Интерфейсы периферийных устройств

Лабораторные работы 2005-2006 г.

Лабораторная работа №1 Работа с COM портом.

Организовать связь между двумя компьютерами с использованием COM порта персонального компьютера при помощи нуль-модемного кабеля.

Программы могут быть выполнены на языках среднего и высокого уровня с использований низкоуровневых подпрограмм при обращении к периферии. Задания 1.1 – 1.3 компилируются под операционные системы типа Windows 9*/ME или DOS, т.к. их выполнение подразумевает загрузку с системной дискеты или в среде виртуальной машины VMware.

1.1) Организовать связь по программному протоколу прямым программированием регистров порта.

Программно COM порт доступен как набор 12-ти регистров, расположенных в пространстве ввода-вывода ПК по 8-ми смежным адресам. Начальный адрес регистров порта обычно называется базовым (BASE) и адресация остальных регистров производится относительно него (напр. BASE+1).

Базовый адрес для порта COM1 = 3F8h, COM2 = 2F8h.

Для достижения цели необходимо:

1) Произвести инициализацию УАПП, т.е.

а) Установить скорость обмена :

Перевести регистры BASE и BASE+1 в режим делителя чатоты тактового генератора записью управляющего байта в регистр контроля линии (BASE+3).

Записать старший и младший байты кода делителя частоты в соответствии с выбранной скоростью обмена (рекомендуется 1200, 4800 бод).

б) Записать управляющий байт в регистр контроля линии:

перевести регистры BASE и BASE+1 в режимы приема и передачи данных и разрешения прерываний соответственно

настроить канал для обмена, рекомендуются следующие параметры:

- количество бит данных – 8
- количество стоп-бит – 1
- контроль на четность с постоянным значением контрольного бита

в) Запретить прерывания

2) Написать приемные и передающие процедуры.

Каждая процедура обязательно должна включать анализ регистра состояния (статуса) линии (УАППа) (BASE+5).

1.2) Организовать связь по программному протоколу используя функции BIOS.

Для работы с COM - портом предназначено 14-е прерывание BIOSа.

Для достижения цели необходимо:

1) Произвести инициализацию УАПП, т.е.

а) Выбрать требуемый порт, задать скорость и параметры обмена (кол-во бит данных, стоп бит, наличие бита паритета) – функция 0 прерывания 14.

2) Написать приемные и передающие процедуры.

Для приема-передачи данных используются 1 и 2я функции 14-го прерывания.

1.3) Организовать связь с использованием аппаратных прерываний.

Коммуникационный адаптер может вызывать специальное аппаратное прерывание, условия генерации которого задаются регистром порта EIR (регистр разрешения прерываний – BASE+3). Возможные причины указаны ниже:

- прерывание по статусу модема;

- … по обрыву или ошибке линии;

- … завершение передачи;

- … приему символа;

- … по тайм-ауту (в режиме FIFO).

Причина указывается в регистре идентификации прерываний- IIR.

Программа должна обеспечить следующее:

1) Установить собственный обработчик прерывания

а) Запретить маскируемые прерывания;

б) Запретить аппаратные прерывания нужного IRQ (4 или 11 для COM1, COM3; 3 или 10 для COM2, COM4);

в) Заменить вектор прерывания своим обработчиком;

г) разрешить аппаратные прерывания;

д) разрешить маскируемые прерывания.

2) Задать конфигурацию порта.

3) Написать приемные и передающие процедуры для собственного обработчика прерывания.

1.4) Организовать связь по программному протоколу при помощи функций Win API.

В операционных системах Windows последовательные порты имеют имена "COM*", где * - номер порта (например COM1), и относятся к так называемым коммуникационным ресурсам. Для работы с портом на программном уровне используются следующие функции программного интерфейса: CreateFile, SetCommState, SetCommTimeouts, WriteFile, ReadFile, TransmitCommChar, CloseHandle.

Последовательность программирования следующая:

1) Открывается выбранный COM порт, как коммуникационный ресурс – функция CreateFile.
2) Произвести инициализацию порта:
- задаются интервалы ожидания по чтению/записи – SetCommTimeouts (если этого не сделать, сохранятся настройки выставленные системой «по умолчанию»).
- задается скорость передачи, количество бит данных, стоп бит, наличие бита паритета – SetCommState (заполняется структура DCB).
3) Написать процедуры приема/передачи данных – ReadFile, WriteFile/TransmitCommChar.
4) Для корректного завершения программы по окончании ее работы COM порт закрывается – CloseHandle (иначе судьба порта зависит от компетенции вашего компилятора и терпения операционной системы).

Лабораторная работа №2 Работа с шиной PCI на низком уровне.

Программы данной работы также могут быть выполнены на языках среднего и высокого уровня с использований низкоуровневых подпрограмм при обращении к периферии. Программы компилируются под операционные системы типа Windows 9*/ME или DOS, т.к. их выполнение подразумевает загрузку с системной дискеты или в среде виртуальной машины VMware.

2.1) Получить сведения о шине PCI и устройствах на ней присутствующих.

Для работы с PCI существуют специализированные функции PCI BIOS, предназначенные для доступа к конфигурационному пространству и генерации специальных циклов шины.

Для упрощения работы, при написании программ, рекомендуется пользоваться 16-тиразрядным реальным или защищенным режимом работы процессора – функция 0B1h прерывания 1Ah. (Лицам сумевшим сделать требуемое в 32 – разрядном режиме особый почет).

Программа должна выполнить следующее:

1) Провести проверку наличия PCI BIOS в системе, в случае успеха вывести данные о шине (поддерживаемый аппаратный механизм, версия интерфейса, номер последней шины в системе).
2) Если пункт 1) завершился неудачей, дальше делать нечего, в противном случае необходимо вывести список всех типов устройств, присутствующих на шине с указанием количества устройств каждого типа.
3) Определить тип IDE контроллера, если он присутствует.
4) Вывести следующие данные произвольно заданного устройства: его номер, номер функции, номер шины подключения. Произвести чтение конфигурационного пространства устройства. Вывести идентификационный номер устройства и производителя, прерывание, закрепленное за устройством.

2.2) Работа с шиной PCI прямым программированием аппаратных регистров.

Для доступа к шине PCI используются адреса 0xCF8—0xCFF. За определенными адресами закреплены аппаратные регистры, позволяющие получить доступ к разнообразным устройствам на шине PCI:

Регистр конфигурации адреса – 0xCF8 – 32-разрядный регистр, доступный для чтения и записи. Позволяет определить конфигурационные параметры устройства на шине, и номер регистра для последующего доступа к устройству. Формат регистра: 0-1-резерв; 2-7-определяет значение регистра для доступа к устройству; 8-10-номер ф-ии для многофункционального устройства; 11-15-номер устройства на шине; 16-23 номер шины PCI; 24-30 резерв; 31- 1-разрешает доступ к конфигурационному пространству шины, 0- доступ блокирован.

Регистр конфигурации данных – 0xCFC -32-разрядный регистр доступен для чтения и записи, предназначен для чтения и записи данных в конфигурационное пространство шины PCI (для обмена данными используются все биты регистра)

Программа должна выполнить следующее:

1) Вывести список устройств, присутствующих на шине (допускается список кодов устройств).
2) Прочитать и вывести область конфигурационного пространства любого устройства из присутствующих. Вывести имя производителя, адреса базовых портов и номер прерывания любого из выбранных устройств.

Лабораторная работа №3 Работа с USB.

Программы данной работы также могут быть выполнены на языках среднего и высокого уровня с использований низкоуровневых подпрограмм при обращении к периферии. Программы компилируются под операционные системы типа Windows 9*/ME или DOS, т.к. их выполнение подразумевает загрузку с системной дискеты или в среде виртуальной машины VMware.

3.1) Монитор состояния шины

Драйвер интерфейса USB управляет работой хост-контроллера через регистры. Регистры универсального хост-контроллера принято разделять на две группы; группу конфигурационных регистров PC1 (USB PCI Configuration Registers) и группу регистров пространства ввода-вывода (USB Host Controller IO Space Registers). Ниже мы будем рассматривать только регистры ввода-вывода, так как непосредственная работа с конфигурационными регистрами из прикладных программ нежелательна.

Для описания режима доступа к данным в регистрах USB используются следующие стандартные обозначения;
- **RO — возможно только считывание данных;**
- **WO — возможна только запись данных;**
- **R/W — разрешено выполнение как записи, так и считывания данных;**
- **R/WC — разрешено считывание данных и сброс отдельных разрядов регистра (запись единицы в некоторый разряд регистра приводит к тому, что этот разряд сбрасывается в ноль).**

Список регистров ввода-вывода хост-контроллера шины USB приведен в табл. 8.2. Доступ к этим регистрам осуществляется через группу портов ввода/вывода, базовый адрес которой задан в конфигурационном регистре USBBA.

Регистры ввода-вывода универсального хост-контроллера шины USB

Смещение	Размер	Доступ	Мнемоника	Наименование регистра
00h	WORD	R/W	USBCMD	Регистр команды USB
02h	WORD	R/WC	USBSTS	Регистр состояния USB
04h	WORD	H/W	USBINTR	Регистр управления прерываниями USB
06h	WORD	R/W	FRNUM	Регистр номера кадра USB
08h	DWORD	R/W	FLBASEADD	Регистр базового адреса списка кадров USB
0Ch	BYTE	R/W	SOFTMOD	Регистр модификатора начала кадра USB
10h	WORD	R/WC	PORTSCO	Регистр состояния и управления порта 0
12п	WORD	R/WC	PORTSC1	Регистр состояния и управления порта 1

Программа должна выполнить следующее:

1) Найти все контроллеры USB типа Universal Host Controller. Вывести параметры их подключения к шине PCI.
2) Отображать в реальном времени состояние регистров ввода-вывода хост-контроллеров ПК.

P.S. Количество заданий будет постепенно увеличиваться, пока самый передовой студент группы не выполнит на 0,5 заданий меньше числа предложенных.

Литература

1. Агуров П.В. Последовательные интерфейсы ПК. Практика программирования – СПб.: БХВ-Петербург, 2004.-496с.
2. Кулаков В. Программирование на аппаратном уровне: специальный справочник. – СПб.: Питер, 2003.-848с.
3. Нежинский В. Программирование аппаратных средств в Windows – СПб.: БХВ-Петербург, 2004.-880с.

Приложение. Примеры реализации программ.

1а) Пример прямого программирования порта

```
\ программа написана на языке Форт с использованием компилятора
\ Small32 Александра Ларионова
\ смотри http:\\www.forth.org.ru

\ задаем константы, определяющие адреса портов для COM2
$2f8 constant dataport          \ базовый - для записи данных и младшего байта
                                        \ кода делителя частоты
$2f9 constant irport            \ регистр прерываний и старший байт делителя частоты
$2fb constant manager   \ регистр управления
$2fd constant statline          \ регистр статуса

\ управляющие слова
$bb constant upr1               \ задаются параметры передачи; 7 бит регистра управления
                                        \ устанавливается в 1
$3b constant upr2       \ сбрасывается  7 бит регистра управления
$0c constant freq               \ код делителя частоты (9600 бод)

\ определяются переменные
variable indata\ символ на передачу
variable outdata \ принятый символ

\ делаем переменные и константы доступными для вызова из ассемблерных
\ определений
>public error indata outdata

>public dataport irport idport manager statline upr1 upr2 freq

\ _____________________________
\ основные низкоуровневые слова
\ _____________________________

\ ______слово инициализирующее порт COM2
code comminit
                mov eax,upr1    ; загружаем первое управляющее слово,устанавливая 0 и
                                        ; 1 регистры в режим приема кода делителя
частоты
                mov edx,manager ;
                out dx,al       ; передаем управляющее слово в регистр управления
; frequense  set
                mov eax,freq    ; читаем код делителя частоты
                mov edx,dataport ;
                out dx,al       ; загружаем младший байт делителя в регистр данных
                mov al,ah       ;
                mov edx,irport  ;
                out dx,al       ; загружаем старший байт делителя в регистр прерываний
; step2             ;
                mov eax,upr2    ; загружаем второе управляющее слово в регистр
                                        ; управления, 0 и 1 регистры COM2
принимают свои
                                        ; рабочие функции
                mov edx, manager ;
                out dx, al      ;
; not interrut          ;
                mov al,0        ;  запрещаем прерывания
                mov edx,irport  ;
                out dx,al       ;
next
endcode

\ ______слово для передачи символа в COM2
code commwrite
                mov eax,[outdata]  ; читаем символ из переменной
                mov edx,dataport ;
                out dx,al       ; передаем его в регистр данных
waitout:
                mov edx,statline ;
                in al,dx   ; читаем регистр статуса
                mov ah,al  ;
                and al,64  ; // 01000000b проверяем, пуст ли буфер передатчика
                jz waitout ; ждем
                mov al,ah  ;
                and al,32  ; //00100000b проверяем, пуст ли регистр передатчика
                jz waitout ; если нет, ждем, да выходим из слова
next
endcode

\ ______слово для чтения символа из COM2
code commread
                mov edx,statline ;
waitread:
                mov edx,statline
                in al,dx      ; читаем регистр статуса
                mov bl,al     ;  проверяем пришли ли данные
                and eax,1     ;
                jz waitread ;   если не пришли, ждем дальше
                mov edx,dataport ; если пришли читаем регистр данных
                xor eax,eax
                in al,dx
; error1:
                mov [indata],eax ; загружем принятый байт в переменную
next
endcode

: writecomm
\ данное слово организует цикл предачи символов,принимаемых с
\ клавиатуры, в COM2

\ ждем ввода символа,выводим его на экран,записываем его в переменную
key dup emit outdata !
\ передаем символ в COM2,проверяем: если переданный символ Esc выходим из
цикла
commwrite outdata @ $1b = if exit then
\ если нет возват к началу слова (организуем цикл)
recurse
;

: readcomm
\ данное слово организует цикл приема символов с COM2 и выводит их на экран

\ читаем COM2
commread
\ проверяем принятый символ: если Ecs - выход из цикла, если нет вывод его
\ на экран
indata @ dup $1b = if exit then emit
\ возврет к началу слова (цикл)
recurse
;

: choose        \ слово выбора режима работы
\ вывод приглашения
cr ." Choose mode: type W to WRITE symbol, type R to READ, X to EXIT :"
\ ждем ввода символа с клавиатуры, выводим его на экран, анализируем
key dup emit
case

                set 87 119 \ код символов W и w (запись)
                of cr ." You choose WRITE. Insert symbol please:"
                writecomm
                cr ." symbols left"
                endof

                set 82 114 \ код символов R и r (чтение)
                of
                cr ." You choose is READ. Wait please"
                cr ." symbols came: " cr
                readcomm
                endof

                set 88 120 \ код символов X и x (выход)
                of
                cr cr ." Programm end work. PRESS any key" key drop
                bye
                endof
endcase
recurse
;

\ main - главное слово в программе, инициирует COM2,вызывает слово выбора
режима
\ работы

: main
comminit \ инициализация порта
cr cr
." Programm for COM2 connection" \ печитаем заголовок
cr ."           was made on Small32 by Kalachev Alexandr. xx.10.2001"
cr cr choose \ вызываем слово выбора режима работы
;

\ создаем EXE файл, выполняющий слово main при запуске
build Comrw commmap
```

1б) Пример программирования порта при помощи функций BIOSa

```
\ программа написана на языке Форт с использованием 16-разрядного компилятора
\ sp-forth Андрея Черезова, версия 2.5.12
\ смотри http:\\www.forth.org.ru

HEX

CODE AT-XY ( x y -- ) \ 93 FACILITY
( установить аппаратный курсор )
   AX, 2 [BP] MOV
   DX, [BP] MOV
   BP, # 4 ADD
   DH, DL MOV
   DL, AL MOV
   BX, BX XOR
   AX, # 200 MOV
   BP  PUSH
   10  INT
   BP  POP
   RET
END-CODE

CODE KEY_PRESSED? ( -- char|ior )
\ проверка нажатия клавиши
  BP, # 2 SUB
  AH, # 1 MOV
```

```
  16 INT
  1$ JNZ
  [BP], # 0 MOV
  2$ JMP
1$:
  AH, # 0 MOV
  16 INT
  [BP], AL MOV
  AH, # E MOV
  BL, # 0 MOV
  10 INT
2$:
RET
END-CODE


VARIABLE PORT

CODE COM_INIT ( code -- ior )
\ инициализация порта
  DX, PORT MOV
  AL, [BP] MOV
  AH, # 0 MOV
  14 INT
  BP, # 2 ADD
  [BP], AH MOV
RET
END-CODE

CODE COM_OUT ( byte -- )
\ вывод байта
  DX, PORT MOV
  AL, [BP] MOV
  AH, # 1 MOV
  14 INT
  BP, # 2 ADD
\  [BP], AH MOV
RET
END-CODE

CODE COM_IN ( -- byte ior )
\ прием байта
  DX, PORT MOV
  AH, # 2 MOV
  14 INT
  BP, # 2 SUB
  [BP], AL MOV
  BP, # 2 SUB
  [BP], AH MOV
RET
END-CODE

CODE COM_STATE ( -- ior )
\ получение регистра статуса порта
  DX, PORT MOV
  AH, # 3 MOV
  14 INT
  BP, # 2 SUB
  [BP], AH MOV
RET
END-CODE

: data_ready? ( -- ior )
\ пришли ли данные?
COM_STATE 1 AND
;
: shift_reg_empty? ( -- ior )
\ регистр передатчика пуст?
COM_STATE 40 AND
;

\ "экранные" переменные
VARIABLE Xout VARIABLE Yout \ переменные на вывод
VARIABLE Xin VARIABLE Yin \ переменнаые вывода на экран
15 Yout !

: out_cursor Xout @ Yout @ AT-XY \ курсор на вывод
;

: input_cursor Xin @ Yin @ AT-XY Xin 1+! \ координаты входящих символов
;

: in_cr D = IF Yin 1+!  0 Xin ! THEN ; \ перевод строки

: out_cr D = IF Yout 1+!  0 Xout ! ELSE Xout 1+! THEN ; \ --//--

: COM_SELECT
\ выбор порта
." select COM: 0 - COM1, else - COM2" CR
KEY 30 = IF 0 PORT ! ELSE 1 PORT !  THEN
." selected" CR
83  COM_INIT
 ." init" CR
 Xout @ Yout @ AT-XY
20 0 ?DO CR LOOP \ очистка экрана
;

: COM_TEST
 COM_SELECT
 BEGIN
   data_ready? IF COM_IN   input_cursor \ Xin @ Yin @ AT-XY
    IF  ." error byte:"
    THEN
   DUP EMIT in_cr
  THEN
  out_cursor
  KEY_PRESSED? FF AND DUP
   IF DUP COM_OUT DUP out_cr
   THEN
  1B = IF 0 -1 ." I'll be back" ELSE 0 THEN
 UNTIL
;

//вторая версия программы с использованием функций BIOS
//Рабочая версия!! она работает!!
//так и не получилось запустить ее по правильному,
//вообщем проблема в том, что после приема наглухо сбрасывается признак
//приема и видимо навсегда. поэтому пришлось отправлять нуль а во время приема
//проверять не нуль ли пришел. вот.
//Псарюк С. гр 501

#include <bios.h>
#include <conio.h>

#define  SETTINGS  (_COM_9600  |  _COM_CHR8  |  _COM_STOP1  |
_COM_NOPARITY)
```

1б) Еще один пример программирования порта при помощи функций BIOSa

```
//вторая версия программы с использованием функций BIOS
//Рабочая версия!! она работает!!
//так и не получилось запустить ее по правильному,
//вообщем проблема в том, что после приема наглухо сбрасывается признак
//приема и видимо навсегда. поэтому пришлось отправлять нуль а во время приема
//проверять не нуль ли пришел. вот.
//Псарюк С. гр 501

#include <bios.h>
#include <conio.h>

#define  SETTINGS  (_COM_9600  |  _COM_CHR8  |  _COM_STOP1  |
_COM_NOPARITY)

void main (void)
{
 unsigned int ch;
 int i;


 _bios_serialcom(_COM_INIT,1, SETTINGS);

 clrscr();
 for (;;)
 {
  if (kbhit())
  {
     if ((ch = getch()) == '\x1B')   break;
     _bios_serialcom(_COM_SEND,1,ch);
     textcolor(4);
     cprintf("%c",ch);
  }
  else
   {
    if (_bios_serialcom(_COM_STATUS,1,0) & 0x100)
    {
                ch = _bios_serialcom(_COM_RECEIVE,1,0);
                _bios_serialcom(_COM_SEND,1,0);
    if(ch!=0)
     {
                textcolor(25);
                cprintf("%c",ch);
     }
    }
   }
 }
}
```

1в) Пример программирования порта при помощи аппаратных прерываний

```
// INTERRUPT
#include <stdio.h>
#include <conio.h>
#include <iostream.h>
#include <dos.h>
                // прототипы функций
void init_Serial_Port(void);
void transmitter (unsigned char);
void interrupt far receiver(...);

void main (void)
{
                 int sym;
_disable();
                 init_Serial_Port();

                 asm{  in al,0x21;

                                          and al,0xE7;

                                          out 0x21,al;

                 };
_dos_setvect(0x0B,receiver);  // установка вектора на программу обработки
прерывания
```

```
_enable();

clrscr();
while(1)
{
                if(kbhit()!=0)
                                { sym=getche();
                                                if(sym=='1') return;
                                                else transmitter(sym);
                                                }

}
}

void init_Serial_Port(void)
{
                asm { mov al,0x80;
                                mov dx,0x2FB;
                                out dx,al;   // 7_bit=1
                                mov al,0x0C;
                                mov dx,0x2F8;
                                out dx,al;   // 9600 bit/s
                                mov al,0x00;
                                mov dx,0x2F9;
                                out dx,al;         // 9600 bit/s
                                mov al,0x3B;
                                mov dx,0x2FB;
                                out dx,al;   // data 8 bit, stop_bit 1, paritet=0
                                mov dx,0x2FC;
                                in al,dx;
                                or al,0x0B;
                                out dx,al;   // setb: DTR,RTS,MCR.3
                                mov al,0x01;
                                mov dx,0x2F9;
                                out dx,al;   // interrupt on!
                };
}

void transmitter (unsigned char symbol)
{
unsigned char flag=0;
while(flag==0)
 {
                asm { mov dx,0x2FD; // 5_bit=1?
                                in al,dx;
                                and al,0x20;  // mask
                                mov flag,al;
                };
 }
                asm { mov al,symbol;
                                mov dx,0x2F8;
                                out dx,al;
                };
}

void interrupt far receiver(...)
{
_disable();
 char sym;
                                asm {   mov dx,0x2F8;
                                                in al,dx;
                                                mov sym,al
                                };
cout<<sym;
                                asm {   mov dx,0x20;
                                                mov al,0x20;
                                                out dx,al;
                                };
_enable();

};
```

1г) Пример реализации функций для работы с COM портом под ОС Windows.

```
\ программа написана на языке Форт с использованием 32-х разрядной системы
\ sp-forth Андрея Черезова, версия 4.007
\ смотри http:\\www.forth.org.ru

\  работа с  последовательным портом под Виндовс

\ константы Виндовс
-2147483648 CONSTANT GENERIC_READ
1073741824 CONSTANT GENERIC_WRITE

\ внутренние переменные
0 VALUE handle
VARIABLE WasRead
VARIABLE ReadBuffer
\ ф-ии Виндовс

WINAPI: TransmitCommChar KERNEL32.DLL \ ( char handle -- ior )
WINAPI: SetCommTimeouts KERNEL32.DLL
WINAPI: GetCommTimeouts KERNEL32.DLL
WINAPI: BuildCommDCBA KERNEL32.DLL
WINAPI: SetCommState KERNEL32.DLL
WINAPI: GetCommState KERNEL32.DLL
WINAPI: WaitCommEvent KERNEL32.DLL
WINAPI: SetCommMask KERNEL32.DLL

0
CELL -- Internal
CELL -- InternalHigh
CELL -- Offset
CELL -- OffsetHigh
CELL -- hEvent
CONSTANT OVELAPPED
HERE DUP >R OVELAPPED DUP ALLOT ERASE VALUE overlap
VARIABLE Event

0
CELL -- ReadIntervalTimeout
CELL -- ReadTotalTimeoutMultiplier
CELL -- ReadTotalTimeoutConstant
CELL -- WriteTotalTimeoutMultiplier
CELL -- WriteTotalTimeoutConstant
CONSTANT COMMTIMEOUTS

HERE DUP COMMTIMEOUTS DUP ALLOT ERASE VALUE CommTimeouts

0
CELL -- DCBlength          \ sizeof(DCB)
CELL -- BaudRate           \ current baud rate
CELL -- Mode
(   биты слова Mode
   ^-- fBinary:1           \ 1 - binary mode, no EOF check
   ^-- fParity:1           \ 0 - enable parity checking
   ^-- fOutxCtsFlow:1      \ 0 - CTS output flow control
   ^-- fOutxDsrFlow:1      \ 0 - DSR output flow control
   ^-- fDtrControl:2       \ 00 - DTR flow control type
   ^-- fDsrSensitivity:1   \ 0 - DSR sensitivity
   ^-- fTXContinueOnXoff:1 \ 0 - XOFF continues Tx
   ^-- fOutX:1             \ 0 - XON/XOFF out flow control
   ^-- fInX:1              \ 0 - XON/XOFF in flow control
   ^-- fErrorChar:1        \ 0 - enable error replacement
   ^-- fNull:1             \ 0 - enable null stripping
   ^-- fRtsControl:2       \ 00 - RTS flow control
   ^-- fAbortOnError:1     \ 0 - abort reads/writes on error
   ^-- fDummy2:17          \ reserved
)
2 -- wReserved             \ not currently used
2 -- XonLim                \ transmit XON threshold
2 -- XoffLim               \ transmit XOFF threshold
1 -- ByteSize              \ number of bits/byte, 4-8
1 -- Parity                \ 0-4=no,odd,even,mark,space
1 -- StopBits              \ 0,1,2 = 1, 1.5, 2
1 -- XonChar               \ Tx and Rx XON character
1 -- XoffChar              \ Tx and Rx XOFF character
1 -- ErrorChar             \ error replacement character
1 -- EofChar               \ end of input character
1 -- EvtChar               \ received event character
2 -- wReserved1            \ reserved; do not use
CONSTANT DCB

HERE DUP >R DCB DUP ALLOT ERASE VALUE MyDCB

: WaitComm overlap Event handle WaitCommEvent DROP Event @ ." Event" . ;

: CommOpen ( addr n -- -1|handle ) \ for example string "com1" or "com2"
DROP >R
0 0 OPEN_EXISTING 0 0 GENERIC_READ GENERIC_WRITE OR R> CreateFileA
DUP -1 = IF DROP 0 ELSE TO handle -1 THEN
;
: CommClose ( -- ior )
handle CloseHandle
;
\ передача символа в открытый порт
: CommOut ( char -- ) \ worked maybe :-)
handle TransmitCommChar DROP ;

\ прием символа из порта
: CommIn ( -- char ) \ WaitComm
0 WasRead 1 ReadBuffer handle ReadFile DROP ReadBuffer C@ ;

\ прием нескольких байт в выбранный буфер
: ReadFromComm' ( addr n -- )
SP@ 0 \ ( addr n &n 0 ) -> ( 0 &N N ADDR )
 SWAP 2SWAP SWAP
 handle ReadFile ." [" . ." ]"
;
: ReadFromComm ( addr n -- )
OVER + SWAP ?DO
CommIn I C!
LOOP
;
: WriteToComm ( addr n -- )
OVER + SWAP ?DO
I C@ CommOut
LOOP
;
: Timeouts ( ms -- flag ) \ установка интервалов ожидания для чтения/записи в порт
10 CommTimeouts ReadIntervalTimeout !
1 CommTimeouts ReadTotalTimeoutMultiplier !
 CommTimeouts ReadTotalTimeoutConstant !
100 CommTimeouts WriteTotalTimeoutMultiplier !
1 CommTimeouts WriteTotalTimeoutConstant !
CommTimeouts handle SetCommTimeouts
;
: CommInit ( -- ior ) \ первоначальная инициализация порта
DCB MyDCB DCBlength !
MyDCB handle GetCommState DROP
9600 MyDCB BaudRate !
0x80000000 MyDCB Mode !
```

```
8 MyDCB ByteSize C!
2 MyDCB StopBits C!
1 MyDCB Parity C!
MyDCB handle SetCommState
;
: SetComm ( BaudRate ByteSize StopBits Parity -- ior )
\ настройка порта
 MyDCB Parity C!
 MyDCB StopBits C!
 MyDCB ByteSize C!
 MyDCB BaudRate !
 MyDCB handle SetCommState 0 <>
;
\ choose port
: COM1 ( -- flag ) S" COM1" CommOpen DUP IF CommInit 1000 Timeouts 2DROP THEN
;
: COM2 ( -- flag ) S" COM2" CommOpen DUP IF CommInit 1000 Timeouts 2DROP THEN
;

\ тестовые слова
: TEST ( n -- ) 0 ?DO I CommOut LOOP ;
```

2a) Пример реализации функций для работы с шиной PCI

```
\ программа написана на языке Форт с использованием 16-разрядного компилятора
\ sp-forth Андрея Черезова, версия 2.5.12
\ смотри http:\\www.forth.org.ru


HEX

CODE TEST
  BP, SP XCHG
  EDX, EDX XOR
  DX, # 80FF MOV
  CX, # 4 MOV
  EDX, CL SHL
  EDX PUSH
  AX, # 7 MOV
  AX PUSH
  BP, SP XCHG
RET
END-CODE

: --     \ создание структур
CREATE OVER , +
DOES> @ +
;

CODE PCI? ( -- last_bus № ap_mech _I CP  0|1  ) \ проверка наличия PCI BIOS в системе
  BP, SP XCHG
  AX, # B101 MOV
  1A INT
  CX PUSH \ номер последней шины
  BX PUSH \ номер версии PCI

  CX, CX XOR
  CL, AL MOV
  CX PUSH \ аппаратный механизм
  EDX PUSH \ сигнатура "PCI "
  AL, AH MOV
  AX PUSH \ есть ли в системе (если 0 - да)
  BP, SP XCHG
RET
END-CODE

0 VALUE PCI

: pci_info \ вывод информации о шине PCI
 PCI?
 IF
   DROP 2DROP 2DROP
  ." нет pci в системе"
 ELSE
  4350 = SWAP 2049 = AND IF \ проверка сигнатуры "PCI "
    \ если все в порядке вывод информации о шине
    ." аппаратный механизм " . CR
    ." номер версии PCI " . CR
    ." номер последней шины " . CR
    -1 TO PCI \ TRUE в переменную,
  ELSE
    DROP 2DROP
    ." ошибка в сигнатуре"
  THEN
 THEN
;

CODE FIND_PCI_CLASS ( base_class subclass_interface number -- number ior )
\ поиск устройства заданного класса
  SI PUSH
  BP, SP XCHG
  SI POP
  ECX POP
  AX, # B103 MOV
  1A INT
  BX PUSH \ BH-номер шины BL-биты [7-3]-номер устройства, [2-0]- н-р функции
  CX, CX XOR
  CL, AH MOV
  CX PUSH \ код возврата (0 - удача, 0x86 устройство не найдено)
  BP, SP XCHG
  SI POP
RET
END-CODE

\ создаем структуру конфигурационного пространства PCI
0
  2 -- .VendorID
  2 -- .DeviceID
  2 -- .Code
  2 -- .Status
  1 -- .RevisionID
  3 -- .ClassCode
  1 -- .CasheLineSize
  1 -- .LatencyTimer
  1 -- .HeaderType
  1 -- .BIST
  16 -- .BaseAddrReg
  4 -- .CardBusCISPointer
  2 -- .SubsysVendorID
  2 -- .SubsysID
  4 -- .ExpanationROMBase
  4 -- .Rev1
  4 -- .Rev2
  1 -- .IntLine
  1 -- .IntPin
  1 -- .MinGNT
  1 -- .Max_iat
  8 -- .DeviceSpec
CONSTANT /pci_config

CODE READ_PCI_BYTE ( PCInumber byte_number -- byte ior )
\ читаем байт конфигурационного пространства устройства
  DI PUSH
  BP, SP XCHG
  DI POP
  BX POP
  AX, # B108 MOV
  1A INT
  CX, 00FF AND
  CX PUSH
  BX, BX XOR
  BL, AH MOV
  BX PUSH \ код возврата (0 - удача, 0x87 некорректный индекс)
  BP, SP XCHG
  DI POP
RET
END-CODE

\ читаем N слов конфигурационного пространства устройства PCInumber
\ запоминаем в addr
: read_pci_config ( PCInumber addr N -- )
  0 ?DO ( PCInumber addr -- )
    OVER I READ_PCI_BYTE ( PCInumber addr byte ior -- )
    IF
      DROP 2DROP ." неверный индекс" CR EXIT
    ELSE
      ( PCInumber addr byte -- )
      OVER I + C!
    THEN
  LOOP
  2DROP
;

CODE WRITE_PCI_BYTE ( PCInumber byte byte_number -- ior )
  DI PUSH
  BP, SP XCHG
  DI POP
  CX POP
  BX POP
  AX, # B10B MOV
  1A INT

  BX, BX XOR
  BL, AH MOV
  BX PUSH \ код возврата (0 - удача, 0x87 некорректный индекс)
  BP, SP XCHG
  DI POP
RET
END-CODE

\ S" pci_list.f" INCLUDED
(
Поиск устройства PCI по коду класса
Поиск устройства определенного типа можно осуществлять по коду класса.
Код класса состоит из трех байтов:
  старший байт задает базовый класс - BaseClass,
  средний байт - подкласс - SubClass,
  младший байт - интерфейс - Interface

23_____16 15_____8 7_____0
Базовый класс   Подкласс   Интерфейс

Рис. 3.3. Общая структура поля кода класса

  В таблице 3.3 приведен перечень базовых классов устройств PCI,


Таблица 3.3. Коды базовых классов
Код класса  Тип устройства
0x00 - Устройство было изготовлено до введения стандарта на коды классов
0x01 - Контроллер устройства массовой памяти
0x02 - Сетевой контроллер
```

```
0x03 - Видеоконтроллер
0x04 - Устройство мультимедиа
0x05 - Контроллер памяти
0x06 - Устройство типа <мост>
0x07 - Простой коммуникационный контроллер
0x08 - Основная системная периферия
0x09 - Устройство ввода
0x0A - Стыковочная станция
0x0B - Процессор
0x0C - Контроллер последовательной шины
0x0D - Контроллер устройства беспроводной передачи данных
0x0E - Интеллектуальный контроллер ввода-вывода
0x0F - Сопутствующий коммуникационный контроллер
0x10 - Контроллер устройства шифрации/дешифрации
0x11 - Контроллер устройства сбора и обработки сигналов
0x12-FE - Зарезервированы
0xFF - Устройство не подходит ни к одному из перечисленных классов
)

(  Полное описание кодов классов, заодно формируем таблицу
 в словаре форт-системы,     )

HEX
0 VALUE PCI_DATA_BASE  \ начальный адрес списка устройств PCI
0 VALUE /PCI_DATA_BASE \ количество строк списка
0
 1 -- .BaseClass
 1 -- .SubClass
 1 -- .Interface
 1 -- .Number
 CONSTANT /CODE_FIELD

HERE TO PCI_DATA_BASE

\ Базовый класс | Подкласс | Интерфейс | Кол-во |
 00 C, 00 C, 00 C, \  Все устройства, выпущенные до принятия стандарта
                    \ , за исключением VGA - совместимых
 00 C, 01 C, 00 C, \ VGA-совместимые устройства
 01 C, 00 C, 00 C, \ SCSI-контроллер
 01 C, 02 C, 00 C, \ Контроллер дисковода гибких дисков
 01 C, 03 C, 00 C, \ IPI-контроллер
 01 C, 04 C, 00 C, \ RAID-контроллер
 01 C, 80 C, 00 C, \ Устройство массовой памяти другого типа
 02 C, 00 C, 00 C, \ Контроллер Ethernet
 02 C, 01 C, 00 C, \ Контроллер Token Ring
 02 C, 02 C, 00 C, \ Контроллер FDDI
 02 C, 03 C, 00 C, \ Контроллер ATM
 02 C, 04 C, 00 C, \ Контроллер ISDN
 02 C, 80 C, 00 C, \ Сетевой контроллер другого типа
 03 C, 00 C, 00 C, \ VGA -совместимый контроллер
 03 C, 00 C, 01 C, \  8514-совместимый контроллер
 03 C, 01 C, 00 C, \  Контроллер XGA
 03 C, 02 C, 00 C, \  3D-контроллер
 03 C, 80 C, 00 C, \  Другой видеоконтроллер
 04 C, 00 C, 00 C, \  Видеоустройство
 04 C, 01 C, 00 C, \  Аудиоустройство
 04 C, 02 C, 00 C, \  Устройство для компьютерной телефонии
 04 C, 80 C, 00 C, \  Мультимедиа-устройство другого типа
 05 C, 00 C, 00 C, \  Контроллер оперативной памяти
 05 C, 01 C, 00 C, \  Контроллер Flash-памяти
 05 C, 80 C, 00 C, \  Контроллер памяти другого типа
 06 C, 00 C, 00 C, \  Мост хоста
 06 C, 01 C, 00 C, \  Мост ISA
 06 C, 02 C, 00 C, \  Мост EISA
 06 C, 03 C, 00 C, \  Мост MCA
 06 C, 04 C, 00 C, \  Мост PCI-to-PCI
 06  C,  04  C,  01  C,  00  C,  \    Мост PCI-to-PCI, поддерживающий вычитающее декодирование
 06 C, 05 C, 00 C, 00 C, \  Мост PCMCIA
 06 C, 06 C, 00 C, 00 C, \  Мост NuBus
 06 C, 07 C, 00 C, 00 C, \  Мост CardBus
 06 C, 08 C, 00 C, 00 C, \  Мост RACEway в режиме прозрачности
 06 C, 08 C, 01 C, 00 C, \  Мост RACEway в режиме оконечного узла
 06 C, 09 C, 40 C, 00 C, \  Полупрозрачный мост PCI-to-PCI, обращенный к хост-процессору <основной> стороной
 06 C, 09 C, 80 C, 00 C, \  Полупрозрачный мост PCI-to-PCI, обращенный к хост-процессору "вторичной> стороной
 06 C, 80 C, 00 C, 00 C, \  Мост другого типа
 07 C, 00 C, 00 C, 00 C, \  Стандартный XT-совместимый контроллер последовательного порта
 07 C, 00 C, 01 C, 00 C, \  16450-совместимый контроллер последовательного порта
 07 C, 00 C, 02 C, 00 C, \ 16550-совместимый контроллер последовательного порта
 07 C, 00 C, 03 C, 00 C, \ 16650-совместимый контроллерпоследовательного порта
 07 C, 00 C, 04 C, 00 C, \ 16750-совместимый контроллер последовательного порта
 07 C, 00 C, 05 C, 00 C, \ 16850-совместимый контроллер последовательного порта
 07 C, 00 C, 06 C, 00 C, \ 16950-совместимый контроллер последовательного порта
 07 C, 01 C, 00 C, 00 C, \ Параллельный порт
 07 C, 01 C, 01 C, 00 C, \ Двунаправленный параллельный порт
 07 C, 01 C, 02 C, 00 C, \ Параллельный порт типа ECP1.X
 07 C, 01 C, 03 C, 00 C, \ Контроллер 1EEE1284
 07 C, 01 C, FE C, 00 C, \ Целевое устройство 1EEE1284 (не контроллер)
 07 C, 02 C, 00 C, 00 C, \ Многопортовый последовательный контроллер
 07 C, 03 C, 00 C, 00 C, \ Стандартный модем
 07 C, 03 C, 01 C, 00 C, \ Hayes-соеместимый модем, 16450-совместимый интерфейс
 07 C, 03 C, 02 C, 00 C, \ Hayes-совместимый модем, 16550-совместимый интерфейс
 07 C, 03 C, 03 C, 00 C, \ Hayes-совместимыи модем, 16650-совместимый интерфейс
 07 C, 03 C, 04 C, 00 C, \ Hayes-совместимый модем, 16750-совместимый интерфейс
 07 C, 80 C, 00 C, 00 C, \ Другое коммуникационное устройство
 08 C, 00 C, 00 C, 00 C, \ Стандартный контроллер прерываний типа 8259
 08 C, 00 C, 01 C, 00 C, \ Контроллер прерываний ISA
 08 C, 00 C, 02 C, 00 C, \ Контроллер прерываний EISA
 08 C, 00 C, 10 C, 00 C, \ Контроллер прерываний ввода-вывода APIC
 08 C, 00 C, 20 C, 00 C, \ Контроллер прерываний ввода-вывода APIC
 08 C, 01 C, 00 C, 00 C, \ Стандартный контроллер ПДП типа 8237
 08 C, 01 C, 01 C, 00 C, \ ISA-контроллер ПДП
 08 C, 01 C, 02 C, 00 C, \ EISA-контроллер ПДП
 08 C, 02 C, 00 C, 00 C, \ Стандартный системный таймер типа 8254
 08 C, 02 C, 01 C, 00 C, \ Системный таймер ISA
 08 C, 02 C, 02 C, 00 C, \ Системные таймеры EISA (два таймера)
 08 C, 03 C, 00 C, 00 C, \ Стандартный контроллер часов реального времени
 08 C, 03 C, 01 C, 00 C, \ Контроллер часов реального времени ISA
 08 C, 04 C, 00 C, 00 C, \ Стандартный контроллер<горячего подключения> PC1
 08 C, 30 C, 00 C, 00 C, \ Системная периферия другого типа
 09 C, 00 C, 00 C, 00 C, \ Контроллер клавиатуры
 09 C, 01 C, 00 C, 00 C, \ Контроллер дигитайзера
 09 C, 02 C, 00 C, 00 C, \ Контроллер мыши
 09 C, 03 C, 00 C, 00 C, \ Контроллер сканера
 09 C, 04 C, 00 C, 00 C, \ Стандартный контроллер игрового порта
 09  C,  04  C,  01  C,  00  C,  \ Контроллер игрового порта с программируемым интерфейсом
 09 C, 80 C, 00 C, 00 C, \ Контроллер другого устройства ввода данных
 0A C, 00 C, 00 C, 00 C, \ Стандартная станция подключения
 0A C, 80 C, 00 C, 00 C, \ Нестандартная станция подключения
 0B C, 00 C, 00 C, 00 C, \ Процессор типа 386
 0B C, 01 C, 00 C, 00 C, \ Процессор типа 486
 0B C, 02 C, 00 C, 00 C, \ Процессор типа Pentium
 0B C, 10 C, 00 C, 00 C, \ Процессор типа Alpha
 0B C, 20 C, 00 C, 00 C, \ Процессор типа Power PC
 0B C, 30 C, 00 C, 00 C, \ Процессор типа MIPS
 0B C, 40 C, 00 C, 00 C, \ Сопроцессор
 0C C, 00 C, 00 C, 00 C, \ IEE1394 (FireWire)
 0C C, 00 C, 10 C, 00 C, \ IEEE1394, следующий за спецификацией 1394OpenHCI
 0C C, 01 C, 00 C, 00 C, \ ACCESS.bus
 0C C, 02 C, 00 C, 00 C, \ SSA
 0C C, 03 C, 00 C, 00 C, \ Устройство USB, следующее за спецификацией Universal Host Controller
 0C C, 03 C, 10 C, 00 C, \ Устройство USB, следующее за спецификацией Open Host Controller
 0C  C,  03  C,  80  C,  00  C,  \ Устройство USB без определенного программного интерфейса
 0C C, 03 C, FE C, 00 C, \ Устройство USB (не хост-контроллер)
 0C C, 04 C, 00 C, 00 C, \ FibreCannel
 0C C, 05 C, 00 C, 00 C, \ SMBus
 0D C, 00 C, 00 C, 00 C, \ IRDA-совместимый контроллер
 0D C, 01 C, 00 C, 00 C, \ IR-контроллер потребителя
 0D C, 10 C, 00 C, 00 C, \ RF-контроллер
 0D C, 80 C, 00 C, 00 C, \ Другой тип,контроллера беспроводной связи
 0E C, 00 C, 00 C, 00 C, \ Буфер сообщений P1P0 со смещением 040h
 0F C, 01 C, 00 C, 00 C, \ TV-контроллер
 0F C, 02 C, 00 C, 00 C, \ Аудиоконтроллер
 0F C, 03 C, 00 C, 00 C, \ Голосовой контроллер
 0F C, 04 C, 00 C, 00 C, \ Контроллер данных
 10 C, 00 C, 00 C, 00 C, \ Сетевой или компьютерный шифратор/дешифратор
 10 C, 10 C, 00 C, 00 C, \ Игровой шифратор/дешифратор
 10 C, 80 C, 00 C, 00 C, \ Шифратор/дешифратор другого типа
 11 C, 00 C, 00 C, 00 C, \ DPIO-модуль
 11  C,  80  C,  00  C,  00  C,  \ Контроллер устройства сбора и обработки сигналов другого типа

\ 01 C, 01 C, 00 C, 00 C, \ IDE-контроллер

\ 0E C, 00 C, xx C, 00 C, \ Интеллектуальный контроллер ввода-вывода (120) с архитектурой, соответствующей спецификации 1.0

PCI_DATA_BASE HERE SWAP - /CODE_FIELD / 1 - TO /PCI_DATA_BASE

: count_class_device ( base_class subclass_interface -- n )
\ возвращает количество устройств
\ заданного класса: base_class subclass_interface
 0
 BEGIN   ( base_class subclass_interface n0 )
   >R
   2DUP R@ FIND_PCI_CLASS NIP ( -- base_class subclass_interface ior )
   R> 1+   ( -- base_class subclass_interface ior n )
   SWAP    ( -- base_class subclass_interface n ior )
 86 = UNTIL NIP NIP 1-
;

: ide_list
  1FF 100 ?DO
    01 I TUCK count_class_device DUP
      IF
        SWAP 01 . 01 . 100 - . .
      ELSE 2DROP
      THEN
  LOOP
;

: refresh_pci_device_list ( -- )
\ обновление списка присутствующих устройств PCI
  PCI_DATA_BASE DUP /PCI_DATA_BASE /CODE_FIELD * + SWAP
    ?DO
     I .BaseClass C@ I .SubClass C@ 8 LSHIFT
     I .Interface C@ OR  \ 2DUP SWAP . .
     count_class_device \ DUP . CR KEY DROP
     I .Number C!
   /CODE_FIELD +LOOP
;

: show_pci_device_list
\ вывод списка кодов классов устройств, присутствующих на шине с указанием их
\ количества
```

```
 ." PCI device code list: " CR
 PCI_DATA_BASE DUP /PCI_DATA_BASE /CODE_FIELD * + SWAP
   ?DO
    I DUP .Number C@ IF
      DUP .BaseClass C@  . DUP .SubClass C@ .
      DUP .Interface C@   . DUP .Number C@ . CR
    THEN DROP
  /CODE_FIELD +LOOP
 ide_list
;
```