**Лабораторная работа №3**

Изучение способов адресации устройств шин PCI и PCI Express. Отработка метода опроса, идентификации и конфигурирования устройств.

**Краткие теоретические сведения**

PCI (Peripheral Component Interconnect) – параллельная шина ввода – вывода для соединения периферийных компонентов.. Она разрабатывалась в расчете на платформу Intel Pentium, но нашла применение и на более современных платформах. Первая версия PCI 1.0 появилась в 1992 г. В настоящее время развитие шины PCI приостановлено ввиду появления последовательной шины ввода - вывода нового поколения - PCI Express(c точки зрения программирования шины PCI и PCI Express идентичны ).

Поначалу шина PCI вводилась как дополнение к системам с основной шиной ISA, став позже центральной шиной: она соединяется с системной шиной процессора высокопроизводительным мостом (северным), входящим в состав чипсета системной платы. Остальные шины ввода-вывода (ISA/EISA или МСА), а также локальная ISA-подобная шина X-BUS и интерфейс LPC, к которым подключаются микросхемы системной платы (ROM BIOS, контроллеры прерываний, клавиатуры, DMA, порты СОМ и LPT, НГМД и др.), подключаются к шине PCI через <южный> мост(рис.1).



Рис.1 Структура шины PCI

Шина является синхронной - фиксация всех сигналов выполняется по положительному перепаду (фронту) сигнала CLK. Номинальной частотой синхронизации считается 33 МГц; начиная с версии 2.1, допускается повышение частоты до 66 МГц при поддержке всех устройств на шине (режим PCI-66).Номинальная разрядность шины данных - 32 бита, спецификация определяла возможность расширения разрядности до 64 бит. При частоте шины 33 МГц теоретическая пропускная способность достигает 132 Мбайт/с для 32-битной шины и 264 Мбайт/с для 64-битной; при частоте синхронизации 66 МГц - 264 и 528 соответственно.

При адресации устройств на шине PCI используются три адресных пространства: **ОЗУ, регистры ввода/вывода, регистры конфигурации;**

Пространство адресов ОЗУ используется для доступа к ячейкам оперативной памяти.

Адресное пространство ввода/вывода используется для обращения к PCI - устройствам по адресам, выделенным этим устройствам во время их начальной установки.

Адресное пространство регистров конфигурации используется для доступа к конфигурационному пространству PCI – устройств, подключенных к шине.

Спецификация PCI требует от устройств способности перемещать все занимаемые ресурсы в пределах доступного пространства адресации. Это позволяет обеспечивать бесконфликтное распределение ресурсов для многих устройств. Многофункциональное устройство, подключенное к шине, может содержать в себе дополнительные «подустройства», которые называются функциями. Каждая функция должна иметь свои регистры конфигурации. Стандарт PCI определяет для каждого устройства конфигурационное пространство размером до 256 регистров (8-битных), не приписанных ни к пространству памяти, ни к пространству ввода-вывода.

В системе может присутствовать несколько шин PCI, соединенных PCI - мостами. Каждая шина имеет свой номер шины (PCI bus number). Шины нумеруются последовательно; главная шина имеет нулевой номер.

С точки зрения конфигурирования устройств, минимальной адресуемой единицей является функция; ее полный адрес состоит из трех частей: номера шины, номера устройства и номера функции.

***Конфигурационное пространство устройств PCI и механизм Plug-and-Play.***

В стандарт PCI заложены возможности автоматического конфигурирования системных ресурсов (пространств памяти и ввода-вывода и линий запроса прерываний).

Автоматическое конфигурирование устройств (выбор адресов и прерываний) поддерживается средствами BIOS и ориентировано на технологию PnP.

После аппаратного сброса (или при включении питания) в PCI – устройствах, подключенных к шине, доступны для операций считывания и записи , только регистры пространства конфигурации. Доступ к этим регистрам осуществляется по специальным циклам шины Configuration Read и Configuration Write. В этих операциях устройства выбираются по индивидуальным сигналам IDSEL и сообщают о потребностях в необходимых системных ресурсах, таких как линия прерывания, пространство занимаемых адресов памяти и портов ввода/вывода. После распределения ресурсов, выполняемого программой конфигурирования ([BIOS](http://ru.wikipedia.org/wiki/BIOS) процедура POST (*Power-On Self-Test)* — самотестирование после включения), в конфигурационные регистры устройства записываются параметры конфигурирования. Только после этого становится возможным доступ к устройствам по командам обращения к памяти и портам ввода-вывода.

***Доступ к пространству конфигурации.***

Для чтения/записи регистров конфигурации любого устройства подключённого к шине PCI выделены два порта:

-порт управления РС1 (CF8h)

-порт данных РCI (CFCh).

Регистры 32х разрядные, т.е. для работы с ними следует использовать 32х битовые инструкции, например:

Mov Dx, CFCh

In Eax, Dx

Операции чтения и записи производятся за два шага. Сначала в порт управления записывается адрес регистра конфигурации, затем регистр считывается или записывается через порт данных. Например, чтение регистра адрес, ко­торого задан в переменной Address может выглядеть следующим образом:

#pragma inline

asm{

p386

mov dx,0x0CF8

mov EAX, Address

out dx,EAX

mov dx,0CFCh

in eax,dx

mov Res,eax

}

или

asm .386p

asm mov dx,0CF8h

asm mov EAX,Address

asm out dx,EAX

asm mov dx,0CFCh

asm in eax,dx

asm mov Res,eax

Результат чтения находится в переменной Res.

Адрес регистра конфигурации состоит из номера шины, номера уст­ройства на шине и номера функции устройства и номера регистра. Формат адреса приведён на рисунке 2:

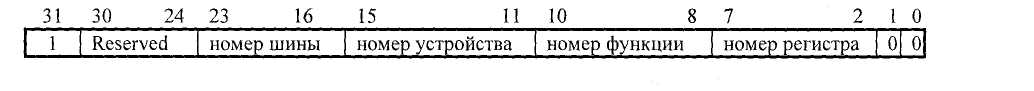


Рис. 2 Формат адреса регистра конфигурации

Все номера считаются от нуля. Стандарт предусматривает возможность подключения к системе нескольких шин PCI. Каждое устройство имеет выделенный номер на шине. Часть номеров резервируются для устройств расположенных на материн­ской плате компьютера. Например для моста который обеспечивает интерфейс шины и процессора. Часть номеров выделяется слотам в которых размещаются платы PCI. Оставшиеся номера не используются. Всего, как следует из рисунка 2 может быть 256 шин. При этом, к шине может быть подключено 32 различных устройства. В вашем проекте следует проверить все 32 номера на предмет наличия устройств. Одно устройство может выполнять до 8 различных функций. В проекте следует проверять все воз­можные номера функций каждого устройства. Назначение каждого реги­стра пространства конфигурации описывается далее.

Пространство регистров конфигурации имеет размер 256 байт. Назначение первых 64 байт жёстко определено в спецификации. Остальные регистры могут использоваться по усмотрению разработчика устройства. Они могут быть вообще не определены. Согласно спецификации каждое устройство PCI должно располагать по крайней мере 16 32-разрядными регистрами (64/4) в начале кон­фигурационного пространства. Эти регистры приведены рисунке 3.

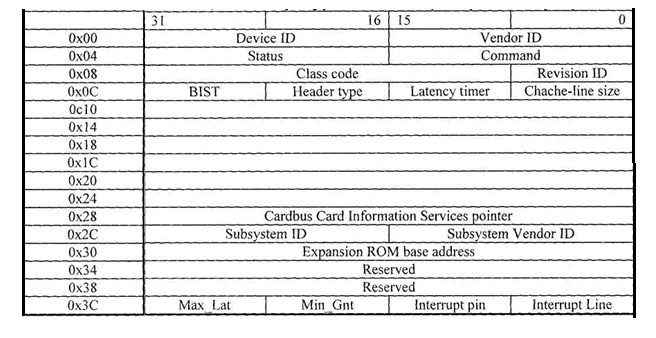


Рис. 3 Пространство конфигурации

Можно выделить следующие основные поля:

Device ID - уникальный идентификатор устройства. Назначается произво­дителем устройства. Запрещённое значение OxFFFF. Вы можете определить факт наличия устройства прочитав этот регистр. Если устройство отсутствует значение регистра = OxFFFF.

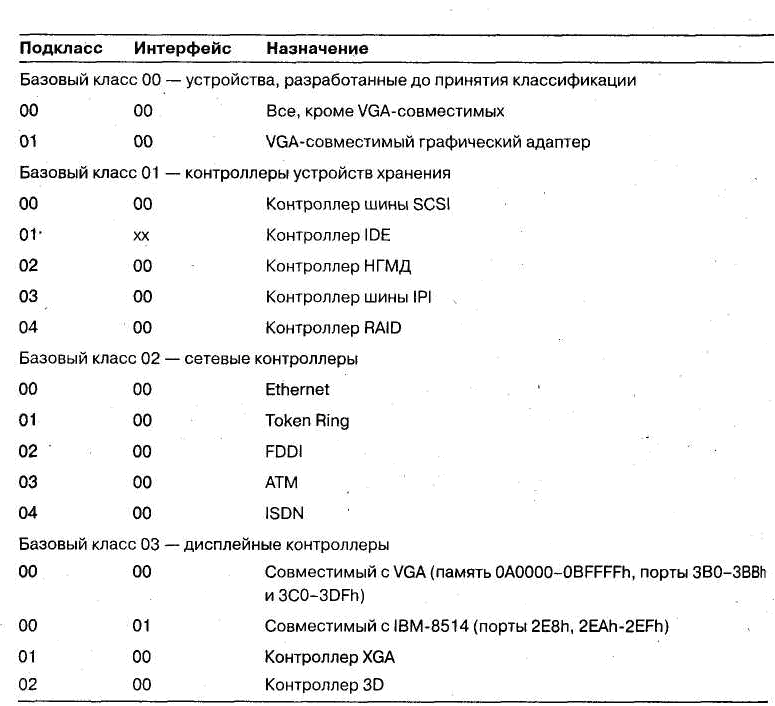
Vendor ID - уникальный идентификатор производителя. Этот идентификатор присваивается разработчиком стандарта «PCI special Interest Group». (PCI SIG /www.pcisig.com).

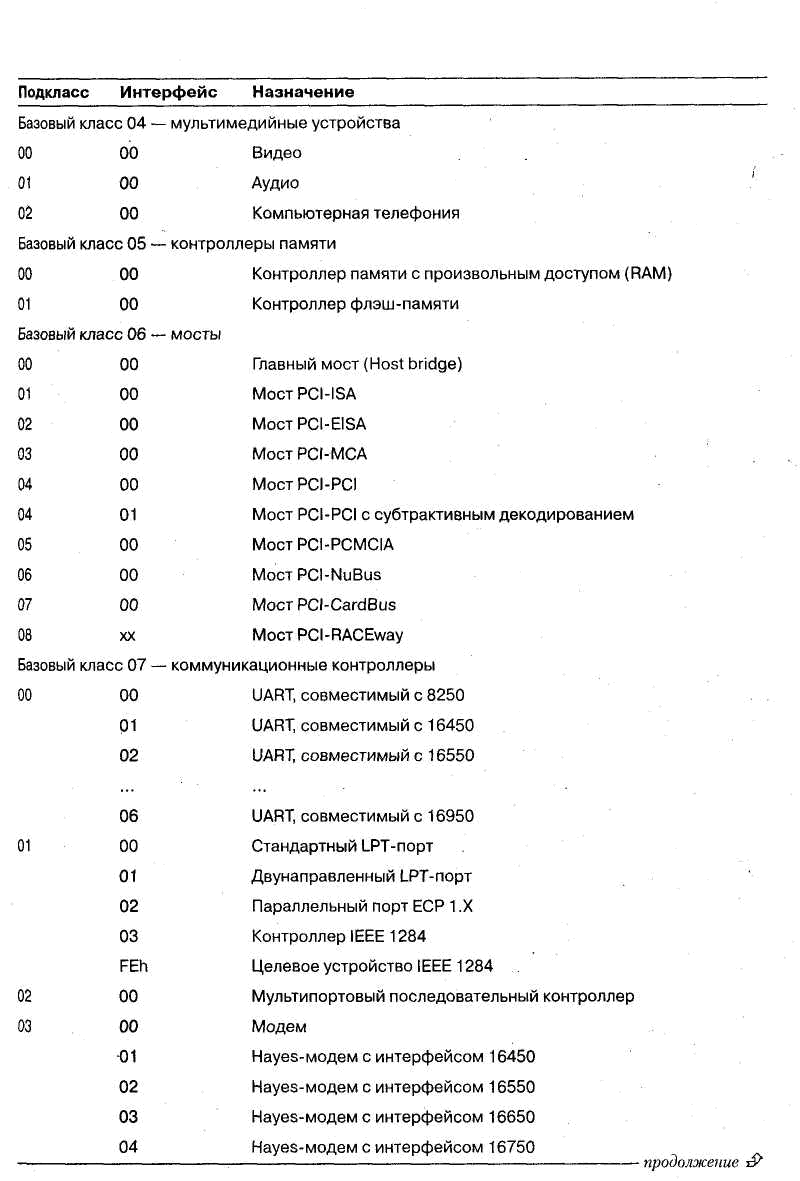
Revision ID — *версия продукта,* назначенная производителем. Используется как расширение поля Device ID;

*Регистр* Status служит для определения *состояния* и *свойств* устройства.

*Регистр команд* Command (RW) служит для управления поведением устройства на шине PCI. Регистр допускает как запись, так и чтение. После аппаратного сброса все биты регистра (кроме специально оговоренных исключений) обнулены.

*Class code* - код класса устройства - определяет к какому классу относится устройство. Значение этого регистра следует декодировать и вывести класс уст­ройства в виде текстовой строки. Код класса разделяется на три поля: Base class - базовый класс(старший байт byte), Subclass - подкласс (средний байт) и specific register level programming interface - специальный регистровый программный интерфейс (младший байт). Коды различных базовых классов приведены на рис. 4 .





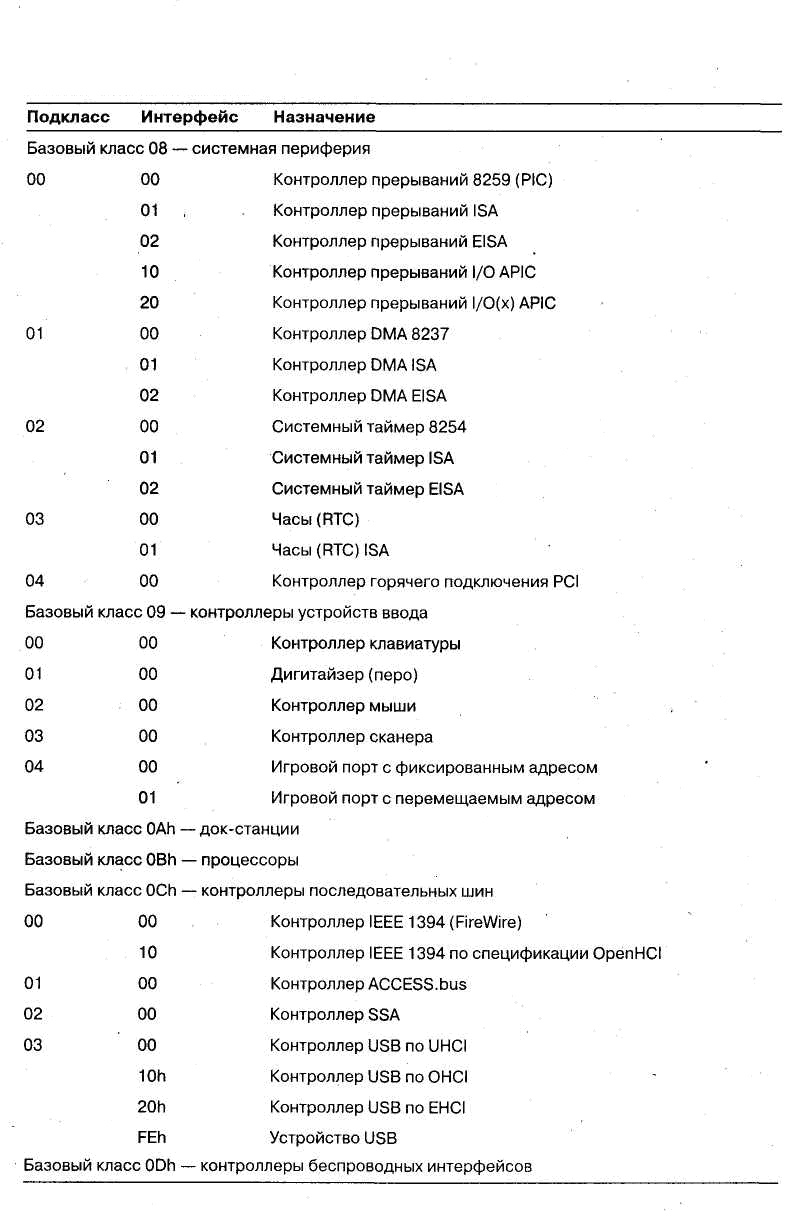


Рис. 4 Коды основных базовых классов

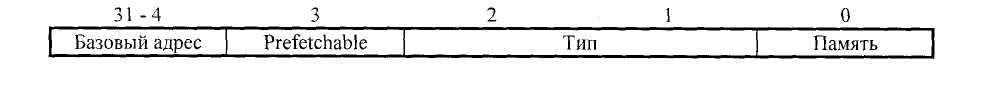
*Revision ID* - код выпуска - задаётся производителем.

*Регистр Cache Line Size (RW)* служит для задания размера строки кэша (0–128). По этому параметру инициатор определяет, какой командой чтения воспользоваться (обычное чтение, чтение строки или множественное чтение).

*Base address registers (BARs)*. Регистры базового адреса**.**

Конфигурационное пространство устройство может включать до шести регистров базового адреса. BARы используются для решения двух задач. Во первых они имеют доступные только для чтения биты, которые указывают какое количество памяти или адресного пространства портов ввода/вывода размещено на плате и требуется отобразить в адресное пространство системы. Во вторых, они содержат доступные для чтения и записи биты предназначенные для хранения базовых адресов пространства памяти и портов ввода-вывода, выделенного устройству системой. Один BAR описывает один непрерывный блок памяти или простран­ства портов. Формат BARa для портов ввода-вывода отличается от формата BARa для памяти. Оба формата приведены далее

*Регистр базового адреса памяти.*



Бит 0 = 0 - признак памяти

Биты [2:1] – тип адресации:

- 00 - память может быть отображена в любое место 32-битового адресного пространства системы

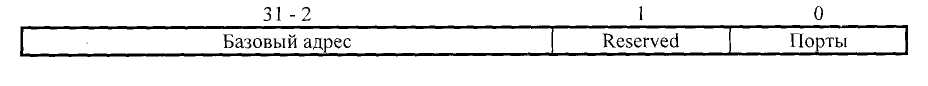
-01 - только ниже 1 MB

-10 - в любое место 64-битового адресного пространства

-11 - зарезервировано.

Биты [31:4] – базовый адрес памяти

*Регистр базового адреса портов.*



Бит 0 = 1 - признак области портов

Бит 1 – резерв

Биты [31:2] – базовый адрес блока портов

Программное обеспечение может определить тип BARa по младшему би­ту. Для BARob памяти биты 0-3 доступны только для чтения, для BARob пор­тов доступны только для чтения биты 0 и 1. Базовый адрес состоит из битов ОЗУ и ПЗУ. Доступные только для чтения биты базового адреса определяют размер адресного пространства относящегося к данному BARy. В биты доступные для чтения и записи система записывает старшую часть базового адреса адресного пространства выделенного устройству. Например BAR который запрашивает 16MB в памяти системы показан на рисунках 4 и 5.

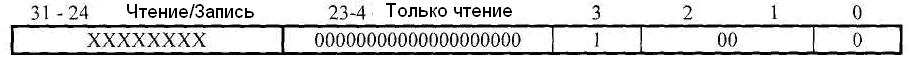


Рис. 5 BAR памяти перед инициализацией.

20 бит в диапазоне 23-4 доступные только для чтения нули. Они опреде­ляют размер памяти который требуется устройству. Этот размер вычисляется как 2(20+4) = 16MB, (добавляются 4 младших служебных бита).

Восемь старших бит доступны для чтения и записи. Они не инициализированы до запуска системы

.

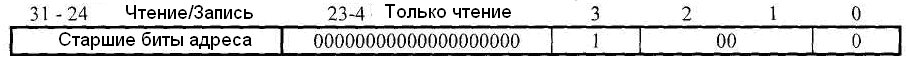


Рис. 5 BAR памяти после инициализации.

Биты 31-24 содержат старшие биты адресного пространства выделенного системой устройству.

Системное программное обеспечение работает по следующему алгоритму:

1.Записать в BAR все единицы OxFFFFFFFF.

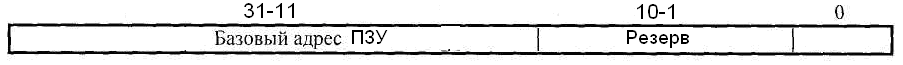
2.Считать значение BAR.

3.Определить сколько нулей содержится в считанном значении и вы­числить размер памяти и адреса портов ввода вывода, требующихся устройству (2(число нулеи + 4)).

4.Выделить системное адресное пространство и адреса портов устройству.

5.Записать в BAR начальный адрес этого адресного пространства. (Инициализация старших битов BARa)

ExpansionROM BaseAddress- регистрбазовогоадресавстроенногоПЗУPCI**-**устройства



Бит 0 - разрешение использования ПЗУ;

Биты [10:1] – резерв:

Биты [10:1] – базовый адрес.:

Бит 0 отвечает за то будет ли задействовано ПЗУ или нет.

Размер ПЗУ и его начальный адрес определяется точно также как и для области памяти.

Важно заметить, что устройство PCI только после инициализации способно воспринимать направленные ему команды чтения и записи памяти и портов. (Устройство сравнивает адрес на шине с содержимым своих BARob)

*Регистр Interrupt Line* (RW) хранит номер входа контроллера прерывания для используемой линии запроса (0—15 — IRQ0-IRQ15 ), который выделяется устройству системой в результате работы алгоритма plug-and-play.В системах с APIC может иметь и большее значение; 255 — неизвестный вход или не используется..

Регистр *Interrupt Pin* **-** задает линию, используемую для запроса прерыва­ния: 0 - не используется, 1 - INTА#, 2 - INTВ#, 3 - INTC#, 4 - INTD#, 5-FFh - резерв.

****

Рис. 6 Распределение прерываний в шинах PCI

*Конфигурационное пространство мостов PCI*

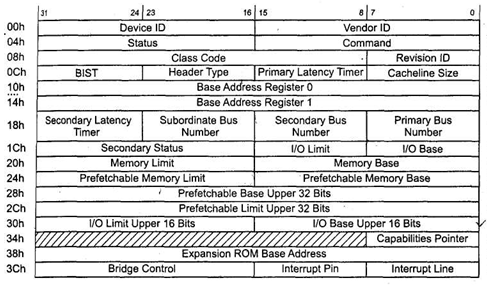


Рис. 7 Конфигурационное пространство мостов

Заголовок конфигурационного пространства мостов PCI-PCI приведен на рис. 7. Регистры в диапазоне адресов 00—17h полностью совпадают с регистрами обычного устройства PCI и описывают поведение и состояние моста на первичной шине, Заметим, что бит 2 регистра команд (Bus Master Enable) управляет возможностью трансляции транзакций с вторичной шины на первичную. Если этот бит обнулен, то мост не должен на вторичной стороне отзываться как целевое устройство в транзакциях записи/чтения памяти и ввода-вывода, поскольку он не сможет транслировать эти транзакции на первичную шину. Регистры BAR описывают только область специфических (зависящих от реализации) регистров моста

Маршрутизирующие свойства моста определяются следующими регистрами

♦Primary Bus Number — номер первичной шины;

♦Secondary Bus Number — номер вторичной шины (это и номер моста);

♦Subordinate Bus Number — максимальный номер подчиненной шины;

♦I/O Base и I/O L imi t — регистры, задающие начальный и конечный адрес пространства ввода-вывода устройств, расположенных за мостом. Эти регистры задают только старшие 4 бита 16-битного адреса ввода-вывода, ♦I/O Limit Upper 16 Bits и I/O Base Upper 16 В i t s — регистры старшей части адреса ввода-вывода, если используется 32-битная адресация ввода-вы­вода (на это указывают установленные биты 0 регистров I/O Base и I/О Limit);

♦Memory Base Memory Limit—регистры, задающие начальный и конечный адрес пространства памяти, на которую отображены регистры ввода-вывода устройств, расположенных за мостом. Эти регистры задают только старшие 12 бит 32-битно­го адреса памяти, так что гранулярность выделения адресов составляет 1 Мбайт;

♦Prefetchable Memory Base и Prefetchable Memory Limit—регистры, зада-ющие начальный и конечный адрес «памяти устройств, расположенных за мостом. Эти регистры задают только старшие 12 бит 32-битного адреса памяти

♦Prefetchable Base Upper 32 Bits иPrefetchable Limit Upper 32 Bits-регистры старшей части адреса памяти, если используется 64-битная адресация (на это указывают установленные биты 0 регистров Prefetchable Memory Base иPrefetchable Memory Limit).

*Регистр* Secondary Status аналогичен обычному регистру состояния (Status), но его признаки относятся ко вторичной шине.

*Регистр* Expansion ROM Base Address, как и для обычного устройства, задает положение ПЗУ расширения BIOS (если это ПЗУ присутствует в мосте).

*Регистры* Interrupt Line и Interrupt Pin относятся к прерываниям, вырабаты­ваемым мостом (если таковые имеются). К линиям прерывания, транслируемым мостом, эти регистры отношения не имеют.

*Регистр* Bridge Control служит для управления работой моста.

**Задание на лабораторную работу**

1. Изучить теоретические сведения о механизме конфигурирования устройств PCI и способам доступа к конфигурационному устройству.

2. Разработать программу формирования цикла опроса и идентификации устройств PCI, которая будет считывать два первых поля конфигурационного пространства - коды Vendor ID (производитель) и Device ID (устройство).

3. Используя файл PCI\_DEVS.TXT или структуры, определенные в заголовочном файле pci\_c\_header.h, произвести расшифровку наименований производителей и устройств.

4. Результатом работы программы должна быть выводимая на экран или в текстовый файл таблица, содержащая следующую информацию:

4.1. адрес устройства (номер шины, номер устройства и номер функции);

4.2. 16-разрядный код производителя (в hex-формате);

4.3. 16-разрядный код устройства (в hex-формате);

4.4. производитель и название устройства.

Дополнительно выполнить индивидуальное задание в соответствии с вариантом

Задачи.

1. Если устройство не мост(0-бит поля Header Type =0) вывести и расшифровать значение поля Сlass Code.
2. Если устройство не мост(0-бит поля Header Type =0) вывести и расшифровать значение полей базовых регистров памяти.
3. Если устройство не мост(0-бит поля Header Type =0) вывести и расшифровать значение полей базовых регистров ввода/вывода.
4. Если устройство не мост(0-бит поля Header Type =0) вывести и расшифровать значение полей базовых регистров постоянной (ROM) памяти.
5. Если устройство не мост(0-бит поля Header Type =0) вывести и расшифровать значение поля *Cache Line Size*
6. Если устройство не мост(0-бит поля Header Type =0) вывести и расшифровать значение поля Interrupt Pin.
7. Если устройство не мост(0-бит поля Header Type =0) вывести и расшифровать значение поля Interrupt Line.
8. Если устройство мост(0-бит поля Header Type =1) вывести и расшифровать значение полей Primary Buss Number.Secondary Buss Number,Subordinate Buss Number
9. Если устройство мост(0-бит поля Header Type =1) вывести и расшифровать значение полей I/O Base и I/O Limit
10. Если устройство мост(0-бит поля Header Type =1) вывести и расшифровать значение полей Mеmory Base и Memory Limit
11. Если устройство мост(0-бит поля Header Type =1) вывести и расшифровать значение поля Сlass Code.
12. Если устройство мост(0-бит поля Header Type =1) вывести и расшифровать значение поля Interrupt Pin и Interrupt Line

Варианты

|  |  |  |  |
| --- | --- | --- | --- |
| № варианта | Номер 1 задачи | Номер 2 задачи | Номер 3 задачи |
| 1 | 1 | 5 | 9 |
| 2 | 2 | 6 | 11 |
| 3 | 3 | 8 | 10 |
| 4 | 4 | 7 | 12 |
| 5 | 8 | 6 | 1 |
| 6 | 5 | 10 | 3 |
| 7 | 7 | 6 | 8 |
| 8 | 2 | 9 | 12 |
| 9 | 3 | 4 | 11 |
| 10 | 6 | 3 | 1 |
| 11 | 5 | 8 | 10 |
| 12 | 2 | 7 | 6 |
| 13 | 4 | 9 | 8 |
| 14 | 2 | 12 | 9 |
| 15 | 3 | 5 | 7 |

**При сдаче работы дать краткую информацию о назначении соответствующих полей конфигурационного пространства.**

**Порядок выполнения работы**

В общем случае программа должна состоять из трех вложенных циклов.

Внешний цикл – номер шины, первый внутренний – номер устройства, второй внутренний номер функции.

!!!!!!!

**Порядок выполнения работы**

Программу рекомендуется разрабатывать на языке программирования C/C++ с применением вставок на языке ассемблера для доступа к порта управления (CF8h) и порта данных (CFCh). Для снятия блокировки доступа к аппаратным портам в ОС Windows XP может потребоваться предварительная установка драйвера доступа к портам ввода-вывода GIVEIO (для соответствующей ОС) (см. папку Драйвер доступа). При этом в программе можно подключить драйвер следующим образом:

HANDLE h=CreateFile(L"\\\\.\\giveio" , GENERIC\_READ, 0, NULL, OPEN\_EXISTING, FILE\_ATTRIBUTE\_NORMAL, NULL);

if (h == INVALID\_HANDLE\_VALUE)

{

printf("Couldn't access GIVEIO driver\n");

printf("GIVEIO driver must be loaded first\n");

return -1;

}

Фрагмент программы

#include "stdafx.h"

#include "pci\_c\_header.h"

#include <iostream>

#include <Windows.h>

#include <tchar.h>

#include <iostream>

using namespace std;

int \_tmain(int argc, \_TCHAR\* argv[])

{

HANDLE h=CreateFile(L"\\\\.\\giveio", GENERIC\_READ, 0, NULL,

OPEN\_EXISTING, FILE\_ATTRIBUTE\_NORMAL, NULL);

if (h == INVALID\_HANDLE\_VALUE)

{

printf("Couldn't access GIVEIO driver\n");

printf("GIVEIO driver must be loaded first\n");

//system("pause");

return -1;

}

}

. При использовании ОС Linux для доступа к этим портам можно использовать команды

## NAME         [top](http://man7.org/linux/man-pages/man2/inl.2.html#top_of_page)

outb - запись байта в порт

outw - запись слова в порт

outl - запись двойного слова

inb – чтение байта из порта

inw – чтение слова из порта

inl - чтение двойного слова

Синтаксис команд

## SYNOPSIS         [top](http://man7.org/linux/man-pages/man2/inl.2.html#top_of_page)

**#include <sys/io.h>**

**unsigned char inb(unsigned short int** *port***);**

**unsigned char inb\_p(unsigned short int** *port***);**

**unsigned short int inw(unsigned short int** *port***);**

**unsigned short int inw\_p(unsigned short int** *port***);**

**unsigned int inl(unsigned short int** *port***);**

**unsigned int inl\_p(unsigned short int** *port***);**

**void outb(unsigned char** *value***, unsigned short int** *port***);**

**void outb\_p(unsigned char** *value***, unsigned short int** *port***);**

**void outw(unsigned short int** *value***, unsigned short int** *port***);**

**void outw\_p(unsigned short int** *value***, unsigned short int** *port***);**

**void outl(unsigned int** *value***, unsigned short int** *port***);**

**void outl\_p(unsigned int** *value***, unsigned short int** *port***);**

Например для записи в порт числа **а** **outl (a, 0xCF8**), для чтения порта **inl(0xCFC**) Для задания уровня приоритета использовать вызов iopl(3)

Пример программы

#include <sys/io.h>

#include "pci\_c\_header.h"

#include <string.h>

#define NOT\_SPECIFIED "Not specified"

int main()

{

//printf("Peripherial devs lab3\n\n");

int i,busid,devid;

if(iopl(3)) //задание уровня приоритета

{

printf("I/O Privilege level change error: %s\nTry running under ROOT user\n",(char \*)strerror(errno));

return 1;

} //granted privileges 3 for port access

Информация о производителях, кодах устройств, класс кодах устройств, командах находится в структурах, определённых в заголовочном файле типа pci.h(или подобных файлах), который находятся в папке Vendor или в папке HeaderFiles и который надо включить в проект для поиска устройств. Похожая информация находится в текстовых файлах pcidevs.

Пример описания структуры кода производителя в заголовочном файле:

**typedef struct \_PCI\_VENTABLE**

**{**

**unsigned short VenId; // код произволителя**

**char \*VenShort; // краткое имя призводителя**

**char \*VenFull; // полное имя производителя**

**} PCI\_VENTABLE, \*PPCI\_VENTABLE;**

**PCI\_VENTABLE PciVenTable[] =**

**{**

**{0x10, "Allied Telesis, Inc (Wrong ID)", ""},**

**{0x1c, "PEAK-System Technik GmbH", ""},**

**{0x3d, "Lockheed Martin-Marietta Corp", ""},**

**{0x59, "Tiger Jet Network Inc. (Wrong ID)", ""},**

**{0x70, "Hauppauge computer works Inc.", ""},**

**{0x71, "Nebula Electronics Ltd.", ""},**

**{0x95, "Silicon Image, Inc. (Wrong ID)", ""},**

**{0xa7, "Teles AG (Wrong ID)", ""},**

**{0x100, "Ncipher Corp Ltd", ""},**

**.**

**.**

**.**

**};**

**// Use this value for loop control during searching:**

**#define PCI\_VENTABLE\_LEN (sizeof(PciVenTable)/sizeof(PCI\_VENTABLE))**

**Пример обращения к полям структуры:**

**for (i=0; i < PCI\_VENTABLE\_LEN; i++)**

**{**

**if (ven\_id == PciVenTable[i].VenId)**

**{**

**/\*поиск кода имени производителя\*/**

**}**

Проверка правильности чтения конфигурационного пространства для ОС Linux c помощью команды lspci – (с ключом) x

**Литература**

ГукМ.Шины PCI,USB и FIREWire. Энциклопедия — СПб.: Питер,. — 540 с.

Контрольные вопросы:

1. Назначение основных выводов процессора 8088/86;
2. Организация сигналов записи/ чтения (память, устройства ввода-вывода);
3. Структурная схема IBM PC XT (с элементами поддержки процессора 8088/86);
4. Контроллер прерываний 8259 и его связь с процессором 8088/86. Последовательность действий при обработке аппаратных прерываний;
5. Контроллер прямого доступа к памяти 8237 и его связь с процессором 8088/86. Последовательность действий при обработке запросов ПДП;
6. Трехканальный таймер 8253 и его связь с процессором 8088/86 для IBM PC XT;
7. Программируемый параллельный интерфейс на базе 8255 и его связь с процессором 8088/86 для IBM PC XT;
8. Интерфейсы компьютера (системная шина, шина ввода-вывода, шины периферийных устройств);
9. Назначение и основные характеристики шин QPI, DMI, Hyper Transport, FSB;
10. Основные характеристики шины ISA. Организация прерываний, ПДП, типы адресных пространств. Архитектура компьютера с шиной ISA;
11. Основные характеристики шины PCI. Организация прерываний, ПДП, типы адресных пространств. Мосты шины PCI Архитектура компьютера с PCI;
12. Назначение и принцип работы расширенного контроллера прерываний APIC, прерывания с помощью сообщений;
13. Основные характеристики шины PCI-Express. Организация прерываний, типы адресных пространств Архитектура компьютера с шиной PCI-Express;
14. Уровневая модель щины PCI-Express (назначение отдельных уровней). Топология шины PCI, масштабируемость шин PCI-Express.
15. Что такое чипсет?