机场导航设备计算机实时监控实现方法

刘 涛 合肥工业大学计算机与信息系 (230009)
钱 军 聂广庆 安徽省民航局通信站导航队 (230051)

Abstract

The document introduces an implementment method of supervise and control to navigation device, which bases on realtime data collection, and dissertates how to direct manipulate COM to implement serial communication in the environment of Visual C++6.0 when PC connects SCM with cables.

Keywords: data collection, serial communication, UTAR, API Function, MSDN

摘 要

本文介绍了一种基于实时数据采集的导航设备监控系统的方法,并论述了基于直接电缆连接方式下在 Visual C++6.0 环境中直接操作串行口进行串行通信的具体实现。

关键词:数据采集,串行通信,UTAR,API 函数,MSDN

导航设备对飞行安全至关重要,需要实时监控,但由于导航设备点多面广且多属于无人值守台站,平时需要维护人员天天到现场巡视,这浪费了大量的人力与物力,安徽民航局通信站为提高设备管理的实时性、高效性、科学性,成立了集中监控科研开发小组,笔者结合安徽骆岗机场通信站的科研开发项目,提出一种实现导航设备实时监控系统的实现方法。

1 系统要求与构成

1.1 系统硬件

导航设备包括双向盲降、信标机、长波机、全向信标、测距仪等,根据不同的工作需求分布在机场的不同位置。本系统要求在中心机房内工作人员可以对各个站点的设备工作状态、机房环境实时监视并且可以进行开关机控制,所以要求系统是一个双工通信系统。导航设备自带的监控器虽然可以对设备进行一定功能的监控,但监控过程无法通过 PC 机来实现,为了实现对导航设备的计算机实时监控,系统采用 PC 机与 8051 系列单片机通过 RS232 接口直接电缆连接的方式来实现计算机对设备监控器送来的设备信号进行监视;通过 PC 机控制设备监控器来达到对各个导航设备进行远程控制操作的目的。设备监控器的采集信号经 A/D 转换后送至 8051 系列单片机预处理,经电缆送至主机 COM 端口,主机对 COM 端口直接进行操作来获取、发送信息,从而实现对导航设备的实时监控。为进一步完善本系统的实时监视的要求,获取更多的信息量,可以直接利用 Modem 进行远程数据传送,这就可以实现主机与远端的设备直接通信,而不是通过设备监控器这一中间环节,为此,有必要在各个设备工作现场采集设备工作信息,对于导航设备我们采集设备发射机工作状态、设备告警信息等电压信号;设备工作现场环境监控采取温度传感器和烟雾传感器采集现场信息,采集信号经 A/D 转换后送至 8051 系列单片机预处理,处理后的信号经 Modem 沿双绞线传送到中心机房,在中心机房对调制信号解调送到主机对信号进行识别判决处理。(关于通过 Modem 实现远距离实现收发数据的实现方法将另行撰文讨论)。

1.2 系统软件

监控软件的操作系统采用 Windows 95/98 或 Windows NT,编程软件采用 Visual C++ 6.0 来实现程序的界面和对计算机串口的通信。

软件部分控制整个监控过程,首先根据用户输入的口令授予用户不同的权限,确定用户的操作等级,比如二级用户可以监视设备运行情况;一级用户可以开关发射机等等。

2 串口通信

2.1 串口通信

本文主要讲述在直接电缆联接方式下实现上、下位机串口通信的实现方法，串口通信时最重要的是对串行口进行初始化，该初始化并非只是在 Visual C++中对串口控件进行重新打开和关闭操作，而是要对通用异步接受发送器（UART）的各个寄存器进行初始化。设串口基地址为 CommAddress（通常为 0x2E8、0x2F8、0x3E8、0x3F8 之一，这个地址可以在 CMOS 中得到）在 C 语言环境下对串口初始化时可以直接利用 _inp()、_outp()函数对 UATR 的以下寄存器进行设置：

* 波特率寄存器（CommAddress; CommAddress+1）：设置通信的波特率因子；

* 线控制寄存器（CommAddress+3）：设置字符的长度、停止位个数和奇偶校验类型；

* 中断控制寄存器（CommAddress+1）：允许发送保持寄存器空、接收数据准备好、Modem 状态改变、接收数据错等多个类型的中断；

* Modem 控制寄存器（CommAddress+4）：将 DTR 和 RTS 位置 1，并允许中断。

设置寄存器时首先设置波特率寄存器，因为它们要求线控制寄存器的第一位为 1，接着将线控制器的第 7 位 0，这样就可以访问后续寄存器。波特率与波特率因子的关系举例：如波特率为 75，其波特率因子为 0x06（MSB），0x00（LSB）波特率为 1200，其波特率因子为 0x00（MSB），0x60（LSB）1800 对应 0x00（LSB），0x40（MSB）2400 对应 0x00（LSB），0x30（MSB）3600 对应 0x00（LSB），0x20（MSB）9600 对应 0x00（LSB），0x0C（MSB）等等，其中 LSB 对应的地址为 CommAddress；MSB 的地址为 CommAddress+1。

我们对串口的初始化是在 Visual C++中调用 API 函数设置 DCB 设备块来实现的，，这样效率更高一些。在 VC 环境下串口通信遵循的步骤为：1）通信端口资源的打开；2）端口参数的初始化设置；3）串口的读写通信操作；4）关闭通信资源。

Visual C++中相应用到的函数主要有：

CreateFile：用于打开通信串行口；

SetupComm：用于设置输入输出队列的大小；

GetCommState：获得端口参数当前配置；

SetCommState：设置端口 DCB 的各个参数；

WaitCommEvent：等待串口事件；

SetCommTimesout：设置读写超时参数；

PurgeComm：刷清缓冲区；

ReadFile：读指定串行端口数据；

WriteFile：写指定串行端口数据；

CloseHandle：关闭指定端口。

以上函数的详细用法可参见 VC++6.0 的在线帮助 MSDN。

在 WIN32 环境下通信函数是中断驱动的：发送数据时，先将其放入缓存区，串口准备好后，就将其发送出去；传来的数据迅速申请中断，使 Windows 接收它并将其存入缓冲区，以供读取。

串口通信中发送过程较易实现，接收处理方式主要有查询和中断方式。中断方式具有效率高、接收准确、编程简单等优点。因此本文采用的是中断接收方式。为了实现串口通信，用到了多线程技术，主线程先做一些必要的初始化工作，然后主线程建立通信监视线程监视通信端口，当指定的串口事件发生时，向主线程发送 WM_COMMNOTIFY 消息，当不需要 WM_COMMNOTIFY 消息时主线程终止监视线程。

以下是串口通信的读操作实现程序部分源代码。

```
//首先在 DhcomView.h 中定义如下变量
HANDLE hCom,hEvent;
BOOL Error,Result,Success,Connected;
DCB dcb;
DWORD dwEvtMask,dwLength;
OVERLAPPED ComRead;
COMSTAT comState;
COMMTIMEOUTS CommTimeOuts;
char ReadBuf[1024]="";
CwinThread * m_Com;
//在头文件中加入:
DWORD FAR PASCAL CommWatchProc(LPVOID);
# define WM_COMMNOTIFY  WM_USER+101
//消息处理函数声明:
afx_msg LONG OnCommnotify(UINT wParam, LONG lParam);

//消息映射:
BEGIN_MESSAGE_MAP(CDhcomView,CFormView)
    //{{AFX_MSG_MAP(CDhcomView)
    ..... //ClassWizard 产生的消息映射
    //}}AFX_MSG_MAP
ON_MESSAGE(WM_COMMNOTIFY OnCommnotify)
END_MESSAGE_MAP()
//Dhcomview.cpp 实现部分
#include "stdafx.h"
#include "DhcomView.h"
#include "math.h"
#include <stdlib.h>
IMPLEMENT_DYNCREATE(CDhcomView,CFormView)
CDhcomView::CDhcomView()
```

```
    :CFormView (CDhcomView::IDD )
{
    //{ {AFX_DATA_INIT (CDhcomView )
        //} }AFX_DATA_INIT
}
CDhcomView::~CDhcomView ()
{
}
/初始化串口资源
void CDhcomView::OnInitialUpdate ()   /首次生成窗口时
打开及初始化串口
    { hCom = CreateFile ("COM1", /指定串口为 COM1
      GENERIC_READ|GENERIC_WRITE, /设置读写模式
      0, /共享模式,此项必须为零
      NULL, /安全属性
      OPEN_EXISTING,
       /产生方式,必须设为 OPEN_EXISTING
      FILE_ATTRIBUTE_NORMAL|FILE_FLAG_
OVERLAPPED,
       /文件类型为异步通信
      NULL ) /通信中此项必须设置为 NULL
    if (hCom = =INVALID_HANDLE_VALUE )
    {    MessageBox ("CreateCommFile  Error",
         /设置错误,检查串口是否正使用
        "警告",MB_OK )
     }
BOOL Success = SetCommMask (hCom, EV_RXFLAG ) //
串口事件为接收到字符
    if (!Success )
     { MessageBox ("SetCommMask Error!",
       "警告",MB_OK )
     }
BOOL Error = SetupComm (hCom,4096,4096 )
/输入、输出缓冲区设为 4096 字节
    if (!Error )
     { MessageBox ("SetupComm Error!",
       "警告",MB_OK )
     }
PurgeComm (hCom, RURGE_TXABORT|RURGE_RXABORT|
PURGE_TXCLEAR|PURGE_RXLEAR )
/刷清缓冲区
GetCommTimeouts (hCom,&CommTimeOuts );
CommTimeOuts.ReadIntervalTimeout = 0xFFFFFFFF;
CommTimeOuts.ReadTotalTimeoutMultiplier = 0;
CommTimeOuts.ReadTotalTimeoutConstant = 1000;
CommTimeOuts.WriteTotalTimeoutMultiplier = 0;
CommTimeOuts.WriteTotalTimeoutConstant = 1000;
SetCommTimeouts (hCom, &CommTimeOuts );
/对 DCD 设备块进行设置
BOOL Error = GetCommState (hCom,&dcb )
    if (!Error )
     { MessageBox ("GetupComm Error!",
       "警告",MB_OK )
     }
    dcb.BaudRate = 9600;
    dcb.ByteSize = 8;
    dcb.Parity = NOPARITY;
    dcb.StopBits = ONESTOPBIT;
    Error = SetCommState (hCom,&dcb )
    if (!Error )
     { MessageBox ("SetCommState Error!",
       "警告",MB_OK )
     }
Connected = TRUE;
memset (&ComRead, 0, sizeof (OVERLAPPED ));
ComRead.hEvent = CreateEvent (NULL,
                              TRUE,   /手工重置事件
                              FALSE, /无信号
                              NULL ); /无名字
hEvent = CreateEvent (NULL, /无安全属性
                      TRUE, /手工重置事件
                      TRUE, /初始状态:有信号
                      NULL /初始状态:无名字 )
  /创建监视线程
if (m_Com )
    {
    m_Com - >ResumeThread ();
    :: WaitForSingleObject (m_Com - >m_hThread, INFI-
NITE )
    delete m_Com;
     }
m_Com = AfxBeginThread (CommWatchProc,
                        &m_hWnd,
                        THREAD_PRIORITY_NORMAL,
                        0,
                        CREATE_SUSPENDED )
m_Com - >m_bAutoDelete = FALSE;
m_Com - >ResumeThread ();
 }
UNIT  CommWatchProc (LPVOID PHWndView )
 {
    while (Connected )
     {
       DWORD dwRead;
       if (!SetCommMask (hCom,EV_RXCHAR ))
        {
       DWORD err = GetLastError ();
       Return (FALSE )
        }
    dwRead = 0;
    WaitCommEvent (hCom,&dwRead,&ComRead )
    If ((dwRead&EV_RXCHAR ) = =EV_EVENT )
      {
     WaitForSingleObject (hEvent,0xFFFFFFFF )
     ResetEvent (hEvent ) /同步事件复位为无信号
     PostMessage (* (HWND * )PHWndView,
WM_COMMNTIFY,NULL,NULL )
     /发送消息
     }
   }
  retrun 0;
 }
```

(下转第 13 页)

性将 FoxPro 的串行通信语句置于 Visual FoxPro 的过程和函数中实现串行通信功能。

(2)DLL 方法之一:

利用 FOXTOOLS.FLL API 函数库中的两个函数 Regfn() 和 Callfn() 来注册和调用 16 位 Windows.DLL 通信函数。该法对一般用户而言难度较大。

(3)DLL 方法之二:

使用 DECLARE---DLL 命令来注册和调用 32 位 Wondows.DLL 函数。该法并不常用,对一般用户而言难度也较大。

(4)MSComm 控件法:

Visual FoxPro 环境下,在菜单项 Tools/Options 的 Controls 栏列表中选取 Microsoft Communications Control 6.0 并打开 windows\system 下的文件 Mscomm32.ocx 就可以获得该通信控件。通信示例如下:

```
PROCEDURE form_Commnication ( )      && 控件名为 comm
   this form.comm.commport = 1       && 通信口为 COM1
   this form.comm.settings = " 9600 , N , 8 , 1 "   && 设置数据格式
   this form.comm.portopen = .T.     && 打开 COM1
   this form.comm.inputlen = 0       && 读缓冲区
   mydata = thisform.comm.input      && 读并清除缓冲区
   ⋮                                 && 通信部分
   thisform.comm.portopen = .F.      && 关闭 COM1
   release thisform
RETURN
```

7 结束语

上面所介绍的各语言的串行通信方式主要是针对 RS-232 接口,作适当调整后也可用于 RS-422 和 RS-485 等接口;机型针对 intel 80x86 作 CPU 的 PC 机和工控机。根据需要和条件,PC 机、工控机、单片机等两两串行通信在硬件上可用 RS-232 等接口或者 MODEM 接口,而上位机中软件编程语言的选择更灵活一些,面向对象的可视化计算机语言可作为首选。

参考文献

1. 邱公伟等,实时控制与智能仪表多机系统的通信技术,清华大学出版社,1996
2. [美]L.ATKINSON M.ATKINSON E.MITCHELL,曹晓峰等译,Microsoft C/C++ 7.0 使用指南,清华大学出版社,1993
3. MSDN Library (光盘版), Microsoft Develop Network, Visual Studio 6.0, 2001

(上接第 7 页)

```
LONG CDhcomView::OnCommntify (UINT wParam, LONG lParam )
    {
     ClearCommError (hCom,&dwEvtMask,&comState );
     dwLength = comState.cbInQue;
     if (dwLength >=7 )
     Error = ReadFile (hCom,ReadBuf,dwLength,
     &dwLength,&ComRead );
     if (!Error )
      {  MessageBox ("读串口不正常 Error!",
         "警告",MB_OK );
      }
     SetEvent (hEvent ) ;/同步事件复位为有信号
     Return 0;
    }
void CDhcomView::OnDestroy ()
    {  CFormView::OnDestroy ();
       CloseHandle (hEvent );
       CloseHandle (hCom ) ;/关串口资源
       Delete m_Com;
    }
```

3 Normark7050 软件系统的加载

因为 Normark7050 设备自带了监控软件,为了避免重复开发,缩短开发周期,可以利用三个 API 函数 winexec、shellexecute、createprocess 其中的一个来对 Normark7500 软件系统进行加载:其中 WinExec 最简单,两个参数,前一个指定路径,后一个指定显示方式,比如用 SW_SHOWMAXMIZED 方式去加载一个程序。具体的函数用法请参阅 MSDN。

4 结束语

我们在项目开发过程中对 Windows 环境下的串行通信及 Windows API 函数的用法有了一定认识,结合软、硬件设计提出了这一实时监控系统的实现方法,在此一并总结出来,希望能和大家交流。

参考文献

1. Marshall Brain,Lance Lovette,MFC 开发人员指南机械工业出版社,1999
2. 戴梅萼,微型计算机技术及应用清华大学出版社,1991
3. Michael J.Young,Visual C++6 从入门到精通,电子工业出版社,1999
4. 王未央等,基于实时数据采集的驾驶桩考系统的实现计算机应用,2000.11