在 Windows95 下 PC 机和单片机的串行通信

烟台海军航空工程学院 301 教研室(264001)　王　亭　李瑞涛　宋召清

摘　要：基于 32 位操作系统 Windows95 的工业控制 PC 机和单片机间的串行通信，给出了用 VC++5.0 和 PL/M-96 语言编写的串行通信程序。

关键词：串行通信　单片机　Windows API 接口

串行口是计算机与外部设备进行数据交换的重要介质，所以串行通信在工程实现中有着广泛的应用。而 Microsoft 公司的 VC++5.0 功能强大，其基础类库(MFC)封装了 Win32 API 中的标准通信函数，可方便地支持串口通信。本文结合笔者在开发转台控制程序中对串行通信编程的一些收获，介绍了在 Win95 下用 VC++5.0 和 PL/M-96 语言开发 PC 机和单片机通信程序的一些经验。

1　系统组成

系统中采用 N80C196KB 单片机作为下位机，工业控制 PC 机为上位机，二者通过 RS-232 串行口接收或上传数据和指令。传输介质为二芯屏蔽电缆，接线图如图 1 所示。

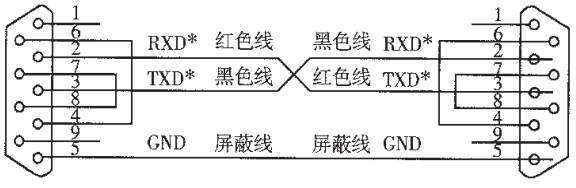

图 1　RS-232 电缆连接图

RS-232 信号的电平和单片机串口信号的电平不一致，必须进行二者之间的电平转换。在此使用的集成电平转换芯片 MAX232 为 RS-232C/TTL 电平转换芯片。它只使用单+5V 为其工作，配接 4 个 1μF 电解电容即可完成 RS-232 电平与 TTL 电平之间的转换。其原理图如图 2 所示。转换完毕的串口信号 TXD、RXD 直接和 N80C196KB 的串行口连接。

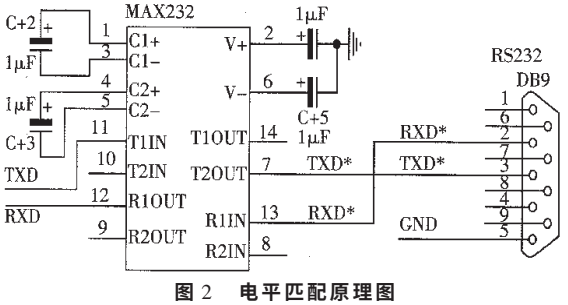

图 2　电平匹配原理图

2　通信协议

项目中工业控制 PC 机承担主控任务，负责转台运动参数的设定，程序由 VC++5.0 编写。单片机接受 PC 机指令，并根据指令信息驱动转台运动或上传数据。通信协议如下：

采用 RS-232 串口异步通信，1 位起始位，8 位数据位，1 位停止位，无奇偶校验，波特率 2 400b/s。传输数据采用 ASCII 模式。指令形式采用 7 个 ASCII 串，格式为“$××××*”，其中‘$’和‘*’分别标明该指令的起始和结束，‘××××’为指令内容。如：“$REMOT*”为远控、“$H±×××*”为转台航向给定命令，“$ASKQ?*”为主机查询倾斜角命令等。下位机按接收到的指令工作。如果主控机发出错误的指令或现在正执行上一条角度给定命令的过程中又收到新的角度给定命令，将不做任何控制，并显示 Error 提示，1 秒钟后自动返回。

3　编程机制

3.1　主控机的 VC++5.0 串行通信程序

在 Win95 环境下提供了完备的 API 应用程序接口函数，程序员通过这些函数与通信硬件接口。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。

发送过程较易实现，接收处理方式主要有查询和中断方式。采用查询方式时，CPU 要不断测试串口是否有数据，以防止接收串口数据出现错误、遗漏、效率低；而采用中断方式则无需测试串口，一旦有数据传至，CPU 终止当前任务，由中断服务程序完成操作。所以，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。

在 Windows95 中，把串口当作文件来操作，取消了原 Windows3.X 下的 WM_COMMNOTIFY 消息，因此在中断机制下，程序员必须像下面这样自定义消息：

```
#define WM_COMM_READ WM_USER+100
```

```
    ON_MESSAGE(WM_COMM_READ,OnCommRead)
                                        //消息映射入口
    LONG OnCommRead(UINT wParam,LONG lParam);
                                        //消息响应函数说明
```

为了实时响应串口事件，必须在主线程之外创建1个辅助监视线程。为防止各线程的共享资源访问出错，在程序中的各线程的动作应同步化。这可利用MFC提供的同步化事件对象。

所以在开始通信前，首先要初始化串口(包含选串口、设置串口掩码、设置缓冲区大小、设置波特率等串行参数)、创建同步事件、创建线程并让辅助线程处于发信号状态，在主线程执行其它操作时，通信监视线程监视串口，一旦输入缓冲区内有数据立即向主线程发送WM_COMM_READ消息，然后由消息响应函数做进一步处理，一次通信即完成。用户编写串口通信程序只需实现如图3所示步骤。

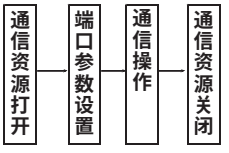

图3 串行通信编程步骤

相应函数主要有：

CreateFile()：用于打开通信资源；

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；

ReadFile()和WriteFile()：读、写指定端口数据；

CloseFile()：关闭指定端口。

以上函数的详细用法可参见VC++5.0的在线帮助及本文给出的例程和参考文献。

3.2 下位机的PL/M-96串行通信程序

N80C196KB单片机程序采用PL/M-96语言编写。PL/M-96语言是一种模块化的高级单片机编程语言，适用于MCS-96系列单片机。首先N80C196KB将串口初始化为1位起始位、8位数据位、1位停止位，波特率为2400b/s。串行口信号输入采用中断方式，输出采用查询方式。其通信程序见4.2节。

4 程序实例

4.1 VC++5.0程序

以下是串口通信的程序部分代码。

```
//首先在ComView.h中定义如下变量
HANDLE hCom,hEvent;
HWND hwComwnd;
BOOL Error,Result,Success,threadFlag;
DCB dcb;
DWORD dwEvtMask,dwLength;
OVERLAPPED ComRead,ComWrite;
COMSTAT comState;
char ReadBuf[14]="";
char *send;
char *receive;
//在头文件中加入：
#define WM_COMM_READ WM_USER+101
//Generated message map functions
//{ {AFX_MSG(CComView)
//消息处理函数声明：
    afx_msg LONG OnCommRead (UINT wParam,LONG
lParam);
        //} }AFX_MSG
        DECLARE_MESSAGE_MAP()
    };
//消息映射：
BEGIN_MESSAGE_MAP(CComView,CFormView)
    //{ {AFX_MSG_MAP(CComView)
        ON_MESSAGE(WM_CCOM_READ,
                                    OnCommRead)
    //}}AFX_MSG_MAP
    END_MESSAGE_MAP()
#include "stdafx.h"
#include "gd.h"
#include "ComView.h"
#include "math.h"
#include <stdlib.h>
#ifdef_DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[]=_FILE_;
#endif
IMPLEMENT_DYNCREATE(CComView,CFormView)
CComView::CComView()
    :CFormView(CComView::IDD)
{
    //{ {AFX_DATA_INIT(CComView)
        //} }AFX_DATA_INIT
}
CComView::~CComView()
{
}
void CComView::OnInitialUpdate()
{    CFormView::OnInitialUpdate();
    //TODO:Add your specialized code here and/or
    //call the base class
hCom=CreateFile("COM2",//指定串口
        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_0
        VERLAPPED,//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);
    if(!Success)
    {   MessageBox("SetCommMask Error!",
        "警告",MB_OK);
    }
BOOL Error=SetupComm(hCom,1024,1024);
    if(!Error)
    {   MessageBox("SetupComm Error!",
        "警告",MB_OK);
    }
BOOL Error=GetCommState(hCom,&dcb);
    if(!Error)
    {   MessageBox("GetupComm Error!",
        "警告",MB_OK);
    }
    dcb.BaudRate=2400;
    dcb.ByteSize=8;
    dcb.Parity=NOPARITY;
    dcb.StopBits=ONESTOPBIT;
    Error=SetCommState(hCom,&dcb);
    if(!Error)
    {   MessageBox("SetupCommState Error!",
        "警告",MB_OK);
    }
hEvent=CreateEvent(NULL,//无安全属性
                   TRUE,//手工重置事件
                   TRUE,//初始状态 ;无信号
                   NULL//初始状态 :无名字);
    if(m_Com)
    {
    threadFlag=TRUE;
    m_Com->ResumeThread();
    ::WaitForSingleObject(
    m_Com->m_hThread,INFINITE);
    delete m_Com;
    }
m_Com=AfxBeginThread(ComReceive,&m_hWnd,
        THREAD_PRIORITY_NORMAL,0,
                        CREATE_SU SPENDED,NULL)
m_Com->m_bAutoDelete=FALSE;
threadFlag=FALSE;
m_Com->ResumeThread();
hwComWnd=m_hWnd;}
UINT ComReceive(LPVOID m_View)
{
  DWORD dwRead;
memset(&ComRead,0,sizeof(OVERLAPPED));
    if(!SetCommMask(hCom,EV_RXCHAR))
    {
    DWORD err=GetLastError();
    Return(FALSE);
    }
while(1)
{
dwRead=0;
WaitCommEvent(hCom,&dwRead,&CoRead);
   If((dwRead&EV_RXCHAR)==EV_EVENT)
   WaitForSingleObject(hEvent,0xFFFFFFFF);
ResetEvent(hEvent);//重启动事件
PostMessage(hEvent,WM_COMM_READ,NULL,NULL);
                                        //发送消息
}
    return 0;
}
LONG CComView::OnCommEvent(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);
    Return 0;
}
    void CComView::sent()
{//例如:
char SendBuf="$ASKQ? *";
send=SendBuf;
Error=WriteFile(hCom,send,7,&dwLength,&ComWrite);
if(!Error)
    {   MessageBox("写串口不正常 Error!",
        "警告",MB_OK);
    }
```

```
}
void CComView::OnDestroy()
{   CFormView::OnDestroy();
    CloseHandle(hCom);//关串口资源
}
```

4.2 PL/M-96 程序

```
START:  DO;
$NOLIST
$INCLUDE(N80C196.PLM)
$LIST
DECLARE RCMD(16) BYTE;/* 接收缓冲区 */
DECLARE INFO(16)  BYTE;/* 发送缓冲区 */
DECLARE RI        BYTE;
DECLARE CHA       BYTE;
COMSET:PROCEDURE;
        DISABLE;/* 关中断 */
        INT_MASK=01000000B;/* 开放串口中断 */
        INT_MASK1=00000000B;
        IOC1=IOC1 OR 20H;
        SP_CON=00001001B;
        BAUD_RATE=9BH;*/波特率为 2400*/
        BAUD_RATE=80H;/*1 个起始位,8 个有效位,1
                        个停止位 */
END     COMSET;
INT6:   PROCEDURE   INTERRUPT 6;
        /* 串口中断服务程序 */
        DH=SP_STAT;
        CHA=SBUF;
        IF CHA='$' THEN RI=0;
        IF CHA='*' THEN
        DO;
           CALL RUNCMD;/* 执行命令 */
           CALL CLRRCMD;/* 清接收缓冲区 */
        END;
        RCMD(RI)=CHA;
        RI=RI+1;
        INT_PENDING1=0;
        INT_PENDING=0;
END     INT6;
COMOCHA:  PROCEDURE (CHA);
          DECLARE (CHA,CHA1) BYTE;/* 查询方
          式发送字符子程序 */
          SBUF=CHA;
LOOP:     CHA1=SP_STAT;
          IF (BITTST(CHA1,5)<>0FFH) THEN
             GOTO LOOP;
END       COMOCHA;
COMOSTR:  PROCEDURE (ADDR,LENTH);/* 发送字
                                  符串子程序 */
          DECLARE ADDR ADDRESS;
          DECLARE LENTH BYTE;
          DECLARE I BYTE;
          DECLARE CHA BASED ADDR BYTE;
          /* 字符串以 ADDR 地址为首地址 */
          DO I=0 TO LENTH-1;/* 字符串以 LENTH
                              为长度(量长 255 个)*/
             CALL COMOCHA(CHA);
             ADDR=ADDR+1;
          END;
END       COMOSTR;
MAIN:     WSR=0;
          INT_PENDING=0;
          INT_PENDING1=0;
          SP=6A00H;
          DISABLE;/* 关中断 */
          IOC0=00000000B;
          IOC1=00000000B;
          IOC2=00000000B;
          CALL COMSET;/* 串口设置 */
          ENABLE;/* 关中断 */
LOOP:     ;
          :::::::::
          CALL COMOSTR(.INFO(0),16);/* 将发送
                              缓冲区的信息发送给主机 */
          ::::::::
          GOTO LOOP;
END START;
EOF;
```

5 结束语

虽然目前讨论串行通信的技术文献很多，但基于 DOS 和 WIN16 的占多数，基于 WIN32 的文献介绍较少，并且对外设的硬件组合较少涉及。我们在对教学设备进行改装的过程中对 Windows 环境下的串行通信及 Windows API 函数的用法有了一定认识，并且通过对外设的改装积累了 PC 机和一般智能设备之间通过串口进行通信的软、硬件设计方法。在此一并总结出来，希望能和大家交流。

参考文献

1 吴华，岳晋生.Windows NT Win32 软件开发使用详解.北京：电子工业出版社，1995

2 Norton P，McGregor R 著，孙凤英，魏军，徐京等译. MFC 开发 Windows 95/NT4 应用程序.北京：清华大学出版社，1998

3 袁涛，孙腾湛. PL/M-96 程序设计语言及其应用.北京：清华大学出版社，1992

(收稿日期：1999-08-26)