**1、 OPC服务介绍**  
  
　　西门子提供的最新软件：Simatic Net PC-Software CD 2005为各种组态软件的开发提供了一个统一的平台，它建立的PC站既为一些组态软件，如：WinCC、Protol等提供了与PLC的通讯平台，也提供了一套编程接口，可使用高级语言编程通过Simatic Net访问PLC数据。本文讨论的主要就是这个编程接口,最新版的Simatic NET支持五种编程方式：  
**<1>、ActiveX控件**  
　　提供了一系列数据访问控件，以便于向VB6这种语言使用控件的方式与PLC通讯。  
**<2>、OPC自动化**  
为VB6、Dephi等语言运用OLE 自动化的方式进行编程。  
**<3>、OPC用户接口**  
　　这是专门为VC++提供的一种高效编程方式，其灵活程度与执行效率比前面的两种方式均要高得多。  
**<4>、针对微软的.NET平台的OPC用户接口**  
　　这也是一种非常灵活的编程接口，不过它针对的是.NET平台，其提供了大量的.NET类库，以便于像VC#、VB.NET等高级语言编程。本文将详细的介绍该接口。  
**<5>、OPL XML接口**  
　　顾名思义，主要是针对XML编程的。  
　　对于<2>、<3>、<4>编程方式，他们各自又可以分为同步访问方式和异步访问方式。按西门子的文档解释：同步通讯指的是当一个客户在访问服务器时，其他客户的访问必须等待，直到服务器处理完该客户的请求，才能继续进行下一个服务，异步访问与之正好相反，本文主要讲的是同步编程篇，异步篇以后再提供。  
**2、 配置OPC服务器**  
　　要进行编程，必须先配置服务器。本文以Prfibus DP网络为例，介绍PC站的配置。其内容主要来自西门子文档。  
**需要的软件：**  
Step7 V5.3  
Simatic Net PC-Software CD 2005   
**需要的硬件：**  
　　至少为CP5611或以上级别，笔记本可以为CP5511,带DP口的S7-300 PLC(若使用Simatic NET的仿真功能可以不需要这些硬件，后面会介绍到)  
　　<1>、组态一个S7站，配置Profibus DP网络，其DP地址设为3，并下载到PLC,然后把网线由MPI口转到DP口。S7站的配置这里就不介绍了。  
　　<2>、在 Step7 V5.3中建立一个新工程，插入一个PC站，并把该PC站的名字改成与你的计算机名字相同。打开该PC站的硬件组态界面。插入OPC服务器和连接卡CP5611(或者CP5511),他们在PC槽中处的位置可以任意，如下图：  
  
　　注：在插入CP5611时，应该选择与组态S7站一样的Profibus网络，并将网络地址设为2，一定不要与PLC的地址冲突。  
  
　　然后点击下面工具条标为红色的按钮：  
  
　　选中”OPC Server”，然后插入一个新的连接，如下图：  
  
　　在弹出的对话框中选择连接类型为S7 Connection,如下图：  
  
　　在OK后，然后在新对话框的红色标志位置输入3，表示PLC的地址，如下图：  
  
　　并选择Address Details…,设置CPU的槽号为2，如下图：  
  
　　OK后，然后编译并保存。  
　　<3>、然后建立OPC服务器，有两种方式，本文介绍较简单的一种。  
　　打开，Simatic Net中的Station Configurator，一般安装后，他会自动启动，并点击Import Station…按钮，找到你刚才在Step 7中建立PC站时创建的  
　　XDBs文件夹下的XDB文件，然后导入成功。  
　　<4>、可以使用Simatic Net中的OPC Scout，并选择Simatic NET服务，然后在它下面创建组，然后在组下创建变量，这样可以监控PLC数据，VC#编程不需要使用该程序，但熟悉使用OPC Scout有利于了解Simatic Net中的编程结构。  
　　说明：打开Simatic Net中的Configuration Console，选中S7进行如下的配置后，可以不需要PLC、CP5611等并可以模拟，如下图：  
  
　　上面的所有步骤，均可在Configuration Console下，PC Station的根树下，选择相应的帮助文档得到。  
**3、 OPC编程  
  
<1>、西门子的变量结构如下：**  
  
----------------------服务器------------------------------  
/ OPC.SimaticNet OPCServer.Wincc .... (一系列类型的服务器）  
/ Group1 Group2 Group3 ...(把更新时间一致的变量统一为一个组)  
/ Item1 Item2 ... (变量：I、Q、M、DB等，指向网络中某个PC站OPC Server服务的某个连接）  
-----------------------------------------------------------------------------------------------------------------   
第一层是不同种类的服务器，如：OPC.SimaticNET类型，OPC.SimaticNET.DP类型，OPCServer.WinCC等一系列类型，这里选择OPC.SimaticNET类型。  
第二层是Group,一个服务器下可以有多个组，可以把组理解为扫描周期相同的一系列变量的集合。在开发组态界面时，可以把一个界面中的所有变量统一到一个组中。  
第三层是Item，项是指向网络中某个PC站OPC Server服务的某个连接的一系列变量，如：I、Q、M、DB等  
  
**<2>、项的命名**  
　　项即Item,在S7连接中针对的直接是PLC中的变量，因此它的命名很重要：  
　　格式： :[]  
其中的protocolID表示连接类型，在上面的组态PC站时可以选择，这里应该与它一致，类型有9种，最常用的为S7，即S7连接，其他类型请参看文档。  
Connectionname：顾名思义，即在上面的组态PC站时产生的连接名，如果使用仿真功能，连接名为DEMO Variablename:变量名有一系列规则，这里举例说明，读者也可以使用OPC Scout创建变量，学习程序是如何生成变量名的。  
S7:[DEMO]MB1 ：表示连接类型为S7,连接名为DEMO（这里为仿真），变量为MB1  
S7:[DEMO]QB0,3: 表示为从QB0开始的三个连续变量。  
S7:[DEMO]DB10,X4.6 ：表示DB10的DBX4.6。  
**<3>、添加引用**  
　　在VC#开发环境中添加对OpcRcw.Da库的引用引用，该库属于.NET库，不属于COM库，西门子虽然编写了类库，以提供对.NET平台的支持，但这些类库仍然难于编程，  
　　里面包含了大量的在托管和非托管区传输数据，因此我们需要在它的基础上再开发一个类库，以简化以后的编程，首先在类的开头使用命名空间：  
using System.Runtime.InteropServices;  
using OpcRcw.Da;  
using System.Collections;  
**<4>、编程  
1、 在类的开头部分生名变量**  
private string serverType="";  
private IOPCServer pIOPCServer; // OPC server接口  
private Object pobjGroup1; // Pointer to group object   
private int nSvrGroupID; // server group handle for the added group   
private System.Collections.Hashtable groupsID=new Hashtable(11); //用于记录组名和组ID号  
private System.Collections.Hashtable hitemsID=new Hashtable(17); //用于记录项名和项ID号  
private Guid iidRequiredInterface;  
private int hClientGroup = 0; //客户组号  
private int hClientItem=0; //Item号  
  
**2、 创建服务器，编写Open()方法**  
///  
/// 创建一个OPC Server接口  
///  
///   
/// 若为true，创建成功，否则创建失败  
public bool Open(out string error)  
{  
error="";bool success=true;  
Type svrComponenttyp ;  
//获取 OPC Server COM 接口  
iidRequiredInterface = typeof(IOPCItemMgt).GUID;  
svrComponenttyp = System.Type.GetTypeFromProgID(serverType);  
try  
{  
//创建接口  
pIOPCServer =(IOPCServer)System.Activator.CreateInstance(svrComponenttyp);  
error="";  
}  
catch (System.Exception err) //捕捉失败信息  
{  
error="错误信息:"+err.Message;success=false;  
}  
Return true;  
}  
  
**3、 在服务器上添加用于添加Group的函数**  
  
///  
/// 添加组  
///  
///   
///   
///   
///   
/// 若为true，添加成功，否则添加失败  
public bool AddGroup(string groupName,int bActive,int updateRate,out string error)  
{   
error="";  
int dwLCID = 0x407; //本地语言为英语   
int pRevUpdateRate;  
float deadband = 0;  
// 处理非托管COM内存  
GCHandle hDeadband;  
IntPtr pTimeBias = IntPtr.Zero;  
hDeadband = GCHandle.Alloc(deadband,GCHandleType.Pinned);  
try  
{  
pIOPCServer.AddGroup(groupName, //组名  
bActive, //创建时，组是否被激活  
updateRate, //组的刷新频率，以ms为单位  
hClientGroup, //客户号  
pTimeBias, //这里不使用  
(IntPtr)hDeadband,  
dwLCID, //本地语言  
out nSvrGroupID, //移去组时，用到的组ID号  
out pRevUpdateRate, //返回组中的变量改变时的最短通知时间间隔  
ref iidRequiredInterface,  
out pobjGroup1); //指向要求的接口  
hClientGroup=hClientGroup+1;  
int groupID=nSvrGroupID;  
groupsID.Add(groupName,groupID);  
}  
catch (System.Exception err) //捕捉失败信息  
{  
error="错误信息:"+err.Message;  
}  
finally  
{  
if (hDeadband.IsAllocated) hDeadband.Free();  
}  
if(error=="")  
return true;  
else  
return false;  
}  
  
**4、 向指定的组中添加变量的函数**  
  
///  
/// 添加多个项到组  
///  
///   
///   
///   
/// 无错误，返回true，否则返回false  
public bool AddItems(string groupName,string[] itemsName,int[] itemsID)  
{  
bool success=true;  
OPCITEMDEF[] ItemDefArray=new OPCITEMDEF[itemsName.Length];  
for(int i=0;i<itemsname.length;i++)  
 {  
hClientItem=hClientItem+1;  
ItemDefArray[i].szAccessPath = ""; // 可选的通道路径，对于Simatiic Net不需要。  
ItemDefArray[i].szItemID = itemsName[i]; // ItemID, see above  
ItemDefArray[i].bActive = 1; // item is active  
ItemDefArray[i].hClient = hClientItem; // client handle   
ItemDefArray[i].dwBlobSize = 0; // blob size  
ItemDefArray[i].pBlob = IntPtr.Zero; // pointer to blob   
ItemDefArray[i].vtRequestedDataType = 2; //Word数据类型  
}  
//初始化输出参数  
IntPtr pResults = IntPtr.Zero;  
IntPtr pErrors = IntPtr.Zero;  
try   
{  
// 添加项到组  
((IOPCItemMgt)GetGroupByName(groupName)).AddItems(itemsName.Length,ItemDefArray,out pResults,out pErrors);  
// Unmarshal to get the server handles out fom the m\_pItemResult   
// after checking the errors  
int[] errors = new int[itemsName.Length];   
Marshal.Copy(pErrors, errors, 0,itemsName.Length);  
IntPtr pos = pResults;  
for(int i=0;i<itemsname.length;i++) 循环检查错误  
 {  
if (errors[i] == 0)  
{  
OPCITEMRESULT result = (OPCITEMRESULT)Marshal.PtrToStructure(pos, typeof(OPCITEMRESULT));  
itemsID[i] = result.hServer;  
this.hitemsID.Add(itemsName[i],result.hServer);  
pos = new IntPtr(pos.ToInt32() + Marshal.SizeOf(typeof(OPCITEMRESULT)));  
  
}  
else  
{  
success=false;  
break;  
}  
}  
}  
catch (System.Exception err) // catch for error in adding items.  
{  
success=false;  
}  
finally  
{  
// 释放非托管内存  
if(pResults != IntPtr.Zero)  
{  
Marshal.FreeCoTaskMem(pResults);  
pResults = IntPtr.Zero;  
}  
if(pErrors != IntPtr.Zero)  
{  
Marshal.FreeCoTaskMem(pErrors);  
pErrors = IntPtr.Zero;   
}  
}  
return success;  
}  
  
　　说明：使用该函数时，在类的开头，应该先声明整数数据，以用于保存由本函数返回的服务器对每一项分配的Item ID号：  
  
**5、 向指定组中指定的一系列项变量写入数据的公开方法**  
///  
/// 一次性写入多个值  
///  
///   
///   
///   
/// 无错误，返回true，否则返回false  
public bool Write(string groupName,int[] itemID,object[] values)  
{  
bool success=true;  
IntPtr pErrors = IntPtr.Zero;  
  
if(GetGroupByName(groupName) != null)  
{  
try  
{ //同步写入  
((IOPCSyncIO)GetGroupByName(groupName)).Write(itemID.Length,itemID,values,out pErrors);   
int[] errors = new int[itemID.Length];   
Marshal.Copy(pErrors, errors, 0,itemID.Length);  
for(int i=0;i<itemid.length;i++) 循环检查错误  
 {  
if (errors[i] != 0)  
{  
pErrors = IntPtr.Zero;   
success=false;  
}  
}  
}  
catch(System.Exception error)  
{  
success=false;  
}   
}  
return success;  
}  
注：参数int[] itemID应该是与AddItems函数中的int[] itemsID参数相对应。  
  
**6、 编写获取变量值的函数**  
  
///  
/// 一次性读取多个数据  
///  
///   
///   
///   
/// 无错误，返回true，否则返回false  
public bool Read(string groupName,int[] itemID,object[] result)  
{  
bool success=true;  
//指向非托管内存  
//指向非托管内存  
IntPtr pItemValues = IntPtr.Zero;  
IntPtr pErrors = IntPtr.Zero;  
if(GetGroupByName(groupName)!=null)  
{  
try  
{ //同步读取  
((IOPCSyncIO)GetGroupByName(groupName)).Read(OPCDATASOURCE.OPC\_DS\_DEVICE,itemID.Length,itemID,out pItemValues,out pErrors);   
int[] errors = new int[itemID.Length];   
Marshal.Copy(pErrors, errors, 0,itemID.Length);  
OPCITEMSTATE[] pItemState=new OPCITEMSTATE[itemID.Length];  
IntPtr pos = pItemValues;  
for(int i=0;i<itemid.length;i++) 循环检查错误  
 {  
if (errors[i] == 0)  
{  
//从非托管区封送数据到托管区  
pItemState[i] = (OPCITEMSTATE)Marshal.PtrToStructure(pos,typeof(OPCITEMSTATE));  
pos = new IntPtr(pos.ToInt32() + Marshal.SizeOf(typeof(OPCITEMSTATE)));  
result[i]=pItemState[i].vDataValue;  
}  
}  
}  
catch(System.Exception error)  
{  
return false;  
}  
}  
return success;  
}  
　　注：同Write()函数一样，参数int[] itemID应该是与AddItems函数中的int[] itemsID参数相对应。  
  
　　通过给类编写上面的几个最重要的函数，我们已经可以读写PLC数据了，下面给出例子。  
  
　　创建一个C#工程，添加对上面开发的类库的引用，并在窗体类的开头，声名：  
  
int[] nt=new int[2];int[] nt1=new int[2];  
S7Connection.SynServer server;  
其中的SynServer即为上面开发的类。  
  
**<1>、创建服务器接口**  
  
在程序初始化处，添加：  
server =new S7Connection.SynServer(S7Connection.ServerType.OPC\_SimaticNET);  
  
**<2>、打开连接**  
  
string err;  
server.Open(out err);  
  
**<3>、添加组**  
  
server.AddGroup("maiker",1,350,out err);  
server.AddGroup("maiker1",1,350,out err);  
  
**<4>、添加项（即变量），同样在程序的初始化中，将一系列项添加到他们各自得组。**  
  
string[] m1={"S7:[DEMO]MB1","S7:[DEMO]MW3"};  
string[] m2={"S7:[DEMO]MB6","S7:[DEMO]MW8"};  
server.AddItems("maiker",m1,nt);  
server.AddItems("maiker1",m2,nt1);  
  
**<5>、读写数据,这里以写数据为例：**  
  
obj[0]=this.textBox2.Text;  
obj[1]=this.textBox3.Text;  
if(radioButton1.Checked)   
{  
server.Write("maiker",nt,obj);  
}  
else if(radioButton2.Checked)  
{  
server.Write("maiker1",nt1,obj);  
}  
  
　　至此并完成了数据的通讯，如何，只要你把类库开发完善，在它的基础上再开发，会异常简单，本人已开发了完善的类库，上面的类库只是把最重要的部分讲解出来，我曾经在网上求助过很多次这方面的知识，无人应答。唉！太不容易了，等待Simatic NET软件花费了我一个月的时间，然后读几百页的英文文档，到开发程序，并测试花费了我一个星期的空闲时间，写这篇文章，又花费了我一个晚上的时间，不过我还是愿意把这些摸索出来的东西发给大家。