PCI 设备配置空间研究

程海全[1,2]，徐抒岩[1]，胡 君[1]
（1. 中国科学院 长春光学精密机械与物理研究所 空间光学研究部，吉林 长春 130033；
2. 中国科学院 研究生院，北京 100039）

摘要： 在开发 PCI 设备驱动程序时，需要访问配置空间来控制设备。研究了 PC 平台和 Windows 系统下访问 PCI 配置空间的机制和方法，分析了配置空间中的寄存器结构，提出了一种在用户模式下访问 PCI 配置空间并通过新能力列表确认 PCI Express 设备的方法，在英特尔 G31 主板 Windows XP 系统的 PC 上的实验结果表明这种方法可行。

关键词： PCI；配置空间；能力列表；PCI Express
中图分类号： TN302.7 **文献标识码：** A **文章编号：** 1674-6236(2010)10-0001-04

Research of PCI device's configuration space

CHENG Hai-quan[1,2], XU Shu-yan[1], HU Jun[1]
(1. *Space Optics Research Department, Changchun Institute of Optics, Fine Mechanics and Physics, Chinese Academy of Sciences, Jilin* 130033, *China*; 2. *Graduate School, the Chinese Academy of Sciences, Beijing* 100039, *China*)

Abstract: It's necessary to access device configuration space in the development of a PCI device driver to control the equipment. The mechanism and ways of accessing PCI configuration space under PC platform and Windows OS were studied in this paper. The register structure in PCI configuration space was analysed. A way of accessing configuration space under user mode was introduced, which can determine a PCI Express device through capability list. Experiment on a G31 motherboard and Windows XP based PC shows this solution is valid.

Key words: PCI; configuration space; capability list; PCI Express

PCI(Peripheral Component Interface)自 1991 年由英特尔公司提出以来发展到现在，已有 20 年历史。PCI 局部总线独立于处理器的独特设计及其高性能、低成本、开放性等方面的优势，使其在 PC 平台上得到了迅速地普及和发展[1]，虽然 PCI Express 总线已经流行，但是由于操作系统发展滞后于硬件的发展(特别是 Windows XP 目前还比较流行，而 Windows XP 系统本身并不支持 PCI Express 设备[2])，以及保持设备相互兼容等方面的考虑，有很多时候还必须使用 PCI 设备。因此，这里对 PCI 设备配置空间进行研究，提出一种在用户模式下访问 PCI 配置空间并通过新能力列表确认 PCI Express 设备的方法。

1 基于 PCI 的 PC 体系

PCI 提供了一组完整的总线接口标准，其目的是描述如何将计算机系统中的外围设备以一种结构化和可控化的方式连接在一起，给出了外围设备在连接时的电气和机械特性以及互联协议。因为 PCI 标准原本是由英特尔提出的，所以与 PCI 规范中内容与 PC 有着天然的联系。

PCI 系统拓扑结构主要包括各种 PCI 设备和 PCI 桥接设备。英特尔 915 芯片组以前整个主板架构都是以 PCI 为基础的，连接 GMCH 和 ICH 的 Hub Link 是为了提高南北桥带宽的一种总线，逻辑上相当于 PCI 总线 0。从 915 芯片组开始，整个主板架构基于 PCI Express，连接 GMCH 和 ICH 的 Hub Link 升级为 DMI(Direct Media Interface)。

2 PCI 配置空间概述

PCI 标准中每个 PCI 设备的每个功能要实现 256 字节的配置空间，这个空间分为预定义头标区 64 字节和设备相关区 192 字节。设备在每个区中只实现必要的和与之相配的寄存器。

头标区的各个字段用来唯一识别设备，前 16 个字节的定义在各种类型的设备中都是一样的，剩余的字节随各个设备支持的功能有所不同。分类代码字段用来标识设备的功能，头标类型 D7 字段指出了设备是否包含多功能；D6~D0 字段规定了头标区从 10h 开始的布局类型，类型 1(01h)用于 PCI-PCI 桥，类型 2(02h)用于 CardBus 桥，类型 0(00h)用于除上述两种设备之外的其他所有设备[1]，如表 1 所示。

2.1 新能力列表

在 PCI 规范 2.1 版发布之后，PCI 又增加一些新的特征，这些特征由增加的一系列称为能力列表的寄存器表示。这种

收稿日期：2010-04-07 稿件编号：201004017
基金项目：国家自然科学基金资助项目(60507003)
作者简介：程海全(1985—)，男，天津蓟县人，硕士研究生。研究方向：空间光学仪器计算机控制。

表 1　类型 0 配置空间头标区

Tab. 1　Header region of type 0 configuration space

配置空间偏移量	寄存器名
00~01h	供应商识别号
02~03h	设备识别号
04~05h	PCI 命令寄存器
06~07h	PCI 状态寄存器
08h	版本识别号
09~0Bh	类代码
0Ch	Cache 行大小
0Dh	延迟时间
0Eh	头标类型
0Fh	自测试
10~27h	基地址寄存器
28~2Bh	CardBus CIS 指针
2C~2Dh	子系统供应商识别号
2E~2Fh	子系统识别号
30~33h	扩展 ROM 基地址
34h	能力指针寄存器
35~3Bh	保留(Reserved)
3Ch	中断线
3Dh	中断引脚
3Eh	最小延迟
3Fh	最大延迟

可选择的能力列表数据结构由 PCI 状态寄存器的能力列表位(Bit 4)指示。

如果状态寄存器中的能力列表位为 1，那么该功能在它的配置空间用偏移地址为 34h 处的字节来实现新能力列表指针寄存器。能力指针寄存器所指的单元是一个或多个配置寄存器组链的第一个入口，每个入口都支持一个新特征并且具有相同的格式，如图 1 所示。

图 1　新能力列表入口通用格式

Fig.1　General layout of new capability list

第 1 个字节称为能力 ID(由 PCI SIG 分配)，表示与这个寄存器相关联的特征；第 2 个字节指向下一个能力指针，如果为 00 表示不存在下一个能力指针了。配置空间中的能力寄存器列表就是通过能力指针寄存器和若干个下一能力指针串联在一起的，这些能力结构在链表中出现的顺序 PCI 协议中没有规定。

特征寄存器组总是紧跟在入口的前 2 个字节后，其长度与格式根据特征的类型而定。目前 PCI 规范 3.0 定义的部分新能力 ID 如表 2 所示[3]。

2.2　PCI Express 能力结构

PCI Express 在 PCI 3.0 兼容配置空间中定义了一个能力结构，来确认一个 PCI Express 设备并提供对 PCI Express 新特征的支持。PCI Express 设备必须实现 PCI Express 能力结构。这个能力结构提供一种在旧操作系统上兼容 PCI 软件的机制。不仅仅是为了确定一个 PCI Express 设备，PCI Express 能力结构还被用来访问 PCI Express 相关的控制状态寄存器和增强的功耗管理[4]。

表 2　PCI 规范 3.0 定义的新能力

Tab. 2　New capabilities defined by PCI spec 3.0

能力 ID	功能描述
00h	保留
01h	PCI 电源管理
05h	消息信号中断(MSI)
10h	PCI Express
11h	MSI-X
12h~FFh	保留

3　访问配置空间的机制及方法解释

X86 处理器可寻址 2 类不同的地址空间：存储器和 I/O 空间。而 PCI 总线主设备(包括 Host/PCI 桥)有 3 种地址空间，除了存储器和 I/O 空间以外，还能够进行配置空间的访问。配置访问用于访问设备功能的配置寄存器。

在系统上电时，系统配置软件(通常为 BIOS)扫描系统的各条总线，以确定总线上存在什么设备以及它们需要什么配置。配置软件读取设备的配置寄存器，确定设备所需的存储器和/或 IO 地址范围，分配中断以及主设备对总线的访问要求等。

在系统启动时，由 BIOS 代码执行设备配置。当操作系统起动以后，便由操作系统控制设备管理。无论在哪种情况下，总是由处理器上运行的配置程序执行系统配置，主处理器需要发指令给 Host/PCI 桥，令 Host/PCI 桥在 PCI 总线上执行配置读和写交易。

在 Windows 系统平台上主要有 3 种访问设备配置空间的方法：通过访问 IO 口访问 PCI 配置空间，通过创建即插即用 IRP (IRP_MN_READ_CONFIG 和 IRP_MN_WRITE_CONFIG)或 BUS_INTERFACE_STANDARD 总线接口访问配置空间，通过使用第三方工具提供的封装函数(类)来访问配置空间[5]。

3.1　CF8/CFC 方式访问 PCI 兼容配置空间

PCI 配置机制使用 2 个位于 Host/PCI 桥的 32 位的 I/O 端口地址 0CF8h 和 0CFCh，0CF8h 称为配置地址 (Config-Address)寄存器，0CFCh 称为配置数据(Config-Data)寄存器。通过对这 2 个 32 位的 I/O 端口地址的访问，形成对配置空间的映射，实现对配置空间的访问。32 位配置地址端口的各数据位定义如表 3 所示。

访问一个 PCI 功能的配置寄存器分二步进行。首先，将目标总线号、设备号、功能号以及表示配置空间中目标寄存器偏移地址连同配置映射使能写入配置地址口，然后将配置

表 3　配置地址寄存器各个位的说明
Tab. 3　Bits description of Config-Address

位	描述
31	配置空间映射使能位,为 1 时允许将处理器对配置数据端口的 I/O 访问映射为 PCI 总线上的配置访问。
30~24	保留,必须为 0。
23~16	标识目标 PCI 总线编号(1~256)。
15~11	标识目标 PCI 设备编号(1~32)。
10~8	标识目标 PCI 设备的功能编号(1~8)。
7~2	标识目标功能配置空间中的寄存器偏移地址。
1~0	只读,恒为 0。

数据从配置数据端口读出或写出。

写配置地址端口以后,Host/PCI 桥立即将指定的目标总线和桥另一边现有的总线数目范围相比较,如果目标总线存在,便启动一次 PCI 配置读或写。配置数据的读/写可以是单、双或 4 字节。

3.2　异步 IRP 方式

设备驱动程序可以使用 IRP_MN_WRITE_CONFIG 请求或者 BUS_INTERFACE_STANDARD 标准的 SetBusData 方法来写配置空间,也可以使用 IRP_MN_READ_CONFIG 请求或者 BUS_INTERFACE_STANDARD 标准的 GetBusData 方法来读取配置空间。如果设备不存在扩张配置空间,那么读请求会返回 0xFFFF,写请求会没有效果[5]。

微软 Windows2000,Windows XP,Server2003 本身不支持 PCI Express,它们把 PCI Express 设备看作 PCI 设备,只是通过 BIOS 和 ACPI 支持 PCI Express 的部分功能。Windows Vista 以及后续的 Windows 7 自身开始支持 PCI Express 的 ASPM (Active State Power Management),存取扩展配置空间以及 PME(Power Management Event) 等功能[2]。

3.3　DriverWork,WinDriver 封装类/函数方式

为了降低开发驱动程序的门槛,许多公司在微软驱动开发包的基础上开发了很多软件来简化驱动程序的开发,有些软件甚至可以在用户模式下开发驱动程序。这其中就包括 Numega 公司推出的 DriverStudio 和 Jungo 公司的 WinDriver。

DriverStudio 工具中的 DriverWork 组件提供了 KPciConfiguration 类用于协助开发 PCI 驱动程序,KPciConfiguration 的成员函数帮助驱动程序完成读写 PCI 配置空间中各个域[6]。WinDriver 工具提供了 WDC_PciReadCfgXXX () 和 WDC_PciWriteCfgXXX()函数来访问 PCI 兼容配置空间以及 PCI Express 扩展配置空间,还提供了低层函数 WD_PciConfigDump()来倾印功能设备的配置空间。

对于驱动开发人员来说,虽然使用 CF8-CFC 方式是最不实用的,但是要注意到其他几种高级的访问方式都是基于这种方式的,只不过是在它们的基础上做了层层的封装。还要认识到这几种方式访问的局限性,比如受版本限制 WDK,WinDriver 以及 DriverWork 并没有对 PCI Express 配置空间中的所有信息提供接口函数或方法,有时候还需要使用最原始的方式来获取相关信息。

4　驱动程序访问配置空间能力结构举例

为了说明上面几种访问 PCI 配置空间方法的应用,这里举出了在用户模式下,使用 I/O 方式访问系统总线上每个设备功能配置空间,来确认是否是一个 PCI Express 设备。

```
void DisplayConfigSpace(int bus,int dev,int fn)
{
    DWORD dwAddr,dwData;
    PCI_COMMON_CONFIG PciConfig;
    PCI_SLOT_NUMBER SlotNumber;
    PCI_CAPABILITIES_HEADER CapabilityHeader;
    UCHAR CapabilityOffset;

    SlotNumber.u.AsULONG = 0;
    SlotNumber.u.bits.DeviceNumber = dev;//设置设备号
    SlotNumber.u.bits.FunctionNumber = fn;//设置功能号

dwAddr = 0x80000000 | (bus <<16) | (SlotNumber.u.
AsULONG<<8);
//得到物理地址
for (int i = 0;i < 0x100;i += 4)
{/* 256 字节的 PCI 配置空间 */
    SetPortVal(PCI_CONFIG_ADDRESS, dwAddr | i,4);
    GetPortVal(PCI_CONFIG_DATA,&dwData,4);
    memcpy( ((PUCHAR)&PciConfig)+i,&dwData,4 );
   }

if(PciConfig.VendorID == 0xFFFF)
{return;//无效设备}
else
{
printf("bus:%d dev:%d fn:%d ",
bus,dev,fn);
printf("%x %x",
PciConfig.VendorID,
PciConfig.DeviceID);
}
   //检查状态寄存器判断是否支持能力指针
if ((PciConfig. Status & PCI_STATUS_CAPABILITIES_
LIST) == 0) {
        printf ("This device does not support capabilities
list\n");
        return;
}
//根据不同的头标类型找到能力指针偏移地址
if ((PciConfig.HeaderType & (~ PCI_MULTIFUNCTION))
```

```
        == PCI_BRIDGE_TYPE) {
            CapabilityOffset = PciConfig.u.type1.CapabilitiesPtr;
        } else if ((PciConfig.HeaderType & (~ PCI_MULTIFUNCTION)) == PCI_CARDBUS_BRIDGE_TYPE) {
            CapabilityOffset = PciConfig.u.type2.CapabilitiesPtr;
        } else {
            CapabilityOffset =PciConfig.u.type0.CapabilitiesPtr;
        }
        //循环查找 PCI Express 能力结构，并判断是否是 PCIe 设备
        while (CapabilityOffset ! = 0) {
        //读 PCI 能力头
            SetPortVal (PCI_CONFIG_ADDRESS, dwAddr | CapabilityOffset,4);
            GetPortVal (PCI_CONFIG_DATA, (PDWORD) &CapabilityHeader,2);
        if (CapabilityHeader.CapabilityID ==PCI_CAPABILITY_ID_PCI_EXPRESS) {
       // Found the PCI Express capability
        printf("Is a PCIe device, PCIe Capability Pointer: %x\n", CapabilityOffset);
        break;
       } else {
        //根据下一个能力地址,继续查找
       CapabilityOffset = CapabilityHeader.Next;
        }
       }
        if (CapabilityOffset == 0) {
       //没有找到 PCIe 能力结构,不是 PCIe 设备
       printf("Isn't a PCIe device\n");
        }
       }
```

在主程序 main 中对每一个可能存在的设备都调用上述函数得到了如下的结果。这里使用了微软 WDK 中定义的 PCI_COMMON_CONFIG 和 PCI_SLOT_NUMBER 等几个数据结构和若干个常量来简化存取 PCI 兼容配置空间,这些数据结构和常量的定义可以在 Wdm.h 中找到。另外为了让在用户模式下运行的程序可以直接访问 IO 端口，这里使用了 WinIO 驱动。上述程序在 Windows XP SP3 系统下调试运行通过,在 G31 主板的 PC 上枚举的 PCIe 设备如图 2 所示。

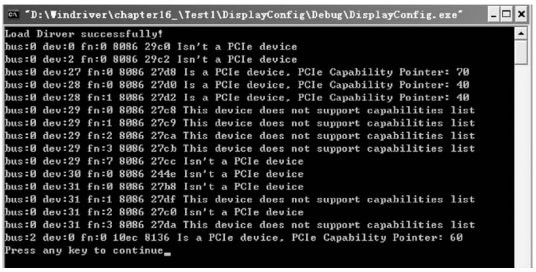

图 2　在 G31 主板上枚举 PCIe 设备图

Fig. 2　PCIe devices enumeration on a G31 motherboard

5　结　论

详细论述了 PCI 设备配置空间的寄存器结构,重点说明了 PCI 能力列表,最后给出了在用户模式下访问 PCI 设备配置空间并确定 PCI Express 设备的方法,此方法可以让应用软件开发人员直接访问系统底层设备。另外由于 PCI Express 向下与 PCI 软件相兼容,该研究设计也适用于 PCI Express 总线设备。

参考文献:

[1] 李贵山,陈金鹏. PCI 局部总线及其应用[M]. 西安:西安电子科技大学出版社,2003:169-171.

[2] Microsoft. PCI, PCI-X, and PCI Express: frequently asked questions[EB/OL]. (2005-11-18)[2010-04-20]. http://www.microsoft.com.

[3] PCI-SIG. PCI local bus specification v3.0 [EB/OL].(2004-02-03)[2010-04-20]. http://www.pcisig.com.

[4] PCI -SIG. PCI.Express. Base. Specification.v1.1 [EB/OL]. (2005-03-28)[2010-04-20]. http://www.pcisig.com.

[5] Microsoft. Windows Driver Kit 7600[EB/OL]. (2009-10-01)[2010-04-20]. http://www.microsoft. com.

[6] 张强,耿爱辉,曹立华,等. CompactPCI 板卡硬件设计与传输速率测试[J]. 光学精密工程,2009,17(8):2047-2052.
ZHANG Qiang,GENG Ai-hui,CAO Li-hua,et al. Design of CompactPCI communication card and its bandwidth test [J]. Optics and Precision Engineering,2009,17(8): 2047-2052.

采用紧凑 0805 封装的 0.5 W 检流电阻 WSLP0805

Vishay Intertechnology, Inc.推出新款表面贴装 Power Metal Strip 电阻——WSLP0805。该器件是业界首款采用紧凑的 0805 封装的 0.5 W 检流电阻,具有 10~50 mΩ 的超低阻值范围,高温性能高达+170 ℃。

WSLP0805 的小尺寸使其能替代更大的检流电阻,节省电路板上的空间,从而为消费者创造出更小、更轻的产品。该电阻适用于计算机的 DC-DC 电源、笔记本电脑的 VRM、锂电池充电管理中的电流检测,以及包括引擎控制、多媒体电子产品、环境控制系统和防抱死制动在内的电子车用系统。

WSLP0805 采用专利技术进行生产,从而实现了极高的功率和低阻值,其全焊接的结构带有一个坚固的镍铬或锰铜合金电阻芯。电阻具有 0.5~5 nH 的超低感值,对 50 MHz 具有优异的频率响应,热 EMF 小于 3 μV/℃。　　咨询编号:2010101001