文章编号:1671-4598(2022)05-0209-06 DOI:10.16526/j.cnki.11—4762/tp.2022.05.036 中图分类号:TP319 文献标识码:A

基于 VxWorks 的 CPCI 多通道卡驱动设计与实现

胡广浩，阮福明，赵希昉，黄德友，井中武
（中海油田服务股份有限公司 物探事业部，天津 300450）

摘要：为解决大规模海上拖缆地震勘探对控制系统的实时性和处理效率问题，提出了采用 CPCI 工控机箱为硬件平台，VxWorks 实时操作系统为软件处理平台，设计和实现了一套 CPCI 多通道卡驱动程序；通过分析 VxWorks 驱动程序结构和 CPCI 总线设备特点，重点给出了从内存映射模块、中断注册初始化模块和中断处理模块等方面进行 CPCI 多通道卡驱动程序的设计方法、实现过程和关键代码；集成 CPCI 多通道卡驱动程序的“海燕”拖缆定位与控制系统具备 12 个通道数据处理能力，多次成功应用到海上生产作业中，其实时性和处理效率满足大规模海上拖缆地震勘探对控制系统的要求；CPCI 多通道卡驱动程序设计合理，易于扩展到其他具有多通道、多任务、实时性要求高的嵌入式数据采集系统中。

关键词：地震勘探；VxWorks；CPCI 驱动程序

Design and Implementation of CPCI Multi-channel Card Driver Based on VxWorks

HU Guanghao，RUAN Fuming，ZHAO Xifang，HUANG Deyou，JING Zhongwu
（Geophysical Department，China Oilfield Services Limited，Tianjin 300450，China）

Abstract：In order to solve the problems of real-time performance and processing efficiency of the control system for large-scale offshore towed-streamer seismic exploration，a set of CPCI multi-channel card driver is designed and implemented by using CPCI industrial control chassis as hardware platform and VxWorks real-time operating system as software processing platform. By analyzing the driver structure of VxWorks and the characteristics of CPCI bus equipment，the design method，the process of implement and key programming code of CPCI multi-channel card driver are emphatically given from the aspects of memory mapping module，interrupt registration initialization module and interrupt processing module. “Haiyan” streamer positioning and control system of integrating CPCI multi-channel card driver has 12 channels data processing capacity and has been successfully applied to offshore production operations for many times，which meets the real-time requirements and processing efficiency requirements of offshore towed-streamer seismic exploration for control system. CPCI Multi-channel card driver designed is reasonable and easy to be extended to other embedded data acquisition systems of multi-channel，multi-task and high real—time requirements.

Keywords：seismic exploration；VxWorks；CPCI driver

0 引言

中国海油开展自主海上高精度地震勘探装备研发[1]，成功研制出具有自主知识产权的“海亮”高精度拖缆地震采集系统、“海燕”拖缆定位与控制系统（控制系统）、“海途”综合导航系统、“海源”气枪震源控制系统，通过多次试验和生产作业，实现了海洋地震拖缆采集装备产业化应用，关键技术跻身国际先进行列，为实现我国海上油气增产、保障国家能源安全、建设科技强国做出积极努力。

控制系统是成套地震勘探装备的重要组成部分，它由船载控制系统和水下控制器组成。它通过控制挂载在拖缆上的罗经鸟的翼板垂直方向移动实现拖缆拖缆沉放深度控制，通过控制水平鸟翼板水平方向移动实现拖缆间距控制，通过控制声学鸟和水平鸟发射声学信号进行声学网络测距，最后通过实时采集深度数据、航向数据、相对位置数据等，给综合导航系统提供原始测量数据用于坐标实时解算，最终给地震数据提供位置信息。随着勘探船拖带能力提升和拖缆地震勘探向深海发展需要[2]，当前我国勘探船拖带的拖缆规模达到 12（缆）*12 km，每种拖缆上挂载的控制器按照 300 米间隔布放，控制系统需要在特定的工作周期内完成 12（拖缆）*120（控制器）规模的实时控制和数据采集，对数据传输的实时性和处理效率要求极高。

1 系统结构及原理

VxWorks 是一款嵌入式强实时操作系统（RTOS，real-time operating system)[3]，以其高可靠、高性能、可剪裁和卓越的实时性被广泛地应用于通信、军事、工业控制、航空航天、地震勘探[4-7]等高精尖技术及实时性要求极高的

收稿日期:2021-11-12； **修回日期**:2022-01-05。
基金项目:国家高技术研究发展计划(863 计划)(2012AA09A211)部分研究成果。
作者简介:胡广浩(1983-)，男，山东滕州人，硕士研究生，工程师，主要从事地震勘探装备软件系统研发方向的研究。
引用格式:胡广浩，阮福明，赵希昉，等. 基于 VxWorks 的 CPCI 多通道卡驱动设计与实现[J]. 计算机测量与控制，2022，30(5)：209-214.

领域中。紧凑型外设部件互连标准（CPCI，compact peripheral component interconnect）总线具有高开放性、高可靠性、可热插拔、带宽高等特点，使其可以广泛应用在通讯、网络、实时系统控制（RTMC，real time machine control）、实时数据采集（RTDA，real-time data acquisition）等需要高速运算、高速数据通信、高可靠度、高扩展性的应用领域。本文控制系统的设计既要考虑到实时性和处理效率问题，又要具备一定的扩展性要求，因此采用紧凑 CPCI 工控机箱作为硬件平台。CPCI 工控机箱的系统槽位运行系统主控板，采用单板计算机和 VxWorks 实时操作系统软件平台，非系统槽位运行定制开发的多通道拖缆控制器接口卡（多通道卡），系统主控板与多通道卡基于 CPCI 背板总线连接，系统结构如图 1 所示。

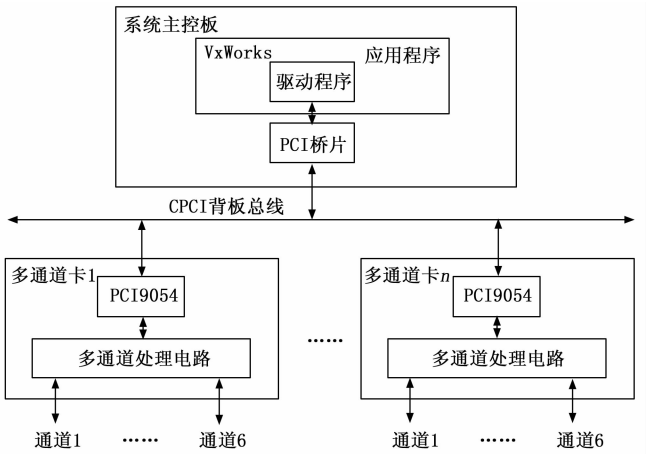

图 1 控制系统结构框图

多通道卡由 CPCI 总线接口芯片和多通道处理电路模块组成，CPCI 总线接口芯片采用 BROADCOM 公司的 PCI9054 芯片，多通道处理电路模块与挂载在拖缆上的水下拖缆控制器通讯，每个多通道处理电路模块拖带 6 个通道。控制系统应用程序运行于系统主控板，多通道卡驱动程序是系统主控板应用程序和多通道卡之间通讯的“桥梁”，它通过对 PCI9054 接口芯片的控制实现应用程序与多通道处理电路 6 个通道的数据通讯，从而实现控制系统应用程序对水下拖缆控制器的控制。运行于系统主控板的多通道卡驱动程序和应用程序均运行于 VxWorks 实时操作系统环境中，多通道卡驱动程序的设计支持多个多通道卡的并发控制，从而为应用程序提供了灵活的扩展能力。

2 VxWorks 下驱动程序结构和 CPCI 总线设备

VxWorks 采用分层设计和结构化设计，由性能高效的微内核 Wind、VxWorks 库、网络协议栈、I/O 系统、文件系统、板级支持包（BSP，board support package）[8-10]、各类驱动程序等组成。BSP 和 PCI 驱动程序位于硬件与操作系统和应用程序之间，它们在系统中的位置如图 2 所示。BSP 是一个软件抽象层，它的主要功能是系统加电以后初始化目标机硬件、初始化操作系统及提供部分硬件的驱动程序，PCI 驱动程序是操作系统和应用程序与具体硬件之间的“桥梁”，对上为 VxWorks 提供标准操作接口和应用程序提供专用接口，对下直接控制硬件设备。对于 BSP 和 PCI 驱动程序共同构成了应用程序和具体物理设备的中间层，这种设计思想可以实现应用层软件与具体物理设备的隔离，提高应用程序的可移植性。本文中系统主控板采用成熟的商用单板计算机，提供了基本的 BSP 开发包，因此只需要对其进行功能扩展，即可满足定制的设备。

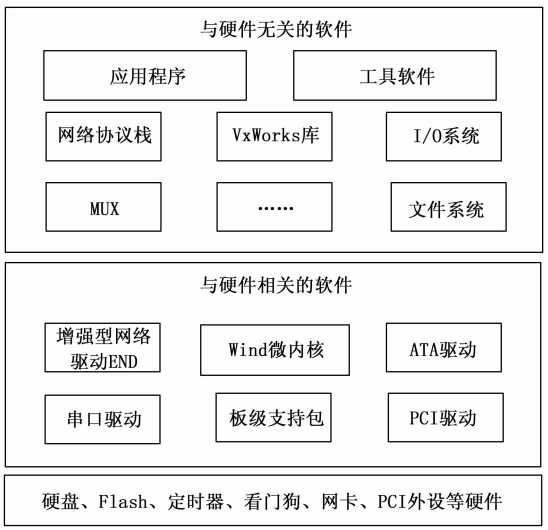

图 2 板级支持包和驱动程序

外设部件互连标准（PCI，peripheral component interconnect）是由外围部件互连专业组（PCI-SIG，peripheral component interconnect special interest group）推出的一种局部并行总线标准，PCI 总线能够根据设备上的配置信息自动为设备分配物理地址空间、输入输出（I/O，input/output）端口号、中断号，并且支持多种外部设备[11]。本文 CPCI 工控机箱采用的背板通信协议是一种基于标准 PCI 总线的紧凑坚固的高性能总线技术，它定义了更加坚固耐用的 PCI 版本，在电气、逻辑和软件方面，它与 PCI 标准完全兼容，因此 CPCI 多通道卡驱动程序的设计完全兼容 PCI 标准协议。

根据控制系统功能需求，结合 VxWorks 驱动程序结构和 PCI 总线特点，将多通道卡驱动程序开发分为内存映射模块、中断注册初始化模块和中断处理模块的设计和实现。

3 多通道卡驱动程序设计

3.1 PCI 配置空间

根据 PCI 规范，每一个 PCI 总线设备有 3 种相互独立的物理地址空间：配置寄存器空间、存储器空间和 I/O 空间。其中配置寄存器空间是其容量为 256 个字节，其中前 64 个字节为配置头标区，该区域是固定结构，后 192 个字节为本地配置空间（设备关联区），主要定义卡上局部总线的特性、本地空间基地址及范围等，寄存器布局因设备而异[12-15]。

配置寄存器空间是 PCI 设备所特有的一个物理空间，支持设备即插即用，因此 PCI 设备不占用固定的内存地址空间或 I/O 地址空间，而是由操作系统决定其映射的基址。配置寄存器空间里最重要的包括厂商标识（Vendor ID）、设备标识（Device ID）、基地址寄存器（BAR，base address register）、中断号（IRQ Line）、中断引脚（IRQ Pin）等。配置寄存器空间里基地址寄存器 PCIBAR［0－5］反映了该 PCI 设备的地址空间映射到系统内存空间或 I/O 空间的起始物理地址。基地址寄存器的最低位 bit0 是只读的，其值为 1 表示该寄存器关联的地址空间是要映射到 I/O 空间，否则映射到系统存内存空间的。本文 PCIBAR0 用于配置本地寄存器、运行时寄存器和 DMA 寄存器到系统内存空间映射，PCIBAR2 用于本地地址空间到系统内存空间映射。基于 PCIBAR0 的运行时状态控制寄存器 INTCSR 用于多通道卡中断状态获取和控制，基于 PCIBAR2 的寄存器用于多通道数据接口和本地中断源管理。中断号的分配由系统在硬件上电阶段自动分配，驱动程序只需要获取到中断号，通过挂接中断服务程序即可处理多通道卡触发的数据请求。主要寄存器设计如表 1 所示。

表 1　主要寄存器定义

偏移地址	寄存器	基址	类型	说明
68 h	INTCSR	PCIBAR0	运行时	中断控制状态寄存器 Bit8：PCI 中断使能/禁用 Bit11：本地中断使能/禁用 Bit15：本地中断激活状态 通常将本地中断标志位和 PCI 中断标志位组合使用
10 h	INTMAP	PCIBAR2	自定义	中断源寄存器 Bit0～Bit5：分别对应 1～6 号通道 对应通道数据读取完后自动复位
14 h	INTMASK	PCIBAR2	自定义	中断屏蔽寄存器 Bit0～Bit5 分别对应 1～6 号通道 用于通道数据传输使能和禁用
0x18	CHN1R_R	PCIBAR2	自定义	通道 1 读数据接口
…	…	…	…	…
0x2C	CHN6R_R	PCIBAR2	自定义	通道 6 读数据接口
0x38C	HN1R_W	PCIBAR2	自定义	通道 1 写数据接口
…	…	…	…	…
0x4C	CHN6R_W	PCIBAR2	自定义	通道 6 写数据接口

基于基地址寄存器 PCIBAR2，分别定义 6 个通道读寄存器和 6 个通道写寄存器用于数据读取和发送，定义 1 个中断源寄存器和 1 个中断屏蔽寄存器，分别用于 6 个通道的中断标志和屏蔽控制。对于 PCI 总线设备的配置和控制，VxWorks 提供了 pciFindDevice（）、pciConfigOutLong（）、pciIntConnect（）等专用应用程序编程接口（API，application programming interface）用于驱动程序开发[16～19]，下面在具体的实现过程中介绍其使用方法。

3.2　内存映射模块

VxWorks 在 BSP 的支持下，自动为 PCI 设备分配 4k＊16 字节大小内存映射空间[8-9]。VxWorks 的内存管理单元（MMU，memory management unit）仅具备基本的内存管理功能[20]，只有将 PCI 设备的存储空间加入到 MMU 中，应用程序才能像操作内存空间一样操作设备存储空间。VxWorks 中页大小在的 BSP 的 VM _ PAGE _ SIZE 定义，默认为 4k 字节，如果需要改变这个值，需要重新重定义 config. h 中的 VM _ PAGE _ SIZE 的值。vmLib. h 中的数据结构 PHYS _ MEM _ DESC 用于定义映射物理内存的参数，内存映射用 sysPhysMemDesc（被声明为一个 PHYS _ MEM _ DESC 的数组）在 sysLib. c 中定义。在 sysPhysMemDesc 中定义的内存区，必须是页对齐的，并且必须跨越完整的页，因此 PHYS _ MEM _ DESC 的最初 3 个域都必须是 VM _ PAGE _ SIZE 的倍数。如果在初始化过程中提供的 sysPhysMemDesc 的元素不是页对齐的，则在 VxWorks 初始化期间将导致系统崩溃。有手动配置和自动配置两种方式实现 MMU 对 PCI 存储空间的映射，前者在 BSP 的 sysLib. c 文件中将设备存储空间手动添加到内存管理表 sysPhysMemDesc 中，后者通过 sysMmuMapAdd（）函数动态将设备的存储空间地址加入到 MMU 中。本文基于后者实现方式，通过扩展板级支持包[8-9]实现内存映射，以获得更好的灵活性和扩展性，实现过程如图 3 所示。

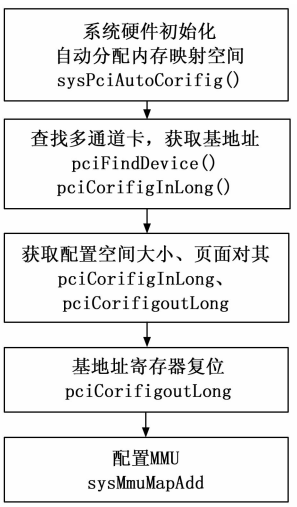

图 3　内存映射模块流程图

1）系统硬件初始化入口：在 BSP 的支持下，VxWorks 操作系统初始化过程中，通过调用 sysPciAutoConfig（）完成系统硬件设备的初始化，主要通过中断号分配、内存映射空间分配、I/O 地址等完成硬件所需资源分配；

2）查找多通道卡：根据设备 vendorID（0x10B5）和 deviceID（0x9054）采用 pciFindDevice（）找到对应的设备，获取到的总线号 busNo、设备号 deviceNo、功能号

FuncNo，总线号 busNo、设备号 deviceNo、功能号 FuncNo 组合在一起可以作为该设备的唯一标识；

3）获取基地址：采用 pciConfigInLong（）函数获取设备的寄存器基地址 PCIBAR2，获得的寄存器基地址与存储器屏蔽位 PCI _ MEMBASE _ MASK 逻辑与，得到内存映射基地址；

4）获取映射空间大小：根据 PCI 规范，采用 pciConfigoutLong（）、pciConfigInLong（）行数将基地址寄存器全部写入 1 再读回，得到映射空间大小，然后对映射空间进行 MMU 页面对齐，确定空间长度；

5）基地址寄存器复位：将 mapBaseCsr 回写到基地址寄存器复位，用于后续其他应用需要。

在 BSP 的 sysLib.c 中的 sysHwInit（）函数中扩展内存映射功能，关键代码为：

```
/* 获取多通道卡设备的总线号、设备号、功能号 */
if(OK == pciFindDevice(VENDERID, DEVICEID, num,
&busNo,&deviceNo,&funcNo)){
/* 获取 PCIBAR2 内容 */
pciConfigInLong(busNo,deviceNo,funcNo,PCI_CFG_BASE_
ADDRESS_2,&PCIBAR2));
/* 将 PCIBAR2 全部写入 1 */
pciConfigOutLong(busNo,deviceNo,funcNo,PCI_CFG_BASE_
ADDRESS_2,0xffffffff));
/* 再读回 PCIBAR2 内容,得到分配内存空间大小 */
pciConfigInLong(busNo,deviceNo,funcNo,PCI_CFG_BASE_
ADDRESS_2,&mapSize))
/* 将分配内存空间大小进行 4k 页面对齐 */
mapSize=(~(mapSize&PCI_MEMBASE_MASK))+1;
mapSize=ROUND_UP(mapSize,VM_PAGE_SIZE);
pciConfigOutLong(busNo,deviceNo,funcNo,PCI_CFG_BASE_
ADDRESS_2,PCIBAR2));
PCIBAR2&=PCI_MEMBASE_MASK;
/* 动态将映射关系加入到 MMU */
sysMmuMapAdd((void*)(PCIBAR2),mapSize,VM_STATE_
MASK_FOR_ALL,VM_STATE_FOR_PCI);
}
```

3.3 中断注册初始化模块

VxWorks 启动过程中，通过板级支持包完成多通道卡存储空间的 MMU 管理后，应用程序就可以像操作内存空间一样操作多通道卡的存储空间。本文为了提高模块的独立性和移植性，将内存映射模块和中断注册初始化模块进行分离，在中断注册初始化过程中，需要再次获取内存映射基地址，之后中断服务处理模块和应用层软件可以通过操作内存空间的方式对多通道卡的存储空间进行存取。

VxWorks 采用中断服务表对中断进行管理，中断服务表维护了中断向量和中断服务程序（ISR，interrupt service routines）的入口地址对应关系。多通道卡有数据到来时采用中断的方式通知 VxWorks 系统，以保证数据能得到及时处理，达到实时性要求[20]。为了处理多通道卡产生的中断，必须获取其在系统中的中断号，这个中断号是 VxWorks 系统启动时通过 PCI 库进行自动分配。根据 PCI9054 配置空间定义，我们首先得到的是一个 8 位的中断号，这种中断号需要通过 INT _ NUM _ GET 转换才能得到该设备在 VxWorks 系统中的中断号[20]。

获取到中断号之后，通过 INUM _ TO _ IVEC 转换成中断向量，最后将中断向量和 ISR 的地址在操作系统的中断管理模块中注册。VxWorks 提供 intConnect（）和 pciIntConnect（）两种注册 ISR 的方式，intConnect（）使用的中断向量是独占的，而 pciIntConnect（）是共享的，即同类型的多个外部设备可以共享同一个中断向量，它在内部使用一个链表管理多个 ISR，发生中断时，链接在一个链表上的各个 ISR 被依次调用，pciIntConnect（）要求每个 ISR 被调用时，应该首先查询是否为自己的设备产生的中断，不是则应立即返回，以继续调用其它 ISR。控制系统在设计上支持多个多通道卡，因此采用共享中断向量的方式对多通道卡 ISR 进行注册，可获得更高的扩展性和复用性。完成 ISR 在 VxWorks 的中断系统中注册后，使能 PCI 中断和本地中断，此后系统就可以处理多通道卡的数据请求。

多通道卡初始化的关键代码为：

```
/* 获取多通道卡设备的总线号、设备号、功能号 */
if(OK == pciFindDevice(VENDERID, DEVICEID, num,
&busNo,&deviceNo,&funcNo)){
/* 获取 PCIBAR2[num] 内容,用于应用程序访问 */
pciConfigInLong(busNo,deviceNo,funcNo,PCI_CFG_BASE_
ADDRESS_2,&PCIBAR2[num]));
/* 获取中断号 */
pciConfigInByte(busNo,deviceNo,funcNo,PCI_CFG_DEV_
INT_LINE,&irqNo))
PCI_DEV[num].irqNo = irqNo;
/* x86 架构采用 8259A 中断控制器,物理中断号转换为系
统中断号 */
intNum = INT_NUM_GET(irqNo);
sysIntDisablePIC(irqNo);
/* 注册中断服务程序 */
pciIntConnect(INUM _ TO _ IVEC(intNum),(VOID-
FUNCPTR)pciFastISR,num);
sysIntEnablePIC(irqNo);
/* 获取中断控制状态寄存器 */
PCI9054_Read_PCIBAR0(num,INTCSR,&intCSR); intC-
SR|= INTCSR_Bit8|INTCSR_Bit11;
/* 使能 PCI 中断和本地中断 */
PCI9054_Write_PCIBAR0(num,INTCSR,intCSR)。
```

3.4 中断处理服务模块

标准的 PCI 设备驱动程序是 VxWorks 体系结构中的 I/O 系统重要组成部分，它通过 BSP 访问 PCI 设备，往上为应用程序提供系统调用入口，从而实现应用层程序与 PCI 设备的交互。本文采用自定义实现方式，通过跨过 I/O 系统，采用中断服务程序和应用层任务的直接通讯，获得更高的运行效率和实时性。

VxWorks 系统的强实时性得益于其独特的中断处理模型，快速的硬件中断处理是其核心的组成部分。为了尽可能快速响应外部中断，VxWorks 的中断服务程序运行在一个不同于任何任务的独立的上下文中，中断处理过程不涉及到任务的上下文切换[8,10]。操作系统捕获到中断后，对注册到同一个中断向量的中断服务程序进行遍历，通过回调函数完成对相应中断服务程序的调用。在执行中断过程中可能会有新的设备中断到来，为了保证新的更紧急的中断能得到及时处理，这就要求我们设计的中断服务程序尽可能的简短而高效。本文设计如图 4 所示处理模型，采用中断快速处理和中断处理任务组合的方式完成一次多通道卡数据传输请求，中断快速处理进行数据传输请求的中断快速响应，中断处理任务进行实际数据传输任务。

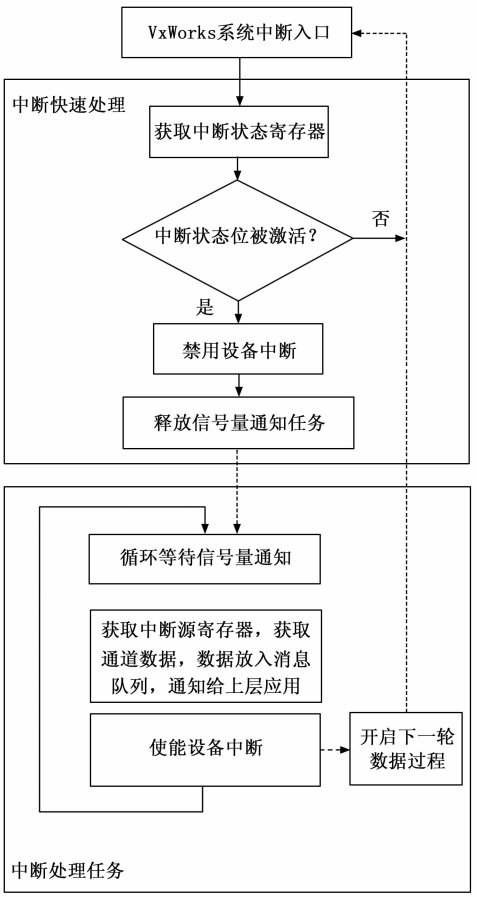

图 4　中断处理模块结构框图

多通道卡有数据传输请求时，产生系统中断，中断快速处理由操作系统中断服务程序自动调用。中断快速处理运行在操作系统中断服务的上下文中，是实现整个中断服务过程的“前半部分”，仅完成必须的工作，包括清除通道中断标记，给中断处理任务发出信号量通知，使能通道中断等，运行极其高效。“前半部分”中断快速处理关键代码如下：

```
void pciFastISR(intnum){
UINT32 intCSR;
/* 获取中断控制状态寄存器 */
PCI9054_Read_PCIBAR0(num,INTCSR,&intCSR);  if (INTCSR_Bit15 & intCSR){
/* 禁用 PCI 中断和本地中断设备中断 */
PCI9054_Write_PCIBAR0(num,INTCSR,intCSR&(~(INTCSR_Bit8|INTCSR_Bit11));
/* 释放信号量,通知上层任务 */
semGive(PCI_DEV[num].semBID);
}}
```

中断处理任务实现中断服务的“后半部分”功能，以高优先级任务的形式由 VxWorks 进行任务调度，处理相对比较耗时操作，结束后使能下一轮数据传输。中断处理任务启动后，循环等待中断快速处理的信号量通知，读取中断源寄存器 INTMAP，依次遍历 6 个标志位判断中断源，获取中断源后，通过读取对应的通道接口获取数据长度和内容，然后将数据放入消息队列通知应用层软件，最后使能 PCI 中断和本地中断，通知多通道卡可进行下一轮数据传输。“后半部分”中断处理任务的关键代码如下：

```
void pciDealTask(int num){
while(1){
/* 等待中断快速处理的信号量通知 */
semTake(PCI_DEV[num].semBID,WAIT_FOREVER));
UINT32 intMap;
/* 获取中断源 */
PCI9054_Read_PCIBAR2(num,INTMAP,&intMap);  /* 依次遍历中断源标志位 */
for(int chl= 0; chl< 6; ++chl){
if(intMap & (0x01 << channel)){
/*通过 PCI9054_Read_PCIBAR2(num,CHN1R_R + chl*4,&data);执行获取数据过程,放入消息队列给上层应用软件使用,不再详述 */
}}
/* 使能 PCI 中断和本地中断,使能下一轮数据传输 */
PCI9054_Write_PCIBAR0(num,INTCSR,(PCI_INT_BIT|LOCAL_INT_BIT));
}}
```

4　实验与应用结果

验证文中多通道卡驱动程序，采用模拟测试和实际应用测试相结合的方式。模拟测试通过在主控板软件编写模拟程序和多通道卡内部编写回环程序，验证 2 个多通道卡共计 12 个通道的数据并发读写功能正确性和数据处理实时性；实际应用测试通过将驱动程序与主控板应用层软件进行功能集成，应用到控制系统的生产应用中，验证驱动程序的可用性。

4.1　模拟测试

模拟程序首先完成测试任务初始化，通过 pciFindDevice () 查找多通道卡，找到后针对每个通道创建测试任务，创建测试任务代码如下：

```
taskId_simu[slotNo][channelNo]= taskSpawn(strcat("simu_",slotNo<<8|channelNo),200,VX_FP_TASK,0x4000,
```

(FUNCPTR)tSimuTask,slotNo,channelNo,0,0,0,0,0,0,0,0)。

由于每个通道对应一个测试任务，测试任务采用随机休眠 10～20 ms（控制系统与水下罗经鸟、水平鸟、声学鸟的通讯间隔为约 80 ms）的形式模拟多个通道的并发访问。测试任务通过随机产生数据，下发至多通道卡所对应通道，接着阻塞在中断处理服务的消息队列中，直到有数据返回，将获得数据后与下发的数据进行比较，实现流程如图 5 所示。

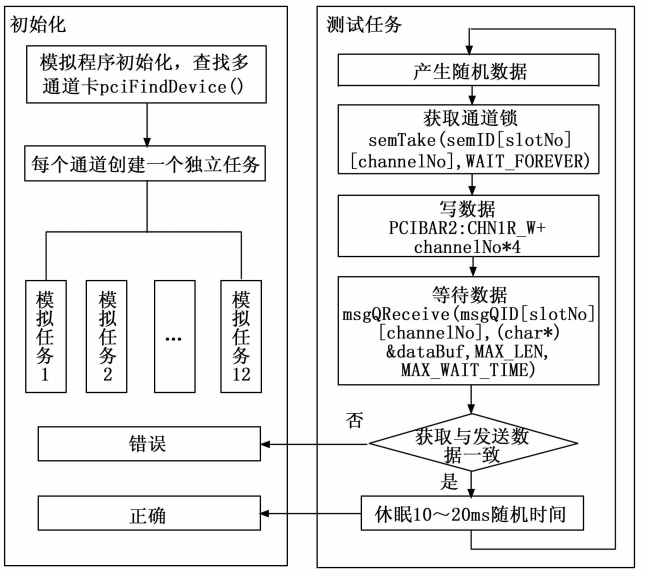

图 5 测试程序流程图

通过对 12 个通道连续并发测试 10 万次，数据传输稳定、正确，验证了驱动程序的正确性和数据处理实时性满足设计预期。

4.2 应用效果

通过将多通道卡驱动程序与应用层软件进行功能集成，形成了“海燕”拖缆定位与控制系统。作为中国海油自主海上高精度地震勘探装备的重要组成部分，控制系统在装配 2 个多通道卡情况下，形成了 12 通道并发处理能力，完成多次实际作业应用，验证其在拖缆深度控制、间距控制、声学测距控制等方面均能满足 12 条拖缆大规模海上拖缆地震勘探作业对数据传输实时性和处理效率的要求。集成多通道卡驱动程序的控制系统在实际生产作业中的成功应用表明，本文设计和实现的多通道卡驱动程序结构合理、功能正确、性能高效，满足实际野外生产需求，达到了设计的预期。

5 结束语

本文从分析 VxWorks 实时操作系统驱动程序结构和 CPCI 总线设备特点入手，依次从 VxWorks 下内存映射模型、中断注册初始化过程和中断处理过程等方面阐述了 VxWorks 下 CPCI 设备驱动程序开发的通用方法，并结合“海燕”拖缆定位与控制系统 CPCI 多通道卡实例给出了具体过程和关键程序代码。本文实现的 CPCI 多通道卡驱动程序集成到控制系统中并成功应用于生产作业中，作业效果表明该驱动程序的设计和实现能满足海上拖缆地震勘探对控制系统的实时性和处理效率要求，达到了设计预期。该驱动程序的设计在不失实时性的前提下，采用了模块块化设计、通用性设计，使之能够灵活的应用到其他具有类似多通道、多任务、实时性要求高的嵌入式系统中。

参考文献：

[1] 阮福明，吴秋云，王 斌，等. 中国海油地震勘探采集装备技术研制与应用 [J]. 中国海上油气，2017，29 (3)：19－24.
[2] 李绪宣，朱振宇，张金淼. 中国海油地震勘探技术进展与发展方向 [J]. 中国海上油气，2016，28 (1)：1－12.
[3] 王学龙. 嵌入式 VxWorks 系统开发与应用 [M]. 北京：人民邮电出版社，2003.
[4] 李艳军，罗兰兵，李兴华. 嵌入式实时操作系统 VxWorks 及其在地震勘探仪器中的应用 [J]. 物探装备，2009，19 (4)：227－232.
[5] 张 朋，李春涛. 基于 VxWorks 的小型无人机飞行控制软件设计 [J]. 计算机测量与控制，2014，22 (8)：2687－2691.
[6] 贾献强，魏延辉，高延滨，等. 基于 VxWorks 的水下机器人通信系统设计 [J]. 计算机测量与控制，2015，23 (10)：82－82.
[7] 赵 洋，吴昭辉，姚跃民，等. 基于 VxWorks 的飞行器模飞测试方法研究 [J]. 计算机测量与控制，2020，28 (3)：9－13.
[8] 隋 霞，许录平. 基于 VxWorks 的 BSP 技术分析 [J]. 微计算机信息，2006，22 (8－2)：86－88.
[9] 周启平，张 扬. VxWorks 下设备驱动程序及 BSP 开发指南 [M]. 北京：中国电力出版社，2004：21－192.
[10] 张 杨，于银涛. Vxworks 内核、设备驱动与 BSP 开发详解 [M]. 北京：人民邮电出版社，2009：140－260.
[11] 王宇磊，周 东. VxWorks 下 PCI 总线驱动设计与实现 [J]. 计算机应用，2005，7 (4)：45－48.
[12] 朱爱红，王国卫，韩建立，等. 基于 FPGA 的 PCI 接口方法研究 [J]. 电子测量仪器学报，2008 (增刊 4)：128－130.
[13] 朱欣华，黄亮亮，夏云翔，等. 基于 PCI 总线的 CAN 接口卡及其 VxWorks 环境下驱动程序的设计 [J]. 电子器件，2006，29 (4)：1275－1279.
[14] 何 莉，龚宗洋，张为公，等. 基于 CPCI 总线的运动控制卡及其 VxWorks 下的驱动设计 [J]. 测控技术，2008，27 (8)：50－52.
[15] 周文庆，曹建福. 通用运动控制卡 CompactPCI 接口的设计 [J]. 电子技术应用，2004，(5)：33－35.
[16] 田志民，刘开华，苏育挺. VxWorks 操作系统下 PCI 总线驱动设计 [J]. 电子测量技术，2006，29 (2)：40－42.
[17] 卞红雨，曹明明，桑恩方. VxWorks 下 PCI 总线设备驱动程序设计 [J]. 声学与电子工程，2005，78 (2)：42－45.
[18] 刘德望，龙 兵，刘 震. 基于 CPCI 总线的雷达发射机自动测试系统的设计 [J]. 计算机测量与控制，2010，18 (6)：1260－1263.
[19] 王子龙，路景泽. 基于 CPCI 总线的雷达导引头测试系统设计与实现 [J]. 计算机测量与控制，2016，24 (7)：141－143.
[20] 孔祥营，柏桂芝. 嵌入式实时操作系统 VxWorks 及其开发环境 Tornado [M]. 北京：中国电力出版社，2002.