用 ActiveX 控件实现 CDT 规约*

陈新之, 傅蕾

(东南大学电气工程系, 210096, 南京)

[摘要]　介绍了用 VC 设计 ActiveX 控件以实现对 CDT 规约处理的方法, 首先分析了 CDT 远动规约的特点, 接着详细介绍了如何根据 CDT 规约的特点进行软件设计, 完整给出了用 MSComm 控件和 VC 设计控件处理 CDT 规约的方法, 并且介绍如何使用多线程实现该通信控件.

[关键词]　VC, ActiveX, CDT 远动规约, MSComm, 多线程

[中图分类号] TM734;　[文献标识码] B;　[文章编号] 1672-1292(2002)03-0046-04

0　引言

循环远动规约(CDT)是实现电力系统调度自动化的一个重要规约, 本文介绍的 ActiveX 控件是一个用在变电站综合自动化系统监控软件里的控件, 该控件的功能是接收主程序送来的数据, 并将数据按照 CDT 规约的要求组织好, 等到调度端拨通电话后, 将数据发送给调度端. 文中详细介绍了如何根据 CDT 规约的特点构造软件, 以及用多线程提高控件工作效率的方法.

1　规约简介

规约根据要传送的数据的类型和重要性不同, 分为: 重要遥测(A 帧)、次要遥测(B 帧)、一般遥测(C 帧)、遥信状态(D1 帧)、电能脉冲数值(D2 帧)、事件顺序记录(E 帧). 数据以帧格式循环发送, 帧格式如图 1 所示.

同步字	控制字	信息字 1	…	信息字 n	同步字	…

图 1　帧结构

同步字有 6 个字节, 是重复 3 遍 EB90H. 控制字也是 6 个字节, 其中最后一个字节为 CRC 检验码. 每个信息字也是由 6 个字节组成的.

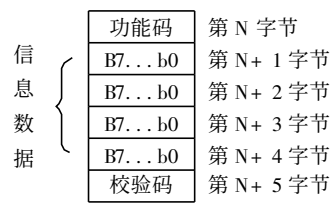

图 2　信息字结构

2　规约分析与软件结构设计

2.1　ActiveX 控件设计

一个 ActiveX 控件由它的一些成员组成: 属性, 方法和事件. 本控件用 MSComm 控件实现对 Modem 的控制, 而使用哪个串口需要根据情况设定, 因此串口号作为控件的一个属性由调用控件的程序修改; 用 Initial 方法实现控件初始化, 用 StopCtrl 方法停止控件, 用 SetData 方法将数据发送给控件.

2.2　分别用数组存放帧数据

每个遥测信息字传送两路遥测信息, 每个遥信信息字传送 32 个状态信息, 每个电能脉冲计数值信息字传送一个电度值. 针对这 3 种格式不同的信息, 要分开处理, 而每个信息字都是 6 个字节, 所以确定

* 收稿日期: 2002-01-28.
作者简介: 陈新之, 1976-, 东南大学电气工程系硕士研究生, 主要从事变电站综合自动化研究.

处理的基本单元是信息字. 为了能使帧发送顺序根据需要调节, 各帧要分别存放. 因此使用 6 个数组分别存放 A、B、C、D1、D2、E 帧的内容. 每个数组的格式依照信息字的格式, 6 个字节一个单元. 在控件初始化时, 首先将数组的内容置为 16 进制的 FF, 用 FF 填充所有单元, 因为在遥测帧中, 一个遥测数据占两个字节, 如果高字节的最高位为 1, 说明该数据无效, 而电能脉冲信息字, 用 4 个字节表示一个值, 最高字节的最高位为 1 时, 表示该信息字无效. 可见在初始化时将数组中填写 FF 可以保证即使控件没有将要发送的数据完全接收到, 而调度端要求传送数据时, 控件发送给调度端的数据不用再次设置是否有效.

2.3 通过数组实现帧类型和帧编号的查询

根据监控软件送来的数据, 要将其按照帧类型和在该帧中的顺序放置到相应数组中. 程序中首先建立了一个枚举类型, 该类型用来表示帧的类型. 接着创建一个结构体, 该结构体有两个成员, 一个是帧类型, 另外一个表示该数据在帧中的编号. 其中 AFrame~ EFrame 表示 A 帧到 E 帧, 而 WFrame 表示错误帧, 即该帧不存在或值不需要处理. 由于采集系统送给该控件的数据是由管理机号、测控单元编号、测量编号和测量数据组成的, 管理机号、测控单元编号和测量编号可以唯一地确定测量值的意义. 根据这一特点, 创建一个结构体数组 Index, 该数组是一个 3 维数组, 3 个维分别为: 管理机号、测控单元编号和测量编号, 从该数组就可以找到一个测量量应该放置到哪一个帧, 是第几个量. 在从主程序获得数据后, 就按照帧的类型和该数据在帧中的编号, 将数据放置到相应数组(下文中称为帧数组) .

```
enum FrameType
    {AFrame,
     BFrame,          struct FrameInfo
     CFrame,              {
     D1Frame,                 FrameType FrmTyp;
     D2Frame,                 BYTE Number;
     EFrame,              };
     WFrame
    };
```

2.4 数据发送设计

数据发送中, 各帧是连续发送的, 所以采用将所有帧放入一个大数组中, 然后再一起发送的方法. 在将各帧数据放入发送数组之前, 先写入同步字和控制字, 然后将帧数据复制到数组, 不断重复该过程, 直到各帧的数据都写完. 整个数据处理流程如图 3 所示.

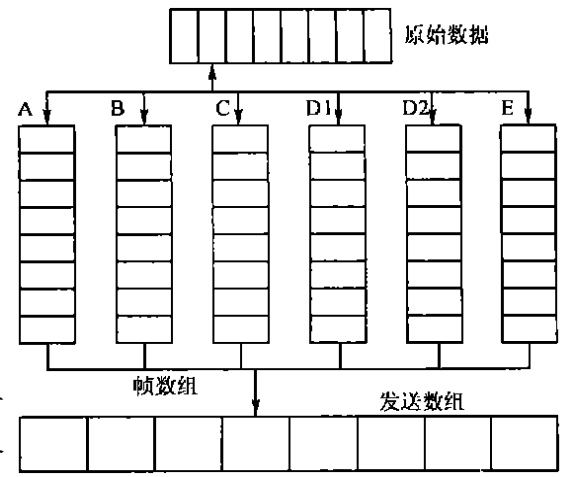

图 3 数据处理图

3 多线程设计

控件中使用了两个辅助线程(Worker Thread) [1] 分别来完成处理主程序送来的数据和向调度发送数据的工作. DealDataThread 线程处理从主程序来的数据, 将数据放置到对应的帧数组, 而 DealOnComThread 线程响应 OnComm 事件, 实现数据的发送.

3.1 线程对共享数据的访问

DealDataThread 线程要将主程序送来的数据放到帧数组, 而 DealOnComThread 要将帧数组的数据放到发送数组中为了实现对共享数据(帧数组) 访问的同步和互斥, 可以使用 CMutex 和 CSingleLock 这两个类[2], 也可以通过设置标志量来实现. 本控件中只有两个线程访问帧数组, 为方便起见, 使用标志量实现同步与互斥。

3.2 线程的创建和结束

一个线程在执行完毕并退出后, 它原本占用的资源就会由系统收回. 但是, 如果不断创建线程然后

再结束线程就会使可用的整块内存变小,影响运行效率.而主程序在一秒钟内,会向控件写数据十几次.所以本控件中采用在控件开始时建立线程,然后通过设置事件来启动线程,在主程序结束时,设置标志让线程退出.下面以 DealDataThread 为例说明设计方法.

```
HANDLE g_hSetData;
……
g_hSetData= CreateEvent(NULL,TRUE, FALSE,NULL);
ResetEvent(g_hSetData);
……
UINT DealDataThread(LPVOID pParam)
{    CCommToUPCtrl * Thread = (CCommToUPCtrl* ) pParam;
     while(! Thread- > bCtrlStop)
     {    WaitForSingleObject(g_hSetData, INFINITE);
          if (Thread- > bCtrlStop)
               break;
          Thread- > DispatchData(Thread- > YCorYX);
          ResetEvent(g_hSetData);
          Thread- > SetDealData(false);
     }
     return 0;
}
```

首先,创建一个 g_hSetData 事件,并复位该事件(即设置该事件未发生),然后创建 DealDataThread 线程,该线程有一个循环,当停止标志 bCtrlStop 为 false 时,线程一直在循环,在循环中用 WaitForSingleObject(g_hSetData, INFINITE)等待事件的发生,如果事件不发生,线程就一直处在等待状态,当主程序向控件写入数据时,程序用 SetEvent(g_hSetData)设置事件,这时控件就开始调用 DispatchData 函数处理数据,在数据处理完毕后,再次复位 g_hSetData 事件,这样,线程又可以进入等待.当程序要退出时,将标志 bCtrlStop 置为 true,然后设置 g_hSetData,线程又启动,当发现 bCtrlStop 为 true,就结束线程.DealOnComThread 线程也同样设计.这样程序就可以安全退出了.

4 防错

程序的健壮性直接影响系统的可靠性.在操作 Modem 的过程中,通过以下一些防错方法,使系统可靠性得到了提高.

4.1 AA 灯熄灭问题

在程序运行当中,遇到的一个问题就是 Modem 从自动应答状态退出了,Modem 从自动应答状态退出后,当 CD 信号发生变化时,Modem 就无法自动响应并转入发送了.直接的反映就是 Modem 的 AA(Auto Answer)灯不亮,为了解决该问题,查阅了 Modem 手册,通过 Windows 的附件超级终端,将自动应答的设置直接写入 Modem,这样设置就不会意外被修改了.

4.2 不响应问题

在实际运行当中,由于调度端的软件没有超时处理功能,当调度和变电站通过 Modem 连上后,就一直等待变电站发送数据.所以如果变电站的监控程序不响应,调度端的程序就不会再向其他变电站要数据了.经过测试,发现是由于在接通后 CD 信号变化后 MSComm 控件没有产生 OnComm 事件,于是采用两个办法:①在处理 OnComm 事件的 Receive 项中,加入判断有没有收到 CONNECT 字符串,因为 Modem 连上后,会有 CONNECT 字符串出现在接收缓冲区中,通过判断在接收缓冲区中有没有该字符

串, 就可以知道 Modem 是否已经连上, 如果连上就可发送数据; ②在控件的主线程中, 加入一个定时器, 时限为 65 s, Modem 从收到信号到连接建立, 大约需要 30 s, 发送数据再等待调度挂断, 最多 10 s, 总共 40 s, 这里用一个比需要的时间长的时限, 以保证可靠. 在定时到达时, 程序检查 CDHolding( ) 信号, 如果连续两次发现该信号为 true, 就认为发生了异常, 接着就将 Modem 挂断, 以保证调度程序能继续运行.

4.3 Modem 返回码问题

当 Modem 处于命令状态时, 每当 PC 机发送一条 AT 命令后, Modem 至少返回一个结果码, 以指示当前命令是否正确执行以及执行结果. 结果码有两种形式: 一种是文本信息(即字符串形式), 另一种是数字码. 虽然可以通过 AT 指令中的 V 命令来设置是采用文本信息还是数字码, 但在调试中发现, 有时候即使设置是返回文本信息, 但实际上收到的还是数字码. 如果仅仅通过在接收缓冲区中寻找 CONNECT 字符串就不能完全正确响应, 所以在程序中增加判断返回码是否为 1(对应于字符串 CONNECT), 是否是大于 10 的数字(10 以上的数字表示分别以不同速率连接), 如果是, 也要进入数据发送.

5 实际运行情况

本控件是在变电站综合自动化系统监控软件投入运行半年后设计的, 添加该控件后, 主程序只做了很少的修改, 包括启动和停止控件、向控件写入数据等操作, 说明本控件有很好的独立性, 发送、挂断电话等操作都独立完成, 不需要主程序控制, 效率较高. 传输的可靠性通过 CDT 规约中规定的 CRC 校验码来保证, 当调度端发现 CRC 校验出错, 会将收到的数据丢弃, 然后再次向变电站要求发送数据, 该过程最多进行 3 次, 如果 3 次数据还没有正确, 就跳到下一个变电站. 由于调度不要求变电站将事件主动上发, 所以控件一直工作在应答状态, 调度软件大约每 10 min 向变电站要一次数据. 控件共发送 A 帧数据 23 个, B 帧数据 32 个, C 帧数据 25 个, 开关量 43 个, 电度量 38 个. 本控件从 2001 年 12 月底开始运行, 到现在(2002 年 2 月) 运行良好, 能正确将遥测、遥信、电度等量上传(经调度和变电站核对, 数据无误), 数次电容器保护动作, 控件也能将开关信息正确传送给调度.

6 结论

本控件依据 CDT 规约的特点设计了合理的数据存储结构, 又合理地安排了线程的启动和停止, 并通过一些方法防止错误发生, 使控件能够高效工作, 现场运行良好.

[参考文献]

[1] David J Kruglinski, Scot Wingo, George Shepherd. Visual C++ 6.0 技术内幕[M]. 希望图书创作室译. 第 5 版. 北京: 北京希望电子出版社, 1999, 261~ 264.

[2] 李予州, 杨宛辉, 许珉, 等. 部颁 CDT 循环规约的 VC+ + 程序实现[J]. 继电器, 2001, 29(4) : 32~ 35.

Design of ActiveX Control to Implement CDT Protocol

Chen Xinzhi, Fu Lei

(Electrical Engineering Department, Southeast University, 210096, Nanjing, PRC)

Abstract: This paper introduces a method of designing an ActiveX control in VC to implement CDT protocol. At first, this paper analyses the characteristics of CDT protocol. Then the paper discusses how to design software according to these characteristics. At last, this paper gives a complete method of designing an ActiveX control to implement CDT protocol by using MS Comm control and VC.

Key words: VC, ActiveX, CDT protocol, MSComm, Multithread

[责任编辑: 刘健]