文章编号:1671-4598(2015)01-0310-03　　中图分类号:TP316.2　　文献标识码:A

基于 RTX 实时系统 ARINC429 总线通信驱动开发

胡　浩
(海军航空工程学院 兵器科学与技术系，山东 烟台　264001)

摘要：某型导弹模拟器采用 ARINC429 总线通信，根据通信协议的实时性要求，通信板卡需要运行在实时环境中，因此设计并实现基于 RTX 实时环境下 PCI 插槽的 ARINC429 总线驱动；在简要介绍 ARINC429 通信板卡和 RTX 实时子系统的基础上，将硬件板卡从 Windows 环境下转换到 RTX 环境，详细介绍了驱动系统结构以及标准驱动的实现步骤和关键技术，并通过实验测试证明，该驱动程序能够准确可靠实现总线数据的接收和发送，且实时性良好。

关键词：RTX；ARINC429 总线；PCI；驱动程序

Development of ARINC429 Bus Driver Based on RTX

Hu Hao
(Department of Armament Science and Technology, Naval Aeronautical and Astronautical University, Yantai　264001, China)

Abstract: A missile simulator uses ARINC429 bus to communicate. Due to the requirement of real-time, the communication card should run in real-time environment. As a result of that, the ARINC429 driver based on PCI slot in RTX is designed and implemented. After the brief introduction of ARINC429 card and the RTX subsystem, the card is switched from Windows to RTX, and then it introduces the architecture of driver system and the development process. Experiments prove that the driver makes the communication precisely and reliably and does well in real-time.

Keywords: RTX; ARINC429; PCI bus; drivers

0　引言

导弹模拟器，作为导弹发控系统的测试设备，能够模拟导弹的电气特性，具备 ARINC429 总线数据传输的能力，主要用于检验、判断导弹发射装置在导弹发射过程中能否正常进行通信，提供飞行参数，下达导弹点火、发射等指令，是考核导弹发控系统性能的重要装置[1]。针对现有的 ARINC429 驱动程序不支持实时扩展系统的问题，进行驱动程序的开发，最终通过实验验证了所设计的驱动程序的性能。

1　ARINC429 通信板卡介绍

Excalibur 公司生产的 EXC-2000PCI 板卡，它具有如下特性[2]：

1) 10 路 429 通信通道，如图 1 所示，每一通道均可以在应用程序中设定为发送或接收；

2) 地址映射，64 k×8 双通道 RAM；

3) 程序可设置缓冲区大小，位速率，奇偶校验；

4) 提供循环或任意次发送数据方式，以及中断或查询接收数据方式；

2　RTX 介绍

RTX (real-time extension，实时扩展) 与 RTOS (实时操作系统) 有很多类似的地方，但是 RTX 并非 RTOS 的一种，它能将 Windows 操作系统扩展到关键时间控制领域，生成一个实时子系统，允许在一台计算机上同时进行高确定性的实时性开发以及非实时性开发[3]。对于 Windows 系统，它不再进行任何的修改和封装，仅在 HAL (硬件抽象层) 添加了实时模块[3]。采用 RTX 平台构建的导弹模拟器既能提高实时性，又具备良好的人机交互能力。

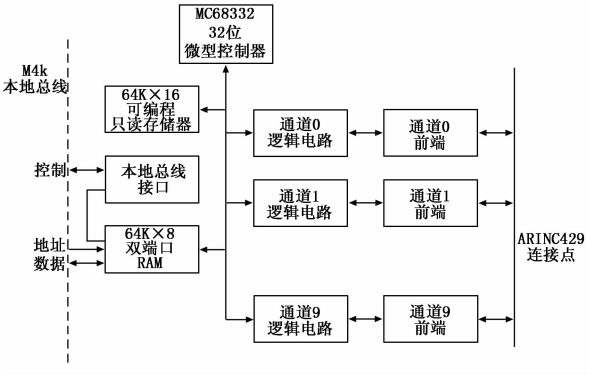


图 1　M4K429RTx 方框图

3　驱动开发

3.1　驱动开发环境配置

使用 Visual Studio 创建驱动开发环境，Visual Studio 能够为 RTX 提供“RTX Application Wizard”、“RTX Driver Wizard”和“RTX Network Driver wizard”向导支持[4]。

计算机内部的硬件默认是工作在 Windows 系统下，而 RTSS 是工作在 Windows 内部的实时子系统，所以需要添加 RTX 的 INF 支持，并将 Windows 下的设备转换到 RTX 子系统中[5]。利用 RTX 提供的“RTX properties”工具，选择目标板卡，单击“Add RTX INF Support”，点击“Apply”即可，如图 2 所示。

收稿日期:2014-05-16；　修回日期:2014-06-23；

作者简介:胡　浩(1989-)，男，湖北襄阳市人，硕士研究生，主要从事导弹发控系统测试及可靠性设计方向的研究。

图 2　RTX 属性工具窗口

3.2　RTX 驱动系统结构

Windows 系统的硬件驱动程序控制硬件上传其状态以及接收的数据。上层应用程序通过驱动程序与底层硬件通信[6]。M4K429RTx 板卡必须通过驱动程序进行参数配置以及处理收发数据，从而实现 ARINC429 总线通信。

RTX 在 Windows 系统下，拥有自己的执行环境和应用程序接口，它可以直接访问硬件资源，使开发驱动程序变得简单[7]。RTX 对硬件的操作仅需要通过遍历命令来找到相应的硬件端口，这样就能跳过驱动层对硬件资源直接操作。

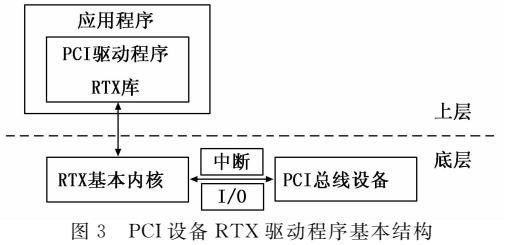
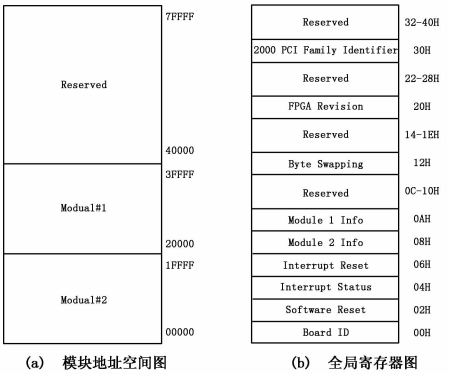


图 3　PCI 设备 RTX 驱动程序基本结构

3.3　板卡实时驱动程序开发

3.3.1　M4K429RTx 存储结构

EXC-2000PCI 板卡有两个存储块：第一个存储块（Base 0）大小为 512 kB，它包含了板卡上所有模块的存储空间，每个模块分配了 128 kB 空间，如图 4（a）所示；第二个存储块（Base 1）大小为 64 B，是全局寄存器，主要提供板卡识别信息，实现软件重置、中断选择等功能，如图 4（b）所示。



(a)　模块地址空间图　　(b)　全局寄存器图

图 4　EXC-2000PCI 板卡存储空间配置

3.3.2　PCI 总线空间结构

PCI 总线设备对配置空间中的参数进行自动配置，实现总线板卡即插即用，配置空间大小为 256 字节，包括 64 字节的头标区和 192 字节的设备相关区[8]，图 5 显示的是头标区的结构定义。

MAX-LAT	MIN-GNT	Interrupt Pin	Interrupr Line	3CH
Reserved=0s				38H
Reserved=0s			Cap.pointer	34H
Expansion ROM BASE Address(not used)				30H
Subsystem ID		Subsystem Vendor ID		2CH
Cardbus CIS Pointer (not used=0s)				28H
Base Address Register#5 (not used)				24H
Base Address Register#4 (not used)				20H
Base Address Register#3 (not used)				1CH
Base Address Register#2 (not used)				18H
Base Address Rgeister # 1 - Global Register				14H
Base Address Rgeister # 0 - Module Memory Space				10H
BIST	Headre Type=0	Latency Timer	Cache Line Size	0CH
Class Code				08H
Status Register		Command Register		04H
Device ID				00H
31 24	23 16	15 08	07 00	

图 5　PCI 板卡配置空间结构

头标区主要包含设备的识别信息、基地址存储位置，规定主机访问方式等。Windows 系统启动后，BIOS 自动配置设备参数。

3.3.3　驱动程序基本函数

1）设备查找：

通过厂家提供的 Vender ID 0x1405 和 Device ID 0x4006 遍历 PCI 插槽，直到找到相匹配的设备[9]。

```
BOOL RTFCNDCL FindDevice ()
{ for (PCIbus; Flag; PCIbus++) {
for (i = 0; i < PCI_MAX_DEVICE && Flag; i++) {
SlotNumber. u. bits. DeviceNumber = i;
for (j = 0; j < PCI_MAX_FUNCTION; j++) {
if ((PciData ->VenderID == 0x1405) && (PciData -> DeviceID == 0x4006)) {
liPhysAddr. QuadPart = PciData -> u. type0. BaseAddresses [2];
liPhysAddr_IOControl. QuadPart = PciData -> u. type0. BaseAddresses [3];
Flag = FALSE;
retVal = TRUE;
break;}}}}
return retVal;}
```

2）设备初始化：

把步骤 1）得到的基地址转换为实时系统下能够访问的虚拟地址，以便于通过基地址和偏移地址组合的方式访问设备上的资源[10]。

```
if (! RtTranslateBusAddress (PCIBus, 0, BAR1, AddressSpace, &BAR1))
{
vBAR1 = (PCHAR) tBAR1. LowPart;
A429_PORT0_BASE0 = (PUCHAR) vBAR1 - 1;
}
```

3）设备读写操作

直接对 IO 端口读写实现设备资源访问。

```
RtWritePortUchar ( (A429_PORT0_BASE0 + 0x58), 0x0);
RtWritePortUchar ( (A429_PORT0_BASE0 + 0x5A), 0x01);
```

```
RtWritePortUchar ( (A429 _ PORT0 _ BASE0 + 0x5B), 0x02);
```

3.3.4 驱动程序功能函数

上节介绍的基本函数主要是对 PCI 设备进行初始化，不能满足应用程序对板卡操作的要求，因此，有必要设计其他的功能函数。

1）设备结构体：

```
typedef struct
{
PCHAR pBase0Addr;
PCHAR pBase1Addr;
UINT32 addrRange; // 板卡的空间地址范围
CARD _ STATUS cardStatus // 板卡状态信息
HANDLE interruptHandle; // 中断句柄
BOOL intIsEnable; // 中断开启标志
PCI _ SPACE _ INFO pciInfo;
} A429Card;
// 将通信板卡的相关信息定义一个结构体
A429Card a429Device;
```

2）读写函数

```
BOOL A429Card _ ReadData (UINT32 offset, UINT32 *data)
{
ULONG DATA1 = 0;
A429Card * pA429Card;
pA429Card = &a429Device;
*data = *((ULONG*)(pA429Card -> pBase0Addr + offset));
return TRUE;
}
BOOL A429Card _ WriteData (UINT32 offset, UINT32 *data)
{
ULONG DATA1 = 0;
A429Card * pA429Card;
pA429Card = &a429Device;
*((ULONG*)(pA429Card -> pBase1Addr + offset)) = data;
return TRUE;
}
```

3）用户接口函数

依据实际所需的通信功能，定义下列函数。

void initA429Card ()，查找 PCI 设备，对板卡进行初始化，填充设备结构体；

void openA429Card ()，打开板卡；

void A429SelfTest ()，板卡进行自检，确定工作状态正常；

void closeA429Card ()，关闭板卡，释放所占用的资源。

3.3.5 驱动程序应用

实时子系统的多个进程间、实时子系统与 Windows 进程之间均采用共享内存。根据功能要求，自定义如下操作共享内存的函数：

1）int CreateMemory (mdata * M, LPCTSTR pSharedName)，创建共享内存，并创建相对应的信号量事件；

2）int OpenMemory (mdata * M, LPCTSTR pSharedName)，打开信号量事件以及共享内存；

3）int CloseMemory (mdata * M, LPCTSTR pSharedName)，关闭信号量事件以及共享内存；

4）int ReadMemory (mdata * M, USHORT * Num)，从共享内存取出数据，并进行解析；

5）int WriteMemory (mdata * M, UINT Num)，向共享内存写入数据；

4 驱动程序测试

根据数据总线通信要求，驱动程序能够准确无误地实现 ARINC429 数据收发，并且保证发送的实时性，即每 10 ms 定时接收和发送数据。

将 M4K429RTx 板卡的通道 0 与通道 1 直接连接，在两个通道之间发送总线数据，形成回绕测试，对比通信数据，判断总线是否能够正常传输数据，图 6 显示了测试的软件界面结果。同时利用高分辨率的示波器监视数据传输过程，观察记录数据传输之间的时间间隔，测试波形如图 7 所示。

图 6 数据传输结果

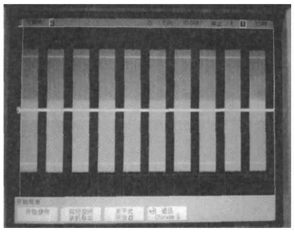


图 7 示波器显示波形

图 6 和图 7 测试结果表明，接收数据与发送数据完全一致，驱动程序准确可靠驱动板卡完成数据发送接收任务。通过对示波器的显示波形长期观测，证明了数据传输的时间间隔稳定在 10 ms，满足了实时性要求。

5 结束语

针对导弹模拟器研制过程中对通信的实时性要求，开发了 Windows XP 操作系统下基于 RTX 实时扩展包的 M4K429RTx 板卡驱动程序。

实验表明，所开发的驱动程序极大发挥了 429 板卡的性能，克服了 Windows 系统在实时性上的局限性。该驱动程序已经成功运用于某型导弹模拟器。

参考文献：

[1] 刘明阳，张建东，庞 敏. 导弹任务器与导弹模拟器仿真系统设计与实现 [J]. 计算机测量与控制，2012，20 (9)：2460 - 2462.

[2] M4K429RTx User's Manual [Z]. Excalibur Systems, Inc. July 2009, Rev A - 5.

[3] 王 丰，刘 娜，肖雅静等. RTX 在实时控制系统中的应用 [J]. 电子工业专用设备，2012，214 (11)：40 - 43.

[4] 罗伟栋，徐广宏，兰宇飞，等. 一种气相色谱仪的实时测控系统设计 [J]. 电子测量技术，2013，36 (2)：123 - 126.

[5] 刘正雄，黄攀峰，闰 杰. RTX 下高分辨率精确授时系统设计 [J]. 计算机测量与控制，2011，19 (7)：1755 - 1757.

[6] 韩玉芹. 基于 RTX 子系统的导弹试验实时测控系统研发 [D]. 西安：西安电子科技大学，2013.

[7] 刘心语，杨玉普，给予 RTX 实施环境的 1394 驱动程序开发 [J]. 微计算机信息，2009，25 (11)：75 - 77.

[8] 程海全，徐抒岩，胡 君. PCI 设备配置空间研究 [J]. 电子设计工程，2010，18 (10)：1 - 4.

[9] 应三丛，汪明寅，张 行. 高性能 PCI 驱动程序的关键技术 [J]. 计算机工程与设计，2012，33 (6)：2208 - 2212.

[10] 杨志强，王厚军，李 力. Linux 系统下的 PCI 串口设备驱动 [J]. 电子测量技术，2011，34 (9)：58 - 68.