OPC Server 在 VxWorks 下的应用研究

赵启升[1],李存华[1],刘小明[2]

(1. 淮海工学院计算机工程学院,江苏 连云港 222005)
(2. 淮海工学院信息中心,江苏 连云港 222005)

[**摘要**] 随着工业生产的不断发展,工业控制软件日益复杂. 传统的基于设备驱动程序的数据传输方式已经不能满足现实的需要,OPC 标准通信协议应运而生. 本文提出了一种在嵌入式实时操作系统 VxWorks 下开发 OPC 服务器应用软件的方法,并开发了一款遵循 OPC 数据存取规范的 OPC 服务器软件. 该 OPC 服务器基于 VxWorks 提供的 VxDCOM 组件,相比基于 Windows COM/DCOM 的 OPC 服务器软件具有更高的稳定性和实时性. 实验证明,该 OPC 服务器可以与 Windows 下或 VxWorks 下开发的的 OPC 客户端软件进行通信,并在功能与性能上都有不错的表现.

[**关键词**] OPC,VxWorks,VxDCOM,工业控制
[**中图分类号**]TP3 [**文献标志码**]A [**文章编号**]1672-1292(2014)04-0066-05

An Implementation Research of OPC Server Under VxWorks

Zhao Qisheng[1], Li Cunhua[1], Liu Xiaoming[2]

(1. School of Computer Engineering, Huaihai Institute of Technology, Lianyungang 222005, China)
(2. Information Centre, Huaihai Institute of Technology, Lianyungang 222005, China)

Abstract: With the continuous development of industrial production, industrial control software becomes more and more complex. The traditional way of data transmission which is based on device driver can no longer meet the real needs, thus leads to the birth of the OPC—the standard communication protocol. This paper presents a method in developing OPC Server applications under VxWorks, a real-time embedded operating system and develops an OPC Server application following OPC Data Access Specification. This OPC Server is based on the VxDCOM component which is provided by VxWorks. Compared with those based on the Windows COM/DCOM, our Server has more stability and real-time property. Experiments show that the OPC server can communicate with standard OPC Clients under Windows or VxWorks operating system, and do well in both functionality and performance security.

Key words: OPC, VxWorks, VxDCOM, industrial control

随着工业生产的不断发展,以及工业自动化与信息技术的紧密结合,工业控制软件不断取得进步. 但是,生产规模的不断扩大和过程复杂度的不断提高,给工业控制软件的研发带来了巨大的难度. 在传统的控制系统中,控制系统软件与现场智能设备之间的信息交换是通过驱动程序来实现的,驱动程序的特殊性导致了控制系统间的互联异常困难.

OPC 即“用于过程控制的对象链接与嵌入技术”,它以 COM/DCOM 技术为基础,为工业自动化软件的开发提供了统一的标准[1,2]. 采用该标准以后,硬件厂商为其生产的硬件编写 OPC 服务器软件,应用软件厂商可以不用考虑不同硬件驱动程序的差异,开发统一的符合 OPC 规范的客户端软件,该软件能够对一切支持 OPC 标准接口的服务器进行数据交互. 目前国内市场上的 OPC 服务器软件大多基于 Windows 操作系统,而 Windows 操作系统固有的缺点,如稳定性比起 Linux、Vxworks 等系统不尽如人意,必须基于 X86 平台,导致整个系统的臃肿、庞大,稳定性低. 本文针对 Windows 平台下 OPC 服务器的这些缺点,讨论了在 VxWorks 下开发 OPC 服务器应用软件的方法.

收稿日期:2014-08-16.
基金项目:国家自然科学基金(61103017)、江苏省教育厅教育改革课题(2013JSJG124)、连云港市社会发展项目(SH1212)、连云港市科技公关项目(CG1215).
通讯联系人:赵启升,副教授,高级工程师,研究方向:信息安全及计算机应用. E-mail:17988417@qq.com

1 相关工作

OPC 基金会是制定 OPC 通信协议标准的国际组织,目前 OPC 基金会已经推出多种 OPC 规范,其中,OPC 数据存取规范是使用最为广泛,也是 OPC 协议体系架构中最为重要的规范.

1.1 OPC 本质——Microsoft COM/DCOM

OPC 技术本质是采用了微软的 COM/DCOM 技术,COM 主要是为了实现软件复用和互操作,并且为基于 Windows 的程序提供了统一的、可扩充的、面向对象的通讯协议[3]. 一个 OPC 数据存取服务器就是一个 COM 组件,实现了 OPC 规范中定义的标准接口. 由于 COM 的平台无关性,理论上可以在任何平台上实现 COM. 但是由于特定的原因,目前 COM 技术仍然以 Windows 操作系统为主,在非 Windows 操作系统上开发 OPC 软件具有极大的难度. 在 VxWorks 中提供了 VxDCOM 组件,使得 VxWorks 下的 OPC 软件开发成为可能. VxWorks 的实时性、稳定性,以及它在工业控制领域的广泛应用使得其与 OPC 的结合具有无限的发展空间和应用前景.

1.2 OPC 服务器结构

一个 OPC 服务器由 3 个对象组成:服务器对象、组对象、项对象[4].

(1)OPC 服务器对象:一个 OPC 服务器对象就是一个 COM 对象,OPC 服务器对象用来提供关于服务器自身的相关信息,并作为 OPC 组对象的容器.

(2)OPC 组对象:OPC 组对象为 OPC 服务器对象所包含,OPC 组对象用来提供关于组对象自身的有关信息,并提供组织和管理数据的方法. 组对象分为两种:公共组对象和私有组对象. 公共组由多个客户共享,而私有组只隶属于一个客户程序.

(3)OPC 项对象:OPC 项对象为 OPC 组对象所包含,一个 OPC 项对象代表了 OPC 服务器到数据源的一个物理连接. 一个 OPC 项不能被 OPC 客户程序直接访问,因此在 OPC 规范中没有对应于项的 COM 接口,所有与项的访问需要通过包含项的 OPC 组对象来实现. 每一个 OPC 项对象包含 3 个属性:值、品质、时间戳,值以 VARIANT 形式表示.

2 OPC 服务器设计与实现

2.1 利用 Microsoft ATL 简化类厂

在 Windows 平台上开发 COM 组件一般使用微软 ATL(活动模板库)进行,使用 ATL 进行 OPC 服务器开发能够极大减小开发难度. 在 VxWorks 开发环境中,可以通过提取 ATL 部分代码,封装 COM 类厂创建过程、IUnknown 函数接口等实现一系列复杂的 COM 特性.

2.2 代理存根代码的生成

OPC 基金会提供了两个 OPC DA 2.05 规范标准接口配套的 IDL 文件:opccomn. idl 和 opcda. idl. 利用 TornadoII 提供的 widl. exe 工具可以生成 opccomn_i. c、opccomn_ps. c、opcda_i. c、opcda_ps. c、opccomn. h、opcda. h,这些文件形成了 OPC 的代理存根模块. 在建立 Tornado 工程时,需要将这些文件包含进工程中一起编译.

2.3 OPC 对象及其接口实现

(1)OPC 服务器对象实现. OPC 服务器对象是 OPC 服务器中最为重要的对象. 客户端通过访问服务器对象的标准接口函数与之交互[5]. 在本 OPC 服务器中定义服务器类 xopc_server,该类继承 IOPCServer,IOPCItemProperties,IOPCBrowseServerAddressSpace 3 个重要的接口,并逐一实现其函数.

(2)OPC 组对象实现. 组对象用于组织管理服务器内部的实时数据信息,它是 OPC 项对象的集合. 在本服务器中定义组对象类 xopc_group,该类继承 IOPCGroupStateMgt、IOPCItemMgt、IOPCSyncIO、IOPCAsyncIO2、IConnectionPoint、IConnectionPointContainer 等接口并实现其接口函数. 在实现 OPC 组对象时应注意以下几点:

①在实现 IConnectionPoint 和 IConnectionPointContainer 接口时,我们采用 ATL 的做法,在类体中使用以下宏定义映射连接点对象,其中,IID_IOPCDataCallback 为客户端回调接口,OPC 组对象通过该接口完成 OPC 异步数据传输.

BEGIN_CONNECTION_POINT(xopc_group)
CONNECTION_POINT_ENTRY(IID_IOPCDataCallback)
END_CONNECTION_POINT()

②通常读写硬件比较耗时,当异步读写请求过多时,应采用多线程对请求队列进行处理,以保证高数据吞吐量[6]. 在本 OPC 服务器的实现中,采用了基于消息队列与多线程的异步数据传输模式,如图 1 所示.

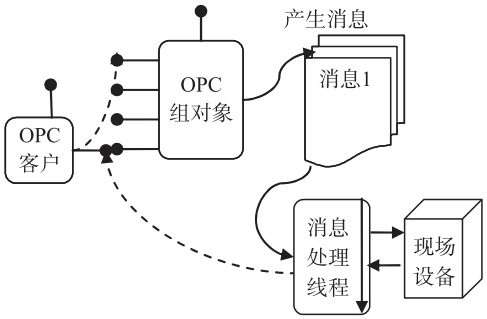

图 1　基于消息队列与多线程的异步数据传输模式

Fig. 1　Asynchronous data transmission mode based on message queue and multi thread

(3)OPC 项对象实现. OPC 规范并没有定义操作 OPC 项对象的标准接口,所有对 OPC 项对象的组织和管理工作都通过 OPC 组对象中定义的 IOPCItemMgt 接口完成,因此,OPC 项对象的实现完全由服务器开发者自己定义. 特别需要注意的是,每一个 OPC 项对象都有一个数据源节点与之对应,该数据源节点为 OPC 项对象的数据值与设备驱动程序的公共缓冲区,设备驱动程序负责从现场设备获得数据写入该缓冲区,或者从该缓冲区获取数据写入现场设备.

2.4　OPC 对象及其接口实现

数据源缓冲区是 OPC 服务器程序中的重要组成部分,一个数据源节点对应一个现场设备节点. 对于每一个数据源,可能有一个或多个 OPC 项对象与之关联,这些项对象可能存在于同一个组对象中,也可能存在于不同的组对象中. 我们可以使用 C++STL 中的 map 容器存储这种关系,数据源节点可以定义为:std::map<xopc_group * ,std::set<xopc_item * >>items.

OPC 服务器导出接口 add_node()函数用于添加数据源节点,每个数据源节点对应一个读写回调函数,该回调函数负责与设备驱动程序产生数据交互. 通过调用 register_callback()函数注册读写回调函数. 当同步/异步读写设备时,该回调函数将被调用.

3　硬件设备驱动

每一个数据源节点都有一个设备驱动程序与其对应. 设备驱动程序负责与硬件设备产生数据交互,并将实时数据传送至 OPC 数据源缓冲中. 在本 OPC 服务器软件中,实现了一个内存读写设备驱动程序和一个蜂鸣器设备驱动程序[7]. 设备驱动程序与数据源节点的关联过程如图 2 所示.

下列伪代码说明了读写回调函数的编写方法:

```
1. int g_beepStatus=0;  //蜂鸣器状态
2. BeepWriteCallback(param){
3.    if(param==1){
4.    Statues=1;
5.    startBeep();  //使能蜂鸣器
6.    set_value();  //修改节点值
7.    g_beepStatus=1;  //设置状态值
8.    }
9.    else if(param==0){
10.   Statues=0;
11.   stopBeep();  //打开蜂鸣器
12.   set_value();  //修改节点值
13.   g_beepStatus=0;  //设置状态值
14.   }
15.   }
16.   BeepReadCallback(param){
17.   Param=g_beepStatus;
18.   set_value();  //修改节点值
19.   }
```

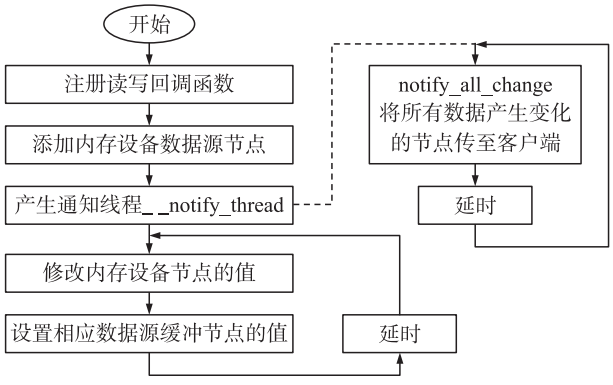

图 2　设备驱动程序与数据源节点的关联过程

Fig. 2　Association of device driver and the data source node

4 测试与结果分析

使用 KepWare 公司的 OPC Quick Client 连接 VxWorks 下的 OPCServer,并添加 OPC 组对象与 OPC 项对象,效果实现. 为测试 OPC 服务器性能,添加 Dev. OpcState. DropPack_Count 模拟数据源标签,用于记录无法及时处理而丢失的数据包个数,通过添加不同的 OPC 项对象个数,测试服务器的回调响应时间和每分钟最大丢包数[8],并与 Windows 端 OPC 服务器(采用第三方 OPC 开发库开发)进行对比,对比结果如图 3、图 4 所示.

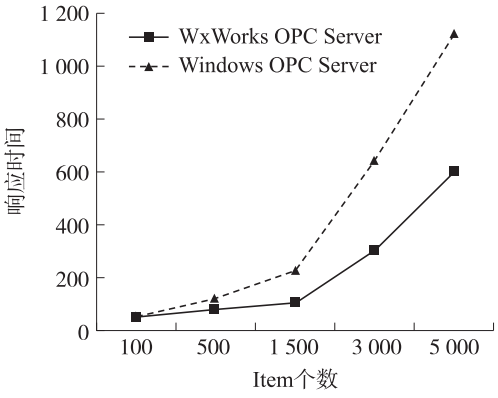

图 3 响应时间曲线对比

Fig. 3 Curve comparison on response time

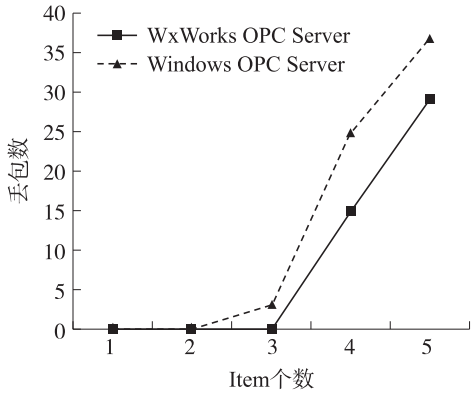

图 4 丢包数对比

Fig. 4 Correlation of the number of lost packets

可以看到,本文描述的基于 VxWorks 的 OPC 服务器在响应时间和平均丢包个数上,都小于基于 Windows 的 OPC 服务器. 在 Item 个数达到 5 000 时该服务器仍能达到 600 ms 的响应时间,能够满足实际应用需要. 在 Item 个数达到 1 500 以上出现了小数据量的丢失情况,亦属于可接受范围之内.

5 结语

本文提出了在嵌入式实时操作系统 VxWorks 下开发 OPC 服务器应用软件的方法,在 VxWorks 下实现了 OPC 数据存取服务器,在功能和性能上都有不错的表现,比基于 Windows 平台的 OPC Server 具有更高的稳定性与实时性;提取 Windows ATL 代码作为开发 COM 组件的工具,代码架构清晰,易于二次开发;突破传统的 OPC 开发模式,具有较高创新性与实用性. 但是,由于 VxDCOM 并不是一个完整的 COM 库实现,并且存在较多 BUG,使得整个系统的稳定性难以得到保证[9]. 因此,在下一步的工作中,需继续深入研究并修复 VxDCOM 的代码,提高底层软件框架的稳定性.

[参考文献](References)

[1] 单寅. 基于 VxWorks 的机载图形显示系统软件研制[D]. 南京:南京航空航天大学通信与信息系统系,2012.
Shan Yin. Software development of airborne graph display system based on VxWorks [D]. Nanjing: Department of Communication and Information System, Nanjing University of Aeronautics and Astronautics, 2012. (in Chinese)

[2] 陶春燕. 基于 VxWorks 的嵌入式实时数据库管理系统设计与实现[D]. 哈尔滨:哈尔滨工程大学控制理论与控制系统系,2012.
Tao Chunyan. Design and realization of embedded real-time database management system based on VxWorks[D]. Harbin: Department of Control Theory and Control Engineering, Harbin Engineering Univeristy, 2012. (in Chinese)

[3] 沈迪. 基于 VxWorks 的可视化文件传输系统的设计与实现[D]. 哈尔滨:哈尔滨工程大学导航、制导与控制系,2012.
Shen Di. The research and implementation for visual file transfer system based on VxWorks [D]. Harbin: Department of Navigation, Guidance and Control, Harbin Engineering Univeristy, 2012. (in Chinese)

[4] 岳潇. 基于 PowerPC 和 VxWorks 的信号处理平台的研究[D]. 沈阳:沈阳航空航天大学信号与信息处理系,2013.
Yue Xiao. Research of signal processing platform based on the power PC and VxWorks[D]. Shenyang: Department of Signal

and Information Processing, Shenyang Aerospace University, 2013. (in Chinese)
[5] 宋晓莉. 基于 VxWorks 的数据通信及控制技术的研究与实现[D]. 西安:西安电子科技大学通信与信息系统系,2009.
Song XiaoLi. Research and implementation of data communications and control technology based on VxWorks[D]. Xi'an: Department of Communication and Information System, Xidian Univerisity, 2009. (in Chinese)
[6] 宋泽帅,王守城,段俊勇,等. 基于 OPC 技术的电能数据采集系统设计[J]. 电气自动化,2013,35(6):9-10,59.
Song Zeshuai, Wang Shoucheng, Duan Junyong, et al. Design of OPC-based power data aquisition system[J]. Electrical Automation, 2013, 35(6): 9-10, 59. (in Chinese)
[7] 田启贲. 基于 VxWorks 的 PXIe 嵌入式控制软件开发[D]. 哈尔滨:哈尔滨工业大学仪器科学与技术系,2013.
Tian Qiben. Software development of PXIe embedded controller based on VxWorks[D]. Harbin: Department of Instrument Science and Technology, Harbin Institute of Technology, 2013. (in Chinese)
[8] 詹俊,龙辛,黄波,等. 基于 VxWorks 的软 PLC 远程监控系统设计与实现[J]. 机械工程与自动化,2013,178(3):3-5,8.
Zhan Jun, Long Xin, Huang Bo, et al. The remote monitor system design and implementation of soft PLC based on VxWorks [J]. Mechanical Engineering and Automation, 2013, 178(3): 3-5, 8. (in Chinese)
[9] Mazinan A H, Kazemi M F. Recent developments on applications of sequential loop closing and diagonal dominance control schemes to industrial multivariable system[J]. Journal of Central South University, 2013, 12: 3 401-3 420.

[责任编辑:顾晓天]

(上接第 65 页)

[参考文献](References)

[1] Li Y, Sun J, Tang C K, et al. Lazy snapping[C]//International Conference on Computer Graphics and Interactive Techniques. New York, 2004: 303-308.
[2] Rother C, Kolmogorov V, Blake A. "GrabCut": interactive foreground extraction using iterated graph cuts[J]. ACM Transactions on Graphics(TOG), 2004, 23(3): 309-314.
[3] Wu Xiaoyu, Wang Yangsheng. Interactive foreground/background segmentation based on graph cut[C]//International Congress on Image and Signal Processing. Sanya, 2008.
[4] Boykov Y, Kolmogorov V. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2004, 26(9): 1 124-1 137.
[5] Boykov Y, Jolly Pi M. Interactive graph cuts for optimal boundary and region segmentation of objects in n-d images[J]. ICCV, 2001(7): 105-112.
[6] Vincent L, Soille P. Watersheds in digital spaces an efficient algorithm based on immersion simulation[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1991, 13(6): 585-598.

[责任编辑:丁　蓉]