наличии 12 линий (и бит), нормально работающих на вывод, на ввод работает только 5 линий состояния. Если необходима симметричная двунаправленная связь, на всех стандартных портах работоспособен режим полубайтного обмена — Nibble Mode. В этом режиме, называемом также Hewlett Packard Bi-tronics, одновременно принимаются 4 бита данных, пятая линия используется для квитирования. Таким образом, каждый байт передается за два цикла, а каждый цикл требует по крайней мере 5 операций ввода-вывода.

Сигналы порта приведены в таблице 2.1, а временные диаграммы на рис. 2.2.

Таблица 2.1 — Сигналы LPT-порта в полубайтном режиме ввода

Контакт	Сигнал SPP	I/O	Бит	Описание
14	AutoFeed#	O	CR.1	HostBusy — сигнал квитирования. Низкий уровень означает готовность к приему тетрады, высокий подтверждает прием тетрады
17	Selectln*	O	CR.3	Высокий уровень указывает на обмен в режиме IEEE 1284 (в режиме SPP уровень низкий)
10	Ack#	I	SR.6	PtrClk. Низкий уровень означает готовность тетрады, высокий — ответ на сигнал HostBusy
11	Busy	I	SR.7	Прием бита данных 3, затем бита 7
12	PE	I	SR.5	Прием бита данных 2, затем бита 6
13	Select	I	SR.4	Прием бита данных 1, затем бита 5
15	Error#	I	SR.3	Прием бита данных 0, затем бита 4

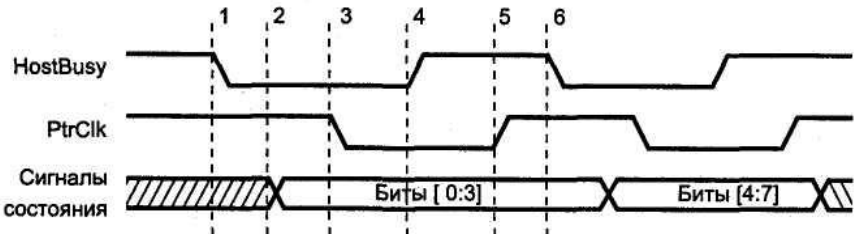

Рисунок 2.2 — Прием данных в полубайтном режиме

Прием байта данных от периферийного устройства (ПУ) к Хосту (рис 2.2) в полубайтном режиме состоит из следующих фаз.

1) Хост сигнализирует о готовности приема данных установкой низкого уровня на линии HostBusy.

2) ПУ в ответ помещает тетраду на входные линии состояния.

3) ПУ сигнализирует о готовности тетрады установкой низкого уровня на линии PtrClk.

4) Хост устанавливает высокий уровень на линии HostBusy, указывая на занятость приемом и обработкой тетрады.

5) ПУ отвечает установкой высокого уровня на линии PtrClk.

6) Шаги 1-5 повторяются для второй тетрады.

Полубайтный режим сильно нагружает процессор, и поднять скорость обмена выше 50 Кбайт/с не удается. Безусловное его преимущество в том, что он работает на всех портах. Его применяют в тех случаях, когда поток данных невелик (например, для связи с принтерами). Однако полубайтный режим редко применяется для связи с адаптерами локальных сетей, внешними дисковыми накопителями и CD-ROM.

2.2. Практическая часть

2.2.1. Реализации протоколов обмена через LPT-порт

Для реализации протоколов обмена через LPT-порт можно воспользоваться стандартными API функциями Windows. Работа с портом представляет собой следующую последовательность действий: открыть устройство; настроить нужный режим; читать из устройства и писать в устройство; закрыть устройство. Рассмотрим указанные действия.

1) Открытие устройства. Функция CreateFile формирует запрос, открывающий устройство.

Пример 2.1. Открытие порта LPT1 для операций асинхронного чтения и записи.

```
hLPT := CreateFile('LPT1', generic_read or generic_write, 0, nil, OPEN_EXISTING, FILE_FLAG_OVERLAPPED, 0);
```

Пример 2.2. Открытие порта LPT1 для операций синхронного чтения и записи.

```
hLPT := CreateFile('LPT1', GENERIC_READ or GENERIC_WRITE, 0, nil, OPEN_EXISTING, 0, 0);
```

где hLPT — дескриптор устройства, который затем используется при обращении к его драйверу (hLPT: THandle;).

2.2.2. Порядок выполнения работы

1) Реализовать программу, позволяющую:

а) определять адрес LPT порта по значению номера порта из выпадающего меню ComboBox1.

б) настроить LPT порт на режим работы NibbleMode.

в) передавать данные по протоколу NibbleMode порта из поля Edit1 по нажатию кнопки Write (Button1).

г) принимать данные по протоколу NibbleMode в поле Edit2 по нажатию кнопки Read (Button2).

2) Используя кабель LPT-NibbleMode, соединить два LPT порта ПК1 и ПК2. Используя программу п.1 осуществить передачу и прием данных по протоколу NIbbleMode.

3) Оформить отчет по лабораторной работе.

2.3. Контрольные вопросы

1) Поясните диаграмму работы по протоколу Centronix.

2) Поясните диаграмму работы по протоколу NibbleMode.

3) Поясните процедуру вывода байта данных по протоколу Centronix.

4) Поясните процедуру принятия и передачи байта данных по протоколу NibbleMode.

Лабораторная работа № 3.
РАБОТА С ПОСЛЕДОВАТЕЛЬНЫМ ПРОТОКОЛОМ ОБМЕНА RS-232

Цель работы — изучить протокол RS-232, а также освоить работу с COM-портом, используя API функции Windows.

3.1. Теоретические сведения

3.1.1. Последовательный интерфейс RS-232

Последовательный интерфейс RS-232 для передачи данных использует одну сигнальную линию, по которой информационные биты передаются друг за другом последовательно. Последовательная передача позволяет сократить количество сигнальных линий и увеличить дальность связи. Характерной особенностью является применение не ТТЛ сигналов. В ряде последовательных интерфейсов применяется гальваническая развязка внешних (обычно входных) сигналов от схемной земли устройства, что позволяет соединять устройства, находящиеся под разными потенциалами.

Последовательный интерфейс COM-порт (Communication Port — коммуникационный порт) появился в первых моделях IBM PC. Он был реализован на микросхеме асинхронного приемопередатчика Intel 8250. Порт имел поддержку BIOS (Int 14), однако широко применялось (и применяется) взаимодействие с портом на уровне регистров.

3.1.2. Системная поддержка COM-порта

При включении компьютера в процессе начального тестирования POST BIOS проверяет наличие последовательных портов (регистров UART 8250 или совместимых) по стандартным адресам 3F8h, 2F8h, 3E8h, 2E8h и помещает базовые адреса обнаруженных портов в ячейки BIOS DATA AREA 0:0400h, 0:0402h, 0:0404h, 0:0406h. Эти ячейки хранят адреса портов с логическими именами COM1…COM4, нулевое значение адреса является признаком отсутствия порта с данным номером.

В ячейки 0:047Ch, 0:047Dh, 0:047Eh, 0:047Fh заносятся константы, задающие выдержку тайм-аута для этих портов. Обнаруженные порты инициализируются обычно на скорость обмена 2400 бит/с, 7 бит данных с контролем на четность, 1 стоп-бит. Управляющие сигналы интерфейса DTR и RTS переводятся в исходное состояние.

3.1.3. Способы последовательной передачи

Последовательная передача данных может осуществляться в асинхронном или синхронном режимах. При асинхронной передаче каждому байту предшествует старт-бит, сигнализирующий приемнику о начале посылки, за которым следуют биты данных и, возможно, бит паритета (четности). Завершает посылку стоп-бит, гарантирующий паузу между посылками (рис. 3.1).

Рисунок 3.1 — Формат асинхронной передачи

Старт-бит следующего байта посылается в любой момент после стоп-бита, то есть между передачами возможны паузы произвольной длительности. Старт-бит, имеющий всегда строго определенное значение (логический 0), обеспечивает простой механизм синхронизации приемника по сигналу от передатчика. Подразумевается, что приемник и передатчик работают на одной скорости обмена. Внутренний генератор синхронизации приемника использует счетчик-делитель опорной частоты, обнуляемый в момент приема начала старт-бита. Этот счетчик генерирует внутренние стробы, по которым приемник фиксирует последующие принимаемые биты. В идеале стробы располагаются в середине битовых интервалов, что позволяет принимать данные и при незначительном рассогласовании скоростей приемника и передатчика. Очевидно, что при передаче 8 бит данных, одного контрольного и одного стоп-бита предельно допустимое рассогласование скоростей, при котором данные будут распознаны верно, не может превышать 5%. С учетом фазовых искажений и дискретности работы внутреннего счетчика синхронизации реально допустимо меньшее отклонение частот. Чем меньше коэффициент деления опорной частоты внутреннего генератора (чем выше частота передачи), тем больше погрешность привязки стробов к середине битового интервала, и требования к согласованности частот становятся более строгими. Чем выше частота передачи, тем больше влияние искажений фронтов на фазу принимаемого сигнала. Взаимодействие этих факторов приводит к повышению требований к согласованности частот приемника и передатчика с ростом частоты обмена.

Для асинхронного режима принят ряд стандартных скоростей обмена: 50, 75, 110, 150, 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600 и 115200 бит/с. Иногда вместо единицы измерения "бит/с" используют "бод" (baud), но при рассмотрении двоичных передаваемых сигналов это некорректно. В бодах принято измерять частоту изменения состояния линии, а при недвоичном способе кодирования (широко применяемом в современных модемах) в канале связи скорости передачи бит (бит/с) и изменения сигнала (бод) могут отличаться в несколько раз. Количество бит данных может составлять 5, 6, 7 или 8 (5- и 6-битные форматы распространены незначительно). Количество стоп-бит может быть 1, 1,5 или 2 ("полтора бита" означает только длительность стопового интервала).

Асинхронный обмен в ПК реализуется с помощью COM-порта с использованием протокола RS-232.

Синхронный режим передачи предполагает постоянную активность канала связи. Посылка начинается с синхробайта, за которым сразу же следует поток информационных бит. Если у передатчика нет данных для передачи, он заполняет паузу непрерывной посылкой байтов синхронизации. Очевидно, что при передаче больших массивов данных накладные расходы на синхронизацию в данном режиме будут ниже, чем в асинхронном. Однако в синхронном режиме необходима внешняя синхронизация приемника с передатчиком, поскольку даже малое отклонение частот приведет к искажению принимаемых данных. Внешняя синхронизация возможна либо с помощью отдельной линии для передачи сигнала синхронизации, либо с использованием само-синхронизирующего кодирования данных, при котором на стороне приемника из принятого сигнала могут быть выделены импульсы синхронизации. В любом случае, синхронный режим требует дорогих линий связи или оконечного оборудования.

На физическом уровне последовательный интерфейс имеет различные реализации, различающиеся способом передачи электрических сигналов. Существует ряд родственных международных стандартов: RS-232C, RS-423A, RS-422A и RS-485.

3.1.4. Назначения контактов и порядок обмена по интерфейсу RS-232

Основными преимуществами использования RS-232C по сравнению с Centronics являются возможность передачи на значительно большие расстояния и гораздо более простой соединительный кабель.

Данные в RS-232C передаются в последовательном коде побайтно. Каждый байт обрамляется стартовым и стоповыми битами. Данные могут передаваться как в одну, так и в другую сторону (дуплексный режим).

Компьютер имеет 25-контактный (DB25P) или 9-контактный (DB9P) разъем для подключения RS-232C. Функциональное расположение выводов приведено в табл. 3.1, а назначение контактов разъема в табл. 3.2.

Таблица 3.1 — Расположение контактов разъема интерфейса RS-232C

Цепь	Контакт (в разъеме DB25P)	Контакт (в разъеме DB9P)	I/O
FG	1	.	—
TxD\	2	3	O
RxD\	3	2	I
RTS	4	7	O
CTS	5	8	I
DSR	6	6	I
SG	7	5	—
DCD	8	1	I
DTR	20	4	O
RI	22	9	I

Таблица 3.2 — Назначение контактов разъемов интерфейса RS-232C

Сигнал	Назначение
CTS	Clear To Send — вход разрешения терминалу передавать данные. Состояние "выключено" аппаратно запрещает передачу данных. Сигнал используется для аппаратного управления потоками данных
DSR	Data Set Ready — вход сигнала готовности от аппаратуры передачи данных (модем в рабочем режиме подключен к каналу и закончил действия по согласованию с аппаратурой на противоположном конце канала)
DTP	Data Terminal Ready — выход сигнала готовности терминала к обмену данными. Состояние "включено" поддерживает коммутируемый канал в состоянии соединения
DCD	Data Carrier Detected — вход сигнала обнаружения несущей удаленного модема
Rl	Ring Indicator — вход индикатора вызова (звонка). В коммутируемом канале этим сигналом модем сигнализирует о принятии вызова
PG	Protected Ground — защитная земля, соединяется с корпусом устройства и экраном кабеля
SG	Signal Ground — сигнальная (схемная) земля, относительно которой действуют уровни сигналов
TO	Transmit Data — последовательные данные — выход передатчика
RD	Receive Data — последовательные данные — вход приемника
RTS	Request To Send — выход запроса передачи данных: состояние "включено" уведомляет модем о наличии у терминала данных для передачи. В полудуплексном режиме используется для управления направлением — состояние "включено" служит сигналом модему на переключение в режим передачи

Интерфейс RS-232C предназначен для подключения аппаратуры, передающей или принимающей данные (ООД — оконечное оборудование данных или АПД — аппаратура передачи данных) к оконечной аппаратуре каналов данных (АКД). В роли АПД может выступать компьютер, принтер, плоттер и другое периферийное оборудование. В роли АКД обычно выступает модем. Конечной целью подключения является соединение двух устройств АПД. Полная схема соединения приведена на рис. 3.2. Интерфейс позволяет исключить канал удаленной связи вместе с парой устройств АПД, соединив устройства непосредственно с помощью нуль-модемного кабеля (рис. 3.3).

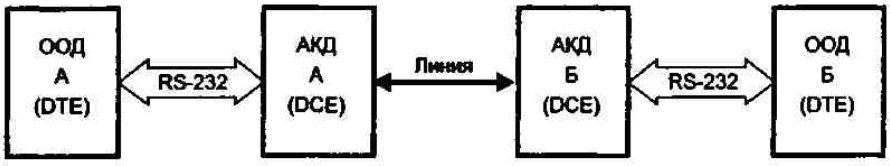

Рисунок 3.2 — Полная схема соединения по RS-232C

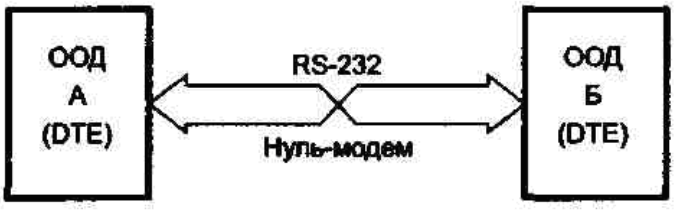

Рисунок 3.3 — Соединение по RS-232C нуль-модемным кабелем

3.1.5. Электрический интерфейс RS-232C

Стандарт RS-232C использует несимметричные передатчики и приемники — сигнал передается относительно общего провода — схемной земли (симметричные дифференциальные сигналы используются в других интерфейсах, например, RS-422). Интерфейс не обеспечивает гальванической развязки устройств. Логической единице соответствует напряжение на входе приемника в диапазоне –12...–3 В. Для линий управляющих сигналов это состояние называется ON ("включено"), для линий последовательных данных — MARK. Логическому нулю соответствует диапазон +3...+12 В. Для линий управляющих сигналов состояние называется OFF ("выключено"), а для линий последовательных данных — SPACE. Диапазон –3...+3 В — зона нечувствительности, обусловливающая гистерезис приемника: состояние линии будет считаться измененным только после пересечения порога (рис. 3.4).

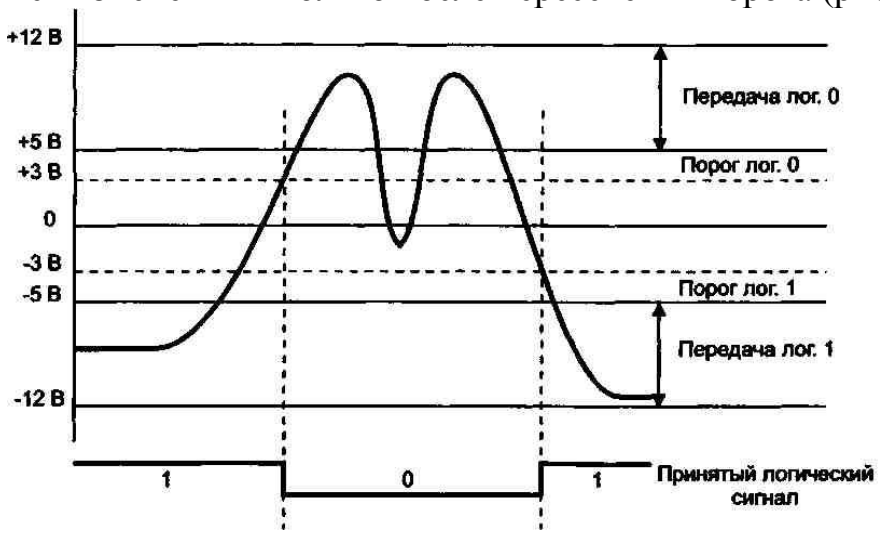

Рисунок 3.4 — Прием сигналов RS-232C

Уровни сигналов на выходах передатчиков должны быть в диапазонах –12...–5 В и +5...+12 В для представления единицы и нуля соответственно. Разность потенциалов между схемными землями (SG) соединяемых устройств должна быть менее 2 В, при более высокой разности потенциалов возможно неверное восприятие сигналов. Интерфейс предполагает наличие защитного заземления для соединяемых устройств, если они оба питаются от сети переменного тока и имеют сетевые фильтры. Подключение и отключение интерфейсных кабелей устройств с автономным питанием должно производиться при отключенном питании. Иначе разность невыровненных потенциалов устройств в момент коммутации может оказаться приложенной к

выходным или входным (что опаснее) цепям интерфейса и вывести из строя микросхемы. Для интерфейса RS-232C специально выпускаются буферные микросхемы приемников (с гистерезисом и передатчиком двуполярного сигнала). Вывести из строя интерфейсные микросхемы замыканием сигнальных цепей маловероятно: ток короткого замыкания передатчиков обычно не превосходит 20 мА.

Стандарт RS-232C регламентирует типы применяемых разъемов. На аппаратуре АПД (в том числе на COM-портах) принято устанавливать вилки (male — "папа") DB-25P или более компактный вариант — DB-9P. Девятиштырьковые разъемы не имеют контактов для дополнительных сигналов, необходимых для синхронного режима (в большинстве 25-штырьковых разъемов эти контакты не используются). На аппаратуре АКД (модемах) устанавливают розетки (female — "мама") DB-25S или DB-9S. Это правило предполагает, что разъемы АКД могут подключаться к разъемам АПД непосредственно или через переходные "прямые" кабели с розеткой и вилкой, у которых контакты соединены "один в один". Переходные кабели могут являться и переходниками с 9- на 25-штырьковые разъемы (рис.3.5).

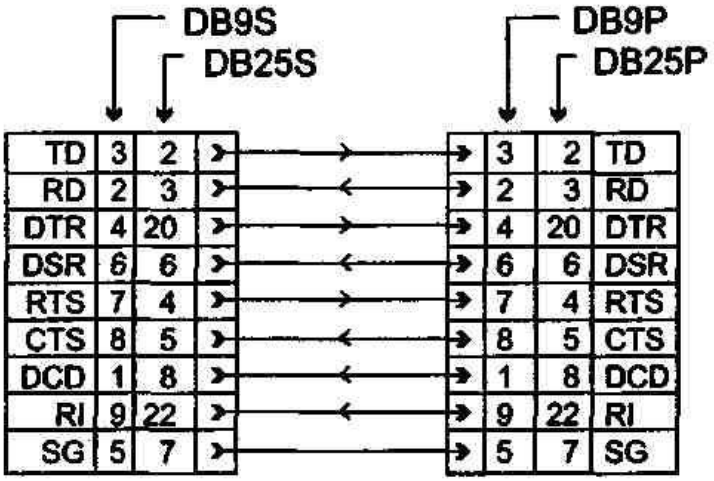

Рисунок 3.5 — Кабели подключения модемов

Если аппаратура АПД соединяется без модемов, то разъемы устройств (вилки) соединяются между собой нуль-модемным кабелем (Zero-modem или Z-modem), имеющим на обоих концах розетки, контакты которых соединяются перекрестно по одной из схем, приведенных на рис. 3.6.

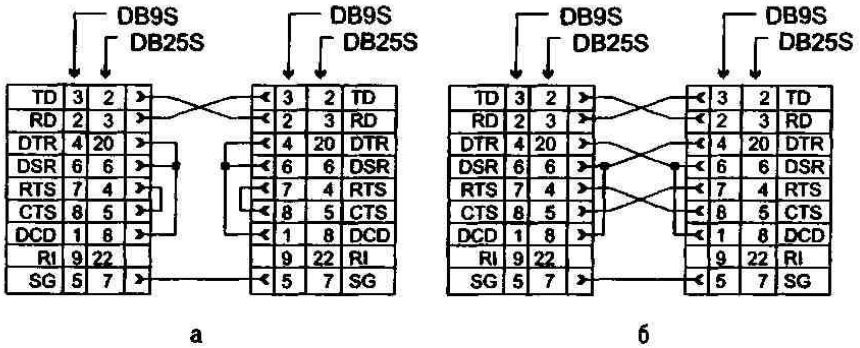

Рисунок 3.6 — Минимальный (а) и полный (б) нуль-модемный кабель

3.2. Практическая часть

3.2.1. COM-порт в операционной системе Windows

Рассмотрим работу с COM портом с использованием API функций Windows. С COM-портом в Win32 работают так же, как и с обычным файлом, используя при этом несколько специфичных функций WinAPI. Однако коммуникационный порт — это не совсем обычный файл. Для него, например, нельзя выполнить позиционирование файлового указателя, или же создать порт, если таковой отсутствует.

3.2.1.1. Открытие порта

Любая работа с портом начинается с его открытия. Для этого используется файловая функция CreateFile. Функция возвращает CreateFile описатель порта (hPort), который будет необходим для вызова других функций работы с портом. Если в результате открытия порта описатель не получен, то возбуждается исключение с соответствующим текстом ошибки.

```
HANDLE CreateFile(
LPCTSTR lpFileName,     {имя открываемого файла}
DWORD dwDesiredAccess,       {тип доступа}
DWORD dwShareMode,  {параметр совместного доступа}
LPSECURITY_ATTRIBUTES lpSecurityAttributes,
                                {атрибут защиты}
DWORD dwCreationDistribution,       {режим автосоздания}
DWORD dwFlagsAndAttributes, {атрибут режима обработки}
HANDLE hTemplateFile          {описатель шаблона}
);
```

Пример 3.1. Открытие порта COM из списка Combobox1 с параметрами, присущими для работы с коммуникационным портом (синхронный режим).

```
var      hComm: THandle;
Err: integer;
         DCB: TDCB;

Begin   {Создание файла для работы с COM портом}
hComm:=CreateFile(
PChar(Combobox1.Items.Strings[Combobox1.ItemIndex]),
GENERIC_READ or GENERIC_WRITE, 0, nil,                Open_Existing,
FILE_ATTRIBUTE_NORMAL, 0);
{Если порт не обнаружен, то выдача сообщения ошибки}
  if hComm=Invalid_Handle_Value then
  begin
```

```
        Err:=GetLastError;
        MessageBox(Handle,PChar('Ошибка открытия порта-
'+IntToStr(Err)),PChar('Ошибка COM-порта'),MB_ICONERROR);
       Exit;
     end;
```

3.2.1.2. Закрытие порта

По окончании работы с портом его следует закрыть, вызвав функцию закрытия порта CloseHandle, где в качестве единственного параметра надо передать полученный ранее описатель порта (hPort).

```
BOOL CloseHandle(HANDLE hObject);
```

Пример 3.2. Закрытие порта.

```
if hComm<>Invalid_Handle_Value then
        CloseHandle(hComm);
```

3.2.1.3. Настройка параметров порта

Для настройки порта используется функция SetCommState.

```
BOOL SetCommState(
HANDLE hFile,   {hFile — описатель открытого порта }
LPDCB lpDCB    {lpDCB — указатель на структуру DCB}
);
```

Основные параметры последовательного порта описываются структурой DCB. Она содержит массу полей, каждое из которых соответствует определенному параметру настройки порта. Рассмотрим несколько полей, которые нам необходимы

а) BaudRate — скорость передачи данных. Возможно указание констант — CBR_100, CBR_300, CBR_600, CBR_1200, …, CBR_256000.

б) Parity — схема контроля четности. Может содержать одно из следующих значений: EVENPARITY, MARKPARITY, NOPARITY, ODDPARITY, SPACEPARITY.

в) ByteSize — число информационных бит в передаваемых и принимаемых байтах.

г) StopBits — количество стоповых бит. Может быть ONESTOPBIT, ONE5STOPBIT, TWOSTOPBIT.

Чтобы не заполнять структуру DCB вручную, ее можно заполнить информацией о текущем состоянии порта вызовом функции GetCommState(), затем изменить необходимые поля и установить настройки вызовом функции SetCommState(). Настройку порта желательно производить сразу после его открытия. Помимо настройки структуры DCB необходимо выполнить настройку программных таймаутов, используя функции GetCommTimeouts и SetCommTimeouts.

Пример 3.3. Настройка параметров порта.

```
if not GetCommState(hComm, DCB) then
    begin
        Err:=GetLastError;
MessageBox(Handle, PChar('Ошибка получения параметров порта-
'+IntToStr(Err)), PChar('Ошибка COM-порта'),MB_ICONERROR);
        Exit;
    end;
  {Выбор скорости работы — 9600}
  DCB.BaudRate:=CBR_9600;

 {**** Установление параметров порта: ******}
  DCB.Parity:=NOPARITY;          {контроль четности}
  DCB.ByteSize:=8;    {кол-во передаваемых бит данных}
  DCB.StopBits:=ONESTOPBIT;    {кол-во стоповых бит}
    {Если параметры не установлены, то выдача ошибки}
  if not SetCommState(hComm, DCB) then
    begin
        Err:=GetLastError;
MessageBox(Handle, PChar('Ошибка установки параметров порта-
'+IntToStr(Err)), PChar('Ошибка COM-порта'),MB_ICONERROR);
        Exit
    end;
```

Пример 3.4. Настройка программных таймаутов.

```
if not GetCommTimeouts(hComm, CTO) then begin
    Err:=GetLastError;
    MessageBox(Handle, PChar('Ошибка получения таймаутов
порта'+IntToStr(Err)), PChar('Ошибка COM-порта'), MB_ICONERROR);
    Exit
  end;
  CTO.ReadIntervalTimeout:=70;
  CTO.ReadTotalTimeoutMultiplier:=0;
  CTO.ReadTotalTimeoutConstant:=5;
  CTO.WriteTotalTimeoutMultiplier:=10;
  CTO.WriteTotalTimeoutConstant:=0;
  if not SetCommTimeouts(hComm, CTO) then begin
    Err:=GetLastError;
    MessageBox(Handle, PChar('Ошибка установки таймаутов
порта'+IntToStr(Err)), PChar('Ошибка COM-порта'), MB_ICONERROR);
    Exit
  end;
```

3.2.1.4. Прием и передача данных

Прием и передача данных выполняется функциями ReadFile() и WriteFile().

```
BOOL ReadFile(
HANDLE hFile,
LPVOID lpBuffer,
DWORD nNumberOfBytesToRead,
LPDWORD lpNumberOfBytesRead,
LPOVERLAPPED lpOverlapped
);
BOOL WriteFile(
HANDLE hFile,
LPCVOID lpBuffer,
DWORD nNumberOfBytesToWrite,
LPDWORD lpNumberOfBytesWritten,
LPOVERLAPPED lpOverlapped
);
```

где hFile — описатель открытого порта;

lpBuffer — адрес буфера;

nNumberOfBytesToRead/nNumberOfBytesToWrite — число ожидаемых к приему или предназначенных для передачи байт;

lpNumberOfBytesRead/lpNumberOfBytesWritten — число фактически принятых или переданных байт;

lpOverlapped — адрес структуры OVERLAPPED, используемой для асинхронных операций;

Пример 3.5. Чтение данных по нажатию кнопки READ (Button2).

```
procedure TForm1.Button2Click(Sender: TObject);
var
i:byte;
Count: Cardinal;
Buffer: array[0..255] of byte;
begin
ReadFile(hcom,Buffer,255,Count,nil);
for i:=0 to Count do
      begin
          Edit1.Text:=Edit1.Text + IntToStr(Buffer[i]);
      end;
end;
```

Пример 3.6. Передача данных по нажатию кнопки WRITE (Button3).

```
procedure TForm1.Button3Click(Sender: TObject);
var
i:byte;
s:string;
Count: Cardinal;
```

```
Buffer: array[0..255] of char;
begin
s:=Edit2.Text;
if Length(s)>0 then
 begin
    for i:=0 to Length(s)-1 do
      begin
          Buffer[i]:=s[i+1];
      end;

    if not WriteFile(hcom,Buffer,Length(s),Count,nil) then
       begin
           MessageBox(handle,
           Pchar('Ошибка при передаче - '+ intToStr(getLastError)),
           Pchar('Ошибка COM-порта'),MB_ICONERROR);
           Exit;
        end;
     end;
 end;
end.
```

3.2.2. Порядок выполнения работы

1) Реализовать интерфейс программы работы с COM-портом рис. 3.7. Назначение кнопок: Open — открытие порта, его настройка и настройка таймаутов; Close — закрытие порта; Transmit — передача данных из поля Edit1; Receive — прием данных в поле Edit2.

2) Выполнить Пуск/Программы/Стандартные/Связь/ Hyperterminal и создать новое подключение. В свойствах «Подключаться через» выбрать требуемый COM-порт. Настроить работу COM-порта (рис. 3.8) в соответствие с настройкой программы п.1.

3) Запустить программу Virtual Serial Port Driver и выполнить виртуальное соединение COM-порта, выбранного в программе п.1 и в Hyperterminal п.2.

4) Проверить возможность чтения и передачи данных программой п.1, выполняя соответствующую передачу и чтение данных в Hyperterminal.

5) Выполнить разрыв виртуального соединения COM-портов используя Virtual Serial Port Driver.

6) Оформить отчет по выполненной работе.

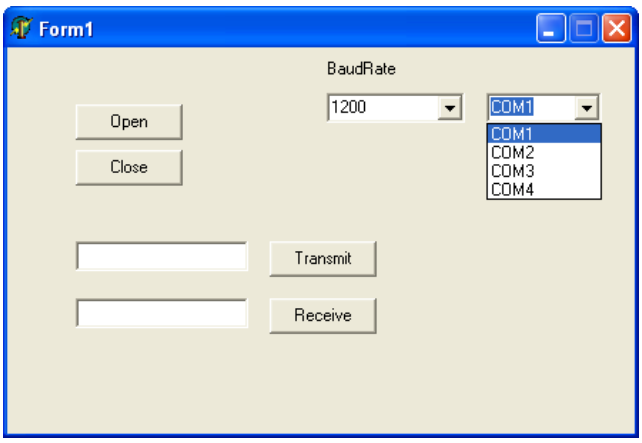

Рисунок 3.7 — Пример программы работы с COM-портом

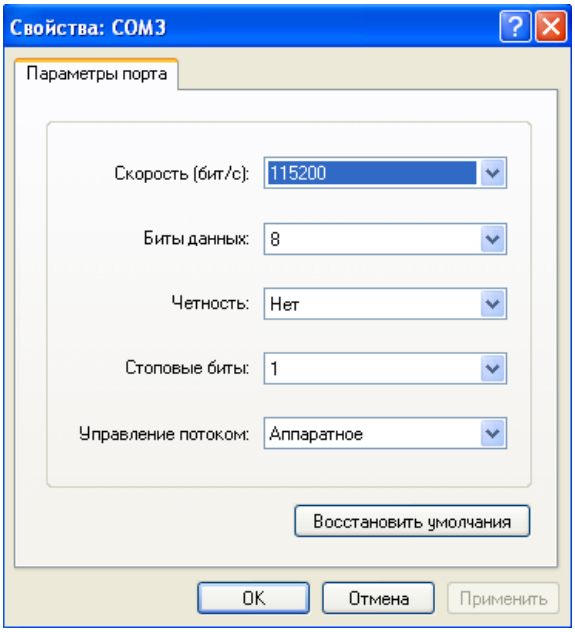

Рисунок 3.8 — Настройка параметров Hyperterminal по работе с COM-портом

3.3. Контрольные вопросы

1) Назовите отличительные особенности COM порта.
2) Перечислите назначения регистров COM порта.
3) Как работать с COM портом используя API функций Windows?
4) Опишите функцию открытия порта CreateFile.
5) Опишите функцию закрытия порта CloseFile.
6) Приведите примеры записи и чтения порта.
7) Как пользоваться программой Hyperterminal?
8) Как пользоваться программой Virtual Serial Port Driver?

Лабораторная работа № 4.
ПРЕОБРАЗОВАНИЕ RS-232-UART

Цель работы — изучить метод преобразования сигнала RS-232 в UART; выполнить преобразование сигналов, используя согласующий буфер MAX232.

4.1. Теоретические сведения

4.1.1. Преобразование уровней RS-232 в TTL

Интерфейс RS232/UART может применяться для сопряжения отдельных электронных устройств. Для этого необходимо:

- наличие аппаратного или программно эмулируемого порта UART в периферийном устройстве;
- наличие порта RS232C или адаптера USB-RS232 на ПК;
- наличие интерфейсной микросхемы-преобразователя уровня или транзисторного преобразователя уровней.

В качестве схемы согласования уровней RS232 в TTL-уровни можно использовать транзисторный преобразователь, схема которого приведена на рис. 4.1.

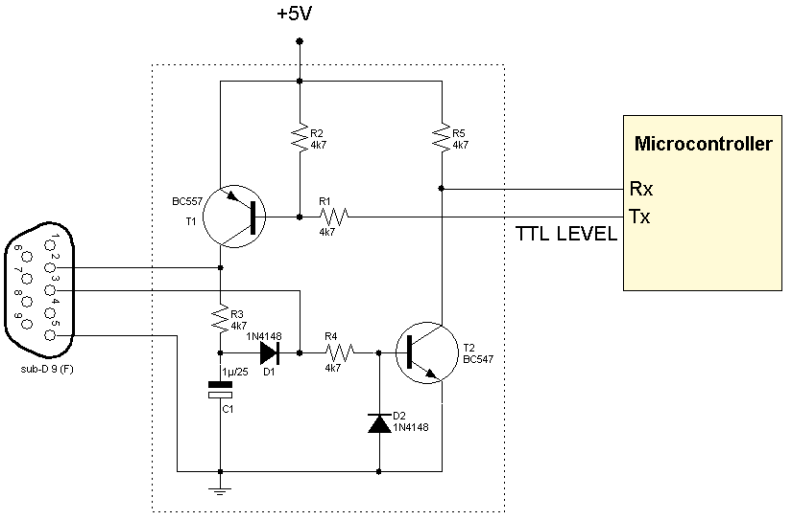

Рисунок 4.1 — Схема транзисторного преобразователя

Транзисторный преобразователь не обеспечивает нужные электрические уровни при передачи данных от периферийного устройства (в данном примере от микроконтроллера) к ПК, из-за чего возможна нестабильная работа канала связи. Поэтому для согласования уровней необходимо применять специальные преобразователи. Например, в качестве преобразователей применяют микросхемы MAX220 - MAX249 — линейные приемо-передатчики, предназначенные для преобразования интерфейсов EIA/TIA-232E и V.28/V.24, особенно в устройствах, где отсутствуют напряжения ±12В. Для сопряжения микроконтроллера и ПК предпочтительно использовать интерфейсную микросхему MAX232 (для систем с напряжением питания +5 В) или MAX3232 (для систем с напряжением питания +3,3 В). Другая альтернатива —

микросхема ICL232 — сдвоенный приемо-передатчик, соответствующий спецификациям RS-232C и V.28. Напряжения +10В и –10В преобразуются из 5 В при помощи двух емкостных преобразователях напряжения.

4.1.2. Преобразователи семейства MAX232

MAX232 — интегральная схема, преобразующая сигналы последовательного порта RS-232 в сигналы, пригодные для использования в цифровых схемах на базе ТТЛ или КМОП технологий. MAX232 работает приемопередатчиком и преобразует сигналы RX, TX, CTS и RTS (рис. 4.2). В преобразователе содержится 2-х канальный приемник (выводы 12-13 и 9-8 соответственно) и 2-х канальный передатчик RS232C (выводы 11-14 и 10-7).

Схема MAX232 обеспечивает уровень выходного напряжения, используемый в RS-232 (приблизительно ± 7.5 В), преобразуя входное напряжение + 5 В при помощи внутреннего зарядового насоса на внешних конденсаторах. Это упрощает реализацию RS-232 в устройствах, работающих на напряжениях от 0 до + 5 В, так как не требуется усложнять источник питания только для того, чтобы использовать RS-232.

Входное напряжение от RS-232, которое может достигать ± 25 В, понижается до стандартных 5 В, используемых в транзисторно-транзисторной логике. Входы имеют средний порог 1,3 В и средний гистерезис 0,5 В. Когда схема MAX232 получает на вход логический "0" от ТТЛ, она преобразует его в напряжение от +3 до +15В, а когда получает логическую "1" — преобразует её в напряжение от –3 до –15В, и по тому же принципу выполняет обратные преобразования от RS-232 к ТТЛ.

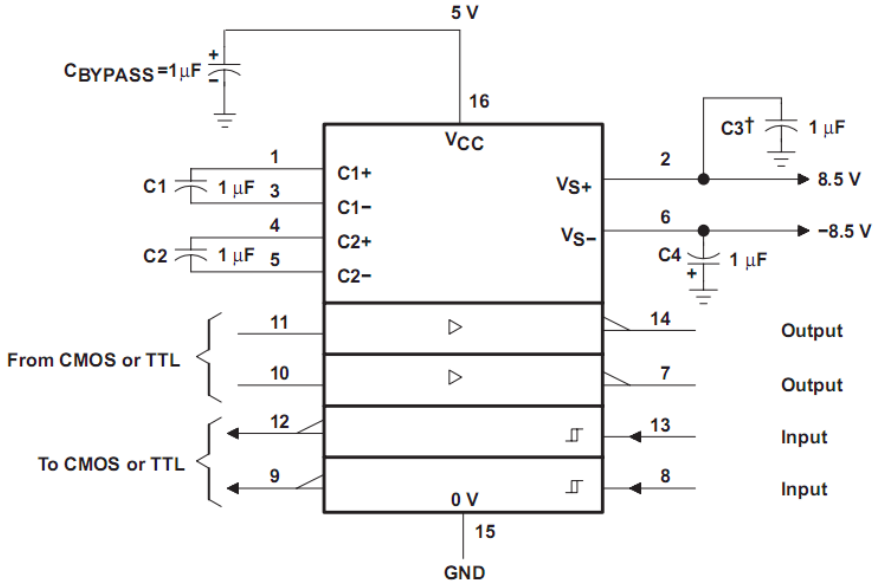

Рисунок 4.2 — Преобразователь MAX232

Модификация MAX232A обратно совместима с MAX232, но может работать на более высоких скоростях, и использовать внешние конденсаторы меньшей емкости — 0,1 мкФ вместо конденсаторов на 1,0 мкФ, используемых с оригинальной схемой.

Модификация MAX3232 также обратно совместима с предыдущими, но работает в диапазоне напряжений от 3 В до 5.5 В. Схема применения MAX3232 приведена на рис. 4.3.

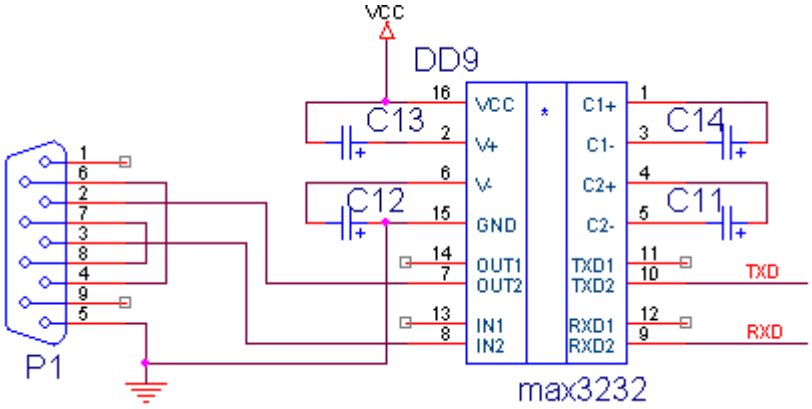

Рисунок 4.3 — Схема применения микросхемы MAX3232

4.1.3. Программное обеспечение для работы с портом

Terminal — это простой эмулятор терминала последовательного порта COM (рис. 4.3). Может применяться для коммуникации с различными устройствами, такими как модемы, роутеры, GSM телефоны. Данная утилита полезна для отладки приложений для соединений по последовательному каналу.

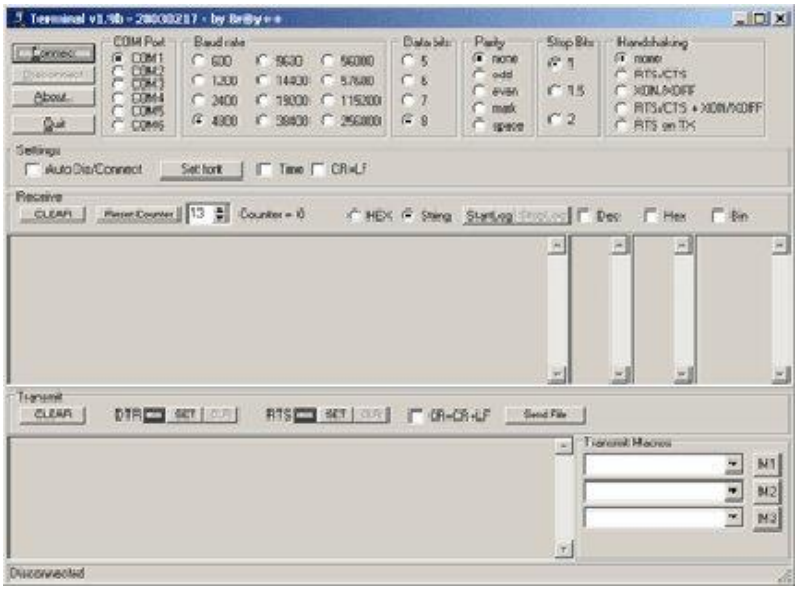

Рисунок 4.3 — Внешний вид программы Terminal

4.2. Порядок выполнения работы

1) Используя нуль-модемный кабель подключить COM порт ПК и COM разъем лабораторного макета.

2) Используя утилиту Terminal, установить соединение между ПК и лабораторным макетом, отправив и приняв данные.

3) Реализовать программу обмена данными через COM порт по примеру лабораторной работы №3 п.1, изменив процедуру чтения данных — чтение данных производить постоянно.

4) Проверить соединение между ПК и лабораторным макетом, отправив и приняв данные по RS-232C.

5) Оформить отчет по лабораторной работе.

4.3. Контрольные вопросы

1) Поясните назначение преобразователей уровней TTL / RS232?

2) Опишите особенности преобразовательных микросхем семейства MAX.

3) Назовите функциональные особенности MAX232.

4) Нарисуйте схему преобразования уровней с использованием микросхемы MAX232.

5) Какие особенности работы с утилитой Terminal?

Лабораторная работа № 5.
ПРЕОБРАЗОВАНИЕ USB-UART

Цель работы: изучить особенности интерфейса USB; выполнить преобразование сигналов, используя согласующий буфер FT232R.

4.1. Теоретические сведения

4.1.1. Интерфейс USB

Универсальная последовательная шина (USB) стала чрезвычайно популярной за счет предоставления ряда удобств конечным пользователям, например, функция "Plug and Play", которая позволяет идентифицировать подключенное устройство без необходимости рестарта компьютера.

Физический интерфейс USB состоит из четырех проводников: 2 для питания внешнего устройства (VCC и GND) и 2 сигнальных проводника (DATA+ и DATA–). Через проводники питания передается постоянное напряжение приблизительно 5 В с нагрузочной способностью максимум 500 мА.

В соответствии со стандартом USB высокий уровень на сигнальных проводниках должен составлять 3,0…3,6В, при этом, напряжение питания Vcc шины USB, поступающее от главного (компьютера) составляет 4,4…5,25В (рис. 5.1).

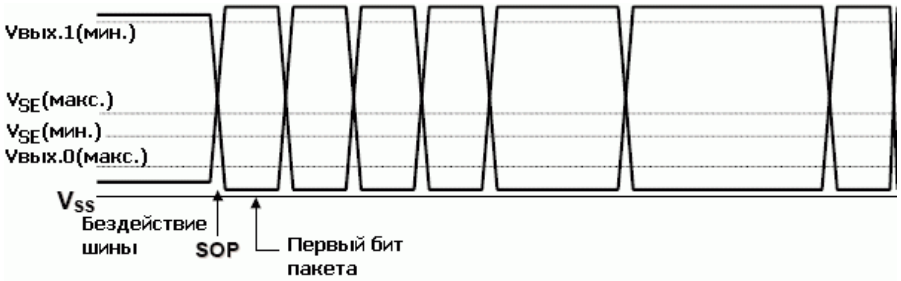

Рисунок 5.1 — Уровни напряжений при передаче пакетов

Принцип детектирования подключения и отключения USB-устройства основан на контроле сопротивления линии USB (рис. 5.2). У низкоскоростных USB-устройств необходим подтягивающий резистор между сигналом DATA– и Vcc. У полноскоростных устройств данный резистор подключается к DATA+. Определяя, на какой линии подключен подтягивающий резистор, главный компьютер определяет какое новое устройство подключено к линии USB.

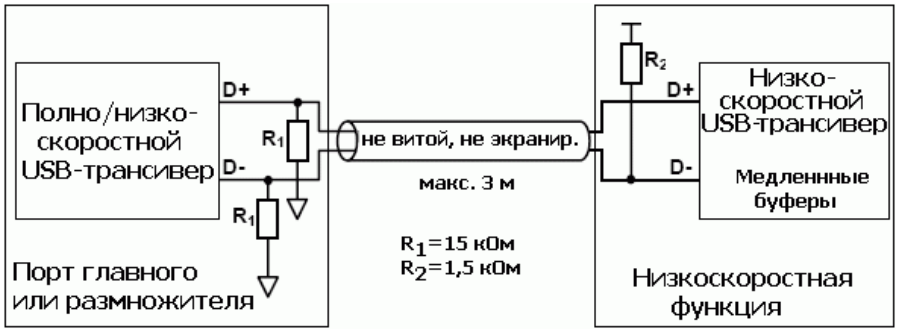

Рисунок 5.2 — Подключения кабеля и резисторов к низкоскоростному устройству

После определения нового устройства ПК начинает связь в соответствии с физическим протоколом USB. Протокол USB, в отличие от UART, основан на синхронной передаче данных. Синхронизация передатчика и приемника необходима для осуществления связи. Синхронизация выполняется путем передачи небольшого заголовка "образцовая синхронизация", который предшествует передаче данных (рис. 5.3). Данный заголовок представляет собой прямоугольные импульсы (101010), за ними передаются два 0, а затем данные.

Уведомление об окончании передачи данных выполняется с помощью передачи сигнала "конец пакета" EOP (рис. 5.4). EOP передается путем установки низких уровней на обеих линиях данных DATA+ и DATA–. EOP передается непродолжительное время (минимум два периода скорости данных). После этого, выполняется следующая транзакция.

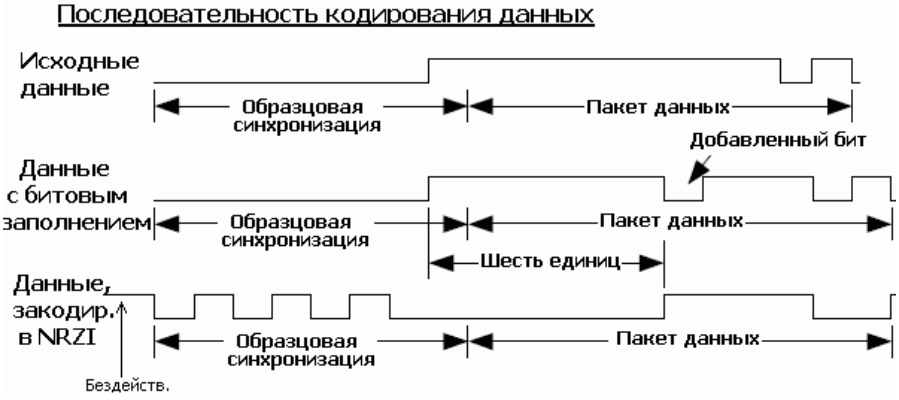

Рисунок 5.3 — Синхронизация приемника и передатчика

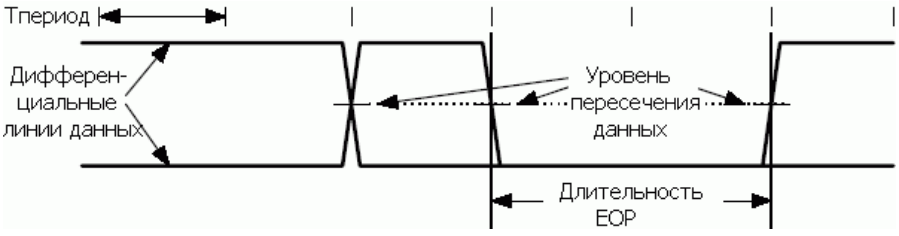

Рисунок 5.5 — Временная диаграмма сигнала EOP

Данные, которые передаются между образцовой синхронизацией и EOP, закодированы в коде NRZI (рис. 5.5). Поток данных состоит из пакетов, пакет в

свою очередь состоит из нескольких полей: поле синхронизации (образцовая синхронизация), идентификатор пакета (PacketID, PID), поле адреса (ADDR), поле конечной точки (ENDP), данные и поле циклического избыточностного контроля (CRC). USB подразумевает четыре типа передачи: передача управления, передача прерывания, изохронная передача и передача потока. Каждый из этих типов передач специфичен для различных требований устройства.

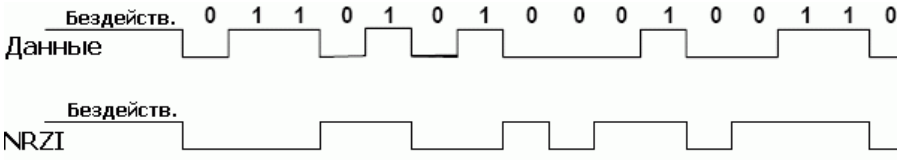

Рисунок 5.5 — Кодирование данных в коде NRZI

Режим передачи управления должен присутствовать у каждого USB-устройства, т.к. он используется для конфигурации при подключении устройства, когда необходимо получить информацию об устройстве, установленный адрес устройства и пр. Каждая передача управления состоит из нескольких стадий: стадия установки, стадия данных и стадия статуса.

Данные по шине USB передаются пакетами, по несколько байт в каждом. Размер пакета определяется каждым устройством, но его предельный размер ограничен. Для низкоскоростных устройств максимальный размер пакета равен 8 байтам. Данный 8-байтный пакет вместе с начальным и конечным полем должны быть приняты в буфер устройства за одну USB-передачу. В аппаратно-реализованных USB-приемниках различные части передачи автоматически дешифрируются и устройство уведомляется, когда все сообщение назначено отдельному устройству. При программной реализации USB-сообщение дешифрируется программно после приема в буфер всего сообщения. Вследствие этого возникают требования и ограничения: устройство должно иметь буфер для хранения всего USB-сообщения, иметь другой буфер для USB-передачи (подготовка данных для передачи), а также выполнять администрирование с дешифрованием и проверкой сообщений.

5.1.2. CP2102/CP2103 — преобразователи интерфейсов USB-RS232/RS485

Микросхемы CP2102 и CP2103 представляют собой преобразователи данных между USB и UART форматами. Эти микросхемы (а также выпускавшаяся ранее CP2101) пин-совместимы и взаимозаменяемы во многих приложениях. CP2103 рекомендована для применения в тех случаях, когда требуется работа с низким UART напряжением, наличие GPIO, а также возможность работы по RS485.

CP2103 представляет собой высокоинтегрированный мостовой контроллер USB-UART (рис. 5.6). Он дает возможность подключать устройства с интерфейсами RS-232 или RS-485 к ПК, имеющим только USB порт. При этом используется минимальное количество дополнительных компонентов и занимается минимум места на печатной плате. CP2103 является

усовершенствованным продолжением серии CP2101/02 и наделен новыми функциями, такими как возможность настройки уровня напряжения портов ввода/вывода (3,3-1,8В) и четыре сигнала ввода/вывода общего назначения, предназначенные для отображения состояния системы и управления.

CP2103 выпускается в компактном корпусе типа MLP-28 (5 x 5 мм) и содержит функциональный контроллер полноскоростного интерфейса USB 2.0, трансивер USB, генератор, EEPROM и универсальный асинхронный приемопередатчик (UART), поддерживающий полный набор управляющих сигналов модема. Никаких других внешних компонентов USB не требуется.

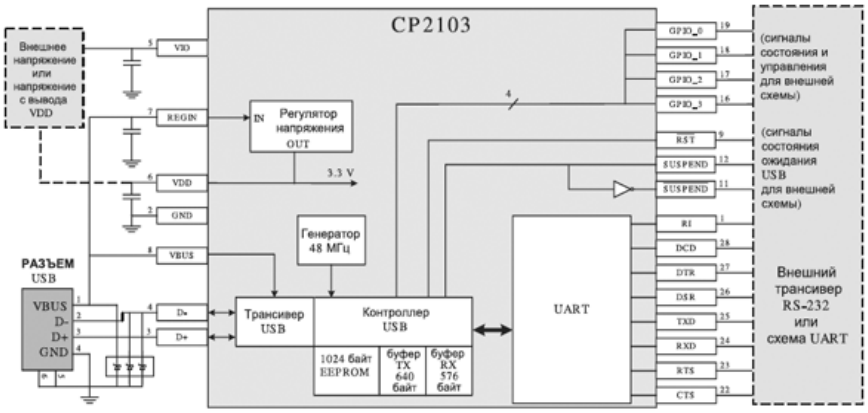

Рисунок 5.6 — Функциональная схема преобразователя CP2103

5.1.3. FTDI FT232R (FT232RL и FT232RQ) — преобразователи интерфейсов USB-RS232/RS485

Микросхема FTDI FT232R (FT232RL и FT232RQ) является высокоинтегрированным переходником USB-COM позволяющая используя минимум внешних компонентов (разъем и пассивные компоненты) организовать последовательный обмен данными между внешним устройством на микроконтроллере и компьютером через шину USB. По сравнению с предыдущими версиями микросхемы у FT232R на кристалл интегрированы тактовый генератор, энергонезависимая память EEPROM, часть внешних пассивных компонентов.

На рис.5.7 представлена функциональная схема FT232. Ее основа — приемопередатчики обоих интерфейсов. Блок UART снабжен полным набором сигнальных цепей стандарта RS-232, приемопередатчик USB — всего двумя информационными выводами USBDP и USBDM, образующие двунаправленный канал передачи данных. Блок последовательного контроллера SIE преобразует поеледовательный код в параллельный и обратно, выполняет процедуры битстаффинга, генерирует (для исходящего потока данных) и проверяет (для входящего) контрольные коды.

Обработчик протокола USB нижнего уровня формирует ответы на запросы хост-контроллера (ПК). Через него же управляют режимом работы UART.

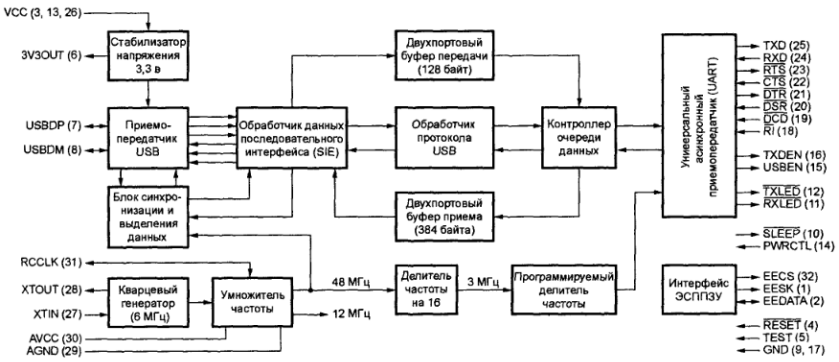

Рисунок 5.7 — Функциональная схема преобразователя FT232R

На рис. 5.8 представлена схема включения FT232R для применения согласования уровней USB и UART микроконтроллера.

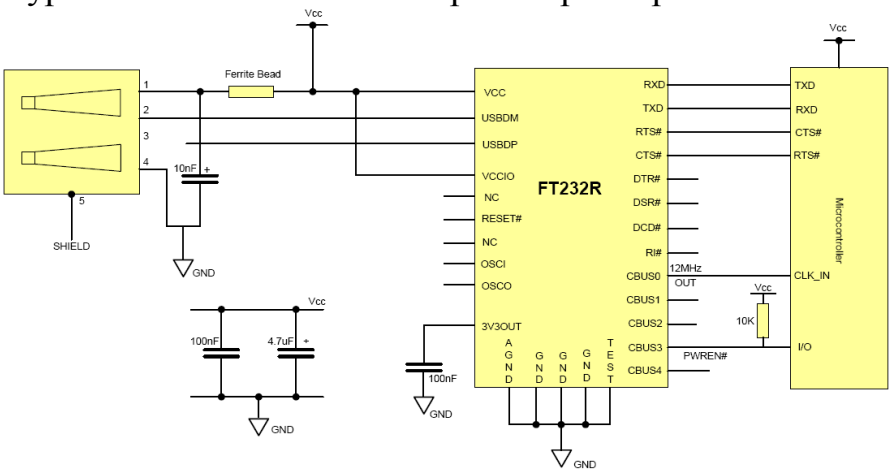

Рисунок 5.8 — Схема преобразования интерфейса USB в UART микроконтроллера

5.2. Практическая часть

5.2.1. Способы реализации интерфейса USB

Для разработчиков интегрировать USB-интерфейс в свои проекты оказалось более сложным по сравнению, например, с интерфейсом RS232. Кроме этого, на стороне ПК также должен быть предусмотрен специальный драйвер устройства. Как следствие, интерфейс RS232 остается очень популярным среди производителей конечных систем. Данный интерфейс хорошо изучен и в достаточной мере поддерживается операционной системой. Однако, как правило, в современных ПК физический порт RS232 не устанавливается и замещается на порты USB.

Реализовать интерфейс USB во внешнем устройстве можно способами.

А) В устройстве установлен приемопередатчик USB, такой как: PDIUSBD12, USBN9603, USBN9604.

Б) С помощью микроконтроллера, у которого интерфейс USB реализован аппаратно. В этом случае необходимо знать, как работает USB и в соответствии с этим написать программу для микроконтроллера. Кроме того, если

операционная система не поддерживает стандартные классы USB, то необходимо написать драйвер для компьютера. Основной недостаток данного способа — ограниченная доступность таких микроконтроллеров и их более высокая стоимость по сравнению с обычными микроконтроллерами, которые используются для связи через RS232.

В) Использование универсального преобразователя интерфейсов: USB и любого другого. В качестве другого интерфейса обычно используется RS232, 8-разрядная шина данных или шина TWI. В этом случае разработка специальной прошивки не потребуется, нет надобности знать, как работает USB и нет необходимости написания драйвера, т.к. производитель преобразователя предлагает свой драйвер. Недостаток — более высокая стоимость завершенной системы, а также повышенные габариты готового изделия.

На рис.5.9 представлена упрощенная схема согласования интерфейсов USB-RS-232 с использованием FT232R.

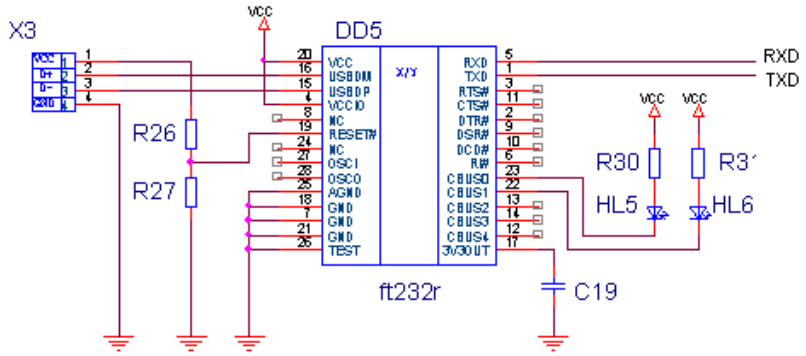

Рисунок 5.9 — Переходник USB-UART

5.2.2. Порядок выполнения работы

1) Запустить программу обмена данными через COM порт по примеру лабораторной работы №3 п.1.

2) Используя USB-B кабель подключить USB порт ПК и USB-B разъем лабораторного макета.

3) По запросу ПК установить драйвер FTDI[2], в системе должен появиться новый виртуальный COM порт (устройство).

4) Проверить соединение между ПК и лабораторным макетом, отправив и приняв данные через виртуальный COM порт. Предусмотреть поиск в системе и выбор найденного виртуального порта. Скорость работы установить 9600 бит/с.

5) Оформить отчет по лабораторной работе.

5.3. Контрольные вопросы

1) Назовите отличительные особенности шины USB.

2) Поясните принцип детектирования подключения и отключения USB-устройства.

[2] Драйвер можно найти по адресу http://www.ftdichip.com/Drivers/VCP.htm

3) Каким образом кодируются данные при передаче по шине USB?

4) Какое назначение преобразователей CP2102/CP2103?

5) Какими характеристиками обладают преобразователи CP2102/CP2103?

6) Поясните схему подключения USB-UART с использованием преобразователя FT232R.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Гук М. Аппаратные интерфейсы IBM-PC / М. Гук. — Энциклопедия, 2-е изд. — СПб.: Питер, 2003. — 928с.

2. Гук М. Интерфейсы ПК : справочник / М. Гук. — СПб.: Питер Ком, 1999. — 416 с.

3. Томпкинс У. Сопряжение датчиков и устройств ввода данных с компьютерами IBM PC / У. Томпкинс, Д. Уэбстер. — М.: Мир, 1992. — 592с.

4. Агуров П.В. Практика программирования USB / П.В. Агуров. — СПб.: БХВ-Петербург, 2006. — 624 с.

5. Бычков Е.А. Архитектуры и интерфейсы персональных компьютеров. — М. : Центр «СКС», 1993. — 152 с.

6. Лысенко, А.А. Преобразователи интерфейса USB на микросхемах FT8U232AM, FT8U245AM / А.А. Лысенко, Р.Н. Назмутдинов // Радио. — 2002. — № 6. — С. 36–39.

7. Новиков Ю.В. Разработка устройств сопряжения для персонального компьютера типа ibm pc / Ю.В. Новиков, О.А. Калашников, С.Э. Гуляев. — М.: ЭКОМ, 1998.

8. Пей Ан. Сопряжение ПК с внешними устройствами / Ан. Пей; пер. с англ. — М. : ДМК Пресс, 2001. — 320 с.

Изд. № 55/18. Объем 2,5 п.л.
РИИЦМ ФГАОУ ВО «Севастопольский государственный университет»