第32卷 第20期 计 算 机 工 程 2006年10月
Vol.32 № 20 Computer Engineering October 2006
·工程应用技术与实现· 文章编号：1000—3428(2006)20—0253—02 文献标识码：A 中图分类号：TP39

基于 Delphi 平台的高效率实时数据采集技术

余 臻，柳培忠
(厦门大学信息科学与技术学院，厦门 362000)

摘 要：在计算机智能控制领域，实时数据采集是一个非常重要的环节。该文以电力系统的数据采集为背景，说明了在 Windows 2000 环境下通过计算机串口实现数据采集的方法。并且介绍了在 Delphi 中多媒体定时器的使用，从而大大提高数据采集的速度和精度，达到了智能控制的要求。

关键词：智能控制；采集；串口；多媒体定时器

Efficient Real-time Data Acquisition Technology Based on Delphi Environment

YU Zhen, LIU Peizhong
(Shool of Information Science and Technology, Xiamen University, Xiamen 362000)

【Abstract】 Real time data acquisition is very important on computer intelligent control. Based on data acquisition of the electric power system, this paper introduces the way of computer serial port data acquisition on Windows environment and the use of the multimedia timer on Delphi. Thus, the speed and the precision will be greatly improved.

【Key words】 Intelligent control; Acquisition; Serial port; Multimedia timer

在信息科学中，数据采集技术已经成为其重要的一个研究问题，它已经与计算机技术、网络技术、传感器技术、信号处理技术共同构成了现代检测技术的基础，随着科学技术的发展和数据采集系统的广泛应用，人们对数据采集的主要技术指标，如采样速率、分辨率、精度、输入电压范围、控制方式以及抗干扰能力等方面，都提出了越来越高的要求。在智能控制领域，实时数据采集是一个最基本且最重要的环节，它关系到整个系统的效能。尤其是在电力系统的实时监控系统中，一旦实时数据采集得到了深入的广泛的应用，将对电力系统的监控起到很大的作用。因此，如何提高采集速度和精度一直是电力系统软件开发人员要解决的难题。

随着 Windows 操作系统的普及应用，尤其是著名 Borland 公司的面向对象开发软件 Delphi 7.0 的出现，为软件开发提供了强大的图形界面功能和强大通信控件，使得对数据采集进行图形化处理更为方便，开发出来的应用程序具有良好的人机交互功能。这里介绍了在 Windows 环境下进行实时高速数据采集的方法。

1 工程应用

随着电力系统自动化程度的不断深入，对电力传输介质的电压、电流、功率、电能、功率因数、频率等参数的采集显得尤为重要。本文以近期完成的福建省厦门演武大桥电力监控项目为例，介绍电力传输介质中的电压、电流数据进行高速实时采集的方法。该项目主要采集设备是河源市雅达电子有限公司开发的 YD2100 智能电力测控仪，它集合了电量变送器、数字式电度表、数显表、数据采集器、记录分析仪、RTU 等仪器的部分或全部功能，每 10ms 对数据进行一次采集。

2 实现方法

针对这一工程应用，重点讨论了在编程过程中的串口操作和定时操作两个方面的问题，它们是实现本工程的关键。

2.1 串口的读写

可以通过以下两种方法实现对串口的读写：

(1)使用 MSComm 串行通信控件。串行通信控件 MSComm32.ocx 提供了使用 RS-232 接口来进行数据通信的所有协议，为该控件提供了标准的事件处理函数过程，并通过属性和方法提供了串行通信的设置。它使用户能够方便地访问 Windows 串行通信驱动程序的大多数特性。包括输入、输出缓冲区的大小及决定何时使用流控制命令挂起数据传输等。

(2)使用 API 函数。API 是附带在 Windows 内部的一个极其重要的组成部分，Windows 的 32 位 API 主要是一系列很复杂的函数和消息集合，它可以看作是 Windows 系统为在其下运行的各种开发系统提供的开放式通用增强功能接口。

2.2 定时器的操作

在 Windows 环境下可以采用 2 种方法进行定时：

(1)采用 Windows 提供的系统计时器。使用这种方法非常安全和简单，可以通过调用 SetTimer 和 KillTimer 函数为应用程序分配一个计时器。当指定了一个时间间隔以后，Windows 系统将每隔指定的时间向应用发送一条 WM_TIMER 消息，从而使应用程序能够实现定时处理。然而，基于 Windows 计时器的硬件计时器每隔 54.915ms 走一次，也就是说，这种方式的定时器只能精确到大约 55ms(本工程要求 10ms 采集一次)，对于 55ms 以下的时间精度便无能为力；另一方面，由于 Windows 是基于消息机制的系统，

作者简介：余 臻(1965－)，男，硕士、副教授，主研方向：计算机智能控制；柳培忠，硕士生

收稿日期：2005-12-14 **E-mail**：liupeizhong0921@163.com

任何事件的执行都是通过发送和接收消息来完成的，这样一旦计算机的 CPU 被某个进程占用，或系统资源紧张时，发送到消息队列中的消息就暂时被挂起，得不到实时处理。

(2)利用 Windows 提供的多媒体定时器。多媒体定时器使用自己单独的线程，来调用一个自己的回调函数。它的优先级高，每隔一定时间就发送一个消息而不管其它消息是否执行完，它的最小时间精度通常可以达到 1ms，一般可以满足实时数据采集的定时精度需求。因此，多媒体定时器是一种极好的选择。

3 在 Delphi 平台的编程实现

3.1 串口的读写

```
        //建立串口句柄
    CommHandle:=CreateFile('COM1',GENERIC_WRITE or
GENERIC_READ,0,nil,OPEN_EXISTING,FILE_FLAG_
OVERLAPPED or FILE_ATTRIBUTE_NORMAL,0);
  if CommHandle=INVALID_HANDLE_VALUE
  then  begin
  MessageBox(0,'串口打开失败!','Notice',MB_OK);
      StatusBar1.SimpleText := '串口打开失败';
      Exit;
    end;
   StatusBar1.SimpleText:='已同 COM1 连接！';
  //设置超时
  SetCommTimeouts(CommHandle,CommTimeOut);
  //设置读写缓存
  SetupComm(CommHandle,ReadBuffer,WriteBuffer);
  //对串口进行配置
  GetCommState(CommHandle,DCB);
  DCB.BaudRate := 19200;
  DCB.ByteSize := 8;
  DCB.Parity := NOPARITY;
  DCB.StopBits := ONESTOPBIT;
  Connected:=SetCommState(CommHandle, DCB);
   //串口的读事件
   if (not SetCommMask(CommHandle,EV_RXCHAR))
  then begin
  MessageBox(0,'SetCommMask Error !','Notice',MB_OK);
      Exit;
    end;
    if (Connected) then
    begin
        CloseHandle(CommHandle);
     StatusBar1.SimpleText := '设置串口成功 19200、8、N、1'
    end;
    else begin
      CloseHandle(CommHandle);
      StatusBar1.SimpleText := '设置串口失败';
    end;
  end;
```

3.2 多媒体定时器的应用

(1)确定最大和最小周期。使用多媒体定时器需要确定它提供时间周期的最大值和最小值。可用 timeGetDevCaps 函数获取这两个值，它的声明如下：

function timerGetDevCaps(lpTimeCaps ： PtimeCaps; uSize:UINT）:MMRESULT;stdcall;

参数 lpTimeCaps 为记录体，它包含两个成员：wPeriodMin 和 wPeriodMax，UINT 类型。wPeriodMin 为定时器可设置的最小值，wPeriodMax 为定时器可设置的最大值，单位为 ms。

函数成功执行则返回 TIMERR_NOERROR 值，否则为 TIMERR_STRUCT。

(2)建立最小时间精度。启动定时器前，必须建立想要使用的最小定时器精度，而在定时器服务事件结束后，必须清除该精度。可用 timerBeginPeriod 和 timeEndPeriod 两个函数来实现它的设置和清除。

```
imerBeginPeriod 函数声明如下：
Function timerBeginPeriod(uPeriod:UINT):
MMRESULT;stdcall;
timerEndPeriod 函数声明如下：
Function timerEndPeriod(uPeriod:UINT):
MMRESULT;stdcall;
```

其中参数 uPeriod 用于设置定时器的精度，一般是 10ms。

(3)启动定时器。采用回调函数的方法，需要编写一个回调函数并指定给该定时器。该定时器会定时地呼叫指定的回调函数，执行设定的代码。TimerSetEvent 函数就是用于设置回调函数，并启动定时器的，声明如下：

```
function timeSetEvent(
uDelay,              //每次之间的间隔，单位 ms
uresolution:UINT;      //定时器的精度
lpFunction:TFNTimeCallBack;//指定的回调函数
dwUser:DWORD;          //用户定义的信息
uFlags:UINT
):MMRESULT;stdcall
```

参数 TFNTimeCallBack 用于指定一个回调函数，声明如下：

```
type TFNTimeCallBack = procedure(
uTimerID,          //返回对应定时器的 ID
uMessage:UINT; //保留
dwUser,            //返回用于定义的信息
dw1,               //保留
dw2:DWORD   //保留
)stdcall;
```

(4)关闭定时器。关闭定时器可以调用函数 timerKillEvent，声明如下：

```
function timeKillEvent(uTimerID:UINT):
MMRESULT;stdcall;
```

其中 uTimerID 是定时器的标志 ID。

4 总结

在该工程的软件开发过程中，对多媒体定时器的使用是重要的步骤之一，它关系到实时的效果，在使用中总结了以下几点：(1)系统计时器回调函数虽然不是中断处理程序，但由于它直接被中断处理程序调用，也将它看作中断代码。在回调函数中应包括尽量少的代码，以使得频繁回调的该函数不至于占用太多的 CPU 时间。(2)多媒体定时器的设置分辨率不能超出系统许可范围。(3)在使用完定时器以后，一定要及时删除定时器及其分辨率，否则系统会越来越慢。(4)多媒体定时器在启动时，将自动开辟一个独立的线程。在定时器线程结束之前，注意一定不能再次启动该定时器，否则将迅速造成死机。

(下转第 275 页)