0 引言
在工业电气传动装置中，变频器因应用方便，性能可靠，所以被作为首选的电机控制器得到广泛应用[1]。计算机控制技术与现代变频器技术相结合，采用变频器的标准RS485通讯接口和内置协议，可以方便地实现微机对下位机的集中控制，并且易于软件实现，费用低廉，具有较高的可靠性和实用价值[2]。
为了便于操作和监控电动机的运行工况，需要对其电压、电流、频率、转速及工作状态等关键信息进行监测，并根据工作需要，修改相应的参数，调整电动机的工作状态。这些信息可通过变频器（下位机）传给微机（上位机），因此需要变频器与微机进行通讯，通过微机监控变频器来控制、协调电动机的运行。本文基于Visual C++下的MSComm控件来实现微机与变频器的串口通讯。
1 变频器通讯机制
1.1 系统硬件总体结构简图
由于多数工业用变频器提供的是符合RS485标准的串行通讯接口，而微机的通讯接口多为RS232C接口[3]，因此在二者之间需要加接RS485-RS232C接口转换器。这样利用该接口通过软件编写的通讯驱动程序就可以实现微机与变频器的信息交换。系统硬件总体方框图如图1所示。



1.2 串行通讯
串行通讯是指外设和计算机间使用一根数据信号线传送数据，这些通讯数据在信号线上是按位进行传输的，每一位数据都占据一个固定的时间长度。
这种通讯方式的优点是使用的数据线少，在远距离通讯中可以节约通讯成本。当然，其传输速度比并行通讯慢。对于RS485 串行接口其最大传输速率为10 Mbit/s，最长通讯距离为1 200 m。
1.3 通讯协议格式
串口通讯传输的基本思想基于帧传输方式，即在向串口发送数据时是逐帧发送。在每帧传输过程中采用应答方式[4]。具体为上位机发送命令参数，变频器首先通过校验判断是否为正常帧，若校验为正常帧，可以接收并做出应答。否则丢弃不予应答。
本文所举例的变频器的通讯帧格式如图2 所示，每一帧都由帧头、帧格式数据块、校验和、帧尾四部分构成。



EOT：起始位，1个字节；
DATA：数据块，n 个字节，包括命令字、参数、工作状态等内容；
BCC：校验和，2个字节，为其前5 个字节数据之和，低位在前，高位在后；
ETX：结束，1个字节。
通讯协议的数据格式为：
1）串口帧采用10位帧，1位起始位，8位数据位，1位停止位，无奇偶校验位，均为十六进制数据，字长为8 bit，采用COM1串口，波特率为9 600 bit/s；
2）变频器根据上位机指令向上传送相应数据；
3）应答发送，微机发出修改命令，变频器接收命令后，修改参数，并把参数修改情况传给上位机；
4）上位机发送其他命令如启动、停止等。
2 MSComm控件
Microsoft Communication Control（简称MSComm）是Microsoft 公司提供的在Windows下进行串行通讯编程的ActiveX控件，它为应用程序提供了通过串行端口接收和发送数据的简便方法。
2.1 MSComm控件的引用
MSComm 控件的引用和其他ActiveX 控件的引用类似，一般都是在Visual C++6.0环境下，新建一个工程，在“工程”菜单下，选择“增加到工程”的Components
and Controls Gallery 选项，在弹出的对话框中打开“Registered ActiveX Controls”，选择“Microsoft Communications Control version 6.0”插入到工程中，这样类似电话机模样的MSComm控件就出现在工具框中，用户可以像使用内部控件那样把它添加到窗体。
值得一提的是，该控件可以放在窗体的任一位置，在运行的时候对外是不可见的。
2.2 MSComm控件两种处理通讯的方式
MSComm控件有两种处理通讯的方式，包括事件驱动方式和查询方式[5,6]。在文献[5，6]中对此做了详细介绍。
2.3 MSComm控件的属性
MSComm 控件中比较重要的几个属性如下：
CommPort 设置并返回通讯端口号，缺省值为COM1，可设置1~16个；
Settings 以字符串的形式设置并返回波特率、奇偶校验、数据位、停止位；
PortOpen 设置并返回通讯端口的状态，也可打开和关闭端口；
Input 读入并清除接收缓冲区的字符；
Output 将发送的字符串或数组写到发送缓冲区；
CommEvent 在通讯错误或事件发生时产生；
OnComm 事件，CommEvent 属性存有该错误或事件的数值码。
3 软件实现
3.1 程序模块
软件实现主要包括上位机初始化程序模块，串口事件处理模块，数据校验模块，数据处理模块等。
3.1.1 上位机初始化模块
打开串口并设置串口参数，同时，预读缓冲区以清空残留数据，对串口初始化。设置的串口参数包括：串口号、波特率、奇偶校验位、数据位、停止位等。
主要代码如下：
If（m\_ctrlComm.GetPortOpen()）
m\_ctrlComm.SetPortOpen（FALSE）；// 检查串口，如果是打开的，则关闭
m\_ctrlComm.SetCommPort（1）； // 根据协议要求，选择COM1端口
m\_ctrlComm.SetSettings（“9 600，n，8，1”）；// 设置波特率9 600，无校验，8个数据位，1个停止位
m\_ctrlComm.SetInputMode（1）；//1：表示以二进制方式检取数据
m\_ctrlComm.SetRThreshold（1）；// 参数1表示每当串口接收缓冲区中有多于或等于1 个字符时将引发一个接收数据的OnComm事件
m\_ctrlComm.SetInputLen（0）；//设置当前接收区数据长度为0，表示全部读取
m\_ctrlComm.GetInput ()；// 先预读缓冲区以清除残留数据
3.1.2 串口事件处理模块
该模块主要是接收下位机发送来的数据包，并对数据包进行数据类型的转换，以便于数据处理模块和数据校验模块对其进行处理。主要代码如下：
switch（m\_ctrlComm.GetCommEvent ()）// 代表控件引起的事件
{
case 2: //comEvReceiv 事件，事件值为2 表示接收缓冲区内有字符
{
VARIANT variant\_inp；
COleSafeArray safearray\_inp；
LONG len，k；
BYTE rxdata[2048]；
CString strtemp，m\_str；
variant\_inp=m\_ctrlComm.GetInput()；//读缓冲区
safearray\_inp=variant\_inp；//VARIANT 型变量转换为ColeSafeArray型变量
len=safearray\_inp.GetOneDimSize ()；// 得到有效数据长度
for（k=0；k<len；k++）{
safearray\_inp.GetElement（&k，rxdata+k）；//转换为BYTE型数组
}
for（k=0；k<len；k++）{
BYTE bt=\*（char\*）（rxdata+k）；
strtemp.Format（“%02X”，bt）；// 将字符以十六进制方式送入临时变量strtemp存放，注意这里应加入一个空隔
m\_str+=strtemp；
//加入接收编辑框对应字符串
}
// 微机控制变频器的参数通过类似COIeVariant的构造函数来实现，COIeVariant 可以接收各种类型的数据（如字符串、整数、浮点数等）并自动将其转换为符合VARIANT要求的数据。然后就可以根据自己的通讯协议要求加入相应的处理代码，实现数据的传递[6]。
…… //在此可以添加通讯协议代码
}
}
3.1.3 数据校验模块
根据约定的通讯协议，对于上位机发送的命令，不符合协议格式（包括数据头，尾，数据长度等）时，下位机将不予应答；对于下位机传送给上位机的数据，处理方式相同。符合协议格式（包括数据头，尾，数据长度等）时，计算校验和，校验和为其前5 个字节数据之和。校验正确，则进入数据处理模块，否则，丢弃不予处理。
3.1.4 数据处理模块
此模块包括数据解包，数据提取，数据转换，如将得到的十六进制的频率数据，以十进制数据方式上传至上位机显示。
此外还有等待超时处理模块，数据传输错误处理模块等。
3.2 通讯协议编程实现
变频器与微机进行通讯时，首先，作为下位机的发送方先将要发送的数据按协议规定的格式组成数据包，并求出检验和，然后连同包头和包尾一同发送出去。作为上位机的接收方收到数据后，通过软件实现通讯协议，主要流程如图3所示。
在软件实现通讯协议的过程中，处理和校验数据时，注意要将接收的数据格式转换为合适的数据格式，否则在调试的时候就会出现错误，或者在上位机的控制界面上不能正确显示。
3.3 界面实现
该软件主要由静态数据区和动态数据区两部分组成。静态数据区包括串口参数的选择和参数数据的显示，并可选择和修改参数命令；由于该变频器采用应答式通讯，所以在动态数据区中，显示下位机传送的数据，同时，在静态工作区可以导入导出参数；
在动态工作区，可以根据需要，选择是否进行定时发送测量命令，从而实时监控下位机的工作状况。静态数据区和动态数据区两部分相互配合，实现对下位机的完整控制。如图4所示。



4 结语
采用Visual C++环境下的MSComm控件对变频器进行软件编程，通过变频器的RS485串口通讯功能，可以实时监控和调整下位机的工作状态，实现对下位机的集中控制。软件界面友好，操作方便，具有较强的实用性。

