Пей Ан

Сопряжение ПК с внешними устройствами

ДМК Пресс

Newnes

- Новые возможности персонального компьютера
- Сетевые приложения и удаленный доступ
- Игровой, параллельный и последовательный порты
- Программы управления экспериментальными платами
- Измерение аналоговых величин с помощью внешних устройств

ББК 32.844.1
А64

Ан П.
А64 Сопряжение ПК с внешними устройствами: Пер. с англ. – М.: ДМК Пресс. – 320 с.: ил.

ISBN 5-94074-076-6

Данная книга посвящена возможностям персонального IBM-совместимого компьютера по сопряжению с внешними устройствами через параллельный, последовательный и игровой порты, которые имеются практически в любом современном ПК. В качестве внешних устройств выступают ЦАП и АЦП, схемы управления электромоторами, трансиверы, модемы, различные индикаторы, датчики и пр.; приводятся тексты программ управления с подробными комментариями.

Книга предназначена для широкого круга читателей, интересующихся информатикой, электроникой и вычислительной техникой. Она будет полезна студентам технических вузов и колледжей в качестве учебного пособия при изучении аппаратной части ПК, а также радиолюбителям, которые стремятся наиболее полно использовать возможности домашнего компьютера. Начинающие программисты найдут здесь большое количество исходных текстов программ, а инженеры-электронщики почерпнут новые идеи для красивой реализации своих профессиональных проектов.

ББК 32.844.1

PC Interfacing, Practical Guide to Centronic RS232 and Game Ports by Pei An.

ISBN 0-24051-448-3 (англ.)

ISBN 5-94074-076-6 (рус.)

СОДЕРЖАНИЕ

1. ПАРАЛЛЕЛЬНЫЙ, ПОСЛЕДОВАТЕЛЬНЫЙ И ИГРОВОЙ ПОРТЫ

Параллельный, последовательный и игровой порты – это наиболее распространенные порты ввода/вывода. В некоторых портативных компьютерах может не быть игрового порта, но параллельный и последовательный входят в стандартную комплектацию для всех типов ПК.

Изначально каждый из этих портов разрабатывался для определенного применения. Параллельные предназначались для соединения компьютеров с принтерами, последовательные – для подключения принтеров, модемов и мыши, а игровые – для присоединения джойстиков. Однако они могут использоваться и для других приложений, связанных с сопряжением компьютера с внешними устройствами. Периферийные устройства, созданные для этих портов, легко подключаются к IBM PC-совместимому компьютеру. Принципиальные схемы отличаются мобильностью и могут применяться для решения проблем сопряжения с любым оборудованием, которое оснащено указанными портами. Таким образом, полезно узнать, как они работают и каким образом обеспечивается наиболее эффективное их использование.

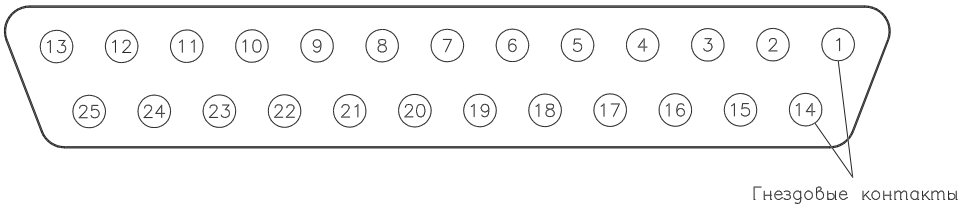
1.1. Параллельный порт

Порт Centronic, или *параллельный*, – это промышленный стандарт для подсоединения принтеров к компьютеру. Компьютер имеет по крайней мере один такой порт, встроенный в материнскую плату или представляющий собой отдельную интерфейсную карту ввода/вывода. Увеличить количество параллельных портов просто и недорого, можно установить четыре параллельных порта с логическими именами от LPT1 до LPT4. Команды управления принтером подробно не описываются.

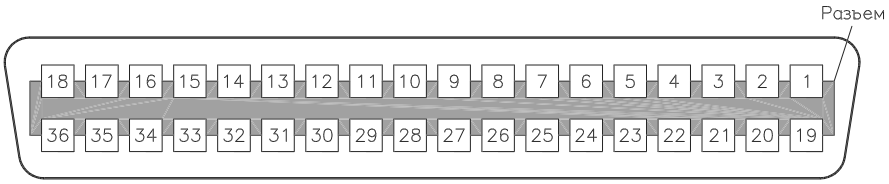
1.1.1. Разъемы

Разъемы порта для компьютера и принтера отличаются друг от друга. Первый – это 25-контактная розетка D-типа (рис. 1.1а), а второй – 36-контактная розетка параллельного типа (рис. 1.1б).

а)



б)



в)

Номера контактов на компьютерах	Номера контактов на принтерах	Направление (относительно ПК)	Наименование	Назначение
1	1	Выход	$\overline{\text{STROBE}}$	Строб данных
2	2	Выход	DB0	Бит данных 0
3	3	Выход	DB1	Бит данных 1
4	4	Выход	DB2	Бит данных 2
5	5	Выход	DB3	Бит данных 3
6	6	Выход	DB4	Бит данных 4
7	7	Выход	DB5	Бит данных 5
8	8	Выход	DB6	Бит данных 6
9	9	Выход	$\overline{\text{DB7}}$	Бит данных 7
10	10	Вход	$\overline{\text{ACK}}$	Подтверждение приема данных, готовность принтера
11	11	Вход	BUSY	Подтверждение занятости принтера
12	12	Вход	PE	Нет бумаги
13	13	Вход	$\overline{\text{SLCT}}$	Принтер подключен к линии
14	14	Выход	$\overline{\text{LF/CR}}$	Автоматический перевод строки после возврата каретки
15	32	Вход	$\overline{\text{ERROR}}$	Ошибка в принтере
16	31	Выход	$\overline{\text{INITIALIZE}}$	Установка параметров по умолчанию
17	36	Выход	$\overline{\text{SLIN}}$	Выбор принтера
18...25	19...30, 33		GND	Витая пара, соединенная с "землей"
	18, 34		Не используется	
	16		LOGIC GND	Логическая "земля"
	17		CHASSIS GND	Заземление на шасси

Рис. 1.1. Контакты на разъемах параллельного порта компьютера и принтера: а – блочная часть 25-контактного гнездового разъема D-типа, вид со стороны задней стенки компьютера; б – блочная часть 36-контактного разъема параллельного типа, вид со стороны задней стенки принтера; в – назначение контактов разъемов параллельного порта

Назначение контактов обоих разъемов представлено на рис. 1.1в. Для соединения компьютера с принтером используется принтерный кабель (рис. 1.2) длиной не более 5 м.

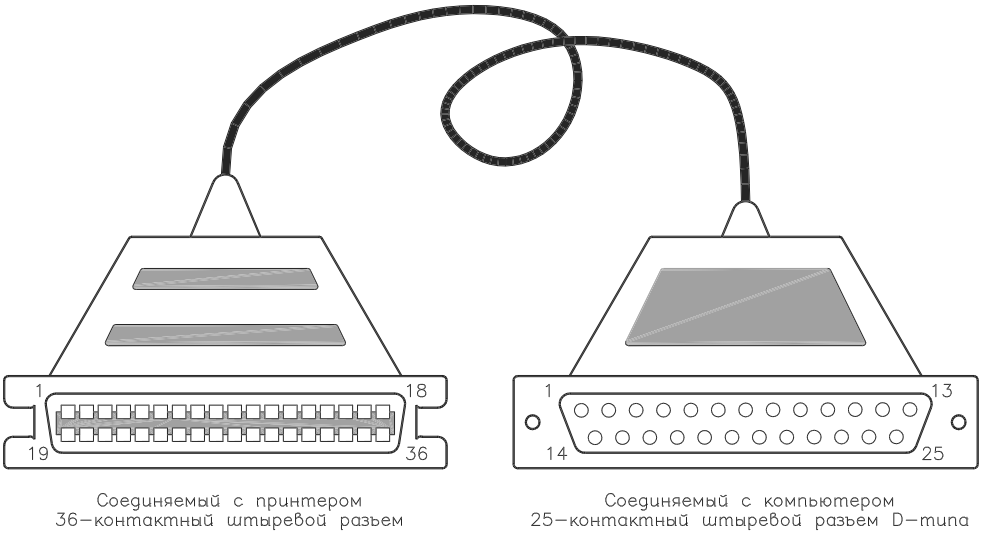


Рис. 1.2. Кабель принтера

1.1.2. Внутреннее устройство

Общая схема параллельного порта внутри ПК представлена на рис. 1.3. Восьмибитовые данные заносятся в DD1 во время записи в регистр с адресом базовый адрес + 0. Операция осуществляется командой WRITE_DATA.

Эти данные образуют группу. Они считываются компьютером из того же регистра через DD2 с помощью команды READ_DATA. Во время чтения выход DD1 должен иметь высокий уровень сопротивления, что достигается подачей на контакт 1 (выход разрешен) DD1 высокого уровня напряжения. Шестибитовое управляющее слово записывается в DD3 через регистр с адресом базовый адрес + 2 при помощи команды WRITE_CONTROL. Биты с 0 по 3 подаются на выход разъема и образуют группу управления. Некоторые биты инвертируются микросхемами с открытыми коллекторами на выходе (DD6 и DD7). Все выходные линии подключены к питанию +5 В через резисторы 4,7 кОм. Состояние этих линий считывается через регистр с адресом базовый адрес + 2 через DD4 посредством команды READ_CONTROL. Четвертый бит управляющего байта разрешает прерывание, а пятый бит открывает или закрывает выход DD1. Состояние пяти контактов разъема порта (группа состояния) компьютер считывает через DD4 с помощью команды READ_STATUS через регистр с адресом базовый адрес + 1. Входы линии подключены к питанию +5 В через резисторы 4,7 кОм, два входа инвертируются.

В первых конструкциях IBM PC контакт «выход разрешен» DD1 соединялся с «землей» для постоянного открывания выходов. Это была однонаправленная версия параллельного порта. Начиная с IBM PS/2, указанный контакт соединили

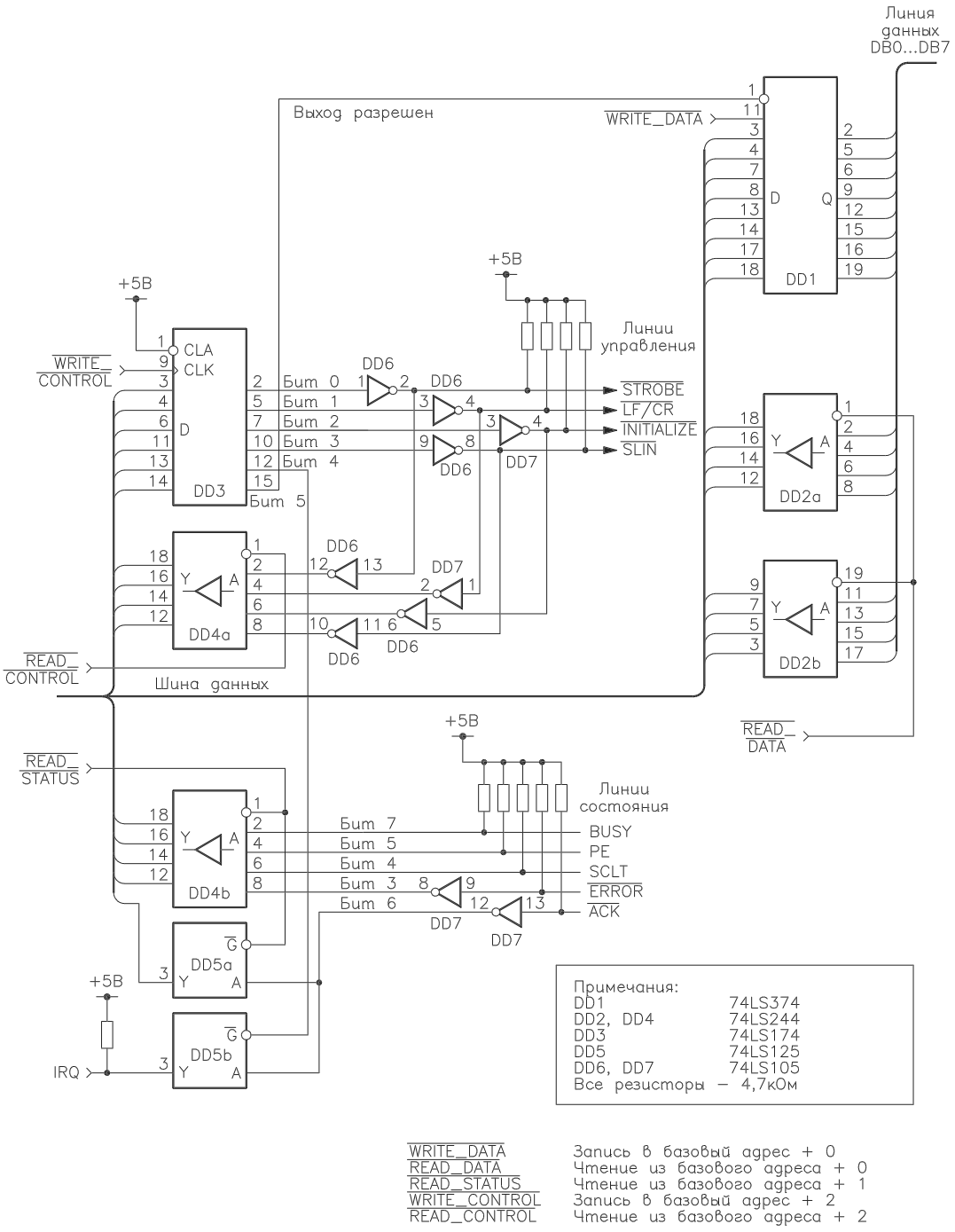


Рис. 1.3. Общая схема параллельного порта

с пятым битом регистра управления DD3 (см. рис. 1.3), и порт стал двунаправленным. Следует отметить, что многие параллельные порты, поставляемые со встроенными картами ввода/вывода, двунаправленные. Для любого контакта следует избегать короткого замыкания и/или соединения с шиной питания. Скорость передачи данных через параллельный порт превышает 1 Мб/с.

В этой главе детально рассматривается однонаправленный параллельный порт. Контакты порта образуют три группы: данных, управления и состояния. На рис. 1.4 представлена логическая структура параллельного порта.

Группа данных

Посылает данные от ПК на внешние устройства. Имеет восемь выходных линий и ассоциируется с байтом в адресном пространстве ввода/вывода процессоров x86. Адрес: базовый адрес.

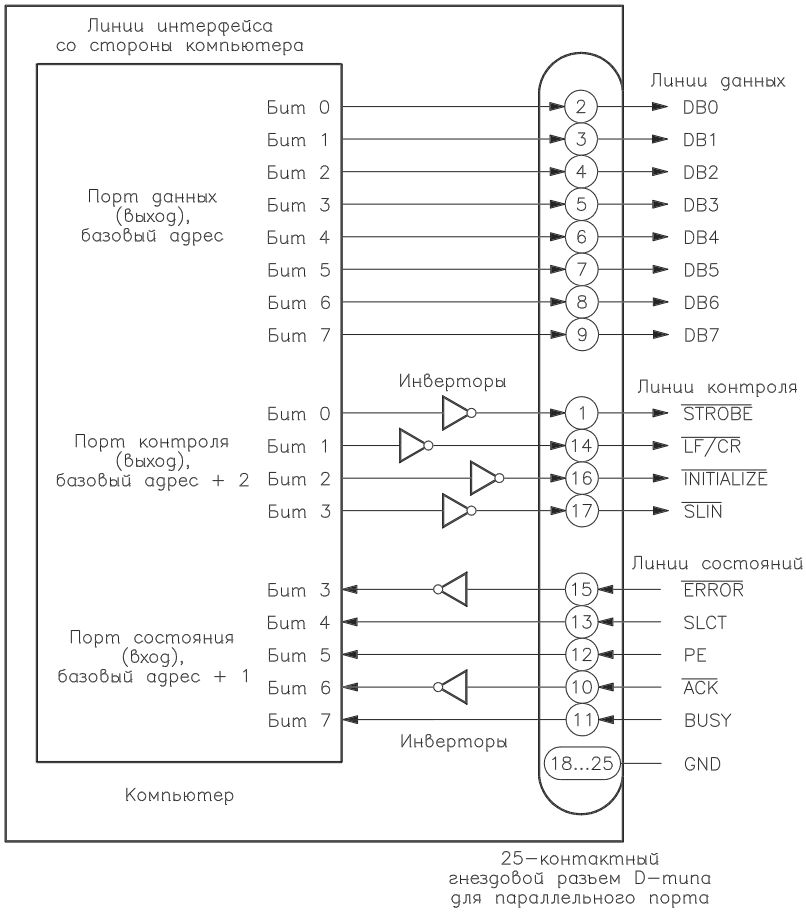


Рис. 1.4. Логическая структура параллельного порта

Группа управления

Контролирует внешнее устройство. Имеет четыре инвертированные выходные линии ($\overline{\text{STROBE}}$, $\overline{\text{LF/CR}}$, $\overline{\text{SLIN}}$ и $\overline{\text{INITIALIZE}}$). Адрес группы управления: базовый адрес + 2.

Группа состояния

Группа может использоваться компьютером для получения текущего состояния внешнего устройства, ее адрес: базовый адрес + 1. Имеет пять линий ($\overline{\text{ERROR}}$, SLCT, PE, $\overline{\text{ACK}}$ и BUSY). Линии $\overline{\text{ERROR}}$ и $\overline{\text{ACK}}$ инвертированы, остальные – нет.

Назначения регистров параллельного порта приведены в табл. 1.1.

Таблица 1.1. Назначения регистров параллельного порта

Группа данных	
Биты 0–7	Биты от 0 до 7, бит 0 – младший
Группа управления	
Бит 0 ($\overline{\text{STROBE}}$)	Строб-импульс данных. Данные фиксируются по низкому уровню сигнала
Бит 1 ($\overline{\text{LF/CR}}$)	Автоматический перевод строки. При низком уровне принтер, получив символ CR (Carriage Return – возврат каретки), автоматически выполняет и функцию LF (Line Feed – перевод строки)
Бит 2 ($\overline{\text{INITIALIZE}}$)	Инициализация принтера. При низком уровне восстанавливаются параметры, принятые по умолчанию; каретка возвращается к началу строки
Бит 3 ($\overline{\text{SLIN}}$)	Выбор принтера. При высоком уровне принтер нечувствителен к остальным сигналам интерфейса
Бит 4 (IRQ)	Разрешение аппаратных прерываний. При высоком уровне разрешается прерывание по спаду сигнала на линии ($\overline{\text{ACK}}$) – сигнала запроса следующего байта
Бит 5 (Data I/O)	Управление направлением передачи (для PS/2 и выше). Запись единицы переводит порт в режим ввода. При чтении состояние бита не определено
Биты 6, 7	Не используются
Группа состояния	
Биты 0–2	Не используются
Бит 3 ($\overline{\text{ERROR}}$)	Ошибка. Низкий уровень свидетельствует о том, что бумага закончилась, о состоянии off-line или о внутренней ошибке принтера
Бит 4 (SLCT)	Выбор принтера. Высокий уровень показывает, что принтер включен и готов к работе
Бит 5 (PE)	Нет бумаги. Высокий уровень означает, что бумага закончилась
Бит 6 ($\overline{\text{ACK}}$)	Импульс подтверждения приема байта (низким уровнем) и запрос на прием следующего. Может использоваться для формирования запроса на прерывание
Бит 7 ($\overline{\text{BUSY}}$)	Принтер занят. Прием данных возможен только при низком уровне сигнала

Базовые адреса портов LPT1 и LPT2 следующие:

LPT1: 956 (3BCh) или 888 (378h)
LPT2: 632 (278h)

Базовый адрес для LPT1 зависит от конфигурации оборудования компьютера. Существует два способа получения базового адреса: проверка конфигурации оборудования вашего компьютера или нахождение адреса непосредственно из пользовательских программ с помощью средств, предлагаемых *базовой системой ввода/вывода* (BIOS) компьютера. При включении или перезагрузке компьютера BIOS проверяет наличие параллельных портов. Если они обнаруживаются, их базовые адреса (двухбайтовые слова) помещаются в определенные ячейки памяти ОЗУ. Для LPT1 это ячейки 0000h:0408h и 0000h:0409h. Первая содержит младший, вторая – старший байт адреса. Базовый адрес LPT1 можно получить, считав содержимое этих ячеек. Ячейки памяти для портов LPT1 – LPT4 приведены ниже:

LPT1: 0000:0408h – 0000:0409h
LPT2: 0000:040Ah – 0000:040Bh
LPT3: 0000:040Ch – 0000:040Dh
LPT4: 0000:040Eh – 0000:040Fh

Кроме того, используется еще одна ячейка памяти: 0000:4011h. Она содержит сведения об общем количестве параллельных портов, установленных на компьютере. Эта информация хранится в битах 6 и 7:

бит 7 = 0, бит 6 = 0	параллельные порты не установлены
бит 7 = 0, бит 6 = 1	установлен один параллельный порт
бит 7 = 1, бит 6 = 0	установлено два параллельных порта
бит 7 = 1, бит 6 = 1	установлено три параллельных порта

1.1.3. Программное управление

В данном разделе приводится информация, необходимая для программирования параллельного порта, и даются начальные сведения по логическим операциям над битами.

Получение базового адреса параллельного порта

Следующая программа написана на языке QBASIC. Она выводит общее количество параллельных портов и их базовые адреса от LPT1 до LPT3. Строка 20 считывает байт, находящийся в ячейке памяти 0000:0411h, используя команду PEEK(). Биты 7 и 6 выделяются с помощью маски AND (128 + 64) . Затем результат сдвигается на 6 бит по направлению к младшему разряду с помощью деления на 64. Строка 30 считывает два байта из двух ячеек памяти, содержащих младший и старший байты базового адреса LPT1. Строки 40 и 50 делают то же самое для LPT2 и LPT3.

```
10 DEF SEG = 0
20 PRINT "Number of Centronic ports:", (PEEK(&H411) AND (128 + 64))/64
30 PRINT "Address of LPT1:", PEEK(&H408)+256*PEEK(&H409)
40 PRINT "Address of LPT2:", PEEK(&H40A)+256*PEEK(&H40B)
50 PRINT "Address of LPT3:", PEEK(&H40C)+256*PEEK(&H40D)
60 INPUT x
```

Следующая процедура на языке TP6 определяет количество установленных параллельных портов и присваивает полученное значение переменной Number_of_LPT. Затем она считывает их базовые адреса из ячеек памяти, где хранятся адреса портов LPT1 – LPT4. Далее программа предлагает указать тот LPT-порт, к которому будет присоединено внешнее устройство. И наконец, она присваивает выбранный базовый адрес переменной Centronic_address. В Turbo Pascal 6 для считывания содержимого ячеек памяти используются функции mem(основание: смещение) и memw(основание: смещение). Функция mem(…) считывает байт из ячейки памяти, а memw(…) – двухбайтовое слово из указанной ячейки памяти и одной ячейки выше.

```
(*-Библиотека ресурсов № A1 (определение базовых адресов LPT-портов)-.*)
Procedure Centronic_Address
(* $000:$0408 содержит базовый адрес для LPT1,
   $000:$040A содержит базовый адрес для LPT2,
   $000:$040C содержит базовый адрес для LPT3,
   $000:$040e содержит базовый адрес для LPT4,
   $000:$0411 содержит количество параллельных портов.*)
var
   lpt:array[1..4] of integer;
   number_of_lpt,LPT_number,code:integer;
   kbchar:char;
begin
      clrscr;
      LPT_number:=1;                          (*Для установки принтера по умолчанию.*)
      number_of_lpt:=mem($0000:$0411);        (*Считывает количество установленных
                                              параллельных портов.*)
      number_of_lpt:=(number_of_lpt and (128+64)) shr 6;        (*Манипуляции с битами.*)
      lpt[1]:=memw($0000:$0408);              (*Процедура считывания из памяти.*)
      lpt[2]:=memw($0000:$040A);
      lpt[3]:=memw($0000:$040C);
      lpt[4]:=memw($0000:$040e);
      textbackground(blue); clrscr;
      textcolor(yellow); textbackground(red); window(10,22,70,24); clrscr;
      writeln('Number of LPT installed:',number_of_lpt:2);
      writeln('Addresses for LPT1 to LPT4: ', lpt[1]:3,' ',lpt[2]:3,'  ',lpt[3]:3,' ',lpt[4]:3);
      write('Select LPT to be used(1,2,3,4): ');
      delay(1000);
      if number_of_lpt>1 then {Выбор порта, если установлено несколько портов.}
      begin
          repeat
              kbchar:=readkey;                (*Считывание значения нажатой клавиши.*)
              val(kbchar,LPT_number,code);    (*Преобразование символа в число.*)
          until (LPT_number>=1) and (LPT_number<=4) and (lpt[LPT_number]<>0);
      end;
      clrscr;
      P_address:=lpt[LPT_Number];
```

```
        writeln('Your selected printer interface: LPT',LPT_number:1);
        write('LPT address: ',P_address:3);
        delay(1000);
        textbackground(black); window(1,1,80,25); clrscr;
end;
```

Функция `centronic(x)` написана на языке Turbo Pascal для Windows. Она может быть вызвана программой, написанной на другом языке программирования для Windows, например Visual Basic или Visual C, если ее оформить в виде библиотеки динамической компоновки DLL. `Centronic(0)` возвращает количество установленных LPT-портов, `Centronic(1)` – базовый адрес LPT1, `Centronic(2)` – базовый адрес LPT2 и т.д.

```
Function Centronic(x:integer):integer; export;
(* $000:$0408 содержит базовый адрес для LPT1,
   $000:$040A содержит базовый адрес для LPT2,
   $000:$040C содержит базовый адрес для LPT3,
   $000:$040e содержит базовый адрес для LPT4,
   $000:$0411 содержит количество параллельных портов.*)
var
   number_of_LPT,LPT1,LPT2,LPT3,LPT4:integer;
   lpt1,lpt2,lpt3,lpt4: integer;
begin
   number_of_LPT:=mem($40:$11);     (*Считывает количество LPT-портов.*)
   number_of_LPT:=( number_of_LPT and (128+64)) shr 6;
   lpt1:=0; lpt2:=0; lpt3:=0; lpt4:=0;
   lpt1:=memw($40:$08);             (*Процедура считывания из памяти.*)
   lpt2:=memw($40:$0A);
   lpt3:=memw($40:$0C);
   lpt4:=memw($40:$0e);
   case x of
       0: centronic:=Number_of_LPT;
       1: Centronic:=lpt1;
       2: Centronic:=lpt2;
       3: Centronic:=lpt3;
       4: Centronic:=lpt4;
   end;
end;
```

Ввод/вывод данных через параллельный порт

Существует несколько способов записи информации в параллельный порт.

Команды принтера и процедуры прерываний BIOS

В QBASIC команда вывода на печать – `PRINT`, в TP6 – `writeln(lst)`. Другой способ управления принтером заключается в использовании прерывания BIOS – `INT 17h`. Временные диаграммы вывода данных через параллельный порт показаны на рис. 1.5.

Сначала компьютер проверяет, готов ли принтер к приему новых данных. Для этого необходимо проконтролировать состояние линии BUSY. Когда на ней низкий уровень («не занят»), ПК записывает данные в регистр данных. Через 500 нс компьютер переводит сигнал $\overline{\text{STROBE}}$ в низкий уровень, который, в свою очередь,

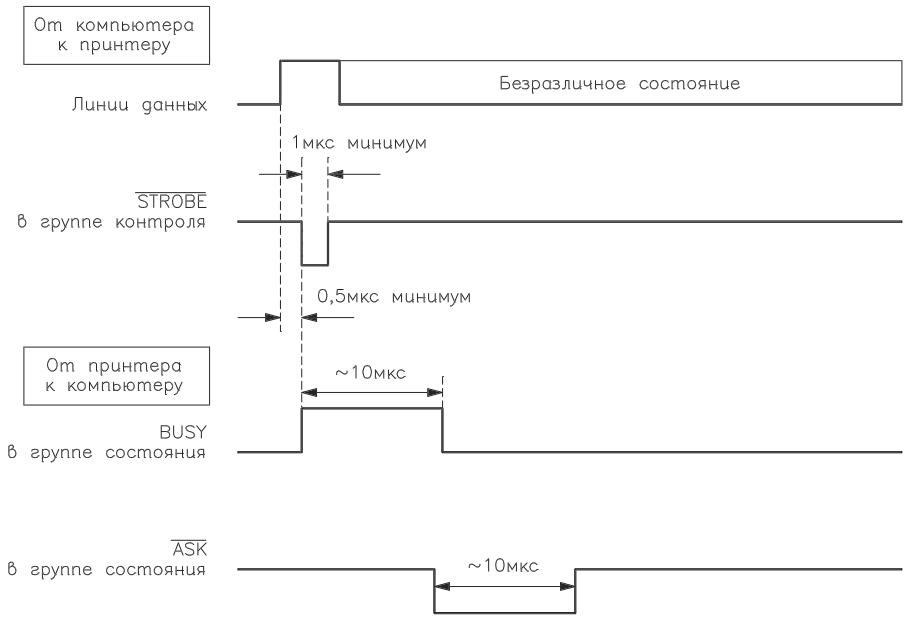


Рис. 1.5. Временные диаграммы взаимодействия компьютера с принтером

переводит принтер в состояние «занят» (BUSY=1). Принтер принимает и обрабатывает данные, а затем переводит сигнал $\overline{\text{ACK}}$ в низкий уровень для индикации, что принятые данные обработаны. В то же время принтер переводит линию BUSY в низкий уровень.

Практически в каждом языке программирования существуют инструкции по управлению принтером. Однако нужно учитывать, что этот метод недостаточно гибок для операций ввода/вывода при сопряжении ПК с внешними устройствами.

Если к компьютеру подсоединена внешняя схема, то в ней должна быть специальная схема для генерации сигналов BUSY и $\overline{\text{ACK}}$. Удобнее всего, если ПК при взаимодействии будет использовать только линии $\overline{\text{ACK}}$. BUSY постоянно соединяется с цифровой «землей» для индикации, что схема готова к приему данных, а PE показывает, что в принтере есть бумага; на линию $\overline{\text{ERROR}}$ подается высокий уровень. Если не соединить таким образом линии PE и $\overline{\text{ERROR}}$, при запросе компьютера на печать будут выдаваться сообщения об ошибке. Более гибкий способ управления параллельным портом – непосредственный доступ к регистрам порта.

Непосредственный доступ к регистрам порта

Это метод управления портом при помощи непосредственного доступа к его регистрам. Параллельный порт рассматривается как три отдельных регистра ввода/вывода, два из которых предназначены для вывода данных, а один – для ввода.

Рассмотрим пример управления портом LPT1. Так как адреса регистров данных, управления и состояния имеют номера 888, 890 и 889 соответственно, для

записи информации в регистры данных и управления применяются следующие команды языка QBASIC:

```
OUT 888, X
OUT 890, X
```

где X – записываемое значение в десятичном представлении. Некоторые линии порта управления инвертированы, что необходимо учитывать при выводе данных. Для чтения данных из регистра состояния можно пользоваться следующей командой:

```
Y=INP(889)
```

где Y – десятичное входное значение. Биты входных данных соответствуют номерам с 3 по 7 регистра состояния, некоторые из них инверсные.

Следующие процедуры на языке TP6 записывают информацию в регистры данных и управления. Обеим процедурам требуются базовый адрес выбранного параллельного порта и значение выходных данных. Выходные данные для регистра управления предусматривают проведение некоторых преобразований над битами.

```
(*-Библиотека ресурсов № A4 (запись информации в порт данных компьютера)-.*)
Procedure Write_data_port(P_address:integer, port_data:byte);
(*Биты регистра данных не инвертированы.*)
begin
  port(P_address):=port_data;  (*Ввод байта данных в регистр данных.*)
end;

(*-Библиотека ресурсов № A5 (запись данных в регистр управления.)-.*)
Procedure Write_control_port(P_address:integer; port_data:byte);
(*Биты 0, 1 и 3 инвертированы. Требуются преобразования над битами.*)
begin
  if port_data and 1=1 then port_data:=port_data and (255-1)
      else port_data:=port_data or 1;
  if port_data and 2=2 then port_data:=port_data and (255-2)
      else port_data:=port_data or 2;
  if port_data and 8=8 then port_data:=port_data and (255-8)
      else port_data:=port_data or 8;
  port(P_address+2):=port_data;    (*Ввод байта данных в регистр управления.*)
end;
```

Следующая функция на языке TP6 считывает биты с 3 по 6 из регистра состояния. Она требует базовый адрес выбранного параллельного порта. Функция также выполняет битовые преобразования и возвращает значение четырехбитовых входных данных.

```
(*-Библиотека ресурсов № A3 (считывание данных в компьютер).-*)
Function Read_status_port(P_address:integer):byte;
var
  byte1:byte;
begin
  byte1:=port(P_address+1);        (*Считывание байта из регистра состояния.*)
  byte1:=byte1 and 120;            (*01111000 (от старшего к младшему) and 0dddd... = 0dddd000.*)
  Read_status_port:=byte1 shr 3;   (*Сдвиг на 3 бита вправо, Read_status_port=0000hhhh.*)
end;
```

В следующих примерах, написанных на языке Turbo Pascal для Windows, представлен ввод и вывод данных через параллельный порт.

```
(*–Библиотека ресурсов № A4 (запись информации в регистр данных).–*)
Function Write_data_port(P_address:integer, port_data:integer); export;
(*Биты регистра данных не инвертированы.*)
begin
  port(P_address):=port_data       (*Ввод байта в регистр данных.*)
end;

(*–Библиотека ресурсов № A5 (запись данных в регистр управления).–*)
Function Write_control_port(P_address:integer; port_data:integer):integer; export;
(*Биты 0, 1 и 3 инвертированы. Требуются преобразования над битами.*)
begin
  if port_data and 1=1 then port_data:=port_data and (255-1)
      else port_data:=port_data or 1;
  if port_data and 2=2 then port_data:=port_data and (255-2)
      else port_data:=port_data or 2;
  if port_data and 8=8 then port_data:=port_data and (255-8)
      else port_data:=port_data or 8;
  port(P_address+2):=port_data;    (*Ввод байта данных в регистр управления.*)
end;

(*–Библиотека ресурсов № A3 (считывание данных в компьютер).–*)
Function Read_status_port(P_address:integer):integer; export;
var
  byte1:byte;
begin
  byte1:=port(P_address+1);        (*Считывание байта из регистра состояния.*)
  byte1:=byte1 and 120;            (*01111000 (от старшего к младшему) and 0dddd… = 0ddddOOO.*)
  Read_status_port:=byte1 shr 3;   (*Сдвиг на 3 бита вправо, Read_status_port=OOOOhhhh.*)
end;
```

Преобразования над битами

В данном разделе рассматриваются некоторые основные приемы битовых преобразований. Рассказывается о том, что такое вес бита, как сделать определенный бит единицей или нулем, описываются процедуры побитовых сдвигов.

Вес бита

Соотношения между битами и их весами приведены ниже:

бит 0	1 (десятичное значение)
бит 1	2
бит 2	4
бит 3	8
бит 4	16
бит 5	32
бит 6	64
бит 7	128

Присвоение биту единичного значения

Следующий пример демонстрирует, как сделать бит 3 (вес равен 8) регистра данных единицей (независимо от его исходного значения), оставив другие биты неизменными:

```
10 X=Original_data OR 8
20 OUT 888, X
```

Строка 10 выполняет операцию логического сложения (OR). Таблица истинности этой операции приведена ниже:

```
0 OR 0 = 0
0 OR 1 = 1
1 OR 0 = 1
1 OR 1 = 1
```

Пример поразрядной операции OR

```
Данные-1:                XXXXXXXX (биты от 7 до 0)
Данные-2:                00001000
Данные-1 OR Данные-2:    XXXX1XXX
```

Присвоение биту нулевого значения

Следующий пример на языке QBASIC показывает, как сделать бит 4 (вес равен 16) регистра данных нулевым:

```
10 X=Original_data and (255-16)
20 OUT 888, X
```

Строка 10 выполняет операцию логического умножения (AND). Таблица истинности этой операции выглядит следующим образом:

```
0 AND 0 = 0
0 AND 1 = 0
1 AND 0 = 0
1 AND 1 = 1
```

Пример поразрядной операции AND

```
Данные-1:                XXXXXXXX (биты от 7 до 0)
Данные-2:                11101111
Данные-1 AND Данные-2:   XXX0XXXX
```

Сдвиг битов вправо или влево

Уже говорилось, что при чтении данных из регистра состояния информативными являются только биты 3–6. Чтобы их выделить, необходимо произвести побитовый сдвиг. В TP6 для этих целей существует две процедуры: SHL – сдвиг битов влево (к старшему разряду) и SHR – сдвиг вправо (к младшему разряду). Следующий пример демонстрирует выполнение обеих операций:

```
Данные:      11111111 (биты с 7 по 0)
255 SHL 3:   11111000
255 SHR 3:   00011111
```

1.2. Последовательный интерфейс RS232

Последовательный интерфейс RS232 – это промышленный стандарт для последовательной двунаправленной асинхронной передачи данных. Он используется в компьютерах при подсоединении принтеров, модемов, мыши и т.д. Максимальное расстояние, позволяющее организовать связь, равно 20 м.

В отличие от параллельного порта, состоящего из восьми информационных линий и за один такт передающего байт, порт RS232 требует наличия только одной такой линии, по которой последовательно передается бит за битом. Это позволяет сократить количество информационных линий для передачи данных между устройствами, но уменьшает скорость.

1.2.1. Последовательная передача данных

Последовательный поток данных состоит из битов синхронизации и собственно битов данных. Формат последовательных данных содержит четыре части: стартовый бит, биты данных (5–8 бит), проверочный и стоповый биты; вся эта конструкция иногда называется *символом*. На рис. 1.6 изображен типичный формат последовательных данных.

Когда данные не передаются, на линии устанавливается уровень логической единицы. Это называется режимом ожидания. Начало режима передачи данных характеризуется передачей уровня логического нуля длительностью в одну элементарную посылку. Такой бит называется *стартовым*. Биты данных посылаются последовательно, причем младший бит – первым; всего их может быть от пяти

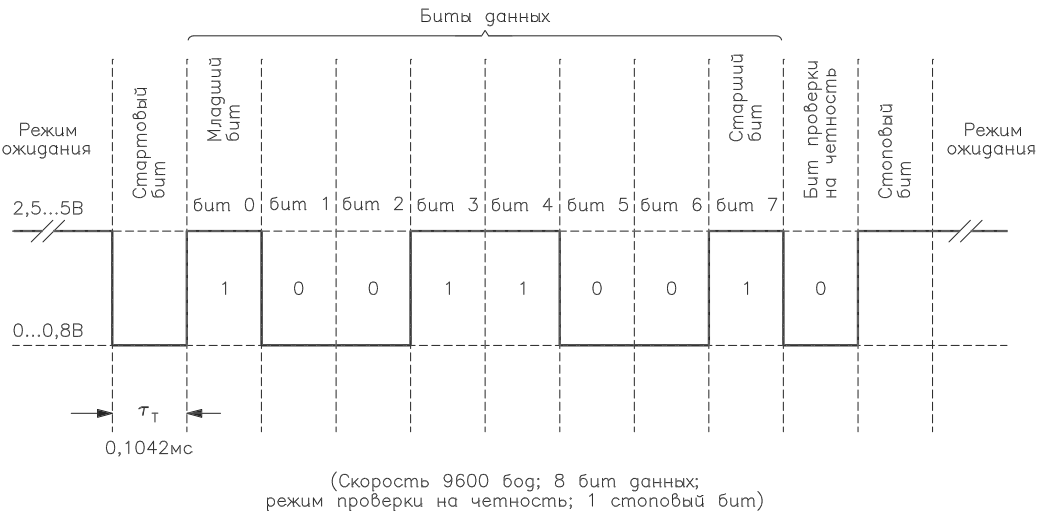


Рис. 1.6. Формат последовательных данных, формируемых UART