组态软件中数据采集模块设备无关性的设计与实现

陈 姝 中南林学院研究生部(412006)

Abstract

This paper introduce a method to design and realization of data collection module which is independent of device in configuration software.It is explained following the analysis of the communication mode of a great variety of control devices,and specify how to program driver in the environment of Visual C++ 6.0.

Keywords:configuration software,data collection,PLC,OPC

摘 要

本文在分析各种控制设备通讯方式的基础上给出组态软件中数据采集模块设备无关性的设计及实现形式,并讲述了如何在 Visual C++ 6.0 环境下编写驱动程序。

关键词:组态软件,数据采集,PLC,OPC

在开发传统的工业控制软件时，当工业被控对象一旦有变动,就必须修改其控制系统的源程序,导致其开发周期长。通用工业自动化组态软件的出现为解决上述实际工程问题提供了一种崭新的方法，而数据采集模块的稳定性及通用性关系到组态软件的整体性能。本文结合作者在实际组态软件中的开发经验给出了一种数据采集模块设备无关性的设计方法。

1 各种控制设备通讯方式介绍

目前底层控制设备中具有数据通讯功能的基本上有以下几种:DCS、PLC、智能模块、智能仪表、板卡、变频器等。虽然各种设备无论从功能上、体系结构上都不尽相同，但是其对外数据接口无非以下几类:OPC、DDE、以太网通讯、现场总线通讯、串口通讯、通过扩展卡进行通讯。

OPC 是工业自动化中的一种软件接口,它是由世界上各大控制设备厂商发起,OPC Foundation 负责协议制定,而最终由 Windows 实现,OPC 应用程序分为服务器程序及客户端程序两种，支持网络及本地数据交换,一般自控设备厂商会提供服务器程序,用户需开发客户程序与服务器程序建立连接，进行实时数据采集；有关 OPC 的详细资料请访问 OPC 基金会网站:www.opcfoundation.org;DDE 也是软件接口的一件,DDE 程序分为 DDE 服务器及 DDE 客户端两种,服务器与客户端通过程序名、话题名、项目名一一对应进行数据通讯,NETDDE 支持网络数据交换；以太网通讯方式就是控制设备支持以太网通讯，该设备既负责现场数据采集同时又可以接入以太网，由于该设备一般处于工业总线网中,实际上起到了网关的功能,使得底层控制网络与上层管理网络可以进行数据交换,用户程序通过 TCP/IP 协议可以与该设备进行通讯,从而采集到现场实时数据；现场总线通讯需在上位计算机的扩展槽中插入一块现场总线卡（如 CAN 卡,LONWORKS 卡等),并且将该上位机连入现场总线网中,做为该网的一个节点,上位机按照一定的协议采用轮转方式对各个节点的实时数据进行采集；串口通讯就是利用上位机的 RS-232 或 RS-485 口按照某种协议与现场设备进行通讯,该通讯方式速度较慢,适用于数据实时性要求不高的场合，一般通讯协议为应答式报文,即由一方发出命令,而由另一方进行响应;通过扩展卡进行通讯是板卡通讯的典型方式，该通讯方式适用于数据量较少的情况下，一般设备厂商会提供 DLL 供用户调用,若没有 DLL,通常会给出数据端口地址、控制端口地址、以及控制字格式,用户程序需对端口进行操作,通讯方式采用中断。

2 实现方式

2.1 数据抽象

数据抽象共分为三个层次的抽象：设备层的数据抽象,用户层的数据抽象,程序接口层的数据抽象。

2.1.1 设备层的数据抽象

现场数据共有两种类型:模拟量与数字量,并且对应数据可能具有控制功能，因而将设备层的数据抽象为以下几种类型:8 位数字寄存器,16 位数字寄存器,32 位数字寄存器,非寄存器数字输入,非寄存器数字输出,非寄存器数字既能输入又能输出,模拟寄存器,非寄存器模拟输入,非寄存器模拟输出。8 位数字寄存器表示现场采集的数据为一个字节，该字节中的每一位对应一个开关量(开或关),同时对应开关量具有控制功能，用户可以对该开关量对应的位进行置位来控制开关的通断;16 位数字寄存器和 32 位数字寄存器同 8 位数字寄存器类同,只是分别表示 2 个和 4 个字节,对应开关量数量不同而已;非寄存器数字输入表示通过置位可以用来控制一个现场开关的通断，该抽象数据不具有数据输入功能；非寄存器数字输出表示该数据对应一个现场开关的状态,不具有控制功能;非寄存器数字既能输入又能输出表示该数据既具有输入功能又具有控制输出功能,但只是对应一个开关量而已；

模拟寄存器表示该数据对应输入值为现场的一个模拟量(如流量,压力,温度等),同时可以对该数据进行赋值控制输出;非寄存器模拟输入表示该数据对应一个现场模拟量的输入值;非寄存器模拟输出表示对该数据进行赋值可以控制一个现场模拟量的输出。

2.1.2 用户层的数据抽象

用户层的数据抽象就是从用户开发一个监控系统的角度看待现场数据的抽象。由于开发一个监控系统也就是对数据进行显示、控制、管理的全过程,因而与数据采集模块相关的用户层的数据抽象为以下几种类型:AA(模拟量报警)、AI(模拟量输入)、AO(模拟量输出)、AT(模拟量读写)、DA(数字量报警)、DI(数字量输入)、DO(数字量输出)、DT(数字量读写)、TX(数据流)。还有一些数据抽象如:CAL(数据计算)、PRG(程序)等属于中间点,与数据采集模块不直接相关,这里就不一一列举了。AA 点的数据结构定义如下:

```
struct AA
{
    BOOL    m_bDRWritten;//数据采集驱动程序是否已写,如果 Net 读取后改为 F
    BOOL    m_bIntialize;//是否已经初始化
    double m_Value;//模拟量原始值 根据值域使用原则为 m_Value[0],m_Value[1],...
    char    m_DeviceName[DRIVE_NAME_LEN];//设备名
    short   m_AddressNo;//对应序号
    short   m_SignalNo;//信号条件
    short   m_HardWareNo;//硬件条件
    char    m_Name[POINT_NAME_LENGTH];//点名
};
```

2.1.3 程序接口层的数据抽象

由于目前的组态软件中的驱动程序一般采用 DLL 调用方式,因而程序接口层的数据抽象也就 DLL 接口函数的定义。驱动程序与监控软件之间的通讯一般采用共享内存的方式,所以驱动程序 DLL 中必须有一个接口函数 BOOL InputMapAddress()用来输入共享内存的地址,同时必须提供一个函数 BOOL Start()供主控程序调用将 DLL 投入运行。配置 DLL 中只需提供一个统一的接口函数 BOOL SetConfig()供用户设置控制设备的参数用。

2.2 现场数据抽象与用户数据抽象之间一一对应关系建立方法

虽然上面就数据的一些共性做了具体的数据抽象,但如果不能建立现场数据抽象与用户数据抽象之间的联系,则驱动程序采集上来的数据监控软件还是无法进行显示,报警和管理。数据流图如图 1。

用户配置 DLL 提供给用户配置设备的一些常用参数,如 C200H 中的节点号,串口号,内存中配置字数,在内存单元中是数字量在前还是模拟量在前等;虽然各配置界面不一样,但都必须有设备逻辑名。图 2 是 Modbus 的配置 DLL 显示的界面。

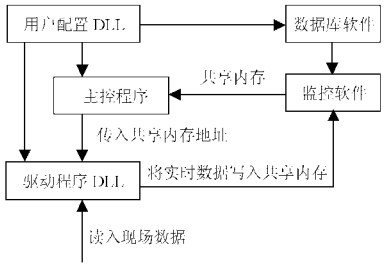

图 1 数据流图

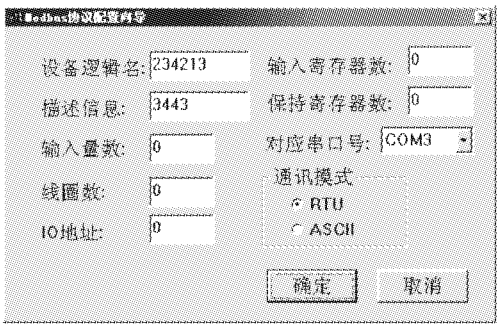

图 2 Modbus 用户配置画面

该配置 DLL 存盘后会生成三个文件,一个文件保存的是用户配置好的设备的参数,传入驱动程序 DLL 使用;另一个文件保存的是该工程中共配置了哪几种设备,该文件传给主控程序,由主控程序根据文件内容调入相应的驱动程序 DLL 进行数据采集;还有一个文件保存的是在已配置的设备中,每种设备有哪几类设备层的数据抽象(也就是前面所说的 8 位数字寄存器等),以及数据抽象的名字,该文件传给数据库程序,供数据库程序在配置数据库点(也就是前面所说的 AI 等)时使用。现场数据抽象与用户数据抽象之间一一对应关系就是在数据库中通过配置建立起来的,即数据库点(DA 等)的数据来源为哪个现场设备的哪个点(8 位数字寄存器等),这个可以从上面的 AA 点的数据结构看出。数据库配置好后会形成一个数据库文件,传给监控软件,监控软件根据数据库点的类型及数量在内存中开辟出一块共享内存空间,然后调入主控程序运行。主控程序初始化时打开共享内存,同时根据用户配置 DLL 生成的存盘文件将相应设备的驱动程序 DLL 调入运行,在调入运行前先传入共享内存的地址及该设备的设备逻辑名。设备驱动 DLL 运行后将共享内存中的各数据点中的 m_DeviceName 与本 DLL 的设备逻辑名相比较,若相同则表示该数据点的数据对应本设备中的一个设备层的数据抽象(8 位数字寄存器等),并且每个数据点有一个序号 m_AddressNo,该序号是按一定顺序排列,每一个序号对应一个相应的设备层的数据抽象(8 位数字寄存器等),因此设备驱动 DLL 采集上来现场数据后就会将该数据传入共享内存中相应数据库点(AA 等),从而建立了现场数据抽象与用户数据抽象之间一一对应关系。

2.3 在 Visual C++ 6.0 中编写设备驱动 DLL

(下转第 40 页)

车道引导板安装在ETC车道的龙门架等处，表示该车道的状况，引导车辆。路侧显示器在入口以驶入车辆为对象，对于非ETC车辆显示“禁止通行”，在出口对于驶出的车辆显示“费额”或“是否可通行”的显示器。自动栏杆为了确实保证收费以及信息处理等，通过自动栏杆对车辆进行交通控制。抓拍摄像机监视ETC车道状态并进行违章抓拍。车道天线是为了进行ETC处理，与安装在车辆上的电子标签进行通信的微波信号收发装置。ETC车道控制器为进行ETC收费计算处理、管理路侧设备的计算机，判别驶入车辆是否为ETC车辆，进行车型判别(入口)、费额计算(出口)、违章检测(出入口)，以及控制各个路侧设备，进行ETC收费站服务器的数据通信。ETC车道监视控制器监视ETC设备状况，车道引导板的手动控制，通过对讲电话进行通话，栏杆的手动控制等设备状况以及实施手动辅助操作。

对于由非ETC车道驶出的ETC车辆、异常ETC车辆等通过IC卡读写器实现无现金的收费处理。车载电子标签为安装在车辆仪表台上或挡风玻璃上的记录有车辆信息的微波装置。插入IC卡后在IC卡与路侧微波设备间进行必要的信息中继传输。IC卡为记录用户的合约信息等的卡片，插入车载电子标签中使用。内部有集成电路进行数据处理、传输、加解密操作。

4 组合式收费技术方案的适用条件及特点

组合式收费技术方案要求IC卡收费系统必须组成一个封闭的网络，支持持通行卡现金付费和持预付卡电子付费两大类业务。在新建系统中实现上述功能比较容易，在已建系统中，需要新增预付卡电子付费业务，有一定的工作量，但不存在重复投资或浪费的问题。在这样一个以IC收费为基础的系统中开展ETC电子付费就非常容易，仅需在交通拥堵的收费站或有特别需求的收费站安装ETC车道系统就可以，没有必要让ETC系统自身单独构成一个封闭式网络，涉及的主要是车道改造任务。组合式收费技术方案特点如下：

1)该方案集中了IC卡收费系统和ETC收费系统的优点，两者互为补充，具有支付手段丰富、交费快捷、通行能力强等特点。

2)IC卡系统作为ETC系统的备份支付系统，使得系统可靠性大大提高，不必再准备手持式ETC读写器进行应急操作。尤其是在ETC系统出现故障时，ETC车道的备用IC卡支付系统将启动运行，解决ETC车辆的通行问题。

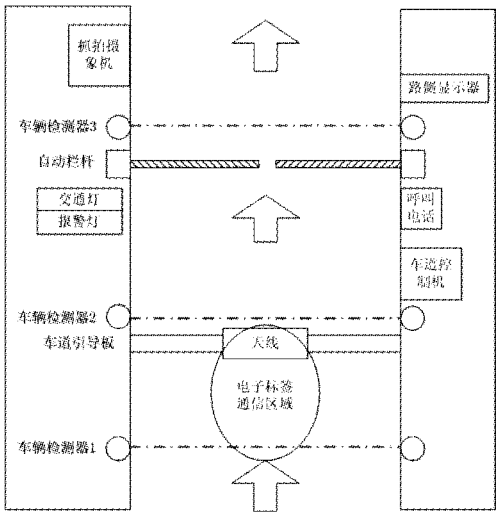

图1 ETC入口车道设备布局示意图

3) 由于采用高安全性的CPU卡作为存储介质并采用规范的双向认证技术，保证为联网收费系统开展预付卡业务提供了安全、可靠的解决途径。

4) 该方案为解决中心城市的环城公路网及区域经济圈内城间公路网的交通瓶颈提供了有效手段；ETC车道可以视实际需求进行逐步扩展，资金投入及系统规模富有弹性、易于试点和推广，运营风险大大降低。

5)该方案符合中国金融卡规范及电子商务等国家鼓励发展方向，为将来在公路服务领域开展扩展支付奠定基础。

参考文献

1 宋燕铭. 高速公路不停车收费系统的发展前景和解决方案[J].湖南交通科技，2000，26(4)：60~61

[收稿日期：2003.5.21]

(上接第34页)

在主控程序调用DLL采用动态调用，即利用函数进行LoadLibrary进行调用；在利用MFC AppWizard(dll)编写DLL时，在DLL类型的选项中需选中MFC扩充DLL，接口函数的导出定义在.def文件，格式如下：函数名 @num，num为一序列号，调用程序可以用函数名或序列号来调用导出函数；前面讲到在设备驱动DLL中必须导出两个函数接口，其函数原型分别为：extern "C" BOOL WINAPI Start (int Sort)，extern "C" BOOL WINAPI InputMapAddress(GatherVarDefInf *pGarthVar,AA *pAA,AI *pAI,AO*pAO,AR *pAR,DA *pDA,DI *pDI,DO *pDO,DR *pDR,CString DeviceName)；在设备驱动DLL中需建立一个或多个线程进行数据采集，这里就不赘述了。

3 结束语

本文介绍的组态软件中数据采集模块无关性的设计思路经过现场检验证明是可行的。作者目前利用这种方法已开发出多个厂家的设备驱动程序，包括：西门子公司的S5，S7系列PLC的驱动程序，欧姆龙公司的C200系列的驱动，研华公司的ADAM智能模块的驱动程序，中泰板卡的驱动以及OPC客户端驱动等。

参考文献

1 MSDN，Microsoft公司，1995 [收稿日期：2003.5.28]