OPC技术在智能仓储系统中的应用

吴 晗,成卫青
(南京邮电大学 计算机学院,江苏 南京 210023)

摘 要:随着工业的不断发展,自动化程度也在逐步提高。工业系统中设备数量逐渐增加,动作逻辑不断复杂,但由于PLC(可编程逻辑控制器)本身的局限性,无法对数据进行大规模的存储且不能提供友好的人机交互界面。随着对现场信息交互与共享能力的需求的不断提高,上位机调度系统变得不可缺少。OPC(OLE for process control)协议是专门为工业自动化提供的一个标准,主要用于解决数据源控制系统和数据源之间的数据交互问题。该文主要研究OPC协议在智能仓储系统中的应用。上位机上运行仓库控制系统(WCS),作为OPC的客户端,而OPC服务器设在PLC或上位机上。介绍了WCS和OPC服务器之间3种通信方式:同步、异步和订阅。设计实现了OPC服务器以及OPC接口,实现了WCS和PLC之间的通信,提升了WCS下指令的速度以及智能仓储系统的工作效率。

关键词:OPC;数据通信;可编程逻辑控制器;智能仓储系统;客户端/服务器模型;仓库控制系统
中图分类号:TP393 **文献标识码**:A **文章编号**:1673-629X(2021)07-0158-06
doi:10.3969/j.issn.1673-629X.2021.07.026

Application of OPC Technology in Intelligent Warehousing System

WU Han,CHENG Wei-qing
(School of Computer,Nanjing University of Posts and Telecommunications,Nanjing 210023,China)

Abstract:With the continuous development of industry,the degree of automation is also gradually improving. In the industrial system,the number of devices is gradually increasing,and the action logic is continuously complicated. However,due to the limitations of the PLC (programmable logic controller) itself,data cannot be stored on a large scale and a friendly human-computer interaction interface cannot be provided. With the increasing demand for the ability of field information exchange and sharing,the upper computer scheduling system becomes indispensable. OPC (OLE for process control) protocol is a standard specially provided for industrial automation,and mainly used to solve the data interaction problem between the data source control system and the data source. We mainly study the application of OPC protocol in intelligent warehousing system. The warehouse control system (WCS) runs on the upper computer and acts as an OPC client,while the OPC server runs on the PLC or the upper computer. Three communication methods between WCS and OPC server,synchronous,asynchronous and subscription,are described. The OPC server and the OPC interface are designed and implemented,and the communication between WCS and PLC implemented improves the speed of WCS issuing instructions and the working efficiency of the intelligent warehousing system.

Key words: OPC; data communication; programmable logic controller; intelligent storage system; client/server model; warehouse control system

0 引 言

当今社会不断进步,工业上的自动化程度也在不断提高,智能仓储也被逐渐应用于各种场景,以达到减少货物存放空间,提高效率的目的。而对各个设备状态的实时监控是整个智能仓储软件最为关键的一部分,在监控、控制整个智能化立体仓储中发挥着十分关键的作用。作为整个系统的核心,WCS(warehouse control system,仓库控制系统)能够根据设备当前的状态,通过对数据库的查询来控制任务信息的下达,可以实时监控当前堆垛机所处的位置、堆垛机状态、是否报警以及堆垛机货叉上是否有货的状态变化,也能够对输送线的状态和是否有报警进行监控,还能够查询到当前货物所存放的位置及库位信息等。为了能够实现上述的功能,第一步需要解决PLC和上位机之间的通

收稿日期:2020-06-30 **修回日期**:2020-10-30
基金项目:江苏省研究生教育教学改革课题(JGZZ19_038)
作者简介:吴 晗(1994-),男,硕士研究生,研究方向为物联网、智能仓储;通讯作者:成卫青(1972-),女,教授,博士,CCF会员(E200019081M),研究方向为网络测量、分布式系统。

信。一般来说,对于一些要求不是很高的设备,串口通信是最常见的方式,除此以外还有总线通信、串口转以太网通信等。但是根据实际使用的情况发现,串口通信和现场总线通信会受到通信距离的限制,而且通信效率不高,常常会有信号延迟或者信号收不到的问题出现。为了解决上述问题,出现了 OPC(OLE for process control)通信。

1　OPC 技术

通信协议规定了实体双方要实现通信所必须遵守的规则或约定。现阶段,随着工业自动化的快速进步,在完成 PLC 和上位机的数据交互方面,OPC 协议起着至关重要的作用[1-4]。由于基于 OPC 通信协议的交互速度快,性能稳定,目前越来越多的交互方式都采取这种协议。根据不同的控制系统,OPC 服务器不仅能够放在 PLC 上成为上位机的远程 OPC 服务器,也能够放在上位机上成为本地 OPC 服务器。此外,接口作为组件之间通信的桥梁[5],OPC 为开发人员提供了一个具有开放式的接口,可以对这个接口进行引用、改写、扩展,从而实现符合自身需求的接口,完成基于 OPC 的上位机与下位机的数据交互。因此,OPC 协议为实现通过 PLC 连接到现场工控电脑的 OPC 客户端提供了一个可行的方式。图 1 为数据源与 WCS 的连接结构。

图 1　数据源与 WCS 连接结构

WCS 与 OPC 服务器的数据通信包含两方面:一方面是 WCS 从 OPC 服务端读数据;另一方面是 WCS 向 OPC 服务端写数据[6]。WCS 从 OPC 服务端读数据有 4 种方法,分别为:异步读取、刷新与订阅、同步读取[7]。WCS 向 OPC 服务器写入数据有两种方法:异步写入和同步写入。WSC 和 OPC 服务器的数据通信有 3 种方式,分别为:异步、同步以及订阅。

1.1　同步访问

同步读和写都属于同步访问,同步访问指的是 WCS(OPC 客户端)调用 OPC 服务器提供的接口,对服务器中的内容进行读取和写入,OPC 服务器根据 WCS 所请求的内容将查找到的数据返回给 WCS,而在此之前 WCS 将会一直处在等待状态。

同步访问的特点是:在 WCS 读取特定数据块内容时,必须要求 WCS 读到数据为止;同样的,WCS 在写入指定的数据块地址之前,会始终处在等待数据的状态。因此,当数据量比较小且 WCS 与 OPC 服务器交互不是很频繁的时候可以使用同步访问的方法。但当数据量大或者有大量用户访问的时候,同步访问会出现延迟情况,造成系统的效率降低。

1.2　异步访问

异步读和写也都属于异步访问,异步访问和同步访问相似,但异步访问的操作过程比同步访问更加复杂,先要 WCS 提出数据请求,然后在 OPC 服务器接收到这个数据请求后,OPC 服务器会将这个请求进行“队列”排序并且为其编号,WCS 所调用的方法就会立即返回,使 WCS 能够继续执行其他任务,不再处于等待状态。在 OPC 服务器操作完成后,WCS 会在一个专门处理外部事件的代码块中处理 OPC 服务器发送过来的数据。

异步访问的特点是:在 WCS 发出读取请求后需要马上返回,然后 WCS 可执行其他任务,当 OPC 服务器读取数据完成后,调用 WCS 的读取完成事件,完成读取过程;同样的,在 WCS 发出写入请求后需要马上返回,然后 WCS 继续执行其他任务,当 OPC 服务器把数据写入完成后,再调用 WCS 的写入完成事件,完成写过程。所以异步访问的效率相比较于同步访问更高,它可以承载更多的用户请求的数据,并且在最大程度上节约通信以及 CPU 资源。

1.3　订阅方式

订阅方式是 OPC 服务器与 WCS 通信中的一种十分特别的数据通信模式[8]。OPC 服务器不需要收到来自 WCS 的数据请求,它是 OPC 服务器在每一个周期内,对于在扫描缓冲区内的数据进行扫描然后对比,若对比结果发现前后数据的变化大于阈值时,则会更新扫描缓冲区中的数据并且向 WMC 提供数据。这样一旦出现数据变化 WCS 就能够及时地收到变化后的数据信息。订阅方式从本质上来说是一种异步方式中的读取方式。使用订阅方式采集数据是按照固定的周期更新数据缓冲区的值,当发现数据出现变化时,则会通过数据变化事件告诉 WCS,并且只有当数据的变化超过规定阈值的时候才会更新数据缓冲区的值并且通知 WCS,由此可以得出,使用此种方式将会忽略数据微小的变化,以降低 WCS 与 OPC 服务器的负载。

订阅方式的特点是:通过 OPC 服务器在每个固定周期中检查缓存区中的数据,当发现数据的变化超过一个固定值之后,就会立即告诉 WCS 应用程序,将数据缓存区的信息传送过去。这种技术是根据“硬件设备-服务器-用户”的模型,如图 2 所示,在服务器内部预先建立一个动态缓存区用来存储数据,当数据出现变化较大时对其进行更新且发送给 WCS。这种方式在数据变化不大的情况下不会向 WCS 发送信息,从而减少数据发送的次数,也能够减少 WCS 对 OPC 服务器的重复访问的次数。在提供数据的设备点十分多的情景中,更能体现出使用这种通信方法的优点。

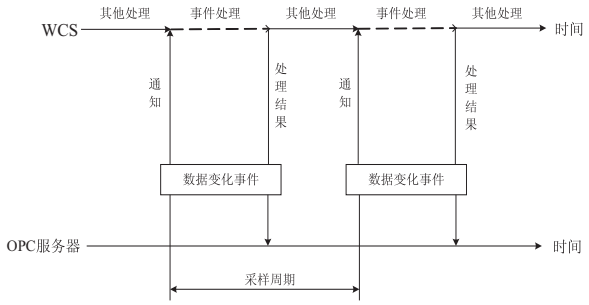

图 2 订阅数据访问交互方式

2 OPC 服务器的搭建

智能仓储一般都使用 C/S 的架构模式,它包含了 OPC 服务器以及 WCS(OPC 客户端)两个部分,该文主要探讨的是 OPC 服务器的实现过程。

2.1 总体设计构思

OPC 服务器的设计需要根据智能仓储的实际情况去考虑,为 WCS 提供一个稳定、流畅的通信平台。具体需要实现 3 个方面的功能。第一,作为一个数据服务器,OPC 服务器需要向 WCS 提供 OPC 服务器的对象以及接口,这样其他软件才能对 OPC 服务器提供的数据进行读操作和写操作,这是作为 OPC 服务器最关键也是最为基本的功能;第二,OPC 服务器应当能够对现场采集的实时数据进行数据管理;第三,为了能便于维护人员进行操作,OPC 服务器应提供灵活多用的交互接口。在开发设计层面,OPC 服务器应当与 WCS 的功能模块相对独立,由数据采集、OPC 对象和接口以及数据管理三个子模块组成。OPC 服务器的总体结构如图 3 所示[9]。

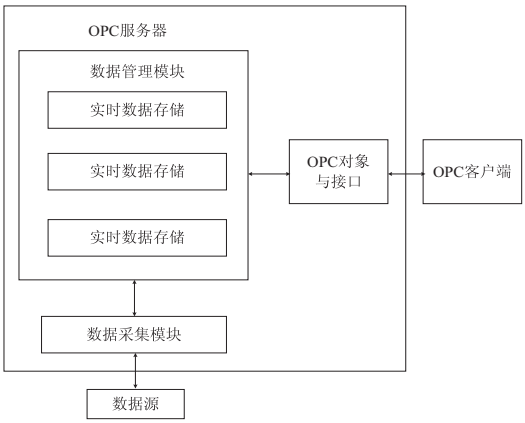

图 3 OPC 服务器的总体结构

2.2 数据采集模块

数据采集与现场设备直接连接,获取到现场设备的实际数值,再将其分配到其对应的通信子模块中。通常来说,每一个设备都会有一个通信子模块与其对应。数据采集模块会有一个固定的采集周期,在每个采集周期内都会从现场实际设备中采集数据。对于数据采集模块来说有两种方式可以实现数据采集,第一种为直接通过 I/O 驱动实现现场设备与通信子模块之间的通信;第二种为使用数据接口,通过这个数据接口不断地读取现场设备中的数据。OPC 服务器数据采集模块的结构如图 4 所示。

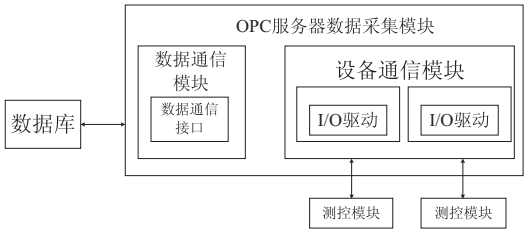

图 4 OPC 服务器数据采集模块的结构

在实际开发应用的时候,工业上很多的控制软件都配备了属于其单独拥有的数据库。通常情况下,都是利用数据库供应商提供的专门的 API 来实现对数据库的操作。除此以外,目前市面上经常使用的数据库如 SQLServer、Oracle、MySql 等,虽然大致相同,但是每个数据库都具备其自身的特点,如果需要访问这些数据库的话,就要对每个数据库单独地编写不同的控制软件。对于日后的系统集成来说,这将会是一个巨大的挑战[10]。

在本系统中 OPC 服务器特别为数据采集模块提供了对数据库进行操作的接口,使得整个 WCS 具有 OPC 服务器的功能,有效地解决了数据库版本不统一,需要单独编写的问题。对于这样的设计方式类似于创建了一个访问 WCS 实时数据库的中间件,对于其他的开发商来说就无需知道这个数据库的特有结构了。OPC 客户端(WCS)只需要根据所提供的 OPC 通信规范[11],就可以利用 OPC 服务器通过间接访问的方式与数据源交换数据。

对于不同的数据源如输送线、传感器、堆垛机等,OPC 服务器必须能够取得这些设施的数据。因此在数据采集模块中,通信子模块中的接口都是以动态链接库(dynamic-link library,DLL)的形式实现的。利用动态链接库能够制定一个统一的通信接口格式,这样就能将 OPC 服务器中上层具体功能和下层数据采集模块的功能分隔开,两部分独立开发降低耦合程度,这样使得 OPC 服务器能够对多种不同的设备实现兼容。除此以外,将通信接口封装成 DLL 来实现,不仅能够在很大程度上降低程序运行所需的资源,增强系统的安全性和可靠性,也使得用户对其二次开发更加方便。

2.3 OPC 对象和接口

WCS 和 OPC 服务器之间的数据通信依靠对象的接口来实现,OPC 对象和接口也是整个 OPC 通信规范的核心。主要有三种结构组成了 OPC 服务器:服务器、组和项[12]。作为所有组对象的集合,服务器对象包含了所有组对象的数据。在同一个组中,本组的组

对象可以获得所有信息,同时也包含了 OPC 数据项。

OPC 的组对象为客户获取数据信息提供了一种方法。客户既可以在客户端当中设置对数据的更新频率,也可以对数据进行读操作或写操作。当在数据缓冲区的数据出现变化后,OPC 服务器会将变化的数据发送给 WCS,它在接收到数据后进行信息处理,无需花费大量的时间来搜索。在 OPC 定义的规范当中,有两组对象:分别是本地组以及公共组。对于公共组来说,它可以被多个 WCS 共同拥有,但是本地组只能有一个 WCS 客户。一般情况下,对于一对已建立连接的 WCS 和 OPC 服务器来说,它们只需要定义唯一的一组对象。但是在每组对象中,WCS 能够将多个 OPC 数据项增加进来。

服务器、组、数据项这三类对象形成了一个分层式的结构模型。在这个结构模型中,作为标准的 COM 对象,OPC 对象包含了服务器对象和组对象,是 OPC 服务器必须实现的两个组件对象[13],它需要通过实现 OPC 规范定义的接口,并且将这些接口提供给 WCS,WCS 再根据所提供的接口,完成对 OPC 服务器的访问[14]。服务器对象拥有自己的类厂,如果需要使用,就必须先行在系统注册表中完成注册;而由于组对象没有类厂,因此不需要向服务器对象一样在注册表中注册,在服务器对象创建的同时也创建了组对象。组对象包括了项对象,它的作用是创建和数据存储区的联系,表明数据源的地址和类型。项对象与数据存储区的数据项这两个项是不同的,数据存储区的数据项主要作用是和数据源通信,获取该数据源中的数据;而项对象主要是和组对象通信,以及获得 WCS 所需要获得的信息。WCS 通过组对象获取到相关对象的名称、属性、值等信息,然后再将项对象与数据存储区的数据项进行关联,从而避免直接操作数据存储区。

OPC 对象和接口模块对外提供 OPC 接口与 WCS 进行交互,对内则通过变量连接机制建立与数据源的连接,其结构如图 5 所示。

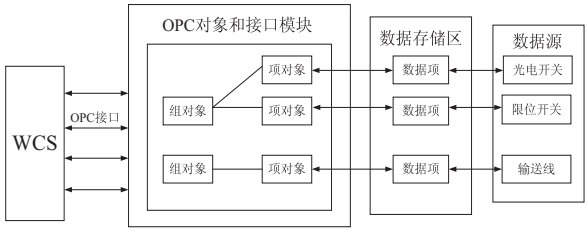

图 5　服务器中对象和接口模块结构

2.4　数据管理

WCS 需要监控很多的变量数据信息,其中就包括了堆垛机的状态,模式,输送线的状态模型,光电开关,报警信息,也能够通过其他程序来获得数据,以上的数据都能作为 OPC 服务器中的数据项,存储在服务器的地址空间中。

3　WCS 与 OPC 服务器通信的实现

WCS 要从 OPC 服务器中获取数据,是数据使用者,而 OPC 服务器是数据的提供者,将数据提供给 WCS。它们两个之间的数据传递、交互是通过 OPC 协议来实现的,也就是说必须要符合 OPC 的接口规范。OPC 技术不仅仅适用于 WCS 和硬件设备之间,也适用于两个软件之间实现数据互通,它们既可以都配置在同一台 PC 上,也能够配置在同一个局域网中的不同 PC 上。

3.1　WCS 与 OPC 服务器的通信流程

WCS 和 OPC 服务器的交互流程如图 6 所示。首先,无论对于 WCS 还是 OPC 服务器第一步都要做的是初始化 COM(组件)库。COM 是微软公司为了计算机工业的软件生产更加符合人类的行为方式开发的一种新的软件开发方式,可以将系统中的组件用新的替换掉,以便于日后的升级,也能够在多个系统中使用多次。然后,OPC 服务器将自身的信息注册到操作系统的注册表中,获取 CLSID,也叫类标识符。接下来是 WCS 根据查询到的 CLSID 向 OPC 服务器提出构建服务器对象的请求,将 WCS 的请求发送给 OPC 服务器后就会构建服务器组件对象,并且向客户端提供返回服务器接口对象的指针。当服务器组件对象构建完成后,WCS 就能根据这个接口指针去调用 OPC 服务器中提供的方法。最后,当 WCS 结束访问后就会将 OPC 服务器的资源进行释放,然后卸载 COM 库,关闭程序。在执行这一项过程当中,是 WCS 主动连接或者断开 OPC 服务器,但是对于 OPC 服务器本身来说,它是并不知道对于任何访问它的客户的信息的。图 6 为 OPC 服务器和 WCS 的交互流程。

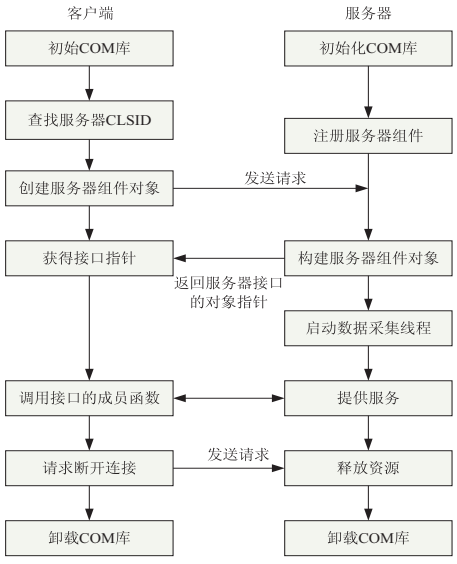

图 6　OPC 服务器和 WCS 的交互流程

3.2 组对象的实现

WCS 是通过控制组对象从而实现更小单位的控制,从本质上来说,是 OPC 服务器将相同类型的数据划分成为一个组而达到对相同类型的数据进行统一管理的目的。比如说,有两个 WCS 需要经常访问的数据,但是它们的数据刷新速率是不同的,那么就可以将这两个数据分别划分到不同组对象当中,这样将大大提高 WCS 读取数据的有效性和效率。

(1)组对象类的定义和方法。

接口中包含的方法与变量如下:

```
Class OpcDaCustomGroup
{
public int ServerGroupHandle;  //输出参数,服务器为新创建的组对象产生的句柄
public int RevisedUpdateRate; //输出参数,服务器返回给客户端的实际使用的数据更新率
public Guid Riid = typeof(IOPCItemMgt). GUID; //引用参数,客户端想要的组对象的接口类型(如 IIDIOPCItemMgt)
public object Group; //输出参数,用来存储返回的接口指针。如果函数操作出现任务失败,此参数将返回 NULL
public Guid Riid; //引用参数,客户端想要的组对象的接口类型(如 IIDIOPCItemMgt)
public object Group; //输出参数,用来存储返回的接口指针。如果函数操作出现任务失败,此参数将返回 NULL。
public GCHandle TimeBias; //指向 Long 类型的指针
public GCHandle PercendDeadBand; //一个项对象的值变化的百分比,可能引发客户端程序的订阅回调。此参数只应用于组对象中有模拟 dwEUType(工程单位)类型的项对象。指针为 NULL 表示 0.0
public int LCID; //当用于组对象上的操作的返回值为文本类型时,服务器使用的语言
public OpcDaCustomItem[ ] OpcDataCustomItems; // OPC 项数组
}
```

(2)组对象的接口实现。

组对象在 Opc 自定义接口-异步管理类(OpcDaCustomAsync)中的添加 OPC 项 AddOpcGroup 中实现,其中部分关键代码如下:

```
//添加 OPC 组
Private void AddOpcGroup(OpcDaCustomGroupopcGroup)
{
InitIoInterfaces(opcGroup); //初始化 IO 接口
if(opcGroup. OpcDataCustomItems. Length>0)
{
//添加 OPC 项
AddOpcItem(opcGroup);
//激活订阅回调事件
ActiveDataChanged(IOPCGroupStateMgt);
}
opcGroup. TimeBias. Free();
opcGroup. PercendDeadBand. Free();  //释放组对象的资源
}
```

其中 IOPCGroupStateMgt 为 OpcDaCustomAsync 类中的成员变量,是 Opc 组管理器。

```
//初始化 I/O 接口方法
public void InitIoInterfaces(OpcDaCustomGroupopcGroup){
int cookie;
    //组状态管理对象,改变组的刷新率和激活状态
IOPCGroupStateMgt = (IOPCGroupStateMgt) opcGroup. Group;
IConnectionPointContainer = (IConnectionPointContainer) opcGroup. Group;
Guidiid = typeof(IOPCDataCallback). GUID;
IConnectionPointContainer. FindConnectionPoint(ref iid, out IConnectionPoint);
//创建客户端与服务端之间的连接
IConnectionPoint. Advise(this,out cookie);}
```

其中 IConnectionPoint 为 OpcDaCustomAsync 类中的成员变量,是连接指针。

3.3 项对象的实现

项对象作为 OPC 服务器之中的私有变量,是将 WCS 和数据源连接起来的核心,又因为是私有变量,但在 OPC 定义的规范中并未为其设定出任何标准接口,WCS 只能通过组对象对其进行操作,但也正因为没有为项对象做出规定,开发者们就能够根据现场的实际情况以及自身的需求来对项对象进行设计与开发[15]。用设计类 OpcDaCustomItem 来描述项对象,包含的主要属性和部分关键代码有:

```
///添加 OPC 项
private void AddOpcItem(OpcDaCustomGroup opcGroup)
{
OpcDaCustomItem[ ] opcDataCustomItemsService = opcGroup. OpcDataCustomItems;
IntPtrpResults = IntPtr. Zero;
IntPtrpErrors = IntPtr. Zero;
OPCITEMDEF[ ] itemDefyArray = new OPCITEMDEF[opcGroup. OpcDataCustomItems. Length];
int i = 0;
int[ ] errors = new int[opcGroup. OpcDataCustomItems. Length];
int[ ] itemServerHandle = new int[opcGroup. OpcDataCustomItems. Length];
//获取每一个数据项的值,存放在一个集合中
foreach (OpcDaCustomItem itemService in opcDataCustomItemsService){
        if (itemService! = null)
{
itemDefyArray[i]. szAccessPath = itemService. Access-
```

```
Path;
            itemDefyArray[i]. szItemID = itemService. ItemID;
            itemDefyArray[i]. bActive = itemService. IsActive;
           itemDefyArray[i]. hClient = itemService. ClientHandle;
           itemDefyArray[i]. dwBlobSize = itemService. BlobSize;
       itemDefyArray[i]. pBlob = itemService. Blob;
       itemDefyArray[i]. vtRequestedDataType = itemService. RequestedDataType;
       }
       //添加 OPC 项组
       ((IOPCItemMgt) opcGroup. Group). AddItems (opcGroup. OpcDataCustomItems. Length, itemDefyArray, out pResults, out pErrors);
       IntPtr Pos = pResults;
       Marshal. Copy (pErrors, errors, 0, opcGroup. OpcDataCustomItems. Length);
       Pos = new IntPtr(Pos. ToInt32() + Marshal. SizeOf(typeof(OPCITEMRESULT)));}
       var result = (OPCITEMRESULT) Marshal. PtrToStructure(Pos, typeof(OPCITEMRESULT));
       itemServerHandle[j] = opcDataCustomItemsService[j]. ServerHandle = result. hServer;
```

4　系统实际应用

OPC 服务器在系统实际应用中主要起以下两个方面的作用:与电子元器件(主要为 PLC)相连获取外界物理数据;与上位机通讯,将获取来的数据提供给上位机进行读写操作,从而实现对外部设备的控制。该系统运行需要的软硬件有:Net Framework 4.0、WinCC_V7.3、西门子 300 或其他类型 PLC。

OPC 服务器从 PLC 处获取数据,需先在本机上安装 WinCC_V7.3,并对站组态编辑器进行配置,如图 7 所示,需要配置好站名、模式,选择本机使用的网卡,以及对应的 IP。

图 7　OPC 服务器站组态编辑器

当以上操作都完成后,需要 PLC 编程人员将程序下载至站组态编辑器之中,这样 OPC 服务器就能与 PLC 建立连接,获取数据。

而作为 OPC 服务器的上位机 WCS 在从 OPC 中获取时,通常情况下都是在文本文件中预先编写好 xml 类型的配置文件,如图 8 所示。

```
<?xml version="1.0" encoding="utf-8" ?>
<System>
  <OpcServer ServerName="OPC.SimaticNET" IPAddress="10.37.236.189">
    <!--堆垛机12003-->
    <OpcGroup GroupName="Stk12001" ClientHandle="1" UpdateRate="0">
      <!--堆垛车模式-->
      <Item ItemID="S7:[S7_Connection_1]DB100,B0" ItemName="V_FoModel" ItemNameCN="堆垛车模式" ClientHandle="1" RequestedDataType="8"></Item>
```

图 8　xml 文件配置示例图

xml 配置文件的第一行为 xml 的版本(1.0 版本),并且使用的是 UTF-8 字符编码。其中的标签从大到小分别为 OpcServer,OpcGroup 和 Item,依次对应 OPC 服务器配置、OPC 组对象配置以及 OPC 项对象配置。在 OPC 服务器配置中需要写明 OPC 服务器名及 IP 地址;在 OPC 组对象配置中需要写明 OPC 组名、客户端句柄以及刷新频率;最后在项对象中则需要写明 OPC 项对象在 PLC 中的地址块、自定义英文名和中文名、客户端句柄以及刷新频率。至此,上位机配置文件基本完成。

由于 PLC 一般适用于工业自动化设备,考虑到软件的稳定性和适用性,以及库文件的多样性,上位机软件开发常使用微软的 VS 平台作为开发环境,.NET 作为开发语言。图 9 为上位机软件 WCS 的变量监控图。

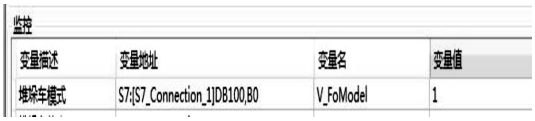

图 9　WCS 监控数据变量图

从图 9 中可以看出,监控的变量为 xml 配置文件中配置好的项对象,其中包含了变量描述、变量地址、变量名以及变量值。其中变量值的刷新频率可以通过 xml 文件的刷新频率(UpdateRate)进行设置,通常情况下数据交互模式都是采用订阅方式进行数据更新,因为这种方式效率高、稳定性好。

5　结束语

该文为智能仓储系统设计实现了 OPC 服务器以及 OPC 接口,实现了基于 OPC 的 WCS 和 PLC 之间的通信。使得 WCS 系统能够对堆垛机、输送线等进行控制并且下发指令,从而实现对货物以及库位的管理。在智能仓储中使用 OPC 通信,相较于传统的串口通信,传输性能大大提高,提升了 WCS 下指令的速度、整个仓储的工作效率以及管理水平。

参考文献:

[1]　龚　勋,王淑营. 基于 C#的 OPC 客户端设计[J]. 计算机系统应用,2020,29(5):239-244.

[2]　王丹豪,彭道刚,张锐锋,等. 基于 JAVA 的 OPC 数据采集和转储系统软件设计[J]. 上海电力大学学报,2020,36(2):117-122.

[3]　李　靖,张继彪,夏　月. 基于 OPC 技术的重型钢构件制作监控系统设计及实现[J]. 工业控制计算机,2020,33(3):92-93.

[4]　LIU Qing,QIU Yongsheng. Development of OPC server in a remote industrial control system[C]//2015 12th IEEE international conference on electronic measurement & instruments

(下转第 170 页)