基于 Win API 串行通信技术的应用研究

尹天明,李也白,张球河,王宇鸽
(北方工业大学 信息工程学院,北京 100041)

摘 要:讨论了基于 Windows API 的串行数据通信,它可以灵活地定制各种串行通信,利用操作系统提供的 API 实现计算机与外部设备的即时通信。该技术已经在出铝控制系统中得到了应用,它能安全有效地实现数据的串行通信。

关键词:Windows API;串行数据;通信
中图分类号:TN919 **文献标识码**:A **文章编号**:1005-3751(2006)04-0193-03

Research and Application of Win API-Based Serial Communication Technology

YIN Tian-ming, LI Ye-bai, ZHANG Qiu-he, WANG Yu-ge
(Department of Computer Engineering, North China University of Technology, Beijing 100041, China)

Abstract: Presents the design and application of Windows API-based serial communication that makes many kinds of serial communication and implements communication between computer and peripheral equipment via API. This technology has been used in the aluminum-control system and implements safely and effectively serial communication.

Key words: Windows API; serial data; communications

0 引 言

在计算机通信方式日益多样化的今天,串口数据通信在企业级应用中仍然占有非常重要的地位。实现计算机串口通信的方式是多样的,而利用 Windows API 实现数据的串口通信则可以根据实际应用来灵活的定制通信方式,文中将结合一个实际的工程来讨论其设计和实现的具体策略,系统环境为 Win2000 professional,使用语言为 Delphi 7.0。

1 串口通信及 Windows API

目前串行通信端口(RS-232)[1]是计算机上的标准配置,在计算机上一般将 COM 口以 9 引脚的接头接出,且计算机上的 RS-232 均是公头,用途上主要是通过调制解调器以及其它外部设备进行通信数据传输。由于其具有广泛的实用性,因此在工业领域也得到了大量的应用。目前,实现串口通信的方式主要有两种:一种是通过各个集成开发环境所提供的串行通信控件,实现起来较为简捷,但是灵活性不够,不易扩展;另一种是用利用操作系统所提供的应用编程接口(API)来实现,其实现起来相对较为复杂,但是十分灵活,可以定制开发出各种串行通信软件。在操作系统所提供的 API(应用程序接口)中有 20 个左右与串行通信有关,利用这些 API 可以灵活操作串口资源。使用串行端口的相关数据前,必须先打开通信端口,在设置好相应的通信参数后即可以开始通信,使用完毕后要及时关闭通信端口,及时释放串口资源。串口打开的流程如图 1 所示。

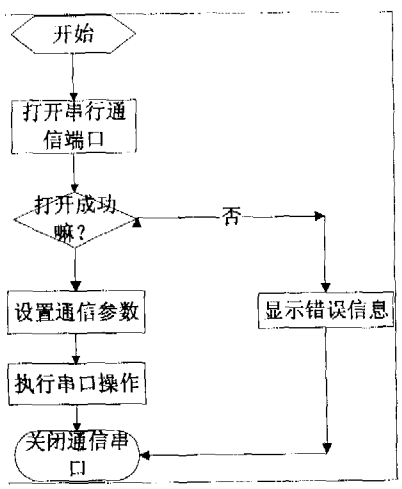

图 1 串口通信流程

2 系统结构设计

将该串口通信系统应用到了一个大型铝业集团的自动出铝控制系统中,该系统由上下位机共同组成,上下位机之间的通信过程如下:

1)上位机将参数分别发送到对应的串口;

收稿日期:2005-07-14
作者简介:尹天明(1975-),男,山东青岛人,硕士研究生,研究方向为网络通信、工作流技术;李也白,教授,研究方向为数据库、电子政务。

2)串口再通过数传电台将数据发送到每个下位机;

3)下位机在处理完生产过程后将获得的数据再通过串口和数传电台传回给上位机,就完成了一次处理过程。

从以上过程可以看出,通过上下位机的通信实现的系统的远程控制,数传电台是可以按照预先设定好的参数自主地处理串口发送来的数据,所以系统实现正常工作的核心就集中到对串口通信的操作。系统结构如图 2 所示。

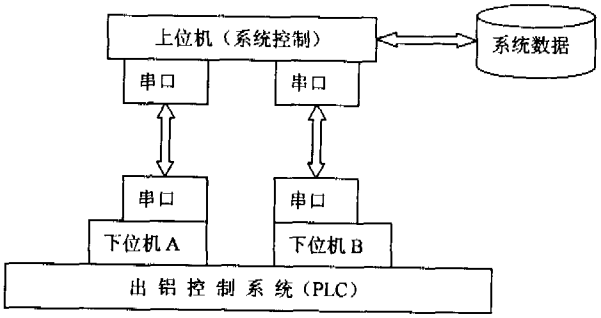

图 2 出铝系统结构

3 串口数据通信实现的关键技术

3.1 通信参数设置

首先要检查一下串口资源的设置[2],例如查看到 COM1 时,其所使用的资源如下:

地址:03F8~03FF

中断:04

那么通过这两个资源,用户就可以控制串行端口,要检查通信端口的冲突设备列表,如果有冲突的话,必须修改其地址和中断资源,否则串口不能正常工作。

串口通信必须要求发送和接收方必须要遵循相同的协议,这其中主要包括:波特率、奇偶校验、数据位数、停止位数等,实现过程如下[3]:

1)定义串口句柄变量,以后可以利用此句柄来进行对串口的操作。

2)以文件的方式打开串口并赋值给串口句柄,主要代码如下:

```
createfile ( hcomm, GENERIC_ READ and GENERIC_ WRITE,0,nil,OPEN_EXISTING,0,0)
```

其中:

hcomm:串口句柄变量

GENERIC_READ and GENERIC_WRITE:以读写方式打开(也可以只定义其中一个)

第三个参数表示打开串口的共享模式,可以以共享读或者写方式打开,在本系统中设置为 0。

OPEN_EXISTING:表示打开一个已经存在的文件,如果不存在,则返回出错。

其余的参数都设置为 0。如果打开出错要进行相应的错误处理。

3)在成功地打开串口后首先要获得目前系统的 COM 状态,设备的状态保存在一个 DCB 结构中,该结构体内定义了对串口通信设备的控制信息,其主要内容有:

```
typedef struct DCB {
  DWORD DCBlength; // DCB 结构的长度
  DWORD BaudRate; // 波特率
  DWORD fParity: 1; // 奇偶校验
  BYTE ByteSize; // 数据位的个数(4-8)
  BYTE Parity; // 没有(0), 奇校验(1),偶校验(2)
  BYTE StopBits; // 停止位的个数 0,1,2=1, 1.5, 2
} DCB
```

在串口数据通信之前必须按照通信双方的协议来设置通信参数,关键代码如下:

```
cc:TCOMMCONFIG; //定义通信变量
GetCommState(hcomm,cc.dcb); //获得 COM 的状态
cc.dcb. BaudRate:=CBR_9600; //设置通信参数
cc.dcb. ByteSize:=8;
cc.dcb. StopBits:=ONESTOPBIT;
```

最后利用 SetCommState(hcomm,cc.dcb)函数将参数写入 DCB 的结构体内。

在打开串口、设置 DCB 结构参数等的过程中要对函数的返回值及时加以判断,根据返回值来进行判断,如果函数的处理结果错误要转入相应的错误处理。

设置串口通信参数是非常重要的一步,如果双方的通信参数不一致就会造成接收到乱码数据或者接收不到数据,从而造成系统瘫痪。

3.2 串口的数据接收

在串口通信的过程中,数据的正确接收是重要的环节,通信的双方要根据系统的实际情况制定出统一的通信格式,包括通信字符的起始字符、结束字符、握手信号等,在本系统里定义的结束字符为'='。

在通信双方定义好以上数据格式后,要依据所定义的格式进行数据的接收工作,文中采用了循环的方式来读取由下位机传来的数据[4,5],主要过程是:

1)首先用 clearcommerror()函数清除串口状态,并得到可以读取的字节数;

2)依据获取的字节数来读串口数据,并将其放入事先定义好的存储区内,在 Delphi 里还要将其转换为字符串,以便以后可以利用字符串处理函数来解析接收到的字符串;

3)依据定义的通信格式来判断读取的字符串是否是所需的完整的字符串,不是则继续读取串口,是则完成一次数据的接收工作。

关键代码如下:

```
repeat
  sleep(100);
  clearcommerror(hcomm,dwerror,@cs);
    if cs.cbInQue=0 then
      exit;
  readfile(hcomm,inbuff,cs.cbInQue,comthread.nbyteread,nil);
  comthread.tempArray:=copy(inbuff,1,cs.cbInQue);//转换为字符串以便处理
until (ansiendstext('=',comthread.tempArray));//如果不是结
```

束字符则继续读

其中clearcommerror()函数将返回接收缓冲区存放的实际的字符数，它是当前缓冲区内的字符的一个快照，并不一定是所需要接收的实际的完整的字符串，所以必须要根据接收的字符串的起始和结束字符来判断，如果字符串不完整，则需要继续读取，并根据情况来将传输中分离的字符串进行组合。还有一点要注意，由于串口是硬件设备，它的数据接收和发送速度相比之下较慢，所以在循环读取和写串口时要用sleep()函数来进行一下延迟，再进行串口的操作，避免造成数据丢失。

3.3 串口的监听

串口监听是串行系统实现通信的关键环节，其实现的方法主要有三种：一种是使用定时器控件，修改其时间间隔属性，使其在设定好的时间间隔内循环执行放在定时器中的代码，从而实现对串口的监听；一种是建立通信事件，设置哪一个串行通信事件被检测，一旦该事件发生则开始相应的执行程序；另一种是建立一个多线程机制，通过主进程来创建一个串口监听的线程来实现实时的数据通信。这里采用了多线程机制，在Delphi里产生一个线程的方式是建立线程对象，在线程单元内有一个Excute子程序区，执行的代码都在其中，在编写代码的时候要在每次执行的时候检查线程的Terminate属性，当为false时执行线程代码。因为线程的建立和执行是由应用程序所产生，往往会使用到一些公共的变量，所以必须要采用同步化的机制，主要的同步方式有：

* Critical Section：指临界区，该方法只能用于单一的应用程序内，在临界区内的程序代码不能被两个线程同时访问。

* Mutex：互斥体，可以用来处理全局变量，后进入的线程必须等到前一个释放后才能执行。

* Semaphore：信号灯，它拥有计数功能，可以指定同时操作公共变量的人的个数。

其关键代码如下：

```
Procedure TReadThread.Execute;
var
  {变量定义}
Begin
While not Terminated do
  Begin
  Synchronize (ReadPort);
    {线程执行代码区}
  end;
end;
```

其中将读串口的过程定义成一个函数，然后利用Synchronize()函数实现对公有变量的访问同步。

4 结束语

本系统充分利用了操作系统所提供的强大的Win API，并且结合实际的工程项目，实现了串口数据的有效通信，同时提高了数据通信设计的灵活性，开发人员可以依据实际情况来实现系统的多方面要求，增强了系统的使用效率。本串口通信技术已经在大型铝厂的出铝过程自动控制系统中得到了很好的应用，也为该系统在今后的产业化过程中打下了良好的基础。

参考文献：

[1] 李也白，全　臻，刘伟立. 基于RS-485的树状异步串行通信系统的研究[J]. 计算机应用研究，1999，16(4)：21-22.

[2] 范逸之，陈立元. Delphi与RS-232串行通信控制[M]. 北京：清华大学出版社，2002.

[3] 崔建华，郭瑞金. Delphi串口通信工程开发实例导航[M]. 北京：人民邮电出版社，2003.

[4] 龚建伟，熊光明. Visual C++/Turbo C串口通信编程实践[M]. 北京：电子工业出版社，2004.

[5] 李现勇. Visual C++串口通信技术与工程实践[M]. 北京：人民邮电出版社，2002.

(上接第192页)

参考文献：

[1] 徐少平，孙　骏，徐少文，等. Web服务核心协议及其实现[J]. 计算机与现代化，2004(5)：32-35.

[2] 胡智文，陈国龙. 新一代分布式计算模型——XML Web Services[J]. 计算机工程，2004，30(18)：1-3.

[3] 柴晓路，梁宇奇. Web Services技术、架构和应用[M]. 北京：电子工业大学出版社，2003.

[4] Holzner S. XML完全探索[M]，陶　阳等译. 北京：中国青年出版社，2001.

[5] Ogbuji U. Simple Object AccessProtocol[EB/OL]. http://www.w3c.org/TR/SOAP，2001-07-09.

[6] Word Wide Web Consortium. Web Services Description Language (WSDL)[EB/OL]. http://www.w3c.org/TR/wsdl，2000-09-25.

[7] Brittenham P. Universal Description, Discovery, and Integration (UDDI)[EB/OL]. http://www.uddi.org.，2001-09.

[8] A brief introduction to the web services protocol stack: Soap, wsdl and uddi, HP Web Services Platform[EB/OL]. http://www.hp.com/go/webservices，2001-08-25.

[9] Dale J. Advanced Web Services[J]. Open Agent Systems，2002(3)：11-20.

[10] Leymann F. Web Services Flow Language(WSFL 1.0)[EB/OL]. http://www-4.ibm.com/software/ solutions/webservices/pdf/WSFL.pdf.，2001-06.

[11] W3C的Web Services架构标准[EB/OL]. http://www.w3.org/TR/2003/WD-ws-arch-，2003-08-08.

[12] Gottschalk K，Graham S，Kreger H，et al. Introduction to Web Services Architecture[J]. IBM Systems Journal，2002，41(2)：10-15.