Visual C++串行通讯控件使用方法与技巧的研究*

杨在春[1]，许传军[2]

(1.山东科技大学 信息科学与工程学院,泰安 271019;2.兖州矿业集团有限公司驻日照办事处,山东 日照 276800)

摘　要:通过对引入通讯控件、发送通讯数据、接收通讯数据等几个问题的分析和研究,阐述了在 Visual C++中使用串行通讯控件的基本方法和技巧,并在此基础上给出了实用的程序实现代码。

关键词:通讯控件; 发送; 接收

中图分类号:TP312VC　　**文献标识码**:A

Study on Method and Technique for the Use of Serial Communication Components in Visual C++

YANG Zai-chun, XU Chuan-jun[2]

(1. College of Info Science and Engg, SUST, Taian 271019, China;
2. Rizhao Office of Yankuang Goup Co., LTD. Rizhao, Shandong 276800, China)

Abstract: By analyzing and studying the problems of introducing communication components sending and receiving communication data, the paper explains the basic method and technique of using serial communication componens in Visual C++, on the basis of this practical programming code is given.

Key words: communication component; sending; receiving

面向对象的程序设计语言 Visual C++其界面的开放性著称,因其功能强大而受到广大软件开发人员的青睐,串行通讯是 Windows 编程中的一个重点和难点,且在现实工作中被广泛采用。因此,利用 Visual C++进行串行通讯的程序开发,在许多方面具有重要的研究意义。在 Visual C++使用串行通讯控件编程,首先应该知道控件工作原理、串行通讯控件基本接口,还必须了解 COleVariant 类型与动态数组应用等一些高级技巧知识。结合这些知识,从引入通讯控件、发送通讯数据、接收通讯数据等方面进行分析和研究。

1　引入通讯控件

如果使用的是其他可视化程度较高编程语言,如 VB、C++ Buider 或 Delphi 等,通讯控件可以被直接引入到任意某个窗体之中[1]。而使用 Visual C++时,只有通过对话框资源编辑器来引入通讯控件到对话框中或由 CFormView 派生而来的视图中。如果选择的对话框是模态的,则在使用通讯控件时必须打开该对话框,这对于基于对话框的应用程序使用十分方便。如果不想使用对话框,则可将程序的视图类设计为由 CFormView 派生而来的视图。CFormView 与相应的对话框资源相关联,具有很多对话框的特点,这样,可以将通讯控件直接引入到程序视图之中。后面这种方法的缺点是视图的使用受到了很大的限制。下面介绍另外一种使用比较方便,且更加有效的方法。

同样,我们还是将通讯控件引入到对话框中,只是将对话框做成非模态的。如果没有其他需要,对话框中只包含通讯控件一个元素,且在程序执行时,该对话框始终处于不可视状态。设计时程序中任何地方,只要能访问到该对话框,就可使用其中的通讯控件。再介绍非模态对话框 CCommDlg 的制作及其使用方法,CCommDlg 由 CDialog 派生而来。由于在程序执行过程中对话框是不可视的,用户不可能对其进行任何操作,因此不需要处理对话框的确定和取消按钮的响应消息。为了方便起见,只需向其中增加一个 BOOL 类型的成员函数 Create(),定义如下:

```
BOOL CCommDlg::Create()
{
  return CDialog::Create(CCommDlg::IDD);
}
```
[2]

由于对话框是不可视的,必须通过对话框资源编辑器将其 Visible 属性设置掉。

由于主框架窗口 CMainFrame 是程序的主窗口,程序其他部分的代码很容易通过函数 AfxGetMainWnd 或其他方法访问到 CMainFrame 类型的成员,并程序的主窗口的 C++ 对象和其窗口对象在程序的运行过程中总是有效的,直到程序运行结束才将其释放和析够。因此,可以在 CMainFrame 中增加一个指向 CCommDlg 类型的指针变量:CCommDlg * m_pCommDlg,在 CMainFrame 的 OnCreate 成员函数构造并创建该对话框,代码如下所示:

```
int CMainFrame::OnCreate(LPCREATESTRUCT lp
CreateStruct)
{
  if(CFrame_Wnd::OnCreate(lpCreateStruct) =
= -1) return -1;
   ……
  m_pCommDlg = new CCommDlg();
  if(!m_pCommDlg) return -1;
  m_pCommDlg->Create();
   ……
}
```

m_pCommDlg 对象创建成功后,其 C++ 对象和窗口对象一直处于有效状态,直到程序结束时,才用下面的代码在 CMainFrame 的析够函数中将其释放:

```
CMainFrame::~CMainFrame()
{
  if(m_pCommDlg)
  {
    if(m_pCommDlg->GetSafeHwnd())
    m_pCommDlg->DestroyWindow();
    delete m_pCommDlg;
  }
……
}
```

在程序的执行过程中,可以通过访问 m_pCommDlg 来访问通讯控件 m_Comm,比如程序语句:m_pCommDlg->m_Comm.SetPortOpen(true) 可以用来打开通讯端口。注意,上述存放通讯控件的对话框是不可视的,若想使其成为可视的,并能够完成一些如设置通讯控件的功能的话,还需要做一些额外的工作。

2 发送通讯数据

在 VisualC++ 中,通讯控件发送数据的任务是由其接口成员函数 SetOutput 来实现的,该函数使用的参数是 constVARIANT&newValue。VARIANT 其实是一个 C++ 结构类型,COleVariant 类型对 VARIANT 做了进一步的封装和扩展,提供了许多新的功能和操作方法,支持 OLE 自动化,且更容易向其数据成员填入数据[3]。由于 COleVariant 类型由 VARIANT 派生而来,因此将 COleVariant 类型的变量传递给 SetOutput 函数更为方便。另外,SetOutput 的参数 newValue 类型必须是存放字节类型数据的动态数组。因此,可以利用 Visual C++ 提供 CByteArray 类型来构造 COleVariant 类型的对象,并将其传递给 SetOutput 函数。对 CByteArray 类型变量的操作相对来说要容易得多,比如其成员函数 SetSize 可用来设置动态数组的大小,下标操作符[]可用来为其元素赋值等等。下面的程序代码可实现将存放在缓冲区 strBuf 中的 100 个字节的数据通过通讯控件发送出去:

```
BYTE strBuf[128];
CByteArray OutBuf;
COleVariant varOutput;
 ……
OutBuf.SetSize(100);
for(i=0;i<100;i++)OutBuf[i]=strBuf[i];
varOutput=OutBuf;
```

```
m_pCommDlg->m_Comm.SetOutput(varOutput);
……
```

利用通讯控件发送数据的关键在于构造 COleVariant 类型的变量,并向其中填入通讯数据,使其能满足通讯控件的成员函数 SetOutput 的需要。上面的程序语句 varOutput = OutBuf 可以直接写成:

```
COleVariant varOutput(OutBuf);
```

但这样必须将变量 varOutput 的定义语句 COleVariant var Output 删除掉。

3 接收通讯数据

通讯控件接收通讯数据的任务是通过其接口成员函数 Get Input 来实现的,该函数的返回值的类型为 VARIANT,同样可以将其赋值给 COleVariant 类型的变量。必须知道,该函数返回值变量的有效成员为 parray,其类型定义为 SAFEARRAYFAR*,即 parray 为一个安全数组指针,且数组的元素类型为 BYTE[4]。因此,对 Get Input 函数返回值的访问,可以通过安全数组操作函数来实现,比如:读取数组的维数、下边界、上边界以及各个元素的实际的值等。该返回值变量所对应的安全数组的维数为 1,下边界为 0。根据上述分析,下面的程序代码基本能够实现利用通讯控件接收通讯数据:

```
CByteArray ByteBuf;
COleVarinat varInput;
long ix,1,u;
BYTE bit;
……
varInput = m_pCommDlg->Comm.GetInput();
if(varInput.parray != NULL)
{
  SafeArrayGetLBound(varInput.parray,1,&1);
  SafeArrayGetUBound(varInput.parray,1,&u);
  for(ix = 1;ix <= u;ix++)
  {
    SafeArrayGetElement(varInput.parray,&ix,&bit);
    ByteBuf.Add(bit);
  }
}
……
```

4 小 结

控件是面向对象程序设计方法的最主要的产物,它的产生对程序设计带来了根本性的变革,使用控件很大程度上提高了程序的通用性和可移植性。因此,利用串行通讯控件进行串行通讯编程越来越成为软件开发人员的首选[5]。在使用通讯控件发送通讯数据时,首先应按照通讯数据的格式要求组织好通讯数据,然后将其赋值给 COleVariant 类型的变量,即可利用通讯控件接口函数 SetOutput 来发送数据;接受通讯数据时,应先将通讯控件缓冲区中的数据接受到 COleVariant 类型的变量中,然后即可以编程对其进行校验、分析与处理等各种操作。本文介绍的使用通讯控件发送与接收通讯数据的方法都是基于二进制来实现的,基于本文的情况基本与此类似。在 Visual C++中用好通讯控件还有其他一些应该注意的地方,比如通讯波特率的设置、接收与发送缓冲区的设置以及通讯过程中的接受延时和发送延时问题的处理等等。

参 考 文 献:

[1]肖勇,等. C++Builder 4.0 应用程序开发[M]. 北京:人民邮电出版社,2000.

[2]David J. Kruglinski,等. Visual C++6.0 技术内幕(中译本)[M]. 北京:希望电子出版社,2000.

[3]李真文,等. Visual C++6.0 编程高手[M]. 北京:希望电子出版社,2001.

[4]赵仕健. Visual C++6.0 编程与实例解析[M]. 北京:科学出版社,2000.

[5]Joseph Schumller. ActiveX 轻松进阶(中译本)[M]. 北京:电子工业出版社,1998.

嵌入式资源免费下载

总线协议：

1. 基于 PCIe 驱动程序的数据传输卡 DMA 传输
2. 基于 PCIe 总线协议的设备驱动开发
3. CANopen 协议介绍
4. 基于 PXI 总线 RS422 数据通信卡 WDM 驱动程序设计
5. FPGA 实现 PCIe 总线 DMA 设计
6. PCI Express 协议实现与验证
7. VPX 总线技术及其实现
8. 基于 Xilinx FPGA 的 PCIE 接口实现
9. 基于 PCI 总线的 GPS 授时卡设计
10. 基于 CPCI 标准的 6U 信号处理平台的设计
11. USB30 电路保护
12. USB30 协议分析与框架设计
13. USB 30 中的 CRC 校验原理及实现
14. 基于 CPLD 的 UART 设计
15. IPMI 在 VPX 系统中的应用与设计
16. 基于 CPCI 总线的 PMC 载板设计
17. 基于 VPX 总线的工件台运动控制系统研究与开发
18. PCI Express 流控机制的研究与实现
19. UART16C554 的设计
20. 基于 VPX 的高性能计算机设计
21. 基于 CAN 总线技术的嵌入式网关设计

VxWorks：

1. 基于 VxWorks 的多任务程序设计
2. 基于 VxWorks 的数据采集存储装置设计
3. Flash 文件系统分析及其在 VxWorks 中的实现
4. VxWorks 多任务编程中的异常研究
5. VxWorks 应用技巧两例
6. 一种基于 VxWorks 的飞行仿真实时管理系统
7. 在 VxWorks 系统中使用 TrueType 字库

8. 基于 FreeType 的 VxWorks 中文显示方案
9. 基于 Tilcon 的 VxWorks 简单动画开发
10. 基于 Tilcon 的某武器显控系统界面设计
11. 基于 Tilcon 的综合导航信息处理装置界面设计
12. VxWorks 的内存配置和管理
13. 基于 VxWorks 系统的 PCI 配置与应用
14. 基于 MPC8270 的 VxWorks BSP 的移植
15. Bootrom 功能改进经验谈
16. 基于 VxWorks 嵌入式系统的中文平台研究与实现
17. VxBus 的 A429 接口驱动
18. 基于 VxBus 和 MPC8569E 千兆网驱动开发和实现
19. 一种基于 vxBus 的 PPC 与 FPGA 高速互联的驱动设计方法
20. 基于 VxBus 的设备驱动开发
21. 基于 VxBus 的驱动程序架构分析

Linux：

1. Linux 程序设计第三版及源代码
2. NAND FLASH 文件系统的设计与实现
3. 多通道串行通信设备的 Linux 驱动程序实现
4. Zsh 开发指南-数组
5. 常用 GDB 命令中文速览
6. 嵌入式 C 进阶之道
7. Linux 串口编程实例
8. 基于 Yocto Project 的嵌入式应用设计
9. Android 应用的反编译
10. 基于 Android 行为的加密应用系统研究
11. 嵌入式 Linux 系统移植步步通
12. 嵌入式 CC++语言精华文章集锦
13. 基于 Linux 的高性能服务器端的设计与研究
14. S3C6410 移植 Android 内核
15. Android 开发指南中文版
16. 图解 Linux 操作系统架构设计与实现原理（第二版）
17. 如何在 Ubuntu 和 Linux Mint 下轻松升级 Linux 内核
18. Android 简单 mp3 播放器源码
19. 嵌入式 Linux 系统实时性的研究
20. Android 嵌入式系统架构及内核浅析
21. 基于嵌入式 Linux 操作系统内核实时性的改进方法研究

Windows CE:

1. Windows CE.NET 下 YAFFS 文件系统 NAND Flash 驱动程序设计
2. Windows CE 的 CAN 总线驱动程序设计
3. 基于 Windows CE.NET 的 ADC 驱动程序实现与应用的研究
4. 基于 Windows CE.NET 平台的串行通信实现
5. 基于 Windows CE.NET 下的 GPRS 模块的研究与开发
6. win2k 下 NTFS 分区用 ntldr 加载进 dos 源代码
7. Windows 下的 USB 设备驱动程序开发
8. WinCE 的大容量程控数据传输解决方案设计
9. WinCE6.0 安装开发详解
10. DOS 下仿 Windows 的自带计算器程序 C 源码
11. G726 局域网语音通话程序和源代码
12. WinCE 主板加载第三方驱动程序的方法
13. WinCE 下的注册表编辑程序和源代码
14. WinCE 串口通信源代码
15. WINCE 的 SD 卡程序[可实现读写的源码]
16. 基于 WinCE 的 BootLoader 研究

PowerPC:

1. Freescale MPC8536 开发板原理图
2. 基于 MPC8548E 的固件设计
3. 基于 MPC8548E 的嵌入式数据处理系统设计
4. 基于 PowerPC 嵌入式网络通信平台的实现
5. PowerPC 在车辆显控系统中的应用
6. 基于 PowerPC 的单板计算机的设计
7. 用 PowerPC860 实现 FPGA 配置

ARM:

1. 基于 DiskOnChip 2000 的驱动程序设计及应用
2. 基于 ARM 体系的 PC-104 总线设计

3. 基于 ARM 的嵌入式系统中断处理机制研究
4. 设计 ARM 的中断处理
5. 基于 ARM 的数据采集系统并行总线的驱动设计
6. S3C2410 下的 TFT LCD 驱动源码
7. STM32 SD 卡移植 FATFS 文件系统源码
8. STM32 ADC 多通道源码
9. ARM Linux 在 EP7312 上的移植
10. ARM 经典 300 问
11. 基于 S5PV210 的频谱监测设备嵌入式系统设计与实现
12. Uboot 中 start.S 源码的指令级的详尽解析
13. 基于 ARM9 的嵌入式 Zigbee 网关设计与实现

Hardware：

1. DSP 电源的典型设计
2. 高频脉冲电源设计
3. 电源的综合保护设计
4. 任意波形电源的设计
5. 高速 PCB 信号完整性分析及应用
6. DM642 高速图像采集系统的电磁干扰设计
7. 使用 COMExpress Nano 工控板实现 IP 调度设备
8. 基于 COM Express 架构的数据记录仪的设计与实现
9. 基于 COM Express 的信号系统逻辑运算单元设计