西安科技大学

硕士学位论文

基于PCI总线的电缆故障数据采集系统的研究

姓名：潘海涛

申请学位级别：硕士

专业：控制理论与控制工程

指导教师：汪梅

20080415

论文题目：基于 PCI 总线的电缆故障数据采集系统的研究

专　　业：控制理论与控制工程

硕 士 生：潘海涛　　（签名）潘海涛

指导老师：汪　梅　　（签名）汪梅

摘　要

本论文主要研究 PCI 总线技术在电缆故障数据采集领域的应用与实现问题。随着电网规模的不断扩大，电缆故障的发生也越来越频繁。但到目前为止，国内外较成熟的电缆故障测距方法大都基于离线状态，电缆故障在线测距仍缺少有效的方法。因此，实现电缆在线故障测距，具有迫切的现实意义。

电缆在线故障测距对数据采集与处理的要求很高，PCI 局部总线的引入，打破了数据传输的瓶颈，以其优异的性能成为微机总线的主流。但是由于 PCI 总线协议十分复杂，直接为它设计相匹配的数字逻辑控制电路难度很大。目前用来实现 PCI 接口的有效方案主要有采用可编程逻辑器件 CPLD 或 FPGA 和采用专用接口芯片两种方法。基于电缆故障在线定位系统的一部分，本论文对基于 PCI 总线的数据采集卡的接口技术进行了探讨和研究，在对两种接口方案进行比较的基础上，采用了第二种方案—PLX 公司的专用接口芯片 PCI9054。

本文在介绍了数据采集系统总体模块的构成、PCI 总线的基本传输协议以及电气规范的基础上给出了系统各个模块的设计思路。在本课题中，完成了硬件电路、印刷电路板(PCB)、驱动程序和应用软件的设计工作。具体的功能模块包括 PCI 接口模块、模数转换模块、逻辑控制模块、时钟电路和配置电路。硬件电路的绘制采用 Protel99SE 电路设计软件，驱动程序采用 DriverWorks 开发工具，应用软件则利用 VC++6.0 编程软件。利用 VC++6.0 和 MATLAB 语言编写的上位机后台分析软件程序，主要完成数据实时传输、数据文件保存、图形绘制、小波分析、频谱分析等工作，并提出了该采集系统性能进一步提高的方法，为下几届研究生提供参考意见。

本文最后对电缆在线故障测距装置进行了多项测试。测试表明，该系统硬件和软件设计达到了电缆在线故障测距的要求，系统运行稳定。

关 键 字：PCI 总线；电缆故障；PCI9054；数据采集；DriverWorks

研究类型：应用研究

Subject : The Research of Date Acquisition System in the Cable Fault Based on PCI Bus

Specialty : Control Theory and Control Engineering

Name : Pan Haitao (signature) Pan Haitao

Instructor : Wang Mei (signature) Wang Mei

ABSTRACT

This paper mainly studied the realization and application of cable fault data acquisition system using the PCI bus.With the development of electric power industry, the possibility of the cable fault happening is also becoming more and more greatly. However, up to now, the mature technology for cable fault location is mostly based on off-line at home and abroad. There are not efficient methods for cable fault location on-line, so this topic has an exigently realistic significance.

Cable fault location on-line bring forward higher request for data gathering and processing. The PCI bus protocol is so intricacy and it is one of the most competitive bus criterions for data acquisition technique, but it's hard to design the appropriate digital logic interface circuit for PCI.There are two effective schemes to realize the interface conveniently at present: One is to adopt CPLD or FPGA and the other is to adhibit special interface chip.Depending on one cooperatiove project,this thesis discusses the technique on the interface of data acquisition board based on PCI bus。Contrast has been made between the above two schemes, and finally the paper adhibits the second—PCI9054 to design the interface circuit based on PCI bus.

In the first part of the paper, the protocol and electrical attribution of the PCI will be introduced.Then in the second, the way and method of realizing the hardware of the data acquisition system will be discussed.The thesis includes the design of hard circuit, PCB (Printed Circuit Board), Driver and application soft.The detailed functional modules consist of analog digital conversion module、digital analog conversion module、PCI protocol conversion module、control logic、clock circuit and configuration circuit. The hard circuit is drawn employing circuit soft Protel 99SE. The driver is developed using developing tool DriverWorks and the application soft is programmed making use of advanced programming

soft Visual C++ 6.0. The upper computer procedures which are compiled by VC++6.0 language and MATLAB language that can accomplish some assignments about real time transmission of data, image-based rendering, file saving, wavelet analysis and spectrum analysis. And parts of the system that can be enhanced in the future by other students are also described.

Finally, the system is tested in this paper. The test results show that the hardware and software can normal work and meet the requirements of cable fault location on line.

Keywords : PCI Bus Cable Fault PCI9054 Data Acquisition DriverWorks

Thesis : Application Research

西安科技大学

学位论文独创性说明

本人郑重声明：所呈交的学位论文是我个人在导师指导下进行的研究工作及其取得研究成果。尽我所知，除了文中加以标注和致谢的地方外，论文中不包含其他人或集体已经公开发表或撰写过的研究成果，也不包含为获得西安科技大学或其他教育机构的学位或证书所使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中做了明确的说明并表示了谢意。

学位论文作者签名：潘海洋 日期：2008.4.18

学位论文知识产权声明书

本人完全了解学校有关保护知识产权的规定，即：研究生在校攻读学位期间论文工作的知识产权单位属于西安科技大学。学校有权保留并向国家有关部门或机构送交论文的复印件和电子版。本人允许论文被查阅和借阅。学校可以将本学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存和汇编本学位论文。同时本人保证，毕业后结合学位论文研究课题再撰写的文章一律注明作者单位为西安科技大学。

保密论文待解密后适用本声明。

学位论文作者签名：潘海洋 指导教师签名：汪梅

2008年 4月18日

1 绪论

1.1 论文选题背景

电缆跟架空线相比具有安全可靠、受气候影响小、隐蔽耐用、利于市容美观等优点。随着国民经济的快速发展和电缆成本的降低，越来越多的高低压输电线路逐步实现电缆化，电缆的应用变得日益广泛。但随着电缆数量的增多和运行时间的延长，电缆故障越来越频繁。由于电缆多埋于地下，一旦发生故障，寻找起来十分困难，往往需要花费很长时间，不仅浪费了大量的人力、物力，而且将会造成难以估量的经济损失以及影响人们的日常生活。因此，如何准确、迅速、经济的找出故障点，显得尤为重要。

目前使用的电力电缆故障测距装置，都为电力电缆故障停电后，解开电力电缆，通过相关的设备离线测量故障距离，即离线测距。这样的故障测距方法有很多弊端，例如测距时间过长；电力电缆在运行中发生的故障相当一部分是瞬时性的故障，利用离线故障测距方法查找故障点前，要用高压设备将故障点击穿，高电压对测试设备和电缆及工作人员造成安全隐患，因此迫切需要电力电缆在线测距方法弥补离线测距的缺陷和不足。为了弥补离线的不足，实现在线故障定位[1]，需要在线对传感器等元件进行数据采集,但故障点产生的行波速度很快,若想采集到数据，则要求采样频率很高。为了适应高速数据采集的需要，就要有高速的数据采集系统。数据采集卡是数据采集系统的主要组成部分，它是外界电信号与计算机之间的桥梁，完成对数据的采集和传输。

本课题是陕西省科技攻关项目(2003K06G19)“智能电缆故障预测与定位装置的研制”的一部分。论文主要是利用计算机 PCI 总线技术设计数据采集卡对电缆故障产生的数据进行采集，然后把数据送到上位机进行分析判断，达到在线检测电缆故障的目的，从而对故障点进行准确定位，保证电缆安全可靠的运行。

1.2 国内外测距方法的现状和发展

1.2.1 电缆离线测距方法

电缆故障离线测距方法有很多，根据其工作原理不同，目前离线测距方法分为两大类[1]：阻抗法和行波法，其实际应用大多为离线状态。

(1)阻抗法

阻抗法是以线路的集中参数模型为基础，根据线路单端或多端的电压、电流测量值，推导出特定的故障方程进行定位。由于此方法原理简单、容易实现，多年来一直受到人们的关注。在实际应用中，电桥法是一种较为经典的测试方法，英国雷迪有限公司生产

的BICCOTEST-T272电缆故障定位仪和英国SPIRENT公司生产的E2020电缆故障定位仪就是利用阻抗法中的电桥电路实现对电缆故障的定位。电桥法又分为直流电阻电桥法和电容电桥法。

直流电阻电桥法适用于低阻故障的探测，一般要求故障点的电阻不超过100千欧。基本原理为：将被测电缆的故障相与非故障相短接，电桥两端分别接故障相和非故障相。调节电桥臂上的可调电阻，使电桥达到平衡，在根据惠更斯电桥原理计算出故障点距离。

电容电桥法的测距原理与电阻电桥法的测距原理基本相同，它用于电缆开路故障的测距。基本原理为：开路故障测得的电缆电容值与电缆故障距离成正比，只要利用电容电桥测出电容值大小，并在已知电缆单位长度的电容值的基础上，就可以计算出电缆的故障距离。

(2)行波法

行波法是电缆故障测距中另一种十分重要的方法，它利用测量行波在故障点和测量端之间往返的传播时间来确定故障位置。根据产生行波的种类和测量方式不同，传统的基于行波法的测距方法可分为A、B、C三种类型。近几年来，随着国外学者对行波法研究的深入，又提出两种新的方法，分别是利用重合闸产生的暂态行波在测量端与故障点之间的传播时间进行测距和在测量端反馈到的故障开断时产生的初始浪涌行波与故障点反射波之间的时延进行测距。

1.2.2 电缆在线测量方法

到目前为止，电缆在线故障测距方法及装置没有架空线路那么成熟，国内外的科技工作者正在努力的填补这一领域的空白，并且取得了一定的进展。

(1)由日本学者提出的采用光纤电流互感器感应出故障时产生的浪涌电流信号，利用采集速度为16MHz的快速A/D技术实现不带分支的电缆在线故障测距方法。

(2)由美国学者提出的利用故障时产生的浪涌电压或电流在开路点发生正或负的全反射，通过设于开路点附近的传感器得到脉冲信号，测出其脉冲间隔时间实现电缆在线故障测距的方法。

(3)由我国学者张群峰等人提出的基于电弧特性的电缆在线故障测距方法。

这些方法对电缆在线故障测距都具有很重要的实用价值，当然也不可避免地存在着一些局限性。然而，从长远发展的战略角度出发，电缆在线故障测距具有光明的前景。

1.2.3 实时专家系统

电缆故障诊断是一种综合的电缆故障测距方法，它对电缆在没发生故障时的状态进行监测：对电缆发生故障后的位置、类型、程度做出判断并进行决策，适用于各种电缆

故障。目前，不少的科学工作者已经将实时专家系统与小波分析技术应用于电缆故障诊断问题的研究中。

其中，实时专家系统是利用计算机模拟专家思维，将专家知识库作为数据库，用一套规则来维护和更新知识库，知识库可以从以往的电缆故障事件中提取，并可以在实际应用中修改。小波分析则是一种现代信号处理技术，它在信号奇异点检测方面有着重要的作用。因此，将实时专家系统和小波分析技术应用于电缆故障诊断中有着巨大的潜能和光明的前景。

1.3 数据采集系统发展概述

数据采集的对象多种多样，如常见的有温度、压力、位移、速度、频率等，和模拟系统相比，数字系统有精度高、稳定性好等一系列优点，但是数字系统只能处理离散的数字信号。外部世界的各种各样的信息经过传感器转换以后，除了极小部分为数字信号和开关信号之外，绝大部分以电压或电流等模拟量的形式存在，所以往往需要将这些模拟信号转化为便于处理和存储的数字信号，这部分工作就需要数据采集系统来完成。所谓数据采集就是将模拟信号转化为数字信号，并进一步予以处理、显示、存储和记录的过程。

随着信息科学的飞速发展，数据采集和存储技术已经是数字信号处理中非常重要的环节，将决定整个系统的性能。它广泛应用于雷达、通信、遥测遥感等领域。它已经成为人们获得外界信息的重要手段。基于总线的数据采集与存储系统，由于可靠且易于实现、经济等优点，得到了广泛的应用。但当数据传输率很高时，保持高速数据存储过程的可靠性、实时性将会成为一个比较棘手的问题。数据采集系统的发展目前集中在传输速度、采集精度、可靠性和成本效益上，因而关键也在这上面要提高数据采集系统的性能。

数据采集系统是一个独立的应用系统，它有自己专门的硬件和软件子系统，为了适应高速数据采集的需要，迫切需要高速数据采集系统的出现。而数据采集卡是数据采集系统的主要组成部分，它是外界电信号与计算机之间的桥梁，完成对数据的测量和传输。数据采集卡发展到现在，已经在速度和接口方式上有了很大的改进。就市场上的数据采集卡而言，它的速度已经突破了 500MHz，其接口方式也已经拓展到 PCI、USB、PXI 等多种接口。目前国内的数据采集卡市场还主要是由国外公司占主导地位，在高速采集卡方面尤为如此。国内公司由于技术积累、研发投入、微电子技术等方面的落后，其产品在稳定性、多样性、通用性等方面较之国外产品有较大的差距。而数据采集系统目前在社会生产、科研中的作用正日益显的重要，因此积极研制和开发拥有自主知识产权的数据采集系统对于我国的科研、生产有着很大的现实意义。

目前市场上的数据采集卡产品基本上是针对数据采集这一功能而研制的，很少涉及

可编程性和数字信号处理功能。而随着虚拟仪器技术发展，目前越来越多的硬件测控仪器被虚拟仪器所代替。虚拟仪器往往要对采集信号完成某些数字信号处理，而该工作若由计算机完成，无疑会大大地降低仪器系统的实时性。解决该问题的最好办法就是把这些工作放到数据采集卡上去快速完成。因此，提高数据采集卡的数据处理功能和可编程性将是以后数据采集卡研究的重要方向。

1.4 论文研究内容和结构安排

本文在具体的应用背景下，提出了基于 PCI 总线的电缆故障数据采集系统的设计方案。这个系统的开发分为两个步骤：首先是基于 PCI 总线的数据采集卡的研制。具体工作包括采集卡的原理图设计、各模块功能设计、PCB 制作、PCI 板卡驱动开发及调试等。其次是在 PCI 数据采集卡成功实现的基础上，设计了后台分析程序，对采集到的故障信号进行分析与处理。

论文的结构安排如下：

第一章 绪论：介绍本论文的选题背景、意义以及本领域的研究现状；

第二章 PCI 总线概述：分别对 PCI 总线的系统结构、操作协议、总线命令、电气特性等几个方面做了简要说明；

第三章 数据采集系统的硬件设计：对系统硬件电路的设计方法及过程进行了详细的论述，并给出了相关模块电路的原理图，通过 Protel99SE 完成了硬件的设计与制作；

第四章 系统的软件设计：利用 DriverWorks 和 PlxMon 对本课题所设计的板卡进行了驱动程序的编写以及测试，并且完成了上位机应用程序的设计；

第五章 实验方法及测试结果：将电缆接入 237V 的交流电压信号中对电缆故障在线测距装置的硬件及软件功能进行测试，并对测试结果进行仔细分析，给出测试结论；

第六章 结论：对全文进行总结，并指出论文中的不足之处及进一步改进方法。

2 PCI 总线概述

随着计算机技术的不断发展，其计算机的体系结构也发生了显著的变化，如 CPU 的运行速度的提高、多处理器结构的出现、高速缓冲存储器的广泛应用等，都要求总线进行高速数据传输，从而出现了多总线结构，多总线结构即指 CPU 与存储器、I/O 等设备之间有两种以上的总线，这样可以将慢速的设备和快速的设备挂在不同的总线上，以减少总线竞争，提高系统效率。

在多总线结构中，局部总线[3]（Local Bus）的发展最令人瞩目。局部总线是指来自处理器的延伸线路，与处理器同步操作，由于局部总线有极高的数据传输率，因此，其在 CPU 与高速缓冲存储器、高速数据采集等场合得到了广泛的应用。

在本论文中，正是利用了局部总线中的 PCI 总线完成数据采集系统的设计，以满足高速数据采集和传输的要求。

2.1 PCI 总线特点

PCI 总线独有的特点[5]：

(1) 高性能

PCI 是一套整体的系统解决方案，较其它只为加速图像或者视频操作的局部总线优越。PCI 总线能提高网络适配卡、硬盘的性能；可以出色的配合全活动影像、图象及各种高速外围设备的要求。PCI 局部总线以 33MHz 的时钟频率操作，采用 32 位的数据线和地址线复用结构，可支持多组外围部件及附加卡。数据的最大传输率可达 132MB/S，远远超过标准的 ISA 总线的速率。即使在 32 位的情况下，PCI 总线也能很好的支持奔腾（Pentium）及计算机的各种数据传送。

(2) 突发性传输

PCI 能支持一种称为线性突发的数据传输模式，可确保总线不断满载数据。外围设备可以从内存的某个地址顺序的接收数据，可以减少无谓的地址操作，这样能够更有效地利用总线的带宽去传输数据。也就是说外围设备可以由某个地址起读写大量的数据，然后每次只需将地址自动加 1，便可以接收数据流内的下一个字节的数据。另外，PCI 总线可以支持突发读取和突发写入，这对进行高速的数据采集尤为重要。

(3) 极小的存取延误

支持 PCI 的设备，存取延误很小，能够大幅度减少外围设备取得总线控制权所需要的时间。例如，支持 PCI 的以太网卡可以及时将数据传至中央处理器，减少所需要的额外内存，从而减少附加卡的成本。

(4) 采用总线控制和同步操作

PCI 的总线控制和同步操作功能有利于 PCI 性能的改善。总线主控是大多数总线都具有的功能，目的是让任何一个具有处理能力的外围设备暂时接管总线，以加速执行高吞吐量、高优先级的任务。PCI 独特的同步操作功能可保证微处理器能够与这些总线主控同时操作，不必等待后者的完成。

(5)不受处理器限制

PCI 独立于处理器的结构，形成一种独特的中间缓冲器设计方式，将中央处理器子系统与外围设备分开。因此用户可以随意增添外围设备，以扩展电脑系统而不会因为在不同的时钟频率下导致 CPU 性能的下降。而且，PCI 总线独立于微处理器结构还可以保证不会因为处理器更新换代过快而使外围设备过时，从而保护了消费者的利益。

(6)适合于各种机型

PCI 总线定义了两种信号环境：5V 和 3.3V。他们之间可以很容易地进行转换，而且 3.3V 信号环境的定义为 PCI 总线用于笔记本电脑开阔带来了方便。在服务器环境下，PCI 支持分级式外围设备的特性，可使一个 PCI 界面支持一组级联的 PCI 局部总线；也可以使拥有多组 PCI 总线的服务器增添额外的扩展插槽。

(7)兼容性强

由于 PCI 的设计是要辅助现有的扩展总线标准，因此它与 ISA、EISA 及 MCA 总线完全兼容。PCI 局部总线可提供“共用插槽”，以便插 PCI、ISA 及 MCA 插头。

(8)预留了发展空间

PCI 总线在开发时预留了充足的发展空间，例如，PCI 总线支持 64 位地址、数据多路复用，PCI 插槽能同时接插 32 位和 64 位插卡。允许 PCI 局部总线扩展卡和元件自动配置。在 PCI 器件上包含有寄存器，上面有配置所需要的器件信息。

此外，PCI 总线还具有协议芯片成本低、高效益，而且 PCI 规范也在不断完善等优点。

2.2 PCI 总线系统结构

在一个 PCI 系统中，高速的外部设备和低速的外部设备可以共存，PCI 与 ISA/EISA 总线并存。

图 2.1 是一个典型的 PCI 系统结构[5][8]，可以看出，CPU 通过主设备/PCI 桥，即北桥可以直接访问映射于存储器空间或者 I/O 空间的 PCI 设备，此时 CPU 只与北桥交付。北桥是一个低延迟的访问通道，能够提供数据缓冲功能，以便 CPU 与 PCI 总线上的设备并行工作而不必相互等待；该桥可以使 CPU 与 PCI 总线上的操作分开，以免相互影响；北桥作为主设备并且提供了 PCI 总线上的所有驱动机制。由此可见北桥在 PCI 系统中具有重要位置。

CPU 通过北桥访问主存和 AGP 设备。PCI 总线上可以有 PCI 显卡、PCI 网卡、PCI

声卡、SCSI 的软盘适配器、PCI 扩张总线桥（即南桥）以及 PCI 插槽。南桥可以是 PCI/PCI 扩展总线桥、PCI/ISA 扩展总线桥、PCI/EISA 扩展总线桥。通过北桥和南桥 CPU 可以访问二级 ISA 总线、EISA 总线甚至次级 PCI 总线上的设备。南桥除了可以扩展总线外，还可以接 IDE 接口和 USB 接口等。可见南桥是为了在 PCI 系统中挂接其它总线以兼容现有设备。

PCI 设备有主从之分，主从设备是相对于总线而言。主设备具有总线控制权，能够驱动地址线、数据线和控制线，但是从设备不能驱动地址线。以图 2.1 为例，北桥和南桥的充当的角色是相对于哪一级总线而言，北桥对于 CPU 总线而言是从设备(target)，对于 PCI 总线而言是主设备(master)；同样在 ISA 总线上南桥是主设备，而在 ISA 总线上的所有其它设备，如声卡适配器、超级 I/O 均为从设备。

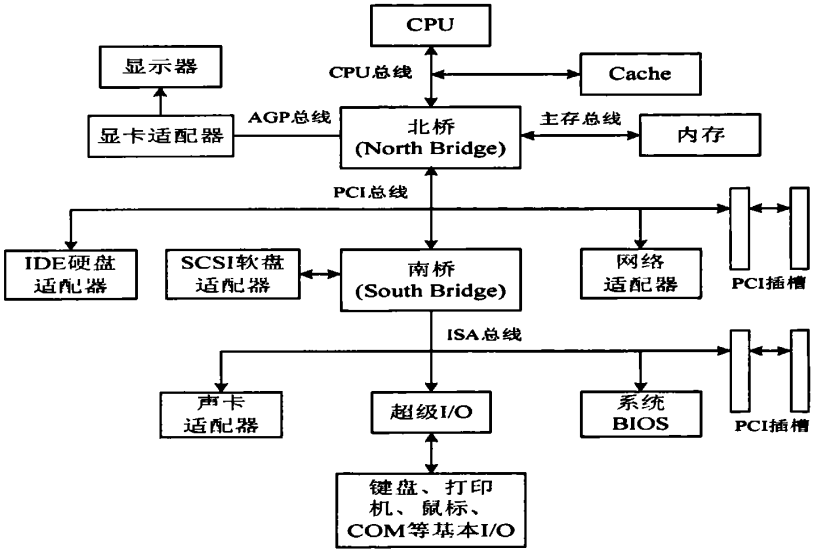


图 2.1 PCI 总线系统结构

2.3 PCI 总线的操作

PCI 总线上的活动表现为 master 和 target 之间的信息传输和交互。总线传输采用命令驱动的方式，总线命令的作用是用来规定主、从设备之间的传输类型。当一个 master 得到总线的控制权后，由它来决定下一次传输的类型。在传输的地址段，C/BE[3:0]#信号线用于给出传输的类型命令，被选中的从设备有效 DEVSEL#设备选择信号表明愿意进行数据的传输。若主设备采样到有效的 DEVSEL#，在地址期结束后，进入数据期，从设备负责锁存起始地址并在后继数据传输中自动递增（假设从设备支持突发传输）以指向下一个单元（双字或四字）。总线命令包括：中断确认，特殊周期，I/O 读，I/O 写，读内存，写内存，读配置空间，写配置空间，双地址周期等。PCI 总线操作比较多，这

里只作简单介绍，详细内容请参阅《PCI 系统结构》一书。

2.4 PCI 总线信号定义

PCI 局部总线信号[5][8]如图 2.2 所示：

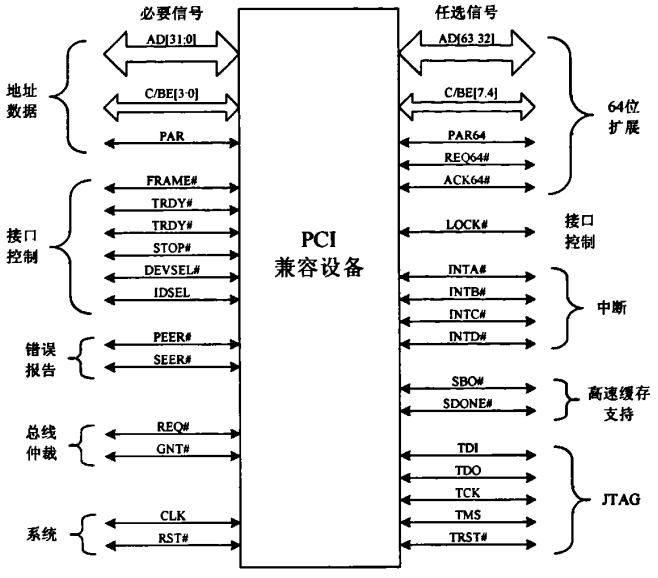


图 2.2 PCI 局部总线信号

注：图中#表示低电平有效，否则为高电平有效。对于仲裁器，REQ#为输入，GNT#为输出，其它适用于仲裁器的信号的方向与主设备和目标设备相同：

IN 表示输入，是标准的只作输入的信号。

OUT 表示输出，是标准的图腾柱式输出驱动信号。

T/S 表示双向的三态输入输出信号。

S/T/S 表示持续的并且低电平有效的三态信号。

O/D 表示漏极开路，以线或形式允许多个设备共同驱动和分享。

(1) 系统信号

CLK：对于系统所有的 PCI 设备都是输入信号。其频率范围为 0~33MHz 或 0~66MHz，这一频率称为 PCI 的工作频率，对于 PCI 信号，除 RST#、INTA#、INTB#、INTC#、INTD#之外，其余信号都是在 CLK 的上升沿进行采样的。

RST#：复位信号。用来使 PCI 专用的特性寄存器、配置寄存器等恢复到规定的初始状态。复位时，PCI 的全部输出一般都驱动到第三态。REQ#和 GNT#必须同时驱动到第三态，不能在复位期间为高或为低。为防止 AD、C/BE#、PAR 在复位期间浮动，可由中央资源将它们驱动到逻辑低，但不能驱动到高电平。RST#和 CLK 可以不同步，但是要保证其在撤消边沿不会有反弹。

(2) 地址和数据信号

AD[31:0]：地址/数据多路复用的输入/输出信号。在 FRAME#有效时，是地址期。在 IRDY#和 TRDY#同时有效时，是数据期。

C/BE[3:0]：总线命令和字节使能多路复用信号。在地址期中，传输的是总线命令。在数据期内传输的是字节使能信号。

(3) 接口控制信号

FRAME#：帧周期信号。由当前主设备驱动，表示当前主设备一次交易的开始和持续时间。

IRDY#：主设备准备好信号。由当前主设备（交易的启动方）驱动，信号的有效表明发起本次传输的设备能够完成交易的当前数据期。

TRDY#：目标设备准备好信号。由当前被寻址的目标驱动，信号有效表示目标设备已经做好了完成当前数据传输的准备工作。

STOP#：停止数据传送信号。信号有效时，表示目标设备要求主设备终止当前的数据传输。

LOCK#：锁定信号。信号有效时，表示一个对桥的原始操作可能需要多个传输才能完成。

IDSEL：设备选择信号。信号有效时，表示驱动它的设备成为当前交易的目标设备。

(4) 仲裁信号

REQ#：总线占用请求信号。信号一旦有效表明驱动它的设备向仲裁器要求使用总线，是一个点到点的信号线。

GNT#：总线占用允许信号。用来向申请占用总线的设备表示其请求已经获得批准，也是一个点到点的信号线。

(5) 错误报告信号

PERR#：数据奇偶校验错误报告信号。只报告除特殊周期之外的所有 PCI 交易期间的数据奇偶错误。

SERR#：系统错误报告信号。是报告地址奇偶、特殊周期命令的数据奇偶错误以及其它可能引起灾难性后果的系统错误。

(6) 中断信号

PCI 局部总线中共有四条中断线，分别为 INTA#、INTB#、INTC#、INTD#，其作用是用以请求一个中断。

(7) 附加信号

PRSNT[1:2]#：卡存在信号。由插件板提供的信号，用来指出 PCI 插卡上是否存在一个插件板，如果存在就为它提供电流。

CLKRUN#：时钟运行信号。可选信号，作为设备的输入信号，用来确定 CLK 的状

态。

(8)64 位总线扩展信号

AD[64:32]：扩展的 32 位地址和数据多路复用线。在地址周期，如果使用了 DAC 命令且 REQ64#有效时，这 32 条线上含有 64 位地址的高 32 位，否则是保留信号；在数据周期，当 REQ64#和 ACK64#同时有效时，这 32 条线上含有高 32 位数据。

C/BE[7:4]：扩展总线命令和字节使能多路复用信号线。

REQ64#：64 位传输请求信号，由当前主设备驱动，表示本设备要求采用 64 位通路传输数据。

(9)JTAG/边界扫描信号

TCK：测试时钟信号。

TDI：测试数据输入信号。

TDO：测试数据输出信号。

TMS：测试模式选择信号。

TRST：测试复位信号。

2.5 PCI 总线命令

总线命令[5]是用来规定主从设备之间的传输类型的，它出现于地址周期的 C/BE[3:0]线上。当一个主设备获得总线的拥有权时，它就可以启动表 2.1 中的任何一种交易类型。在一个交易的地址周期，命令/字节使能总线 C/BE[3:0]用于表明交易命令和类型。

表 2.1 PCI 总线命令编码

C/BE[3:0]	命令类型说明	C/BE[3:0]	命令类型说明
0000	中断应答	1000	保留
0001	特殊周期	1001	保留
0010	I/O 读	1010	配置读
0011	I/O 写	1011	配置写
0100	保留	1100	存储器多行读
0101	保留	1101	双地址周期
0110	存储器读	1110	存储器一行读
0111	存储器写	1111	存储器写无效

2.6 PCI 总线的电气规范

2.6.1 PCI 总线的信号环境

PCI 局部总线的电气规范[6]中提供了 5V 和 3.3V 两种信号环境，二者不能混合使用，

即对某一 PCI 总线系统，所有器件必须使用同一信号规则。但是，通过设计是可以使5V 的元件工作于 3.3V 的信号环境的，反之亦然。元件可以混合使用，但信号环境必须是 5V 或 3.3V 中的一个。

2.6.2 PCI 总线对负载的要求

PCI 总线通常只允许 3 个插卡，任何超出限制的设计都需要 PCI-PCI 桥来保证系统的可靠性。PCI 总线采用无端接方式，信号的传输通过反射波实现。当总线驱动器驱动某一信号时，往往只将信号电平驱动到实际所需电平的一半，信号传送到终点时反射回来，而使得信号电平加倍，达到驱动所需电平。当总线工作于 33MHz 时，信号往返的时间不得超过 l0ns，这种信号传输要求驱动器的输出阻抗与被驱动总线的特性阻抗相匹配。系统为每个连接器提供+12V、-12V、5V 和 3.3V 四个电源，要求扩展卡从四个电源中取得的总功率限制在 25W 以内。

2.7 小结

本章主要说明了 PCI 局部总线的系统架构，对 PCI 局部总线的接口信号和总线操作做了比较详细的论述。这样，我们可以进一步的了解 PCI 局部总线的工作原理，为后面的设计奠定扎实的基础。

3 数据采集系统设计

对电缆故障产生的行波进行采集分析是实现电缆故障在线测距系统的基础，其采集模块设计的好坏，将直接影响电缆故障测距的精度。本章重点介绍了数据采集系统的总体结构和 PCI 数据采集卡各个功能模块的硬件电路设计。

3.1 系统总体设计方案

电缆在线故障测距系统是一种智能化的监测装置，它安装于线路的一端，夜以继日地监视着电缆线路，如图 3.1 所示。线路一端的电压、电流信号经互感器后送入电缆故障测距系统。当线路工作正常时，实时监测各个电压、电流信号；当线路发生故障或者大扰动时，记录各个故障波形。在采样过程结束后停止记录，并将数据传送至上位机，进行分析、处理和存储，为检修人员及时发现和排除故障提供强有力的支持。本文的主要工作集中在 PCI 采集卡的开发。

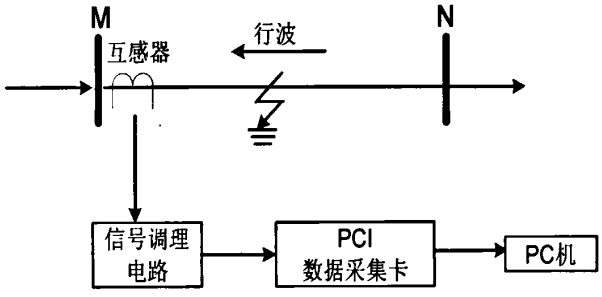


图 3.1 数据采集系统结构图

3.2 PCI 接口模块的设计

3.2.1 PCI 接口的实现方案

PCI 总线复杂的总线逻辑和电气规范使人们不可能像开发 ISA 总线控制卡那样通过简单的逻辑就可以实现计算机与外围设备的接口。由于 PCI 总线规范十分复杂，再加上 PCI 总线时钟高达 33MHz，信号线应该按微波传输线对待，其时序要求非常严格，从而导致自行设计总线接口难度很大，需要的开发周期也很长。目前适宜 PCI 总线接口开发的芯片主要分为两类[7]。

(1)采用可编程逻辑器件 CPLD

这种方法是用符合 PCI 总线规范的 PLD 芯片如 CPLD 或 FPGA 来做。

采用 CPLD 或 FPGA 等可编程逻辑器件实现 PCI 接口最大的优点在于其灵活的可编

程性，对于一个典型 PCI 设计来说，并非一定要实现 PCI 规范中的所有功能，只需实现其中的一个子集即可，其调试方便，易于修改，不受插卡功能限制，当系统升级时，只需对可编程器件重新进行逻辑设计，而无需更新 PCB 版图。但是，要实现 PCI 规定的功能如完成逻辑校验、地址译码、实现配置等所需的各类寄存器，大致需要 10000 个门电路，并且可编程器件生产厂商提供的经过严格测试的 PCI 接口功能模块价格昂贵，同时要求对 PCI 总线的操作时序非常了解，使得设计 PCI 控制接口难度较大，对于产品不大而又有时限的工程项目来说，成本过大。这样，在小规模的应用中，就限制了利用可编程逻辑器件来实现 PCI 接口。

(2)采用专用接口芯片

采用 PCI 总线专用接口芯片，如 PLX 公司的 PCI9054 和 PCI9052，AMCC 公司的 S5933 和 S5920 等，设计者不需花大量的精力去了解 PCI 总线的详细工作机理，也不用考虑 PCI 接口芯片的具体构造，而只需要把精力集中于整个硬件系统的设计上，完成 PCI 芯片和微处理器的硬件接口设计即可，从而可以将复杂的 PCI 总线接口转换为相对简单的接口设计，使设计者可以将整个的精力放在系统设计上，从而大大缩短设计周期，这种芯片也具有较低的成本和通用性。

通过对这两种接口实现方案进行比较后可知，采用 CPLD 或 FPGA 等可编程逻辑器件实现 PCI 接口优点在于其灵活的可编程性，但是技术上和经济上要求比较高，而且为了达到 PCI 规范的严格要求，需要作大量的逻辑验证、时序分析和程序调试，对于小范围的使用是完全没有必要的。而采用专用的 PCI 接口芯片无论从技术上还是从成本上来说都是比较理想的选择。本课题就是采用的第二种方法。在设计中，选用了 PLX 公司的 PCI9054。

3.2.2 PCI9054 主要功能

PCI9054 的主要功能特点如下[10][14]：

(1)符合 PCI V2.1、V2.2 规范。兼容 PCI V2.2 电源管理规范；

(2)采用通用总线主控接口。包含两个独立的 DMA 通道；

(3)支持 PCI 双地址周期(DAC)。地址空间可达 4GB，支持 TYPE0，TYPE1 配置周期；

(4)内含可编程中断控制器。能实现可编程的突发传输操作的 6 个零时间等待可编程 FIFO，8 个 32bit 的 Mailbox 寄存器和 2 个 32bit 的 Doorbell 寄存器；

(5)支持局部总线与 Motorola MPC860，PowerQUICC 和 Intel I960 系列以及 IBM 的 PPC401 等 CPU 直接相连；

(6)支持局部总线与 PCI 总线异步工作，可编程控制局部总线等待状态，支持可编程预读取计数器。

3.2.3 PCI9054 的引脚分配

PCI9054 是本设计电路的核心部分，它有 2 种封装模式：176 引脚的 PQFP 和 225 引脚的 PBGA。本次设计采用 176 引脚的 PQFP 封装的 PCI9054 芯片，其信号引脚可分为四组：电源、地引脚、串行 EEPROM 接口引脚、PCI 系统总线接口引脚和本地总线接口引脚，引脚分配如图 3.2 所示：

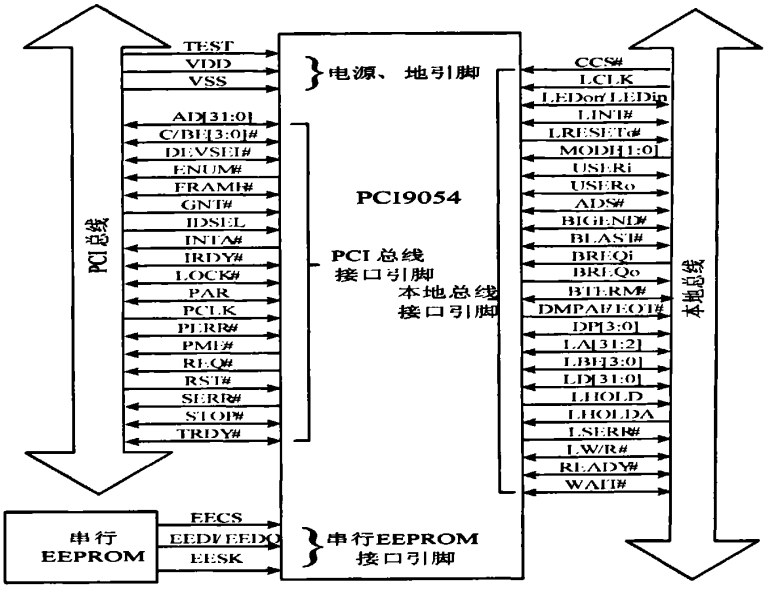


图 3.2 PCI9054 的引脚分配

PCI9054 是 32 位、33MHz 的通用 PCI 总线控制器专用芯片。该芯片符合 PCI 总线规范 2.2 版，突发传输速率达到 132MB/S。Local 总线支持复用/非复用的 32 位地址/数据，可为 M 模式、C 模式、J 模式中的一种。PCI9054 内部有六个可编程的 FIFO，实现零等待突发传输及局部总线和 PCI 总线间的异步操作。PCI9054 支持主模式(initiator)、从模式(target)、DMA 传输方式，可用于适配卡和嵌入式系统。其工作电压为 3.3V，频率 33.3MHz，局部总线频率可达 50MHz。

该 PCI9054 接口的硬件电路主要分为 3 部分：第一部分是 PCI9054 和 PCI 插槽间的连接信号线（PCI9054 接口电路原理图详见附录）；第二部分是与串行 EEPROM 的连线；第三部分是 PCI9054 与 XC95144 的连接（详见附录）。XC95144 被选择作为该设计的控制器，利用其内部逻辑单元设计局部总线控制器，可用于实现局部总线的状态控制。

3.3 数据采集卡硬件设计

硬件设计部分主要包括数据采集系统中的各硬件模块及功能模块，关键器件的选择到硬件电路的设计。首先给出系统的硬件框图，如图 3.3 所示，然后按框图结构对各个

模块予以详细的介绍，其中着重介绍了控制电路的工作原理。

基于 PCI 总线的数据采集卡主要由 PCI 接口模块，数据存储模块，AD 转换模块以及 CPLD 逻辑控制模块等组成，其基本工作原理是通过高速 A/D 将外部模拟信道的信号进行采样，先将采样数据存储在 FIFO 中，当 FIFO 半满时，会产生一个半满信号 HF 通知 CPLD 使其产生控制信号用来控制 PCI9054 执行 DMA 传送，将数据读入到电脑内存中，这样就可以在电脑中对数据进行分析处理。

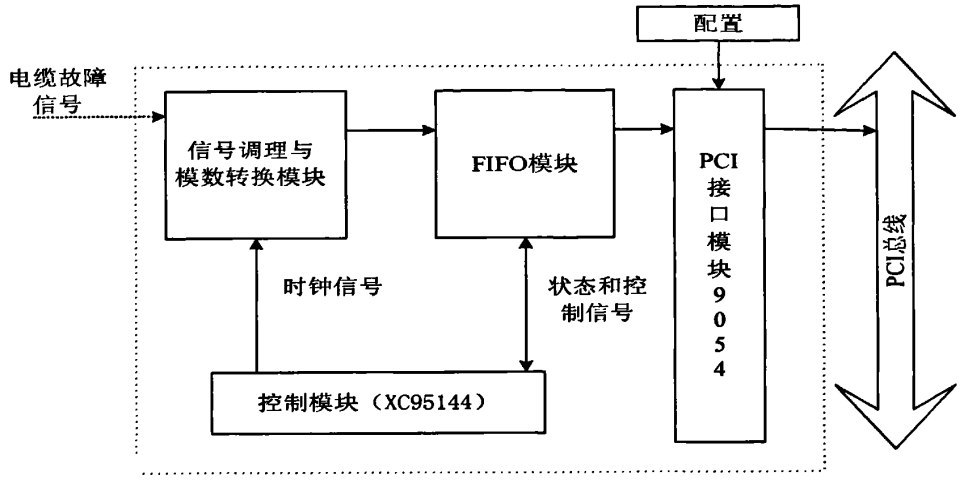


图 3.3 PCI 数据采集卡硬件设计框图

3.3.1 信号调理电路设计

采集卡的 AD 模块采集电缆的电压信号，为了得到电压信号，需对电流互感器的输出电流进行 I/V 变换，电路如图 3.4 所示：

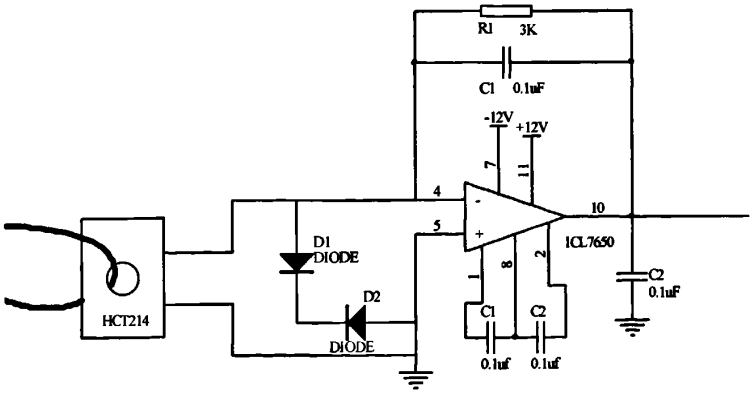


图 3.4 I/V 变换电路

从图中可以看出，电流互感器的副边电路是由运放 ICL7650 进行 I/V 变换的，该运放精度较高且稳定性良好；反馈电阻 R1 用来得到所需要的输出电压；电容 C1 用来补

偿相移，电容 C2 用来去耦合滤波，通常 C1、C2 取 0.1 μF；两个反接的二极管用来保护运算放大器。

由于采用的 A/D 要求其输入信号范围在 0～3.3V 之间，因此还需要外加偏置电路，其功能是将双极性交流信号转换为单极性信号，以便 A/D 采样。具体电路如图 3.5 所示，为了使电路简单，选用-12V 电源作为加法电路的一个输入端。它是由运放所构成的加法电路和反相电路的组合。

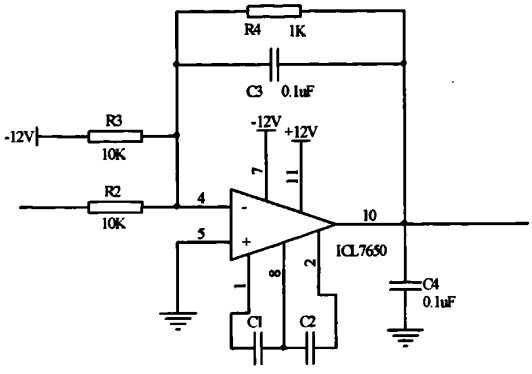


图 3.5 偏置电路

3.3.2 PCI9054 与 EEPROM 的接口设计

PCI9054 的上电配置可以为默认设置，用户若没有对 PCI9054 进行设置，PCI9054 将加载默认设置，但默认设置一般不符合用户的特定的需要。PCI9054 预留了 EECS、EESK、EEDI/EEDO 三个引脚，这三个引脚可以加一个串行的 EEPROM，计算机上电时首先读取的值，写入对应的寄存器。此种方法简单，易于实现，但要注意串行 EEPROM 的选取，选取的 EEPROM 必须有串行化的读写功能。可以是 2K 的，也可以是 4K 的。本文中选用了 PLX 公司推荐的兼容串行 EEPROM 芯片：NM93CS56N 芯片。NM93CS56N 芯片容量为 2048 位(128×16 位)，8 引脚。NM93CS56N 的外围电路如图 3.6。

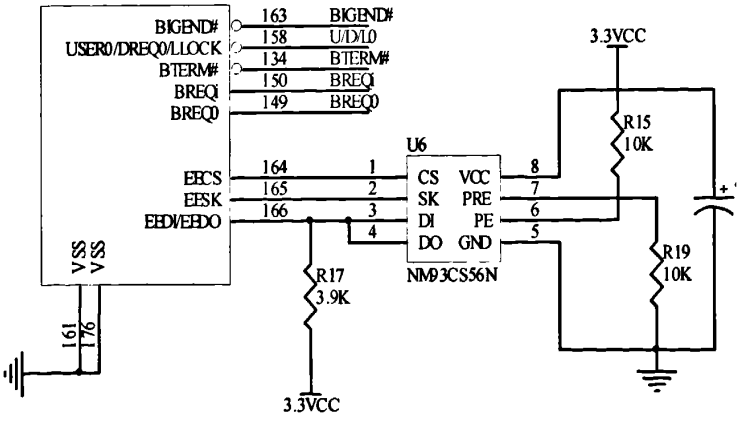


图 3.6 EERPOM 的配置电路

设计好硬件电路后，要实现预期的功能就需要进行 PCI9054 的寄存器配置[13]。寄存器配置包括 EEPROM 初始化、LOCAL 功能寄存器和 PCI 配置寄存器的配置。其配置过程如下：

(1) EEPROM 初始化

PCI9054 的长配置方式要求 68 个字节的配置信息，主要包括以下信息：设备识别号、供应商代码号、Local 总线三个空间的大小以及三个空间的基址等。可以利用编程器事先将要配置的信息写入配置存储器中，也可以在 PLxSdk 中的 PLXMON 下对 EEPROM 进行配置。这样，在计算机启动时将配置好我们申请的系统资源。具体配置过程如下：

在计算机的加电自检期间，PCI 总线的 RST#信号复位，PCI9054 内部寄存器的默认值作为回应。PCI9054 输出本地 LRESET#信号并检测串行 EEPROM。如果串行 EEPROM 中的前 33 个 bit 不全为 1，那么 PCI9054 确定串行 EEPROM 非空，用户可通过向 PCI9054 的寄存器 CNTRL 的 29 位写 1 来加载 EEPROM 的内容到 PCI9054 的内部寄存器。如果串行数据开始位为 1，则表明串行 EEPROM 不存在。对于不存在串行 EEPROM 的情况，需要把 EEDO 引脚拉低以阻止错误检测。同时 PCI9054 停止串行 EEPROM 的加载，并转换到默认值。当串行 EEPROM 有效位 CNTR[28]=1 时，在串行 EEPROM 中会读到真实或随机的数据。

PCI9054 芯片加载 EEPROM 中的信息，先加载高 16 位[31:16]，并且是从高有效位 31 开始，然后再加载低 16 位，从 15 位开始。因此加载顺序为器件 ID 号、生产商 ID 号类码等等。混合寄存器 CNTRL 允许每一时刻对 EEPROM 的一位进行编程。串行 EEPROM 中的数值也可以通过 PLXSdk 中的 PLXMON 中来配置。

(2) PCI9054 寄存器配置

PCI9054 配置寄存器的配置要根据具体的硬件设备进行配置。其配置的正确与否是硬件设备能否正常工作的关键，并且关系到硬件设备是否能够充分的发挥其性能，可以说其配置寄存器的配置是使用 PCI9054 芯片进行硬件设计能否成功的关键。

配置 PCI 配置寄存器，主要是填写十六进制的生产商 ID 号、器件 ID 号、类码子系统 ID 号和子系统生产商 ID 号。对于 PCI9054，生产商 ID 号为 l0B5，器件 ID 号为 9054，子系统号为 9054，子系统 ID 号为 l0B5，类玛号为 0680，表示其为桥设备中的其他桥设备类。

(3) Local 寄存器的配置

对于 Local 配置寄存器的配置其实就是对本地地址空间及其本地总线属性的配置。这些配置要根据实际开发的硬件板卡的硬件资源进行配置。例如若设计的接口卡只使用一个本地地址空间，则只需对着一个地址空间向计算机申请得到相应的 PCI 地址空间即可。PCI9054 在本质上是一个桥设备，它把 PCI 总线对某一段 PCI 总线地址空间的各种

操作包括读、写等转换为相应的地址总线上的操作。因此配置寄存器的任务就是要把某一段本地地址映射为 PCI 地址，换一种说法就是当主机 CPU 要访问本地地址空间时，要知道其对应的 PCI 总线地址。

PCI9054 数据手册中给出的需要配置的寄存器及其在 EEPROM 中的排列顺序，我们将根据这个图表提供的地址顺序在 EEPROM 的配置文件中依次配置各个寄存器。在 PLXMON 下的配置界面如图 3.7 所示。

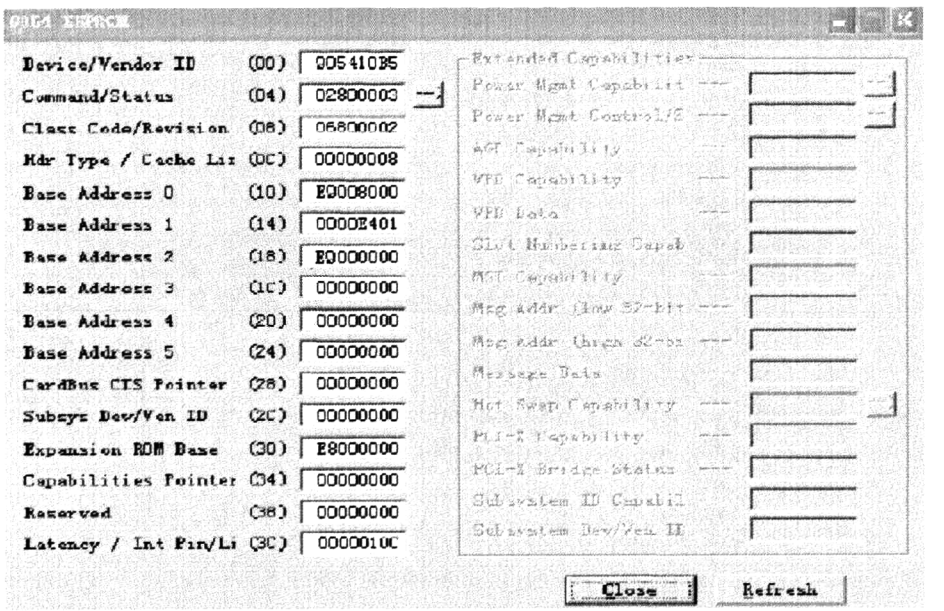


图 3.7 PCI9054 的寄存器配置

3.3.3 数据缓存设计

因为 PCI 总线接口数据传输效率非常高而 A/D 采集数据效率较低，所以为了实现数据的高速传输，采样后的数据必须经过缓存，再由 PCI 总线接口控制器读取。通用的 PCI 接口芯片内部都有专用的 FIFO，但其存储深度较小，通常只有几十个字长(例如 PCI9054 就只有 12Byte 的写 FIFO 和 64Byte 的读 FIFO)，不能起到很好的缓冲效果。这样才能处理更多的模拟信号输入。缓存的设计方案[14]一般有三种：共享 RAM 方式，双口 RAM 方式，FIFO 缓存方式。第一种方式因为涉及到 RAM 的同时访问，逻辑比较复杂；第二种方式芯片价格昂贵；第三种既简单性价比又高。

FIFO[14][17](First In First Out：先进先出)芯片是一种具有存储功能的高速逻辑芯片，在高速数字系统中经常用作数据缓存，在通信中获得广泛应用。由于微电子技术的飞速发展，新一代 FIFO 芯片容量越来越大，体积越来越小，价格越来越便宜，适合在数量大、密度高、速度快的数据处理系统中应用。作为一种新型大规模集成电路，FIFO 芯片以其灵活、方便、高效的特性，逐渐在高速数据采集、处理与传输中得到越来越广泛的应用。因此在本文设计中采用第三种方式，将 A/D 采样来的数据经过 FIFO 缓存然后

传递给主机。

选用的芯片是 IDT72V2105。它的容量是 256K×18bit。这部分是将 AD 采集来的数据进行存贮，半满后将自动产生 HF(半满信号)告诉 CPLD，CPLD 将产生控制信号 LINT#(中断信号)，从而系统在中断服务程序中将数据行采集。

3.3.4 AD 转换模块设计

由于 FIFO 的存储空间很小，它只能用作数据缓存的作用，而不能存储大批量的数据，所以必须保证 PCI 读取数据的速度稍大于模数转换的速度，PCI 支持的最大时钟频率是 33MHz，如果采用 32 位的 DMA 传输，对于 14 位的转换精度的模数转换芯片而言，为了不丢失采集的数据，同时又最大限度的利用模数转换器的转换速度，模数转换的速度应选择在 66Msps，但考虑到 Windows 并非实时的操作系统，计算机响应中断和建立 DMA 传输都需要一定的时间，因此选择了 AD9240,是可以满足要求的。

由 AD 公司推出的模数转换芯片 AD9240，此芯片最高的转换率为 10Msps，转换精度为 14Bits，供电电压为 5V，输出为 TTL 电平，AD9240 在设计上将诸多的外围电路如采样保持、基准参考电压电路等都集成到了内部，将用户所需的工作降到了很低的程度。

AD9240 采用低成本的 CMOS 处理技术，由四级宽带输入取样保持电路(SHA)的流水线结构组成。每个流水线结构单元至少包括一个低分辨率的快闪 A/D 转换器，一个与 A/D 转换器相连的开关控制的 DAC 和一个级间剩余电压放大器(MDAC)。剩余电压放大器放大重构的 DAC 输出与快闪输入端间的电压差。最后一个流水线仅由快闪 A/D 电路组成。AD9240 原理图如图 3.8。

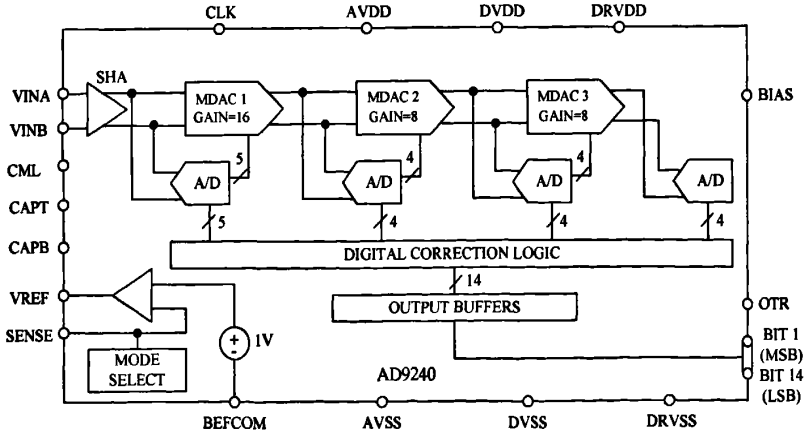


图 3.8 AD9240 的功能框图

AD9240 芯片的时序控制与传统的低速 AD 有所不同，它完全依靠时钟来控制其采

样、转换和数据输出。在第一个时钟的上升沿开始采样转换，当第三个时钟上升沿到来时，数据将出现在输出端口上，图 3.9 所示为其数据转换的时序图。由于 AD9240 是分级型模/数转换，所以，其数据输出时刻比相应的采样开始时刻要晚三个时钟周期。这一点在设计电路时应当注意。

AD9240 的转换过程是完全在采样时钟 AD_CLK 信号控制下进行的，无需其他信号进行控制。对 AD_CLK 信号的质量有较高的要求，它的占空比要求在 45%至 55%之间。AD9240 在每一个时钟的上升沿采样一次。

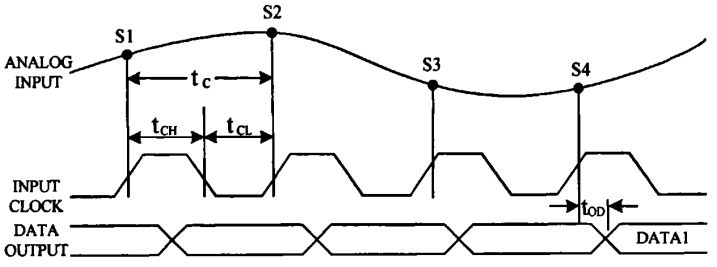


图 3.9 AD9240 数据转换时序

这部分的功能是将外部的模拟信号数据采集进来，再将其转化为数字信号，最后将其暂时存贮在 FIFO 中。

3.3.5 逻辑控制模块设计

逻辑控制模块是本系统设计的核心，负责协调各个模块之间的关系，完成系统的逻辑控制，它有以下几个具体功能：

(1) 产生 A/D 的采样时钟和 FIFO 的写时钟。

(2) 控制 FIFO 的复位及读写。

(3) 产生 FIFO 的读时钟和 PCI9054 的本地时钟 LCLK。

(4) 根据外部触发及 FIFO 的空满情况产生 PCI9054 的本地中断 LINT# 。

(5) 寄存主机发送的命令控制字：包括采样率、触发方式等。

(6) 作为本地处理器控制 PCI9054 本地端的数据传输。

本设计采用复杂可编程逻辑器件 CPLD 来实现以上功能。

CPLD 是 20 世纪 80 年代末 Lattice 公司在 EPLD 基础上推出的 PLD 器件，它采用 EECMOS 工艺，增加了内部连线，改进了内部结构体系，因而性能得到了很大的提高，设计也更加灵活。

本设计中，将采用高密度 PLD 对整个系统进行控制，由于此模块要完成的控制逻辑并不是很复杂，并综合考虑芯片资源量、功耗等因素，本设计最终选择选用 Xilinx 公司 TQFP 封装的复杂可编程逻辑器件 XC95144XL 作为核心芯片来完成整个系统的逻辑

控制。Xilinx 公司的 XC9500 系列可编程逻辑器件是一款高性能、有特点的可编程逻辑器件。从结构上看，它包含三种单元：宏单元、可编程 I/O 单元和可编程的内部连线。它的主要特点是：

(1) 高性能：在所有可编程引脚之间 pin-pin 延时 5ns；系统的时钟速度可达到 100MHz；

(2) 容量范围大：XC9500 系列可编程逻辑器件的容量范围为 36～288 个宏单元，可用系统门为 800～6400 个

(3) 5V 在系统可编程，可以编程 10000 次；

(4) 具有强大的强脚锁定能力；

(5) 每个宏单元都有可编程低功耗模式；

(6) 没有用的引脚有编程接地能力。

XC95144XL-10TQ100C 内部有 144 个寄存器，3200 个可用门，宏单元为 144 个，系统的最高工作效率为 83.3MHz。

在数据采集卡中，CPLD 主要控制三个部分：A/D 转换模块、数据缓冲模块和 PCI 接口模块，以及时钟输入及外部触发输入等。图 3.10 为 CPLD 与外围电路的连接框图。

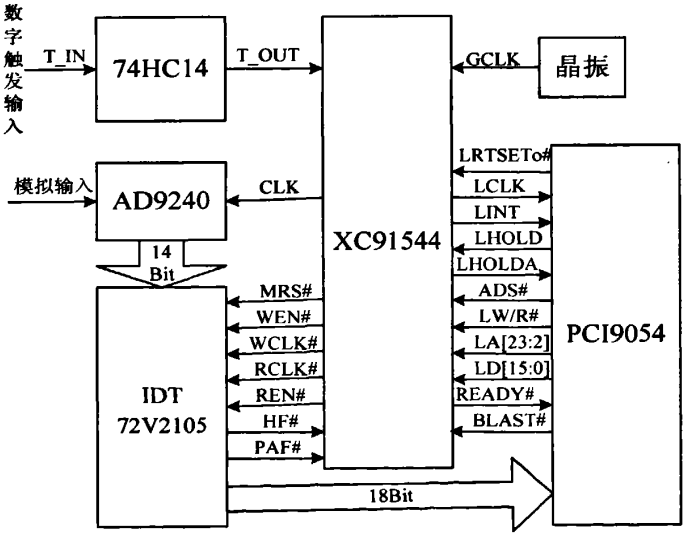


图 3.10 CPLD 与外围电路连接框图

本数据采集卡的时钟源采用 40MHz 的有源晶振，由 CPLD 的 GCLK1 脚输入，分频后得到 AD9240 的转换时钟、FIFO 的读写时钟以及 PCI9054 的本地时钟。外部数字触发 T_IN1 经过 74HC14 整形后输入 CPLD，作为整个系统的触发信号。

数据采集卡的 CPLD 要完成两种功能：第一种功能是从主机接收并寄存数据采集卡的命令控制字，确定采集卡的采样率、触发方式等参数。第二个功能是协调各模块的工作，根据各种状态信息和命令产生各模块的控制信号，保证在采集开始后，数据可以通

过 PCI 接口完整有序的传送到 PCI 总线上。

在设计中，CPLD 内部设定了分频、触发、命令、状态和延迟等五个寄存器，并能够通过计算机对这些寄存器进行读写，确定数据采集卡的工作方式。这五个寄存器分别为：

(1) 分频寄存器：共 2 位，用于设定 A/D 的采样率及 FIFO 的写时钟频率。

(2) 触发方式寄存器：共 2 位，低位控制触发源："0" 为内触发源，"1" 为外触发源。高位控制触发方式："0" 为同步触发，"1" 为延迟触发。同步触发是指当触发信号到来时，数据采集同步进行；延迟触发是指当触发信号到来时，数据要延迟一段时间以后才开始采集。内触发源是指将内部时钟经过 CPLD 分频后作为触发源；外触发源是指从采集卡外部引入一个触发信号，作为触发源。

(3) 命令寄存器：共 2 位，低位为 FIFO 的复位信号使能 "1" 有效；高位为 Local 总线的中断使能 "1" 有效。

(4) 状态寄存器：共 4 位，都是 "1" 有效。第 0 位为采集板的 DMA 工作状态；第 1 位为采集卡的寄存器读写状态；第 2 位为 FIFO 的半满状态；第 3 位为 FIFO 的将满状态。

(5) 延迟寄存器：共 6 位，用于设置在延迟触发方式下数据采集的延迟时间，最大延迟为 64 个采样周期

图 3.11 和图 3.12 分别为数据采集卡初始化及 DMA 传输的算法流程图。

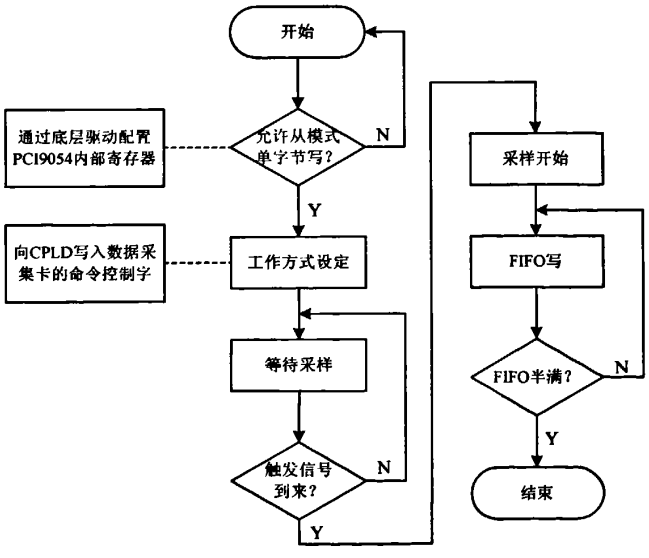


图 3.11 数据采集卡初始化流程图

数据采集卡要进行数据采集的工作，首先由主机以 PCI9054 从模式单字节写方式向 CPLD 写入命令控制字，确定采集卡的采样率、触发方式等参数，使能 A/D 转换及 FIFO

写操作，改写相应的状态控制字。在触发信号到来之后，数据采集开始，A/D 转换模块输出的数据在 FIFO 写时钟 FIFO WCLK 的控制下逐一写入 IDT72V2105，转换为 14 位数据(通过将 FIFO 的 D15~D12 置 0)。

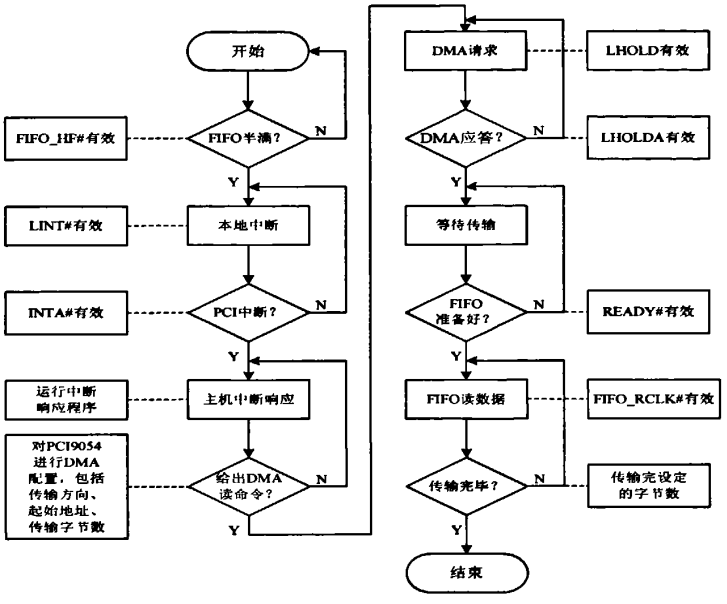


图 3.12 数据采集卡进行 DMA 传输的算法流程图

数据采集开始后，FIFO 将分别经历全空，将空，半满等状态，当 FIFO 的半满信号 FIFO HF#由“1”变为“0”时，CPLD 将使 LINT#有效，引发 PCI9054 的本地中断，接着会引发 PCI 中断 INTA#，向计算机发送中断请求信号。主机 CPU 响应中断，在中断响应程序中给出 DMA 读命令，包括起始地址、传输字节数及传输方向等。这时主机将 DMA 控制权交给了 PCI9054，然后 PCI9054 通过 LHOLD 向本地控制器 CPLD 申请 Local 总线的控制权，CPLD 通过 LHOLDA 响应，这样 PCI9054 也取得了 Local 总线的控制权，成为 PCI 总线和 Local 总线两端的主控设备。根据其内部寄存器的配置，PCI9054 将本地地址空间映射到 PCI 内存空间，接着启动本地总线的 DMA 读周期，CPLD 在收到 PCI9054 发出的地址选通信号 ADS#、读信号 LW/R#及地址信号 LA0 后，使能 FIFO 读使能信号 FIFO_REN#，同时有效 READY#信号通知 PCI9054，开始 DMA 传输。主机将通过 DMA 方式读取 131072(IDT72V2105 的容量为 256K×18bit，半满时已写入 128K 个字)个数据，并在最后一个数据传输前，将 BLAST#信号置为有效，CPLD 在一个时钟周期后使 FIFO 的读使能信号 FIFO_REN#无效，完成一次数据传输。

3.3.6 逻辑控制功能的 VHDL 实现

本设计中，选用了当前流行的 Altera 公司的 EDA 工具 QuartusII 来完成采集卡的逻

辑设计和时序仿真，它支持原理图编辑方式、VHDL 文本编辑方式、VerilogHDL 文本编辑方式等。这里，采用原理图编辑与 VHDL 文本编辑相结合的方式，先用 VHDL 分别编写各个控制模块，编译后产生独立的图形逻辑符号，再使用原理图编辑的方法，将各个模块连接起来，完成整个数据采集控制系统。用 VHDL 编写的系统电路模块主要包括命令控制寄存器模块、时钟发生模块、地址译码模块、FIFO 读写状态机模块和本地总线状态机模块等。

我们将控制模块的功能分为三部分：地址译码部分、时钟发生部分和逻辑控制部分。

(1)地址译码部分：对于 Local 总线的读写而言，通过地址译码，可以确定想要的操作对象。通过设置 PCI9054 的 Local 总线的寄存器组，可以使这些资源的地址固定在 Local 总线的某一区域。再通过设置 PCI 配置寄存器可以将 Local 总线的这一段区域映射到计算机内存空间，这样我们就可以通过计算机对存储空间的读写来完成对 Local 总线存储空间的操作。在 Target 传输中，因为其是单周期传输，我们只要在 CPLD 中定义 Local 总线侧地址空间的某一个地址位，并将它送给 PCI9054 相应的寄存器，在 DMA 传输中，我们要将在 CPLD 中定义的首地址给 PCI9054 相应的寄存器。

(2)时钟发生部分：时钟发生模块:数据采集卡的采样频率是可以通过命令控制字选择的。数据采集前，先将分频信息写入分频寄存器，分频电路将根据分频寄存器中的数据对外部时钟源输入的时钟进行分频，然后将分频后的时钟分别输出到模数转换电路和数据缓冲电路。本模块输入为 40MHz 的 GCLK 信号，输出包括 AD 转换时钟 AD_WCLK，FIFO 写时钟 FIFO_WCLK，FIFO 读时钟 FTFO_RCLK 和 PCI9054 本地时钟 LCLK。主要程序为：

```
ENTITY CLK IS
PORT(GCLK,A,B:IN STD_LOGIC;
        CLK1,CLK2:OUT STD_LOGIC);
END CLK;
ARCHITECTURE RTL OF CLK IS
SIGNAL Counter_A: STD_LOGIC_VECTOR(6 DOWNTO 0);
SIGNAL Counter_B: STD_OGIC_VECTOR(6 DOWNTO 0);
SIGNAL Counter tmp: STD_LOGIC;
BEGIN
  PROCESS(A,B,Counter_B)
  BEGIN
      IF(A='0' AND B='0') THEN
      Counter_B<=" 1010000";
      ELSIF(A '1' AND B '0') THEN
```

```
Counter B<="O101000";
ELSIF(A='0' AND B=' 1') THEN
Counteree B<="0011010";
ELSIF(A ='1' AND B='1') THEN
Counter B<="0010100";
ELSE NULL;
END IF;
END PROCESS;
PROCESS(GCLK,Counter tmp,Counter_A,Counterwe_B)
BEGIN
  IF(GCLK='1' AND GCLK'EVENT) THEN
      IF(Counter_A=Counter_B) THEN
      Counter_A<=(OTHERS=>'0');
      Counter trnp<=NOT Counter tmp;
      ELSE
      Counteres A<=Counteres A+1;
      END IF;
    END IF;
  CLKI<=Counter_tmp;
  CLK2<=Counter_A(0);
  END PROCESS;
END RTL;
```

(3) 逻辑控制部分[16]：主要是用来根据 FIFO、PCI9054、状态寄存器和命令寄存器的信息来产生等待、中断和传输终止等控制信号。主要程序为：

①产生 Local 总线仲裁允许信号(LHOLD 与 LHOLDA 直接相连)：

```
always@(posedge LCLK0)
    begin LHOLDA<=LHOLD;
end
```

②完成地址译码，产生目标设备准备就绪信号(如 FIFO 的读写信号、主复位信号以及 READY 信号)：

```
always@(posedge LCLK0)
    begin
    F[2] <=(F[2] && !F[1]) || (!F[2] && !F[1] && F[0] && BLAST);
    F[1] <=(!F[1] && F[0] && !BLAST) || (!F[2] && F[1] && !F[0]);
```

```
F[0] <=(!F[2] && F[1] && !F[0]) || (F[2] && !F[1] && !F[0])|| (F[2] && !F[1]
&&BLAST) || (!F[2] && F[1] && !ADS) || (!F[2] && !F[1] && !F[0] && !ADS
&& !LA[18])|| (F[2] && F[1] && !F[0] && !ADS);
R[2] <=(R[2] && !R[1]) || (!R[2] && !R[1] && R[0] && BLAST);
R[1] <=(!R[1] && R[0] && !BLAST) || (!R[2] && R[1] && !R[0]);
R[0] <=(!R[2] && R[1] && !R[0]) || (R[2] && !R[1] && !R[0]) || (R[2] && !R[1]
&&BLAST) || (!R[2] && R[1] && !ADS) || (!R[2] && !R[1] && !R[0] && !ADS &&
LA[18])|| (R[2] && R[1] && !R[0] && !ADS);
READY <=( F[2] || !F[1] || F[0]) && (!F[2] || F[1]) && ( R[2] || !R[1] || R[0]) &&
(!R[2]|| R[1]);  //产生 READY 信号
RD <=(F[2] || !F[1] || F[0]) && (!F[2] || F[1])|| L_WE_RD || LCLK;; //产生 FIFO
的读信号
Reset_CS<=(R[2] || (!R[1]) || R[0]) && ((! R[2]) || R[1]);
Reset_pr <= Reset_CS || L_WE_RD || LCLK;
Reset_clr <= Reset_CS || (! L_WE_RD) || LCLK;
Reset1 <= !(Reset_clr && Reset2);
Reset2 <= !(Reset_pr && Reset1);MR<=Reset2; //FIFO 的复位信号
end
```

3.4 PCB 设计与硬件调试

在上面中我们详细探讨了数据采集卡功能模块的硬件电路设计[25]方案。本节将讨论有关 PCI 板卡 PCB 设计的注意事项，最后简单介绍 PCI 数据采集卡的硬件调试过程。

3.4.1 PCI 板卡及其连接器的类型

PCI 局部总线定义了两种扩展板连接器，一种是基于 5V 环境的，另一种是基于 3.3V 环境的。同时定义了三种电器类型的扩展板，分别为 5V 板、3.3 V 板和通用板。5V 的扩展板是按 SV 信号环境设计的，只能插于 5V 的连接器，同样道理，3.3V 的扩展板是按 3.3V 环境设计的，只能工作在 3.3V 环境。通用板则能够检测出当前所用的信号环境属于哪一种，并且能够自适应该环境，因此它可以随便插入任意一种连接器中。所有的 PCI 连接器和板卡上都有相应的定位键以防止将板卡插入不适当的位置。

三种类型的扩展板都具有 5V 和 3.3V 两种电源供电方式，至于扩展板上的元件，有可能包含 5V 的，也有可能包含 3.3V 的，或者两者都有。各种类型板子之间的区别在于它们所采用的信号协议，而不是电源线的连接，也不是板上所包含的元件技术。

通用板上的 PCI 元件必须使用符合 5V 或者 3.3V 信号环境的 I/O 缓冲，这种符合双

重环境要求的缓冲有多种实现方法。一般来说，总是希望它是一个具有双重电压的缓冲，也就是说，利用两种电源的任何一种都可以工作。它们的电源应该来自于标有“$V^{I/O}$”的电源引脚上，在 PCI 局部总线上，这样的电源引脚总是与所用信号环境相对应的电源汇流排相连接。在 5V 信号环境下，这些缓冲将从 5V 电源上得到供电，而当同一块板插入 3.3V 连接器上，这些缓冲便由 3.3V 电源供电。这样就使通用板符合任何一种信号环境。

由于所选用的接口芯片 PCI9054 内核供电电压为 3.3V，它本身就具有双重缓冲—I/O 接口兼容+5V 和+3.3V 两种电压，因此，本数据采集卡被设计成能适应两种信号环境的通用板，在板卡上 PCI 接口芯片到 PCI 连接器之间不需要加专用的电压缓冲芯片，也不需要从标有“$V^{I/O}$”的电源引脚供电，就可以插入 5V 或 3.3V 任意的 PCI 扩展板连接器。

3.4.2 PCI 扩展卡的走线、布局及去耦

PCI 规范推荐 PCI 卡要采用多层板。在印制板上要设计单独的电源层与地层，可以采用四层板，如果板上元件较多，走线复杂可以采用六层板或八层板，另外在印制板布线时要充分考虑电磁兼容的需要，合理布线，对电源要充分去耦，连接器的每个 VCC 引脚必须要接退耦电容，均值不小于 0.01μF。当然，如果板上的元件较少，你可以采用两层板，这时候会增加布线的难度，同时也降低了板卡的抗干扰能力，更需要做好电源的去藕和抗干扰工作。

为增强板卡的抗干扰能力和工作稳定性，这里采用了六层板的设计，除了三个信号层之外，还分别设计了一个独立电源层和一个独立地层。其中电源层采用了“分裂的电源层”技术，在电路中把电源层分为两部分，其中外围的为+5V,里面的为+3.3V；而地层也分为模拟地和数字地两部分，并且通过一个 4μH 的磁珠在一点接地，起到了良好的抗干扰效果。

在画电路图的过程中我们发现 PCI9054 的引脚分配和板卡金手指的引脚分配基本上都是对应的，这是因为 PCI 连接器上的引脚已经考虑到六层板的布线和布局，这大大简化了电路板的布局，降低了布线的难度。

PCI 总线信号的传输通过反射波实现。当总线驱动器驱动某一信号时，往往只将信号电平驱动到实际所需电平的一半，信号传送到终点时反射回来，从而使得信号电平加倍，达到驱动所需的电平。当总线工作于 33MHz 时信号往返的时间不得超过 l0ns，这种信号传输要求驱动器的输出阻抗与被驱动总线的特征阻抗相匹配。PCI 规范对信号时序也作了详细的规定。例如，当设备工作于 33MHz 时，要求总线上输入信号的建立时间小于 7ns，输出信号满足时钟上升沿到输出信号有效小于 11ns 等。

对 PCI 插卡上的 PCI 信号线的最大走线长度也有规定：

(1) PCI 总线的 32 位部分的所有信号线最大走线长度必须限定在 1500mil 以内。

(2) PCI 总线的 64 位部分的所有信号线最大走线长度必须限定在 2000mil 以内。

(3) PCI 时钟信号线的长度必须是 2500mil±100mil，并且只能和插卡上的一个负载连接。

在实际电路中，另外，CLK 信号建议走成蛇形线并用地线屏蔽。蛇行线不但可以进行阻抗匹配，另外还能起到滤波电感的作用。

由于数字电路的高速信号在高低电平之间迅速变化时会引入噪声，导致其电源不纯净，而模拟器件往往具有较高的灵敏度，容易引入干扰。因此，必须将数字电源和模拟电源分开，以免数字信号干扰模拟信号。另外，良好的去耦和滤波也是获得纯净电源的关键。每个芯片的电源引脚附近并联去耦电容和旁路电容，去耦电容为芯片提供局域化的直流，这样,瞬态电流就可以取自去耦电容；旁路电容能消除高频辐射噪声和抑制高频干扰。为了保证电路板在极其恶劣的情况下仍可以稳定的工作，我们在 PCB 板电源处(金手指附近)为+3.3V、+5V、+12V、-12V 以及+$V^{I/O}$ 等电源引脚都加了 10μF 以及 0.1μF 的电容来去除高频和低频干扰，另外在每个芯片的电源与地之间都加了 0.1μF 的去耦电容，起到了良好的去耦效果。

3.4.3 PCI 采集卡的硬件调试

(1) 接口电路的测试：

在板卡制做好了后，首先对板卡进行检查，看看表面有没有焊错的元件以及有没有短路和虚焊的现象。用万用表的欧姆档进行分析，一个笔头接电路板的地，另一个笔头对各个点进行分析，参照电路图就知道有没有短路和虚焊的现象了。

(2) 用 PlxMon 进行硬件测试：

软件调试工具 PlxMon，是 PLX 公司提供了一个 905X 的专用调试软件，可以从网上免费下载。PlxMon 包括以下功能：PCI 总线的探测与选择；配置寄存器的检查和修改；内存空间的显示、修改和填充；EEPROM 内容的读写等。利用这个工具，可以很清楚地看到 EEPROM 的内容以及 PCI 配置寄存器和局部配置寄存器的内容，另外，用户还可以进行内存和 I/O 端口的读写以及 DMA 传输。PlxMon 的安装程序里还提供了 PCI9054 的 Windows 驱动。

在确定没有短路和虚焊后就可以将板卡插到电脑的 PCI 插槽上，系统加电后，BIOS 会利用总线枚举器（Bus Enumerator）对总线上的设备进行遍历，通过读各个设备的信息，以确定它们的资源请求，再将这些信息通知系统。如果布线或 EEPROM 内容设置不当的话，会导致系统死机。如果布线正确且内容设置得当，系统正常启动，会出现“找到新硬件”的向导界面。

在本设计的测试过程中，Windows 可以正确识别数据采集卡。查看设备属性，已分配了相关的硬件资源。如图 3.13 所示。

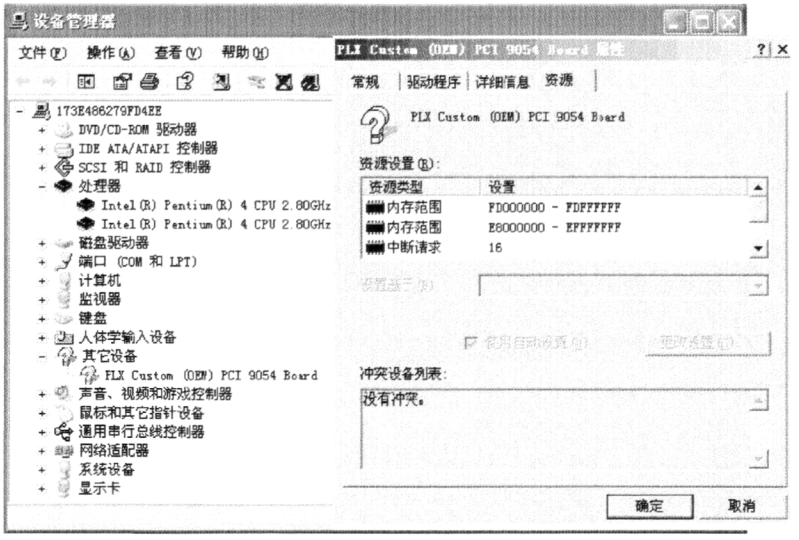


图 3.13 WINDOWS 检测到 PCI9054

3.5 小结

本章提出了系统硬件的总体设计方案，对系统各个模块的设计要求、芯片选择及其特点作了深入介绍，在此基础上完成了 PCI 接口模块、信号调理模块、A/D 转换模块、数据缓存模块和逻辑控制模块的硬件电路设计，通过 Protel99SE 设计线路板，完成了采集卡的硬件制作，并利用 PlxMon 对采集卡进行了硬件测试，已经达到预期目标。

4 驱动程序设计

数据采集卡硬件设计制作完成以后，如果没有驱动程序的支持，还是无法应用的。在本系统中，也就是开发 PCI9054 作为接口芯片的 PCI 采集卡的驱动程序。

设备驱动程序的设计是硬件开发过程中的一个重要环节，好的驱动程序是硬件功能能够正常实现的重要保证，它不仅与所要设计的硬件设备有关，而且还涉及操作系统的内核。Windows 以其友好的用户界面，不仅成为办公环境首选的操作系统，也受到工程技术人员的关注，被广泛运用于某些控制领域。在设计和使用 PCI 设备时，常要在计算机的软件中访问和控制硬件设备，进行中断响应、I/0 端口读写或存储器访问等。但在 Windows 操作系统(包括 Windows95/98，WindowsNT，Windows2000/XP 等)下，为了保证系统的安全性、稳定性和可移植性，对应用程序访问硬件资源加以限制，这就要求设计设备驱动程序以实现 PC 机的软件对 PCI 设备的访问。

4.1 驱动程序开发概述

从广义上来说，设备驱动程序就是控制硬件设备的一组函数。设备驱动程序提供连接到计算机的硬件的软件接口，它是操作系统的一个信任部分。当驱动程序装入后就称为操作系统内核的一部分。驱动程序总是使设备看起来像是一个文件，可以打开设备的一个句柄，然后应用程序可以在设备句柄最后关闭之前向驱动程序发出读写请求 PCI 驱动程序的开发，就是取得 PCI 板卡所占用的各种资源(内存、端口、中断和 DMA 等)，并提供给用户一条可以访问这些资源的途径。对于所有的 PCI 设备，基本上都可以用下面的方法来开发驱动程序。

(1)取得 PCI 板卡所占用的资源

PCI 设备是即插即用的，它占用的所有资源都是由系统浮动分配的，要访问这些资源，就必须先对这些资源定位并进行锁定，进而为我所用。

(2)建立应用程序和驱动程序之间的映射关系

因为在 Windows98 和 Windows NT 等操作系统下，各种硬件资源的访问对用户来讲是透明的，用户不能在上层应用程序里直接访问硬件资源，只能靠底层驱动程序来访问它。因此，用户如要实现对板卡的控制，就必须在应用程序和底层驱动程序之间架起一座桥梁，用以实现应用程序和驱动程序之间的联系。通过这个桥梁，用户可以实现参数的传递和硬件资源的访问。

(3)通过驱动程序实现对硬件资源的访问。

用户最终的目的就是实现对硬件资源的直接访问，而在上层应用程序中却根本就做不到。驱动程序提供了可以直接访问各个资源的函数，用户可以通过驱动程序来实现对

所需资源的访问。

驱动程序需要完成的主要任务有：系统识别、寄存器配置、本地端的操作等。

4.2 驱动程序设计方案

4.2.1 Windows 驱动程序模型（WDM）

Windows 驱动程序模型有两个重要功能[33][34]。其一是描述设备驱动程序标准结构的核心模型；其二是 Microsoft 为常见设备提供的一系列总线和类驱动程序。核心模型描述设备驱动程序如何安装和启动，以及如何为用户请求服务，如何和硬件打交道。Microsoft 提供的系统驱动程序为许多标准型设备的服务提供所需的所有基本功能，有支持不同总线的系统驱动程序、实现标准 Windows 功能的类驱动程序、提供静态图像和扫描仪等处理框架的静态图像体系结构（STI）。

在 Windows 环境下的设备驱动程序有三种类型：

VxD(Virtual Device Driver)——运行在 Windows95/98/Me 操作系统中；

KMD(Kernel Model Driver)——运行在 WindowsNT 操作系统中；

WDM(Win32 Driver Model)——微软公司的全新的驱动程序模型，支持即插即用，电源管理和 WMI（Windows 管理诊断），运行平台为 Windows98/2000/XP 操作系统。目前，在 Windows 平台下，WDM 已成为主流的驱动模式[22]。

由于 WDM 这种新兴驱动程序模型的目的是提供一种灵活的方式简化驱动程序开发，实现对新硬件的即插即用管理，减少并降低驱动程序开发的数量和复杂性，所以在系统的驱动开发中，选择 WDM 驱动程序模型。其模型工作原理如图 4.1。

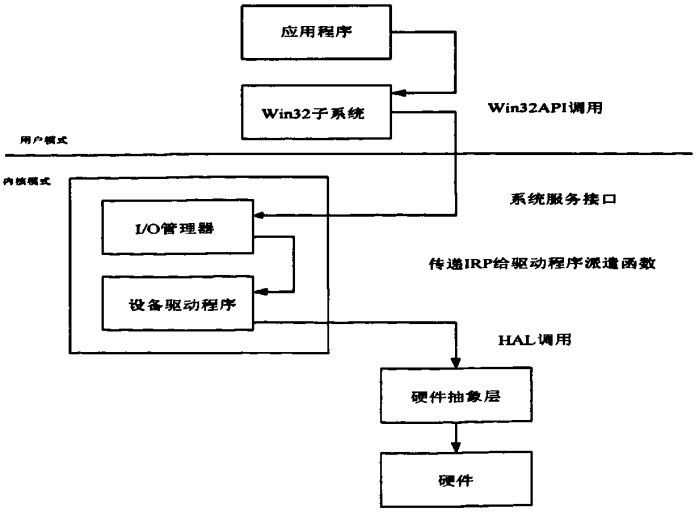


图 4.1 WDM 驱动模型原理图

在上图中，应用程序对设备的 I/O 请求通过调用 Win32API 函数，提交给操作系统内核，该调用由 I/O 系统服务接口接收，然后 I/O 管理器会为这个请求构造一个合适的 IRP，并将其传递给设备驱动程序，设备驱动程序再与硬件或底层驱动程序交互，完成对该 IRP 的处理，最后由 I/O 管理器根据该 IRP 的状态将 I/O 请求的结果返回给应用程序。

4.2.2 WDM 驱动模型的特点

(1)平台之间的可移植性

内核模式驱动程序的源代码应该可以移植到所有 Windows NT 平台，WDM 驱动程序在其定义中就规定了其源代码可以在 Windows98、Windows2000、WindowsXP 之间相互移植。为了实现这种可移植性，驱动程序应该全部用 C 语言来编写，避免使用编译器厂商专有的语言特征，这样将使得仅需要重新编译就可以跨平台使用了。

(2)硬件和软件的可配置性

避免对设备特征或某些系统设置做绝对假设，为了实现可配置性，首先应该在代码中避免直接引用硬件，应该使用 HAL 工具或调用低级总线驱动程序，或者实现一个标准或定制接口，并通过控制面板程序与用户交互。

(3)永远抢占优先和永远中断

在大部分时间中，CPU 执行在 PASSIVE_LEVEL 级上，所有的用户模式代码也运行在 PASSIVE_LEVEL 级别，并且驱动程序的许多活动也都发生在 PASSIVE_LEVEL 级上。当 CPU 运行在 PASSIVE_LEVEL 级时，当前运行的线程可被任何优先级大于它的抢先。然而，一旦 CPU 的 IRQL 大于 PASSIVE_LEVEL 级，线程抢先将不再发生，此时 CPU 执行在使 CPU 越过 PASSIVE_LEVEL 级的任意线程上下文中。

(4)多处理器平台上的多处理器安全

使用对称多处理器模型，即所有的处理器都是相同的，系统任务和用户模式程序可以执行在任何一个处理器上，并且所有处理器都平等地访问内存。

(5)基于对象

驱动程序和内核例程使用的许多数据结构都有公共的特征，这些特征集中由对象管理器管理。

4.2.3 驱动程序开发工具的选择

开发设备驱动程序有多种开发工具[35][36]可以选择，主要包括：

(1)DDK

DDK 是 Microsoft 出品的设备驱动程序开发工具 DDK（Device Developer Kit）。该软件包中包括了有关设备驱动开发的文档、编译需要的头文件和库文件、调试工具和程

序范例。另外在 DDK 中还定义了一些设备驱动程序可以调用的系统底层服务。它有 Windows98 DDK 和 Windows 2000 DDK 两个版本。前者能够开发 Windows95/98/Me/NT 下的 VxD，KMD 和 WDM 驱动程序，后者可以开发 Windows98/Me/NT/2000 下的 KMD 和 WDM 驱动程序。

(2) VtoolsD

Vireo 公司出色的 VtoolsD 由可视的 VxD 代码生成器 QuickVxD，C 运行库、VMM/VxD 服务库、C++类库以及 VxD 的装入程序等组成。利用 QuickVxD 生成的框架程序和经充分测试过的C运行库或C++类库可以绕过DDK用C或C++来编制驱动程序，这就大大地简化了开发的难度，提高了可靠性。框架程序可以直接在 Visual C++集成开发环境中用 NMAKE 编译为 VxD。

(3) WinDriver

WinDriver 是美国 KRFTech 公司出品的用于编写驱动程序的另一种工具包。它包括一个类似 QuickVxD 的代码生成器 WinDriver Wizard、一个 WinDriver 发行包、两个公用程序。与 VToolsD 一样，WinDriver 工具包的优点在于允许编程人员用 MSDEV Visual C/C++、Borland Delphi 或者其他 Win32 编程工具软件在用户模式（User Mode）上编写设备驱动程序，而不是将大量精力放在编写那些复杂的，难于调试的内核模式代码。WinDriver 生成的代码可以运行于 Windows95 /98/NT3.51/NT4.0 和 Windows2000 等各平台。

(4) DriverWorks

NuMega 公司的 DriverWorks 以面向对象的(OOP)的方式，将编写 WDM 及 WINNT 驱动程序所需的对内核模式访问及硬件的访问封装成类。这样只要在它的向导程序的指引下，根据硬件的具体参数填写必要步骤，就可以很方便地完成所需驱动程序的框架。最后根据具体的要求添加新的类对象和所需的代码就可以成功地完成自己的驱动程序。DriverWorks 开发 WDM 时，其 DriverWizard 向导将自动生成 WDM 工程文件，还自动生成 INF 安装信息文件和一个 Console 测试程序，用来检测驱动是否正常工作。DriverWorks 是基于 VC++的，它生成的是标准的 VC 工程，只要将所建的工程在 VC 下编译，就可以生成最终的设备驱动程序。

以上介绍的各种工具各有优点。DDK 功能强大，编程灵活，适用范围广，可应用于各类硬件驱动程序的编写，但对编程人员的要求较高，编程难度较大。VToolsD 主要工作环境是在 Windows95/98 下，它具有较强的开发能力和较高的开发效率，是编程人员常用的工具。WinDriver 的适用面比前两者窄，它主要针对 ISA/PCI 插卡，而对其他类硬件的技术支持较少，但它编写的程序可同时工作在 Windows95/98/NT 操作系统。DriverWorks 对于 Windows NT 下和 Windows98 与 Windows2000 共同支持的 Win32 驱动模型（WDM）设备驱动程序的开发提供完全的支持。DriverWorks 中包含一个非常完善

的源代码生成工具（DriverWizard）以及相应的类库和驱动程序样本，它提供了在 C++下进行设备驱动程序开发的支持。另外，DriverWizard 还能生成专为特殊设备定制的代码，比如 USB 设备、PCI 设备、即插即用设备、ISA 设备等等。本课题选用 Numega 公司的 DriverWorks 作为驱动程序开发工具开发 Win2000 下的 WDM 模型的驱动程序。

4.3 驱动程序框架的建立

4.3.1 驱动程序开发环境的建立和使用

(1) 首先安装 Microsoft Visual C++6.0；

(2) 安装针对 Windows2000 的 DDK；

(3) 安装 DriverStudio；

(4) 运行 NuMega DriverStudio 下 Tools 下的 Setup DDK and Start MSVC 程序；或自己在 VC IDE 中手动设置 BASEDIR 和 CPU 等环境变量；

(5) 用 VC6.0 打开 NuMega\Driver Studio\DriverWorks\Source\VdwLibs.dsw 工程。编译此工程得到以后需要的库文件 vdm_wdm.lib；

(6) 设置 VC6.0 的 Compile\Config 的配置为 WDM Free 或 Checked；

(7) 编译 VdwLibs.dsw 库建立 DriverWork 库文件，对 Free 和 Checked 都要编译；

(8) 开始 WDM 驱动程序的开发。

到此编译环境就完全建立好了，在以后设备驱动程序的编译过程中，只要选择编译成扩展名为.sys 的文件即可。

4.3.2 使用 DriverWizard 生成驱动程序框架

步骤 1：从程序组中选择或 VC6.0 的主菜单 DriverStudio 中选择 Driver Wizard 菜单项，便会弹出 DriverWorks NT/WDM 驱动程序的基本框架生成向导，如图 4.2 所示：

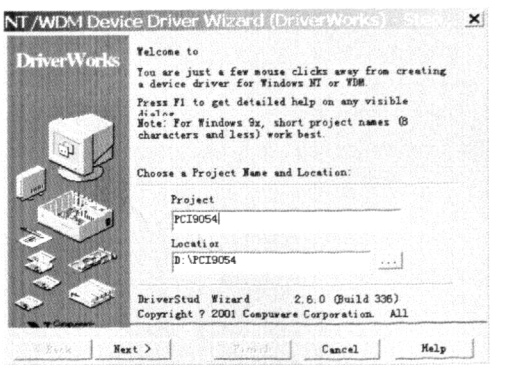


图 4.2 框架生成步骤 1

步骤 2：在 Project 下填入工程名：PCI9054。在 Location 选择工程文件存放的目录，再点 Next 进入下一个对话框。选择 WDM 方式，再点 Next 进入硬件总线选择对话框。

步骤3：在 Select Hardware Bus 单选框中选择 PCI，并在下面的栏目中填入 PCI Vendor ID（如 10B5），PCI Device ID（9054）。这两项必须与你要访问的 PCI 设备的 VID 和 DID 一致。PCI Sub system ID 和 PCI Revision ID 可以不填。如填，也须与设备的 SVID 和 SDID 一致。如不填，须手动删除 INF 文件中的有关项。否则向导生成的 INF 文件不能完成该设备驱动程序的正常安装。如图 4.3 所示。

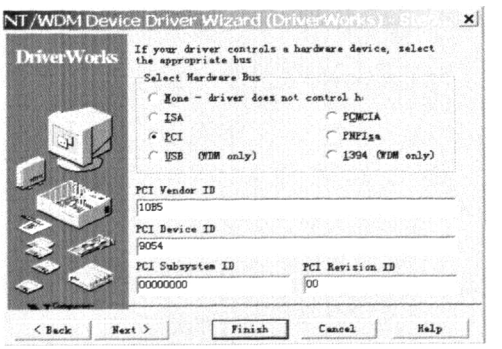


图 4.3 框架生成步骤 3

步骤 4：通过对设备类的定义和资源、接口、缓冲、电源的处理，选择是否产生应用层的测试程序以及调试操作等。此例采用的是默认方式。点击 Finish 按钮，如图 4.4 在接下来的提示框中点击 OK。接着向导会问你是否现在就要在 VC 中打开此工程。如选择是，向导不但会直接在 VC 中打开该工程，而且还会替你将编译环境给你设置好。接下来用户可以根据自己的需求添加相关内容了。

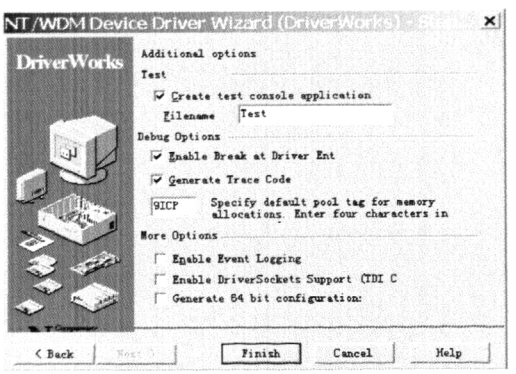


图 4.4 框架生成步骤 4

4.3.3 添加代码

生成了工程框架后，需要对设备具体操作编写和修改代码，添加设备控制例程。接下来编译驱动程序，然后安装驱动程序，安装成功后用测试程序进行测试功能。

4.4 驱动程序的功能实现

按照上节提示逐步完成各操作后，此时已经具备了一个驱动程序以及做测试用的应

用程序的基本框架，可以在集成环境下修改有关程序，增加相关的具体操作代码，然后就可以编译和调试了。设计设备驱动程序主要的面临问题是如何进行硬件操作，这是根据设备的不同而不同的。而硬件驱动程序的功能虽然千差万别，但基本功能就是完成设备的初始化[39][42]、设备对象创建、配置空间的访问以及对内存或 I/O 端口的访问等。

4.4.1 设备初始化例程

WDM 驱动程序和一般的 DOS/Windows C 语言程序不一样，它没有 main（）或者 WinMain（）函数入口，它向操作系统显露一个与 DLL 类似的名称为 DriverEntry（）的例程。DriverEntry（）例程是 WDM 驱动程序的初始化入口点，主要用来产生一个物理设备对象 PDO。使用硬件资源如 I/O 端口、内存、DMA 通道或者 IRQ 的驱动程序，必须在初始化的阶段说明系统资源的使用情况。当 WDM 驱动程序被装入时，内核调用此入口例程。在这个入口函数中，必须作必要的初始化设置，并设置必要的回调函数。函数部分代码如下：

以下是本驱动程序中 DriverEntry（）例程的源代码：

```
NTSTATUS DriverEntry(IN PDRIVER_OBJECT DriverObject  IN PUNICODE_
STRING RegistryPath)
{
    DriverObject->DriverUnload = DriverUnload;
    DriverObject->DriverExtension->AddDevice = AddDevice;
    DriverObject->DriveStartIo = StartIo;
    DriverObject->MajorFunction[IRP_MJ_PNP] = DispatchPnp;
    //设置各个 IRP 的处理函数，这三个 IRP 是每一个 WDM 驱动程序必须处理的。
      DriverObject->MajorFunction[IRP_MJ_POWER] = DispatchPower;
DriverObjet->MajorFunction[IRP_MJ_SYSTEM_CONTROL]=DispatchWmi;
   servkey.Buffer=(PWSTR)ExAllocatePool(PagedPool,
   RegistryPath->Length+sizeof(WCHAR));
   if(!servkey.Buffer)
   return STATUS_INSUFFICIENT_RESOURCES;
   servkey.MaximumLength = RegistryPath->Length + sizeof(WCHAR);
   RtlCopyUnicodeString(&servkey，RegistryPath);
   return STATUS_SUCCESS;
}
```

在 DriverEntry（）例程中，驱动程序要向操作系统登记并注册一些消息处理器，通过 RegistryPath 来找到位于注册表中的驱动程序参数，当驱动程序正确安装后，在

KEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Service 下可以找到我们的采集卡PCI9054。

4.4.2 创建设备例程

大多数的 WDM PDO 都是在 PnP 管理器调用该程序入口点时被创建的。插入新设备后，系统启动时，总线枚举器会发现总线上的所有设备，会自动寻找并安装设备的驱动程序，并由驱动程序中的处理 PnP 功能模块自动处理 AddDevice（）例程及其它 PnP 消息。AddDevice（）例程使用 IoCreateDevice（）函数创建设备对象，再使用 IoRegisterDeviceInterface（）函数将设备组成为一个特定的设备接口，然后使用 IoAttachDeviceToDeviceStack（）函数关联设备栈。采集卡设备创建例程的定义如下：

```
NTSTATUS Pci9054::AddDevice(PDEVICE_OBJECT Pdo)
{    t << "AddDevice called\n";
     Pci9054Device * pDevice=new(
          static_cast<PCWSTR>(KUnitizedName(L"Pci9054Device",m_Unit)),
// 设备名（DeviceName）=PCI9054Device
            FILE_DEVICE_UNKNOWN,
          NULL,
          0,
          DO_BUFFERED_IO| DO_POWER_PAGABLE // I/O 传输方式。
  )
           Pci9054Device(Pdo, m_Unit);
     if (pDevice == NULL)
     {t << "Error creating device Pci9054Device"
                << (ULONG) m_Unit << EOL;
          return STATUS_INSUFFICIENT_RESOURCES; //返回构造函数是否成功的状态代码。}
     NTSTATUS status = pDevice->ConstructorStatus();
     if ( !NT_SUCCESS(status) ) ) //如果不成功，返回错误状态并删除指针。
          { t << "Error constructing device Pci9054Device"
        << (ULONG) m_Unit << " status " << (ULONG) status << EOL;
       delete pDevice; }
    else //如果成功，向系统报告设备的电源状态发生了变化。
   {m_Unit++;
       pDevice->ReportNewDevicePowerState(PowerDeviceD0); }
```

return status;}

4.4.3 打开设备

驱动程序为设备的硬件层编程服务，但同样需要提供和应用程序进行通信的能力，从而达到应用程序控制设备的目的。我们的采集系统正是基于上述考虑，设计了 WDM 驱动程序的 I/O 控制，并提供与上层总控软件的 I/O 接口。在 Win2000 OS 中，应用程序实现与 WDM 驱动程序通信的过程是：应用 Win32CreateFile（）函数打开设备。然后用 DeviceIoControl（）函数和 WDM 进行通信，包括从 WDM 中读取数据和写数据给 WDM 两种情况，也可用 ReadFile（）函数从 WDM 读数据或用 WriteFile（）函数写数据给 WDM。当应用程序退出，用 CloseHandle（）函数关闭设备。这将产生对应于此设备对象的相应的 IRP。

在创建设备后，为了使 Win32 上层总控软件可见，我们必须为每个设备创建符号链接。创建符号链接有两种方法：第一种方法是采用显示的“硬编码”符号链接名，用户态程序必须类似的把设备名硬编码到源代码中，这种方式适用于 Windows NT OS；另外一种方法是使用“设备接口”，每个设备接口由一个 128 位的全局唯一标志符标志 GUID。用户态设备可以取得拥有此 GUID 的设备。本驱动程序采用第二种方法。

这里需要重点介绍的是 OpenByInterface（）函数，它用 GUID 接口方式与 WDM 通信，我们的程序中 OpenByInterface（）函数在 OpenByIntf.cpp 文件中，源代码如下：

```
HANDLE OpenByInterface(
GUID* pClassGuid,  // 设备的 GUID。
DWORD instance,  // 设备的序号。
PDWORD pError
// 存放应用程序打开设备时的错误编号。
)
{// 返回值为设备的句柄号。
HANDLE hDev;
CDeviceInterfaceClass DevClass( pClassGuid, pError );
if ( *pError != ERROR_SUCCESS )
return INVALID_HANDLE_VALUE;
CDeviceInterface DevInterface( &DevClass, instance, pError );
if ( *pError != ERROR_SUCCESS )
return INVALID_HANDLE_VALUE;
hDev = CreateFile(
// 类 DevInterface 描述了一个设备的接口，通过成员函数 DevicePath（）
```

```
// 可以返回一个指向设备路径的指针，传递给 CreateFile（），打开设备。
dwStatus=WDC_PciDeviceOpen(&hDev, pDeviceInfo, pDevCtx, NULL, NULL,
NULL);
if (WD_STATUS_SUCCESS != dwStatus)
{ ErrLog("Failed opening a WDC device handle. Error 0x%lx - %s\n",
dwStatus, Stat2Str(dwStatus));
goto Error;
}
/* 返回一个指向新设备的句柄*/
DevInterface.DevicePath（），
GENERIC_READ | GENERIC_WRITE，
FILE_SHARE_READ | FILE_SHARE_WRITE，
NULL，
OPEN_EXISTING，
FILE_ATTRIBUTE_NORMAL，
NULL
）；
if ( hDev == INVALID_HANDLE_VALUE )
*pError = GetLastError（）；
return hDev；
}
```

在上层总控应用程序中对应的语句为：

```
hDevice=OpenByInterface ( &ClassGuid，0，&Error)；
```

4.4.4 关闭设备

在退出应用程序之前，应当用 CloseHandle（）函数关闭设备，这将产生两个 IRP：IRP_MJ_CLEANUP 和 IRP_MJ_CLOSE。事实上，一个应用程序调用 CloseHandle（）函数，驱动程序首先收到 IRP_MJ_CLEANUP，驱动程序应当在“清除”例程中清除和此设备对象有关的 IRP。然后收到 IRP_MJ_CLOSE，关闭设备对象。在应用程序中对 CloseHandle（）函数的调用如下：

```
void CloseIfOpen( void )
{
 if ( hDevice != INVALID_HANDLE_VALUE )
  {
```

```
    if ( !CloseHandle( hDevice ) )
    hDevice = INVALID_HANDLE_VALUE; }
}
```

4.4.5 LOCAL 总线寄存器组、运行寄存器组的读写

系统上电时，由 BIOS 提供配置信息分配相应的主机资源。欲对 LOCAL 总线、运行和 DMA 寄存器组进行操作时，驱动程序只需去读取 PCI9054 的 PCI 配置空间的寄存器 PCI BAR0 或 PCI BAR1 的值，就可以获取该寄存器组在映射空间的首地址。寄存器 PCI BAR0 中的值为该寄存器组在 PCI 总线存储空间的基址；寄存器 PCI BAR1 中的值为在 PCI 总线 I/O 空间的基址。知道了以上寄存器组的基址，结合各个寄存器的偏移地址就可以从 PCI 的 I/O 空间或存储空间对以上寄存器进行读写操作。

对于 PCI9054 来说，本地控制寄存器被映射到了 PCI 配置寄存器的 BAR0 中。在用驱动程序进行调用的时候，首先要选择：

Pci9054.BaseAddressIndexToOrdinal(0)

这样，当前的 m_MemoryRange0 就被定义到了 PCI 的 BAR0 中，也就是说，对 PCI9054 的功能操作，完全由 m_MemoryRange0 类的函数完成。例如数据的读、写等，就会用到下面的操作：

m_MemoryRange0.ind(), m_MemoryRange0.outd()

4.4.6 DeviceIoControl（）函数的调用

调用 CreateFile（）函数成功后，应用程序就可以调用 DeviceIoControl（）函数与 WDM 进行通信了。对于 DeviceIoControl（）函数的调用，驱动程序根据 I/O 控制命令来决定该如何获取应用程序的缓冲器地址。我们在驱动程序中建立了 5 个函数来将采集数据传送至上层总控应用程序及进行底层硬件控制，他们分别是：

(1) 设备初始化函数 PCI9054_IOCTL_800_Handler;

(2) 启动采集函数 PCI9054_IOCTL_801_Handler;

(3) 停止采集函数 PCI9054_IOCTL_802_Handler;

(4) PCI 读 SRAM 函数 PCI9054_IOCTL_805_Handler;

(5) PCI 发出读数通道函数 PCI9054_IOCTL_806_Handler;

这里以停止采集函数的调用为例来介绍源代码。应用程序调用停止采集例程的函数是 Test_PCI9054_IOCTL_802(void)：

```
void Test_PCI9054_IOCTL_802(void)
{
    CHAR bufInput[IOCTL_INBUF_SIZE]; //设置输入缓冲区
```

```
    CHAR bufOutput[IOCTL_OUTBUF_SIZE]; // 设置输出缓冲区
    ULONG nOutput;// 调用驱动程序中设备 IO 控制接口
    printf("Issuing Ioctl to device - ");
    if (!DeviceIoControl(hDevice, //CreateFile 函数返回的设备句柄。
        PCI9054_IOCTL_802, //停止采集例程
        bufInput,//应用程序传递给驱动程序的数据缓冲区地址。
        IOCTL_INBUF_SIZE, //应用程序传递给驱动的数据字节数。
        bufOutput, //存放驱动返回数据的缓冲区地址。
        IOCTL_OUTBUF_SIZE, //存放驱动程序返回数据的缓冲区字节数。
        &nOutput, //存放驱动程序实际返回数据字节数的变量地址。
        NULL)//一个指向 OVERLAPPED 结构的变量地址，同步时置为 NULL。
)
  {
        printf("ERROR: DeviceIoControl returns %0x.", GetLastError());
        Exit(1);
  }
}
```

4.5 驱动程序的安装与调试

4.5.1 驱动程序的安装

在电缆故障在线定位系统的测试实验中，实验平台为 P42.8G，操作系统为 Windows2000Professional，内存 512M。硬件为 PCI 数据采集卡及相关设备，软件系统在 Visual C++6.0、Windows2000DDK、DriverStudio 下编译通过。

Windows2000 是依靠 INF 文件来得到硬件设备驱动程序的安装信息的。DriverWorks 的 DriverWizard 已经给我们在所建工程的 sys 目录下生成了一个设备信息文件(INF)。只要将文件中双引号中的提示改为相应的内容即可生成采集卡的设备信息文件。

[Strings]

ProviderName=“Your Company Name here” //公司名称

MfgName=“Name of Hw Manufacture here” //硬件制造商名称

DeviceDesc=“Desrcription of Device here” //设备描述

DeviceClassName=“Desrcription of Device class here” //设备类的描述

当系统加电时 Windows 操作系统会自动检测所有外设，当第一次检测到我们的设备时系统会提示用户指定新硬件的驱动程序。根据提示指定了我们修改过的 inf 文件，

以及编译后生成的.sys 系统文件系统就自动安装好了新硬件的驱动程序。或者可以用控制面板中的添加新硬件来搜索新硬件。

安装完成后，此时在设备管理器中可以找到该设备，I/O 管理器已经为该设备分配了存储器资源和中断资源，本文所设计板卡其属性界面如图 4.5 所示。

驱动程序安装好以后，在应用程序中就可以象打开其他驱动程序一样用：

CreateFile 打开设备

CreateFile(sLinkName,GENERIC_READ|GENERIC_WRITE,

FILE_SHARE_READ,NULL,OPEN_EXISTING,0,NULL);

用 DeviceIoControl 与设备通讯，以及 ReadFile/WriteFile 来读写设备。

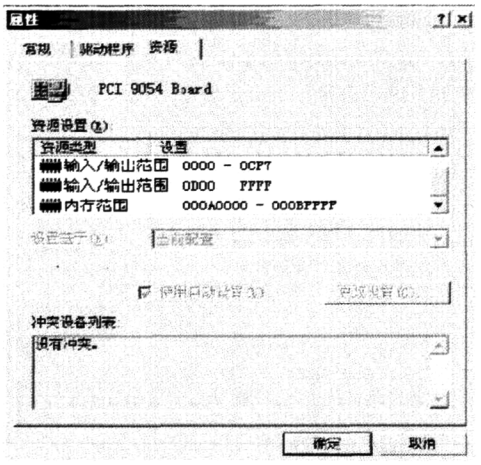


图 4.5 数据采集卡属性

4.5.2 驱动程序的调试

编写设备驱动程序和调试都需要对操作系统底层有一定的了解，并且需要程序员有一定的硬件知识背景。而且驱动程序工作在具有最高权限的 Ring0 层，程序行为处在无保护状态，不当的驱动程序可能使设备工作不正常，干扰其他设备，严重的可以引起系统崩溃。所以必须保证驱动程序可靠。

驱动程序的调试还是用到了硬件调试时的调试软件 PlxMon。PlxMon 包括以下功能：PCI 总线的探测与选择；配置寄存器的检查和修改；内存空间的显示、修改和填充；EEPROM 内容的读写等。

在本次设备驱动程序调试过程中，主要遇到了以下一些问题：

(1) 崩溃

所谓崩溃就是指计算机蓝屏死机，在设备驱动程序的调试过程中，多次遇到计算机崩溃的情况，经过多次的调试，总结出导致系统崩溃的主要原因是错误的指针引用，我们知道在用户层应用程序开发中，引用一个空指针或者无效的指针会导致程序抛出无效

的地址异常，而在设备驱动程序中，一旦发生访问无效地址的操作，系统会直接蓝屏死机，所以在设备驱动程序开发过程中要严格判断指针的有效性。

(2) 驱动程序不启动

当更新驱动程序时，设备管理器可能指出必须重新启动计算机才能使驱动程序生效，这说明驱动程序在装入时返回了一个错误。这样的错误有两种原因，常见的是忘记删除设备或设备的符号链接，另一种是在驱动程序重新装入时，Windows 需要调整系统资源的分配。在这种情况下，如果必须重新启动才能满足资源分配的要求，Windows 将不会启动驱动程序。

4.6 应用程序设计

前文已经介绍了 WDM 驱动程序的实现方法。但是，对于一个完整的数据采集系统来说，编写驱动程序并不是最终目的，而是需要应用程序来调用驱动程序并实现一些功能。所以我们对上层应用软件进行研究和实现是非常必要的。

上层应用软件是整个系统中直接面对用户的重要部分。在实际应用中一般要求提供良好的可视化人机对话界面（MMCI，Man-Machine Conversation Interface），用户能够通过它根据实际情况的需要发出命令，调动系统底层软、硬件来准确、可靠的执行相应的操作，从而完成实际任务

为了处理数据方便的需要，将采样得到的数据上传至 PC 机。因此，本文开发了一套电缆故障定位的后台分析软件，该软件是以 WindowsXP 为平台，测试模块界面用 VC++6.0 编写。图 4.6 是软件的主界面。

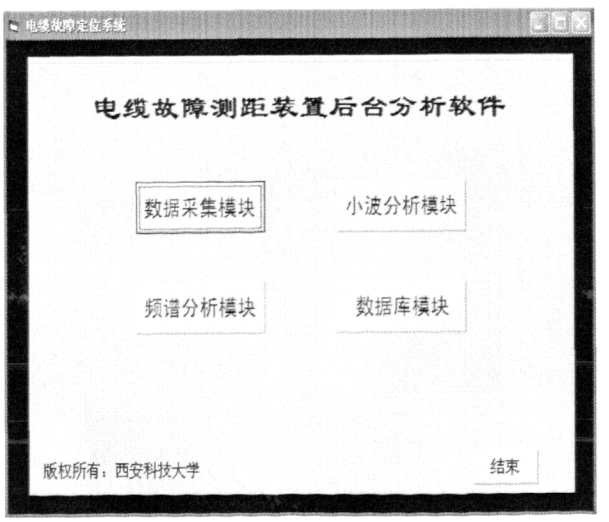


图 4.6 电缆故障测距装置后台分析软件主界面

其中数据采集模块界面设计如图 4.7 所示，其主要功能如下：

(1) 支持在线调整各种通信速率、奇偶校验、通信口参数；

(2) 支持 16 进制、10 进制的数据显示；

(3)能够绘制图形；

(4)能够以.txt 形式保存数据文件；

(5)能够实时接收来自下位机的数据；

(6)能够进行远程数据采集与指令发送。

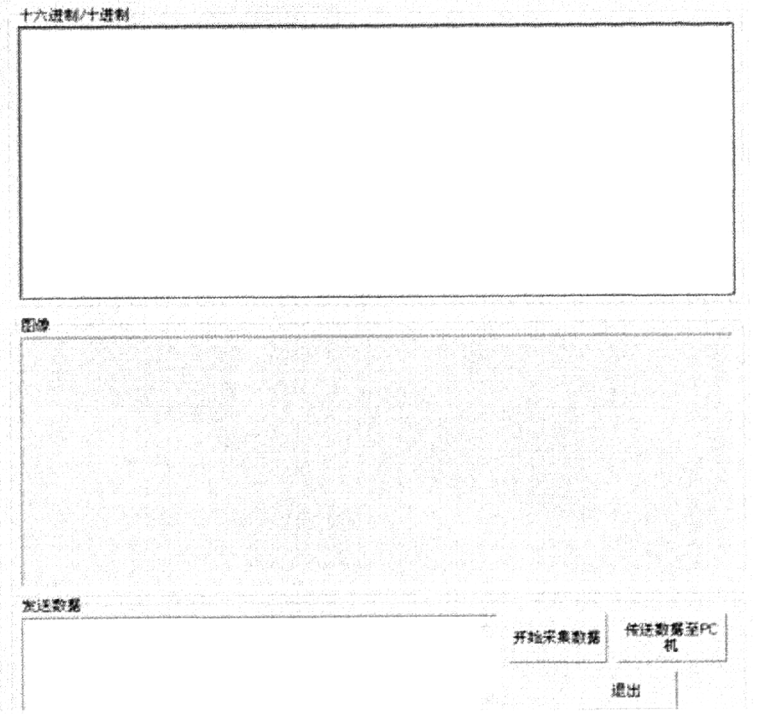


图 4.7 数据显示模块界面

4.7 小结

本章主要完成了 PCI 数据采集卡的驱动开发和上位机软件的设计。使用 WDM 驱动模型开发驱动程序，程序结构清晰，开发周期较短，效率高；并基于系统要求，编写了上位机界面。通过对信号发生器产生信号的采集与分析，验证了该系统各种功能的正确性与准确性。从信号测试的结果来看，该数据采集系统的基本功能已经实现，可直接应用于电缆故障在线测距系统或其他领域的数据采集工作。

5 系统测试

为了验证电缆故障测距系统的硬件和软件工作的正确性和可靠性，并应用于实际现场中，完成了以下测试。系统的测试平台如图 5.1 所示。

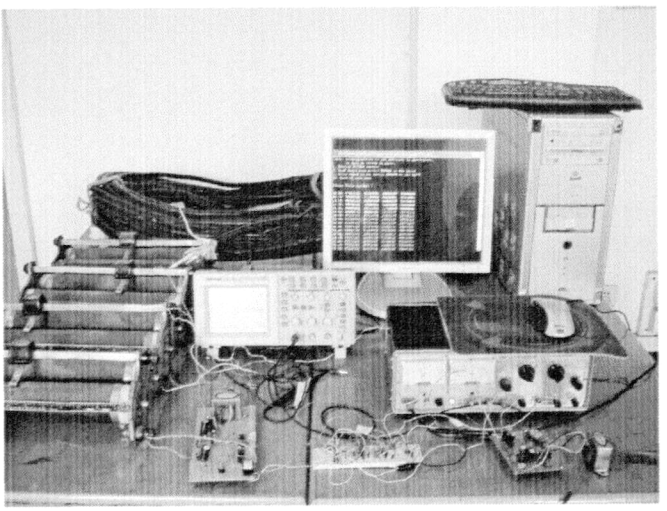


图 5.1 系统测试平台

5.1 信号调理电路测试

实验中，将电缆接入 237V 的交流电压信号中，数字示波器记录信号调理板的输出端波形，测得其幅值为[0.4V，2.16V]，如图 5.2 所示。符合 A/D 输入的范围要求，这说明信号调理电路工作正常。

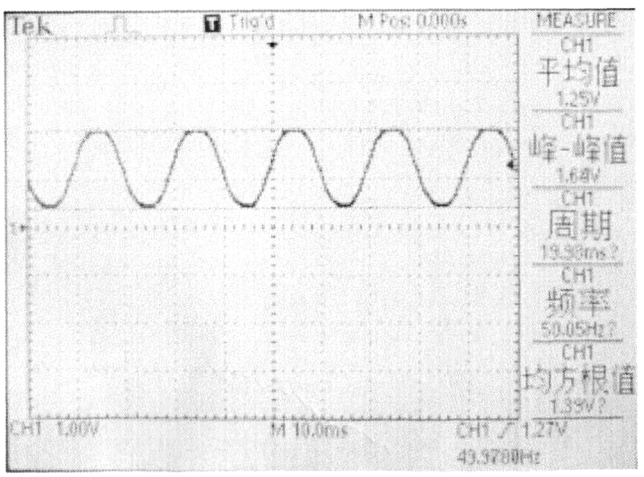


图 5.2 信号调理板电压输出波形

5.2 采集卡采集测试

采集卡采集测试主要有两个内容：电缆正常时 A/D 模块能否采集和电缆发生故障时

能否采集。

由于缺少实际现场试验条件，因此采用的测试方案为：在电缆中间加一个开关和滑阻并联的电路，若开关闭合，电缆正常工作；若开关打开，滑阻接入电缆中间，此处电缆阻值突然增大，电缆发生故障，电压幅值会降低。上位机程序可把采集到的数据通过图形显示，所以通过此功能显示采集的信号，如图 5.3 所示：

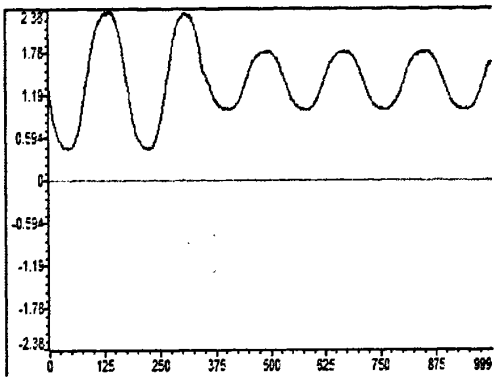


图 5.3 电压采集部分数据波形

从图中可以看出，信号由两部分组成：一是电缆正常工作时采集到的数据，此时电压幅值不变；二是电缆发生故障后采集到的数据，此时电压幅值降低。由此可以说明，A/D 转换模块在电缆正常运行时和发生故障后都能正常工作。

5.3 上下位机通讯测试

系统上下位机通讯测试主要实现两个内容：

(1) 测试数据采集卡 PCI 接口能否将采集到的数据正确传送至上位机；

(2) 测试上位机的数据采集模块能否正确接收来自下位机的数据及绘制图形。

本实验通讯数据以 16 进制显示，上位机数据显示及图形绘制界面如图 5.4 所示。

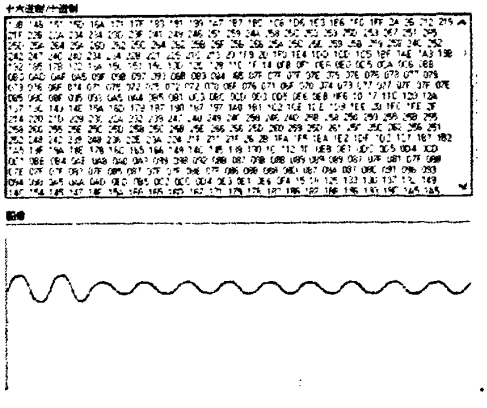


图 5.4 数据显示及图形绘制界面

从图中可以看出，上位机能够实时地接收来自下位机传送的数据，并以图形方式绘制出来。

5.4 上位机后台分析软件测试

5.4.1 小波分析测试

小波分析模块是整个电缆故障定位后台分析软件的核心。该模块主要对上传的电缆故障数据进行小波分析，找出奇异点，从而实现故障测距任务，其软件开发的好坏直接影响到测距的精度。本文通过 VC++6.0 调用在 MATLAB7.0 软件中编写的小波分析程序 wavelet.m，充分利用了 MATLAB7.0 软件强大的数值分析和图形处理功能。

数据采集模块实时接收下位机上传的数据后，进入小波分析，小波分析模块对该信号进行多尺度分析，分析后的结果如图 5.5、5.6 所示：

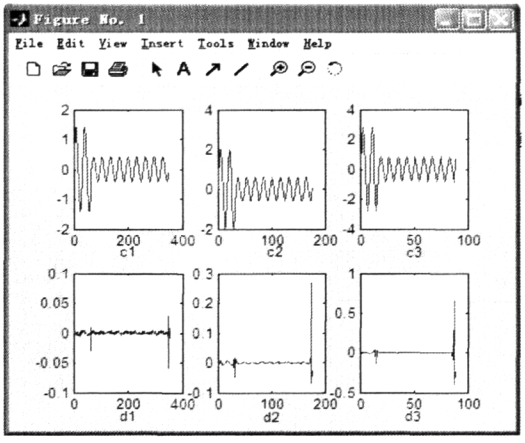


图 5.5 小波分解后的近似系数与细节系数

图 5.6 小波分解后返回的细节系数

信号经 db5 小波 3 层分解后会在不同尺度空间和小波空间上反映出来。其中，近似系数 c1、c2、c3 反映了信号的整体结构，细节系数 d1、d2、d3 反映了信号的奇异点和奇异点的突变程度。

通过测试可以说明：选用的 db5 小波能够很好的对信号进行奇异性分析，这对提高电缆故障定位精度有着重要意义。

5.4.2 频谱分析测试

由于电缆在线故障测距系统在实际工作中将接入电网，难免会受到谐波、噪声等因素的影响。因此，需要对信号进行频谱分析，从而查看各次谐波以及噪声所占基波的比例。由于 MATLAB7.0 软件中提供了一种快速傅里叶变换的函数 fft[48]，对信号进行频谱分析十分方便。

对上传的信号进行频谱分析，分析后的结果如图 5.7、5.8 所示。图 5.8 中 f1 是频率，

mag1 是频率所对应的能量。

图 5.7 频率与能量谱

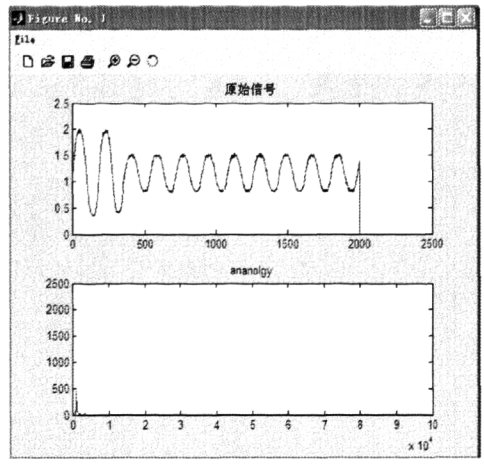


图 5.8 频谱分析图

从图 5.7 可以看出被采样的信号中含有频率成份为 8500Hz 的噪声信号，为我们设计滤波器提供了重要依据。由于上位机具有很强的运算处理能力，可以在很短的时间内完成信号的 FFT 运算，获得每一频率成分的幅度值或归一化值和相位。然后处理结果可以在主机端显示出来。这实际上也就成为了一个数字信号处理平台。

5.5 数据库测试

数据库模块测试主要有两个内容：一是数据读取；二是列表刷新。

数据按照时间顺序进行存储，并且自动保存为.txt 的格式，便于工作人员日后进行查找，如图 5.9 所示。

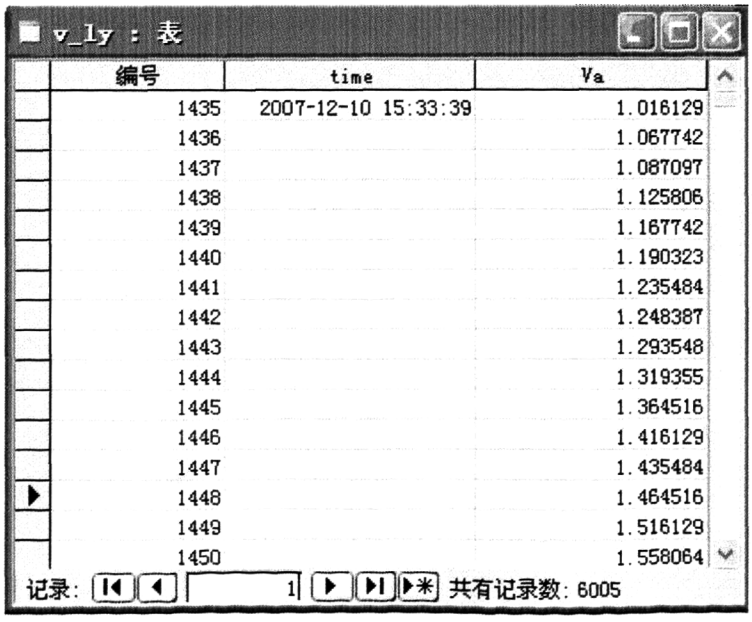


图 5.9 数据库

通过测试可以说明，本文所编写的数据库程序运行正常。

5.6 小结

为了更好的模拟实际现场条件，将电缆接入 237V 交流信号中，并将电缆在线故障测距系统安装在电缆的一端。通过下位机测试，验证了信号调理电路、数据采集卡能够完成 I/V 变换、数据采集、数据显示、数据传输等各项任务；通过将数据上传给 PC 机测试后，验证了电缆在线故障测距系统的后台分析软件能够实现信号的正常通讯、小波分析、频谱分析和数据存储等各项功能。

6 结论

本文设计了基于 PCI 总线的数据采集系统。整个系统的设计用到了多种软、硬件技术，本文所做的工作也是多方面的，归纳起来有以下几个方面：

(1) 深入研究 PCI 总线接口技术，并在系统设计中实现了与计算机的 PCI 总线接口。

(2) 详细阐述了信号调理、模数转换、数据缓冲、PCI 接口和逻辑控制等五个功能模块的设计方案和硬件电路实现方法。

(3) 重点分析了 PCI9054 的数据传输方式，用 VHDL 结合原理图的方法设计了 PCI 数据采集卡的控制逻辑。

(4) 在深入研究驱动程序模型的基础上，用 DriverWorks 开发了采集卡的驱动程序。

(5) 使用 VC++6.0 编写了系统上位机软件，整个软件系统界面友好、可移植性强，并顺利通过了系统测试。

本系统属于应用开发项目，涉及高速数据传输、数据采集、PCI 总线运行机制、计算机操作系统运行的底层机制、驱动程序以及应用软件等方面。系统可以在 Win2000、WinNT 操作系统下使用，但是，由于本课题的研发试验性，整个系统离商品化和实际运用还有一定的距离，为此，还作以下两个方面的考虑：

(1) 本系统采用单通道采样，而 PCI 总线接口的传输带宽远远没有达到它的极限，还有很大的利用空间，这就允许系统进一步增加数据采集通道数和提高采样频率以适应更多的实际应用需求。

(2) 对于运行在 0 级的驱动程序，选择用 DDK 开发工具来开发，会使系统性能有很大的提升。

致 谢

值此论文完成之际，谨向导师汪梅教授致以最诚挚的感谢。本文的研究工作自始至终都得到了汪老师的悉心指导和热心关怀。导师的严谨治学态度、渊博的知识，以及强烈的社会责任感和无私的奉献精神，使我深受启迪，这将使我终身受益。从导师身上，不仅学到了扎实、宽广的专业知识，也学到了做人的道理，更学到了如何去获取知识。“送人一鱼，仅供一饭之需；而授之以渔，则终生受用无穷。”借古人云，送给尊敬的汪老师，向汪老师表示崇高的敬意，并致以衷心的感谢！

课题在设计过程中得到了电控学院郭秀才老师的无私帮助，在此表示深深的谢意。

感谢实验室的各位同学，特别是吴建华、温卫敏、邹慧荣等在课题研究中对我的大力帮助。他们伴我渡过研究生的求学生涯，因为有了和他们在一起的共同学习与相互扶持，三年的研究生涯变得丰富多彩。

感谢我的父母！是他们默默无闻的付出，给我创造了良好的学习条件，他们的理解和支持使我能够专心完成我的论文和学业。

最后，在论文完成之际，谨向所有给予我帮助和关心的老师、亲人、朋友、同学致以我最衷心的谢意！

参考文献

[1] 张群峰.基于电弧特性电力电缆故障在线测距方法[J].高电压技术.2003,30(7):12-14

[2] 李贵山等.PCI 局部总线开发者指南[M].西安:西安电子科技大学出版社,1997

[3] 李贵山,陈金鹏等.局部总线及其应用[M].西安:西安西安电子科技大学出版社,2003.

[4] 尹勇,李宇.PCI 总线设备开发宝典[M].北京:北京航空航天大学出版社,2005

[5] 尚利,安迪生.PCI 系统结构[M].北京:电子工业出版社,2002

[6] 许兴存,曾琪琳著.微型计算机接口技术[M].北京:电子工业出版社,2003

[7] 戴紫彬,孙万忠,张永福.PCI9054 局部总线设计与应用[J].微电子学与计算机,2003:15-20

[8] 索拉里,威利斯.PCI&PCI-X 硬件和软件结构[M].北京:电子工业出版社,2003

[9] 全子一,周利清,门爱东.数字信号处理基础[M].北京:北京邮电大学出版社.2002:12-13

[10] PCI9054 Data Book V2.1[M].PLX Technology.Inc.2000

[11] PCI9054 RDK-LITE Hardware Reference Manual[M].PLX Technology .Inc. 2001

[12] 瞿世尊,陈健.PCI9054 在 PCI 总线高速数据采集系统中的应用[J].电子技术,2004:59-61.

[13] 王树志,周希德.基于 PCI9054 总线控制器的数据接收和存储系统[J].电子设计应用,2002(11):16-17

[14] 陈露晨.PCI9054 性能分析及外部 FIFO 的扩充[J].电子技术应用,2001(4):68-69

[15] PLX Rdk Hardware Development Kit User' s Manual.PLX Technology.Inc. 2000

[16] 熊松.基于 PCI 总线的数据采集卡的实现[D].厦门：东南大学,2006

[17] 齐春东,王华,匡镜明.基于 PCI 总线高速数据采集卡传输模块设计[J].电子测量技术,2004:61- 62

[18] 潘海涛,汪梅,李雯雯.基于 PCI 总线的高速数据采集卡的设计[J].仪器仪表学报,2007(4):232-234.

[19] 方粮,尹桂斌,黄克勋.PCI 总线卡设计与实现的几个关键问题[J].计算机工程与科学,2001:15-18.

[20] 蔡建隆,苏涛.用同步 FIFO 实现 PCI 机箱板卡间的高速数据传输[J].电子设计应用,2003(12):32-34

[21] 赵世霞.VHDL 与微机接口设计[M].北京:清华大学出版社,2004

[22] 王钿,卓兴旺.基于 VerilogHDL 的数字系统应用设计[M].北京:国防工业出版社,2006,2-3

[23] 黄正谨.徐坚系统设计技术入门与应用[M].北京:电子工业出版社,2002,25-28
[24] 张正茂,左维.PCI 总线数据采集卡设计[J].光电技术应用,2004(4),38-42
[25] 王力,张伟.Protel99 SE 典型实例实例[M].北京:人民邮电出版社,2006：209-212
[26] 尤晋元等编著.Windows 操作系统原理[M].北京:机械工业出版社,2001
[27] 陈章进,冉峰,汤立华等.PCI 目标设备的配置空间实现[J].计算机应用,2002,22(1):45-48
[28] 丁祥,余小清.PCI 语音片及 WDM 驱动程序设计[J].电子技术应用,2003,1:5-8
[29] 乔林，杨志刚，刘文杰.Visual C++程序设计提高篇[M].中国铁道出版社,2004
[30] 郑阿奇.Visual C++实用教程[M].第 2 版.北京:电子工业出版社,2003
[31] 严华峰.Visual C++课程设计案例精编[M].北京:中国水利水电出版社,2002:50-60
[32] PLX Sdk Software Development Kit User' s Manual[M].PLX Technology.Inc.2000
[33] 李海.PCI 设备 Windows 通用驱动程序的设计[J].电子技术应用 2000(1):19-22
[34] 王俊锋,冯志彪,陈耀.如何编写 WDM 设备驱动程序[J].微机发展 2005(2):25-28
[35] Chris Cant.Windows WDM 设备驱动程序开发指南[M].北京:机械工业出版社,2000
[36] 武安河等著.Windows2000/XP WDM 设备驱动程序开发[M].北京:电子工业出版社,2004
[37] 郑伟绩.用 DriverStudio 开发 PCI 设备驱动程序[J].微型机与应用 2002,(9):13-14
[38] 孙守阁,徐勇.Windows 设备驱动程序技术内幕[M].北京:清华大学出版社.2003
[39] 杨浩.基于 DSP 和 PCI 总线技术的数据采集卡的改进[D].成都:电子科技大学，2003
[40] PCI Bus Master I/0 Accelerator Chip Highlights[M].PLX Technology.Inc
[41] Microsoft Corp, Microsoft Developer Network6.0. Microsoft Press. 1999，34-36
[42] 王浩.基于 PCI 总线的数据采集卡的研制及其应用[D].石家庄:华北电力大学,2007
[43] 施诺等译.Windows2000 设备驱动程序设计指南[M].北京:机械工业出版社,2001
[44] 尤晋儿编著.Windows 操作系统原理[M].北京:机械工业出版社,2001
[45] PLX Mon Lser' s Manual[M].PLX Technology.Inc.2000
[46] Walter Oney. Programming Windows Driver Model[M].Microsoft Press,2000:44-46
[47] Jeffrey Richt 著，Programming Applications for Microsoft Windows Fourth Edition [M].北京:机械工业出版社,2004
[48] 曾尚璀,沈华.基于 MATLAB 系统的信号 FFT 频谱分析与显示[J].科技通报,2000,16(4):241-246

附 录

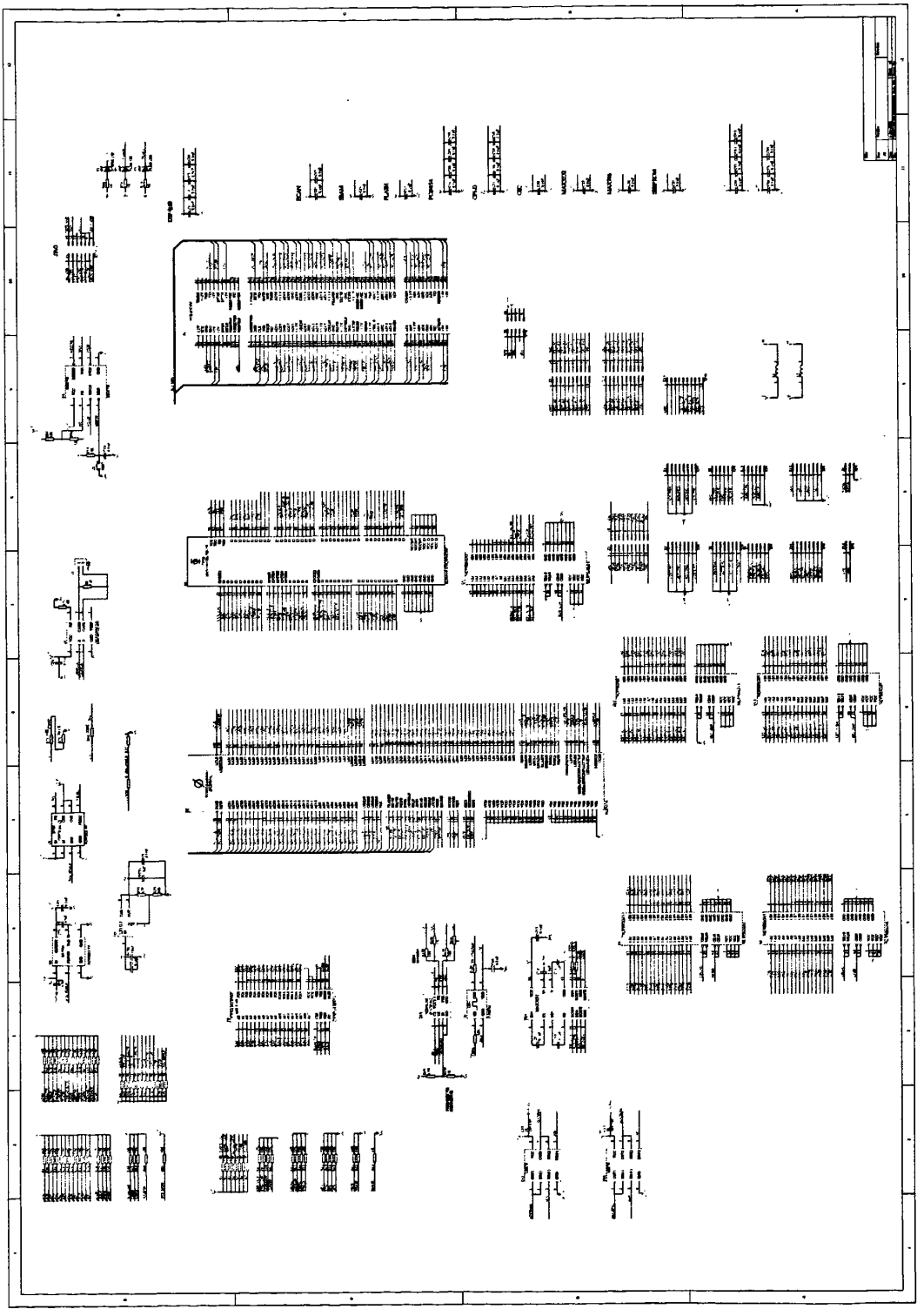
攻读硕士学位期间发表论文

[1] 潘海涛,汪梅.基于 PCI 总线的高速数据采集卡的设计,仪器仪表学报,2007 年 4 月

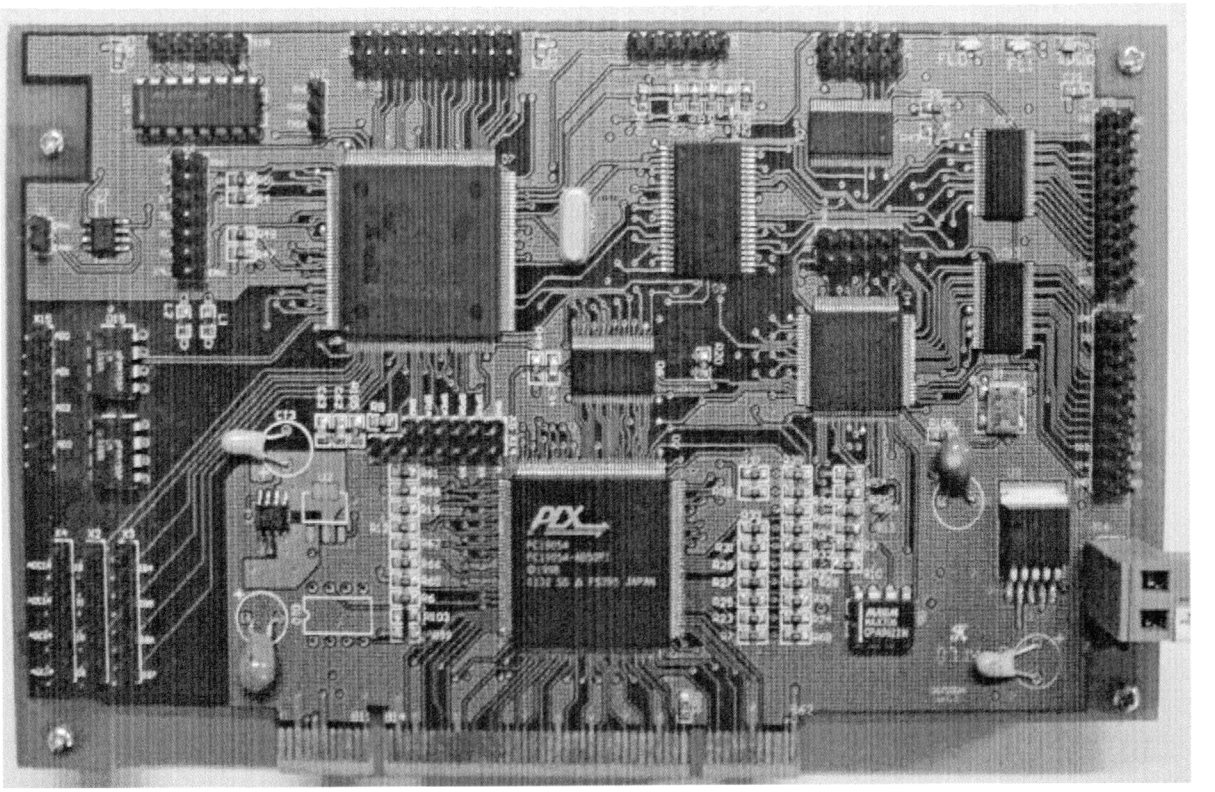
参与的科研项目

[1] 陕西省科技攻关项目(2003K06G19)，智能电缆故障预测与定位装置的研制.

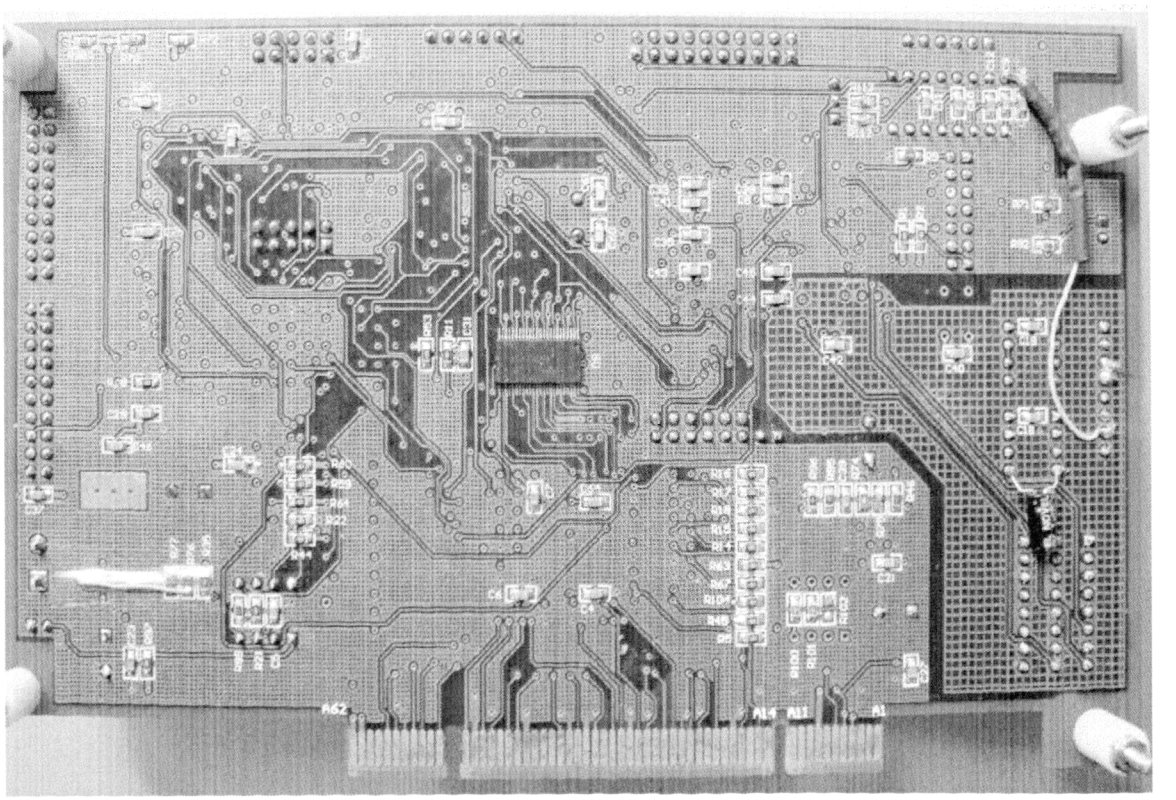
[2] 陕西省教育厅自然科学基金项目(05JK259)，便携式室内环保监控仪的研制.



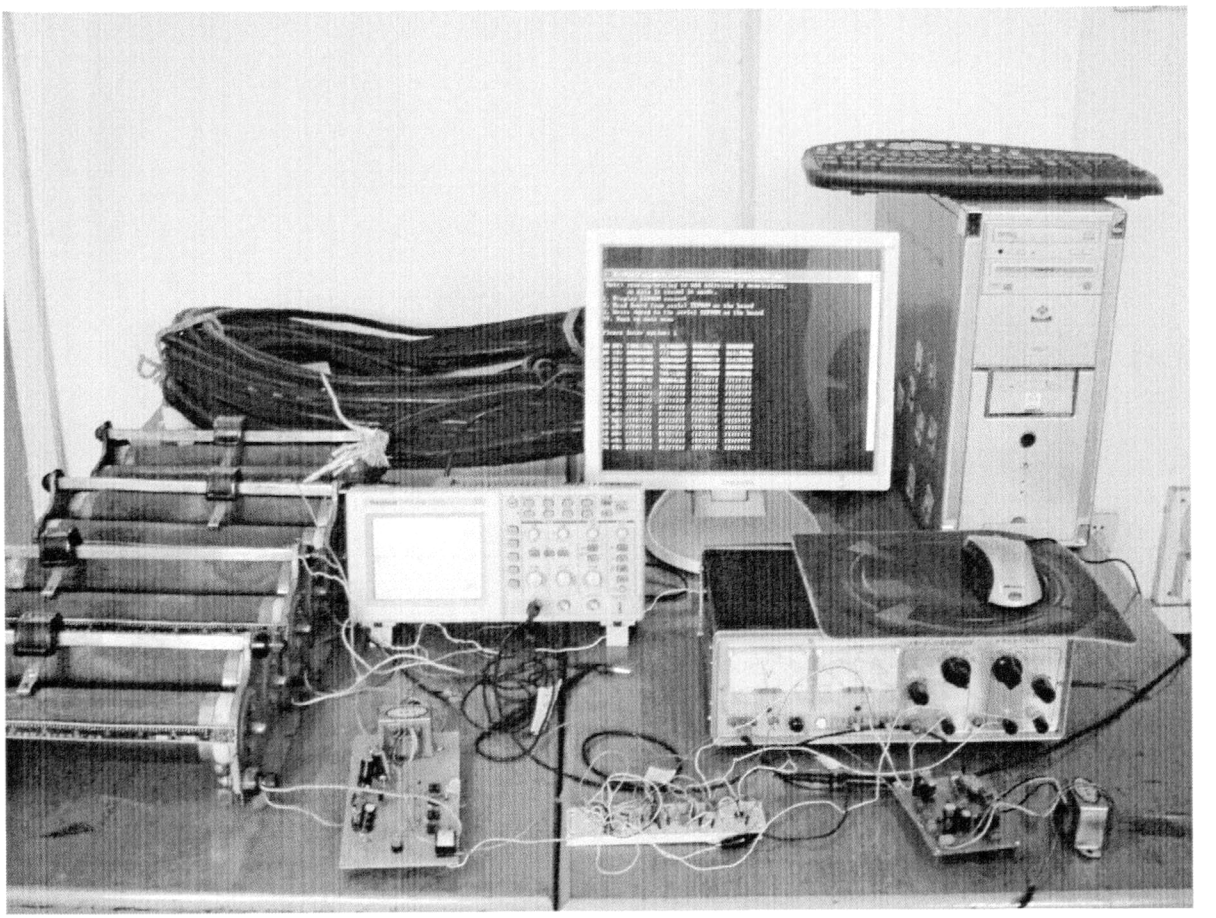
附图 1 数据采集卡原理图



附图 2 数据采集卡正面



附图 3 数据采集卡背面



附图 4 系统测试平台

基于PCI总线的电缆故障数据采集系统的研究

作者：潘海涛
学位授予单位：西安科技大学