实现IT时代的测量控制系统必须的

OPC应用程序入门

-附有ＯＰＣ模拟服务器和示范源程序光盘-

编著: 日本OPC协会
翻译: 郑 立
后援: OPC中国促进会

日文版序

OPC（OLE for Process Control）不仅是生产系统中的关键基础技术之一，并且也是被终端用户广泛采用的软件标准技术。作为推进这种技术的标准化和普及活动的国际组织OPC基金会和其日本地区组织日本OPC协会决心为新时代制造业的革新作出贡献。

在1980年阿尔温·托夫勒，提出了继农业革命，产业革命之后将出现大的社会体系变革的所谓“第三次浪潮”理论。即是一次从集中向分散，从体力劳动向脑力劳动，从机械向信息转移的社会变革。这不仅仅只是实现信息化社会。由于来自旧社会势力的抵抗和现存技术基板的制约，不可能马上实现这种变革。但是20年后的今天，在制造业进行的管理过程革新以及IT的飞跃发展正在逐步克服这些障碍。21世纪将是真正地受到第三次浪潮的冲洗而出现的新地表的世纪。在这其中，OPC正是在制造业的仪表控制领域内所发生的“第三次浪潮”的主流。

自1996年秋OPC基金会和日本OPC协会创立以来，经过五年时间，把OPC确立为一个世界标准规范的活动阶段已告一段落。在迎接新世纪的今天，为了使终端用户真正地得到进一步的益处，我们正在放眼于超越目前为止的活动范围，目标于创立基于21世纪制造业新蓝图的生产系统而努力。为此目标，工业，教育，政府的各个企业和团体以及个人的合作乃是必不可少的。

作为在全世界首次出版的有关OPC应用程序编程书籍的本书，正是日本OPC协会技术委员会活动的总结，也可以说是来自迄今为止为OPC作出贡献的各位成员的总结报告。我希望本书可以成为读者在进行OPC实际应用时的入门参考，并进一步成为今后全球化技术合作的新起点。

ＯＰＣ基金会亚州理事 日本ＯＰＣ协会 秘书长　島貫　洋
（日本东芝公司）

本书作为日本OPC协会技术委员会五年间的努力成果，是一本按照OPC数据访问标准进行产品开发和系统组态的技术说明书。

OPC基金会继数据访问之后，还制定了警报和事件的标准，批处理的标准，安全性的标准等制造自动化和过程自动化所必须的一系列标准。同时为了迎接即将到来的第３代的互联网，现在正在制定OPC-XML等新标准。本书介绍的数据访问标准是这些标准的基础。这次把迄今为止作成的《数据访问标准的技术说明书》，《开发指南》以及技术讲座，对接实验等成果总结成本书正式出版，对于技术委员会来说一件十分值得庆贺的事情。

本书试图从OPC应用程序编程者的观点出发，提供了OPC模拟服务器和演示示范程序，以便通过编程实践理解OPC标准的内容。但是由于是首次出版这样的OPC书籍，难免有遗漏和不足之处，所以敬请各位读者多多给以批评指正，以便在今后再版时予以补正。

OPC通过上述的持续发展，已经得到了走在时代前列的事实上的世界标准的地位。今后正在从OPC基金会成立初期的“工厂内集成化的基础技术”向“企业生产系统间集成化的基础技术”扩展。与此同时，我确信日本OPC协会 技术委员会的作用将越来越更加重要。如果本书不仅仅在开发OPC应用程序时起到抛砖引玉的作用，而且可以成为各位读者于我们合作的桥梁的话，我将感到十分欣慰。

日本ＯＰＣ协会 技术委员会主任　中川 博之
（日本横河电机公司）

译者序

我刚刚接触OPC时，最初的担心是这种新技术使用复杂的微软的COM技术作为基础，恐怕为一般的仪表控制工程师所接受。但是通过对OPC技术的了解逐步深入，我渐渐明白了OPC的终端用户不必十分了解COM技术也可以完成OPC应用程序的编程。

于是我提出了编写一本OPC应用程序的入门书以促进OPC技术推广的提议。这个建议得到了日本OPC协会的同意和支持，并由我组织和参加本书的编写，在日本OPC协会同人的鼎立协作下，终于使本书的日文版得以正式出版。

我认为任何一种新技术即使再优越，如果得不到普及，它也没有生命力。OPC技术也是一样，如果得不到多数仪表控制工程师的承认和采用，也不会继续在世界工业标准中占有一席之地。

众所周知，学习计算机软件技术最好的方法不是阅读而是自己动手编写程序。OPC作为仪表控制的一种专用软件技术也是一样。所以本书与其说是直接说明OPC技术，不如说是通过示范程序使读者理解OPC技术。

本书的中文版可以在我的祖国发行，应该感谢中国OPC促进会的大力支持和协作。虽然我和中国OPC促进会的大多数成员并非面识，有关本书出版的工作也都是通过电子信件或电话联系的，但是正是由于他们的努力才使有关OPC技术的第一本中文技术参考书得以和中国的读者见面。

最后，我希望本书的读者对本书的谬误和欠缺之处予以斧正，以便使本书内容得以完善。

日本ＯＰＣ协会 技术顾问 工学博士 郑 立
（日本山武公司）

前言

本书的目的

这本书告诉你怎样按照OPC数据访问自动化接口标准（版本2.0）去实现OPC客户程序。即使你是OPC技术的初学者，你也可以利用本书顺利地实现你的第一个OPC客户程序。

本书的读者

这本书是为希望学习和开发OPC客户程序的终端用户编写的。本书的读者应该具有一些有关微软Visual Basic或者应用程序的Visual Basic的基础知识。然而我们并不要求你懂得COM和OPC的预备知识。

本书的内容

第１章说明了开发使用OPC数据访问自动化接口应用程序所需的OPC基础知识。首先这一章解释了什么是OPC和为什么需要OPC，并且讲述了OPC的开发历史和现状。接下来解说了有关OPC的一些重要概念，比如OPC服务器（server），OPC包装（wrapper），OPC标签（item），服务器句柄（handler）等。最后详细地讲述了OPC对象。在阅读了本章之后，读者应该可以掌握所有开发OPC应用程序所需的基础知识。

第２章是说明怎样使用微软Visual Basic去开发OPC自动化接口的客户应用程序。利用本章说明的应用程序，可以实现OPC自动化接口（版本2.0）的同步和异步的生产过程数据读写。

第３章是在第２章的基础上，解释怎样使用微软Visual Basic去开发使用OPC自动化接口的ActiveX控件。ActiveX控件是一些可以多次被使用的软件组件。例如当你利用Visual Basic开发去类似设备的控制监视程序时，你可以只开发一个ActiveX控件去监视控制一个设备，然后你可以使用这个ActiveX控件监视控制其他类似设备。你仅仅需要调试好你的ActiveX控件就可以了，这样一来可以大大减少你的开发成本和开发时间。

第４章是说明怎样使用微软应用程序的Visual Basic去开发OPC自动化接口的客户应用程序和使用OPC ActiveX控件的客户应用程序。 使用Excel可以使开发控制系统数据的图形表示或打印应用程序更为简单。

第５章是解释怎样设置你的计算机才能使你的OPC应用程序正常运转，特别是当你希望连接一个远程OPC服务器。当OPC应用程序使用于一个自控系统时，你经常会遇到麻烦的分布式COM安全性机制的设置问题。 但是幸运的是大多数自控系统并不要求安全性机制，因为这些系统是运行在不与外部网络连接的独立的网络。因此本章告诉你怎样设置你的计算机可以使分布式COM安全机制无效。 当然，本章也告诉你可以使分布式COM安全性机制有效的计算机设置方法。

第６章包括了当开发OPC应用程序时对你非常有用的一些信息，比如OPC符号，OPC错误码和OPC数据型。 此外，本章还将告诉你怎样使用本书配套光盘中提供的OPC示范源程序。在本章的最后，还告诉你怎样从本书配套光盘和互联网中找到有关OPC的其他详细信息。

怎样使用本书

你可以按照下面的流程图来阅读本书。

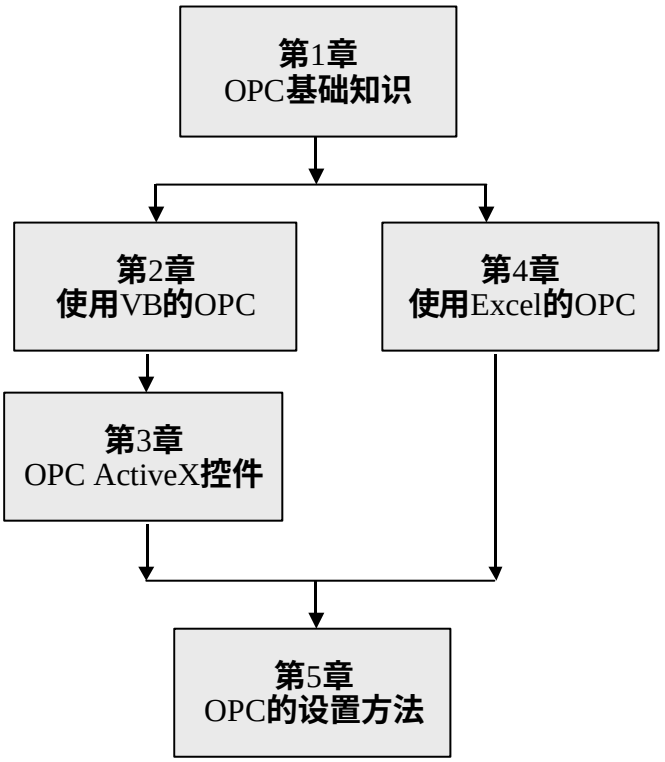
因为第一章是说明有关开发OPC客户应用程序的基础的并且重要的背景知识，所以我们推荐所有的读者从第１章开始阅读本书。

下面你要决定使用哪种编程语言，是Visual Basic（VB）还是应用程序的Visual Basic（Visual Basic for Applications, VBA）。VB的功能当然比VBA更强，但是VBA可以在广泛被使用的Excel等微软的办公室软件中使用。

如果你决定使用VB，那末你应该接着读第２章去学习怎样编写你的基本的OPC程序。第３章是为具有一定编程知识的用户编写的，主要讲述怎样开发他们的OPC ActiveX控件，这些ActiveX控件是可作为多次使用的软件组件。

如果你决定使用VBA，你可以在理解了第１章中讲述的概念后直接开始阅读第４章去学习怎样使用VBA建立你的OPC应用程序。

第５章的内容对于VB和VBA开发者都是需要的，特别当你希望在其他的计算机上运行你完成的程序。



执笔者

郑　　立　（日本山武公司）	【第１，５，６章】
寺冈　义则（日本日立制造所）	【第２章】
北山　胜　（日本Intellution公司）	【第３章】
乌山　幸嗣（日本欧姆龙软件公司）	【第４章 4.1节】
大西　辉生（日本欧姆龙软件公司）	【第４章 4.2节】

目录

1 OPC的基础知识

1.1 OPC的开发背景和历史

1.1.1 为什么需要OPC

对于早期的计算机系统，为了实现不同的硬件和软件所构成的计算机之间的数据交换和通信，必须要花费很多时间去开发独自的通信程序。但是正是由于现在有了数据交换和通信的工业标准，才有可以实现象互联网那样，使不同的计算机相互连接的巨大网络。所以在开发企业的信息系统时，采用符合工业标准的数据库和客户－服务器接口，可以使有效的精力更多地投入到应用程序本身功能的开发中去。

工业制造系统也存在同样的问题。也就是使由不同的供应商提供的机器设备无须特别的软件开发就可以互相连接。例如在实现象图 1-1那样的多层生产控制信息系统时，从处理设备数据的现场设备层，到进行过程处理的过程控制系统层，以至最上层的生产管理层，建立和普及一个有效的数据交换工业标准乃是当务之急。在这种情况下，利用微软Windows视窗中的OLE/COM技术实现工业制造系统过程控制中的数据交换标准化，正是OPC（OLE for Process Control）本来的目的所在。

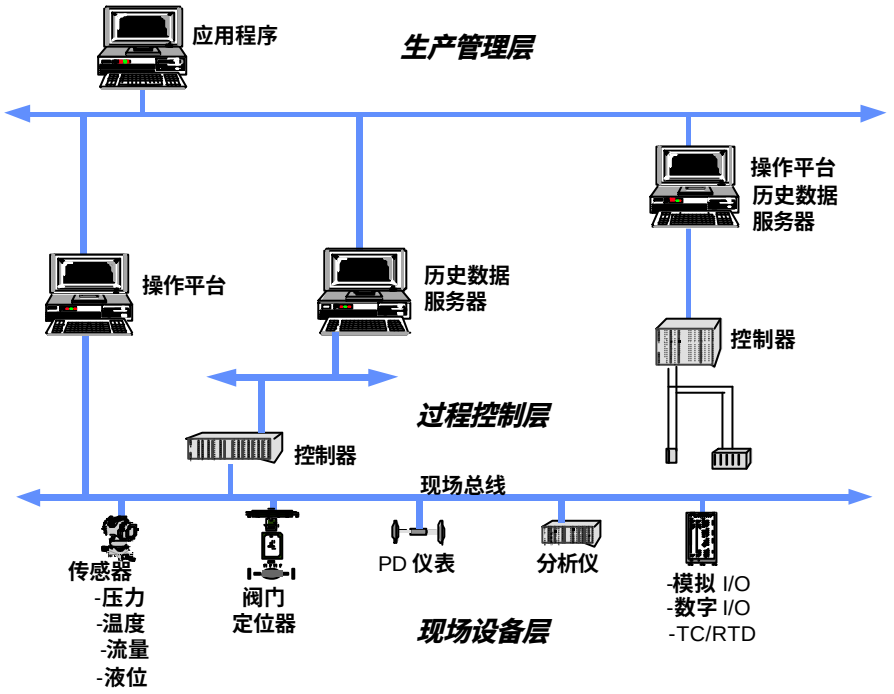


图 1-1 生产控制信息系统的构成

1.1.2 OPC怎样解决你的问题

到目前为止，硬件的驱动器和与其连接的应用程序之间的接口并没有统一的标准。例如，在FA（Factory Automation）领域，连接PLC（Programmable Logic Controller）等控制设备和 SCADA/HMI软件，需要不同的FA网络系统构成。根据某调查结果，据说在控制系统软件开发的所需费用中，各种各样机器的应用程序设计占费用的７成，而开发机器设备间的连接接口则占了３成。此外，在PA（Process Automation）领域，当希望把DCS（Distributed Control System）中所有的过程数据传送到生产管理系统时，

必须按照各个供应厂商的各个机种开发特定的接口（例如，利用C语言DLL连接的DDE服务器或者利用FTP的文本文件传送等）。

例如，在图 1-2所示的由４种控制设备和与其连接的监视，趋势图以及表报３种应用程序所构成的系统时，必须花费大量时间去开发分别对应设备A，B，C，D的监视，趋势图以及表报应用程序的接口软件共计１２种驱动器。同时由于系统中共存各种各样的驱动器，也使维护运转环境的稳定性和信赖性更加困难。

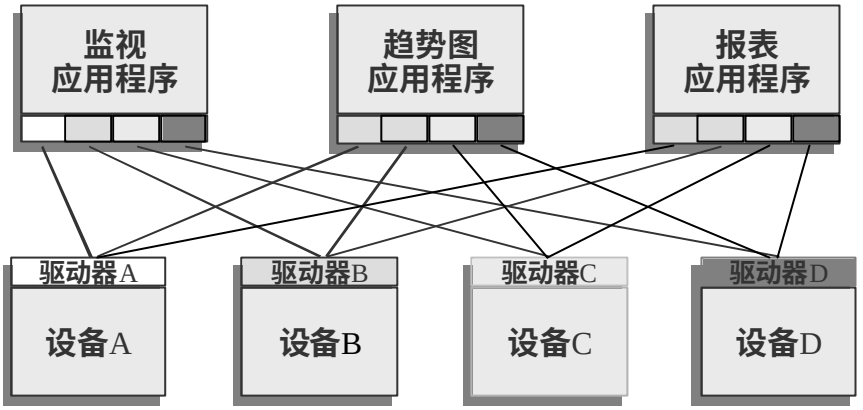


图 1-2 利用驱动器的系统连接

OPC是为了不同供应厂商的设备和应用程序之间的软件接口标准化，使其间的数据交换更加简单化的目的而提出的。作为结果，从而可以向用户提供不依靠于特定开发语言和开发环境的可以自由组合使用的过程控制软件组件产品。

利用OPC的系统，是由按照应用程序（客户程序）的要求提供数据收集服务的OPC服务器，使用OPC服务器所必需的OPC接口，以及接受服务的OPC应用程序所构成。OPC服务器是按照各个供应厂商的硬件所开发的，使之可以吸收各个供应厂商硬件和系统的差异，从而实现不依存于硬件的系统构成。同时利用一种叫做Variant的数据型，可以不依存于硬件中固有数据型，按照应用程序要求提供的数据格式。

利用OPC使接口标准化可以构成如图 1-3所示的系统。用户可以不依存于设备A，B，C，D的内部结构及它的供应厂商，选用监视，趋势图以及表报应用程序。

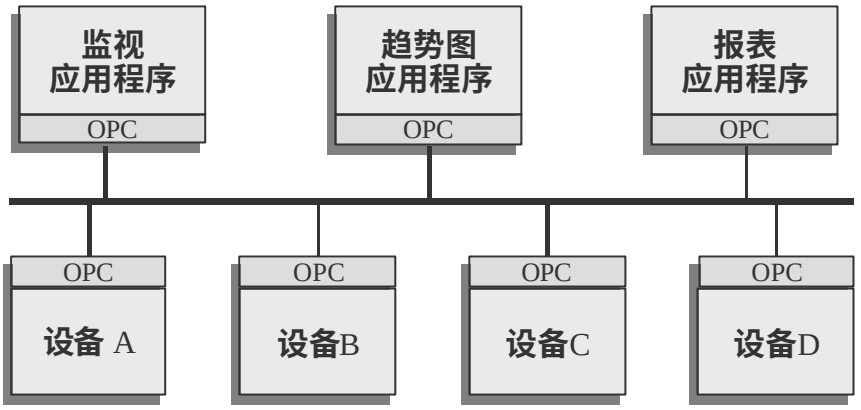


图 1-3 利用OPC的控制系统构成

1.1.3 OPC的历史

早期的OPC标准是由提供工业制造软件的５家公司所组成的OPC特别工作小组所开发的。Fisher-Rosement，Intellution，Rockwell Soft well，Intuitive Technology以及Opto22早在1995年开发了原始的OPC标准，微软同时作为技术顾问给予了支持。

OPC基金会（OPC Foundation，OPC-F），是在1996年9月24日在美国的达拉斯举行了第一次理事会,并在同年10月7日在美国的芝加哥举行了第一次全体大会上宣告正式成立的。之后为了普及和进一步改进于1996年8月完成的OPC数据访问标准版本1.0，开始了全球范围的活动。现在的OPC基金会的理事会是由Fisher-Rosement，Honeywell，Intellution，Rockwell Software，National Instrument以及欧洲代表的Siements和远东代表的东芝所组成。

在日本为响应以美国为中心的国际标准活动，由１１家公司作为发起人，于1996年6月开始基金会成立的准备活动，并于1996年10月17日正式成立了日本OPC协会（OPC-J）。几乎与其同时欧洲OPC协会（OPC-E）也相继成立。在中国也于由５家公司作为发起人于2001年12月正式成立了中国OPC促进会（OPC-C）。

OPC基金会从成立开始会员逐年增加，到目前为止在全球范围内已有近３００家公司加入了这个国际标准组织。同时由控制设备厂商和控制软件供应商提供的OPC产品也日益增加，目前已有６００种以上的OPC服务器产品和OPC应用程序产品出现在由OPC基金会发行的OPC产品目录上。

1.1.4 OPC现状和发展

现存的和正在开发的OPC的标准如表1-1所示。

表 1-1 OPC标准

标准	版本	内容

Data Access	1.0A，2.0 3.0正在开发	数据访问的标准
Alarm and Events	1.0 2.0正在开发	警报和事件的标准
Historical Data Access	1.0	历史数据访问的标准
Batch	1.0	批处理的标准
Security	1.0	安全性的标准
Compliance	1.0	数据访问标准的测试工具
OPC XML	正在开发	过程数据的XML标准
OPC Data eXchage	正在开发	服务器间数据交换的标准

本书是依据OPC数据访问标准（版本2.0），利用微软Visual Basic（以下简称为VB）或者应用程序的Visual Basic（以下简称为VBA）开发客户应用程序的入门书。本书以后所提到的“OPC标准”和“OPC服务器”，其实是指OPC数据访问标准（版本2.0）和OPC数据访问服务器（版本2.0），而“OPC应用程序”是指OPC数据访问客户应用程序。

1.2 什么是OPC

1.2.1 基于COM技术的OPC

微软公司为了提供商业应用程序和特定用途的软件包间的相互连接性，开发了所谓的组件对象模型（Component Object Model, COM）技术。COM是一种软件组件间相互数据交换的有效方法。COM技术具有以下特长：

- 所谓COM并不是一种计算机语言，而是于运行的机器（只要互相连接），机器的操作系统（只要支持COM），以及软件开发的语言无关，任意的两个软件组件之间都可以相互通信的二进制和网络的标准。
- COM服务器是根据COM客户的要求提供COM的服务的执行可能的程序，可以作为Win32上执行可能的文件发布。
- COM客户程序和COM服务器可以用完全不同的语言开发。这样使利用C++，Visual Basic，以及Excel中作为宏使用的应用程序的Visual Basic等不同语言所开发的程序可以相互连接。
- COM组件可以以二进制的形式发布给用户。
- 与过去DLL的版本管理非常困难的问题相比，COM技术可以提供不同版本的COM服务器和COM客户程序之间的最大的兼容性。
- 作为COM技术扩展的分布式COM（Distributed Component Object Model, 分布式COM）技术，更可以使COM组件分布在不同的计算机上，并通过网络互相连接并互相交换数据。所以对于COM客户程序来说，同样像连接本地计算机上的COM服务器一样，去连接远程计算机上的COM服务器，当然通信的速度不太一样，但是重要的是不必对服务器程序进行修正就可以在网络上自由构成（图1-4）。

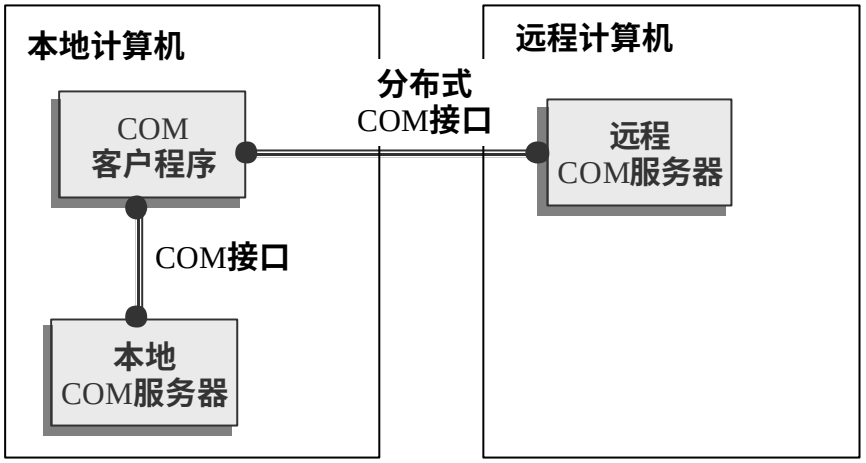


图 1-4利用COM和分布式COM达成的组件间的互相连接

COM技术的出现使简单地实现控制设备和控制管理系统之间的数据交换提供了技术基础。但是如果不提供一个工业标准化的COM接口，各个控制设备厂家开发的COM组件之间的相互连接仍然是不可能的。这样的工业标准的提供，乃是OPC的目的所在。总而言之，OPC是作为工业标准定义的特殊的COM接口。

OPC是以提供移植容易并具有可以满足大多数设备厂家要求的灵活性和高水平的机能性为目标而开发的，对于制造厂商和用户来说，分别可以从OPC得到以下的实惠：

- 设备开发者：可以使设备驱动器开发的单一化成为可能。
- 应用程序软件开发者：可以使用通用的开发工具。不必开发特别的接口，使得设备接口的开发更为简单易行。
- 用户：可以选用各种各样的商业软件包，使得系统构成的成本大为降低。同时可以更加容易地实现由不同供应厂商提供的设备所混合构成的工业控制系统。

随着基于OPC标准的控制组件的推广和普及，不仅使控制系统的增设和组件的置换就更加简单，而且使过程数据的访问也变得容易。比如，过程控制程序可以直接和数据分析软件包或电子表格应用程序连接，从而达成高度的工厂控制系统的信息化。

你可能经常听说过COM的编程非常困难，需要高度的计算机知识。确实对于OPC服务器开发者来说，对COM的理解和掌握是必不可少的。但是本书的目的是讲述怎样利用VB或者VBA开发OPC应用程序，所以我们并不要求读者必须掌握COM的知识。一般来说，只要懂得Visual Basic的基本使用方法，进行OPC应用程序的编程则是完全可能的。

1.2.2 OPC和DDE的比较

在OPC技术出现以前，DDE（Dynamic Data Exchange）技术曾经对过程控制作出巨大贡献。但是DDE是基于Windows的信息（Message）传递而建立的技术，所以DDE技术存在以下问题：

- 数据的传送速度较慢
- 没有安全性管理机制
- 开发困难

- 功能缺乏柔软性
- 可靠性也难以令人满意

所以基于先进的COM技术的OPC技术将逐渐取代现在在过程控制中广泛使用的DDE的位置乃是顺理成章的事情。随着OPC技术的导入，和过去的DDE技术相比，在以下方面显示出它的优越性：

- 高速的数据传送性能
- 基于分布式COM的安全性管理机制
- 开发成本的降低
- 实现具有高度柔软性功能的系统
- 实现具有高可靠性的系统

图 1-5是分别利用OPC和DDE进行数据传送性能的实验结果的例子。从这里也可以看出OPC技术在传送速度上的优越性。

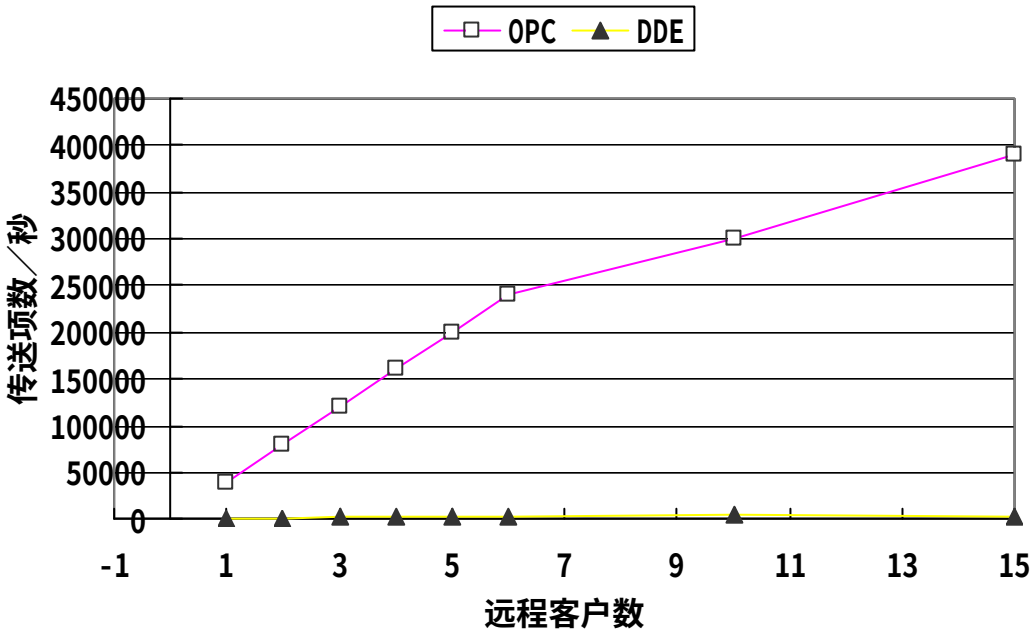


图 1-5 利用OPC和DDE的数据传送性能的实验结果

1.2.3 OPC适用于哪些地方

OPC是为了连接数据提供源（OPC服务器）和数据的使用者（OPC应用程序）之间的软件接口标准。

数据提供源可以是PLC，DCS，条形码读取器等控制设备。随控制系统构成的不同，作为数据提供源的OPC服务器即可以是和OPC应用程序在同一台计算机上运行的本地OPC服务器，也可以是在另外的计算机上运行的远程OPC服务器。

如图 1-6所示，OPC接口既可以适用于通过网络把最下层的控制设备的原始数据提供给作为数据的使用者（OPC应用程序）的HMI／SCADA，批处理等自动化程序，以至更上层的历史数据库等应用程序,也可以适用于应用程序和物理设备的直接连接。所以OPC接口是适用系统的很多场合的具有高度柔软性的接口标准。

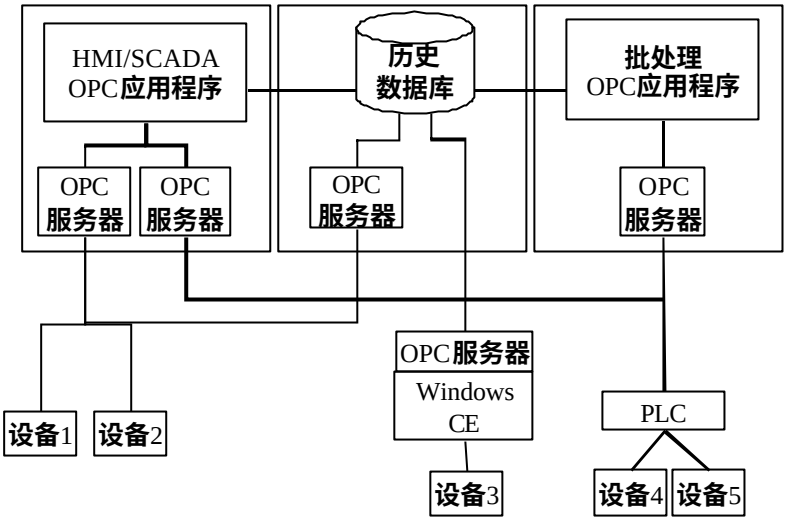


图 1-6 在控制系统中OPC所占的位置

1.3 OPC的概要

1.3.1 OPC服务器和包装DLL

利用VB开发的OPC应用程序的系统主要由下列软件组件构成：

1) OPC服务器：OPC服务器一般是由DCS或者I/O驱动器等硬件供应商，或者由独立的软件供应商提供，即可以在与应用程序计算机相同的本地计算机上运行，也可以在与应用程序计算机不同的远程计算机上运行。OPC服务器的主要目的是提供过程数据。
2) OPC代理－占位DLL：因为VB或者VBA的OPC应用程序是运行在与OPC服务器不同的计算机进程空间，所以不能直接调用OPC服务器的接口进行数据交换，需要通过代理－占位DLL并利用操作系统提供的通信功能进行数据交换。按照COM的术语，代理是起着代表别的组件作用的组件的意思。首先在应用程序侧，代理把向OPC服务器接口传递的数据进行格式变换。然后在OPC服务器侧，占位把OPC客户程序送过来的数据的格式变换解除，同时也对返回到客户程序的数据进行格式变换。实际的OPC代理－占位DLL是同一个DLL。随着被设置的计算机的不同，起着代理的作用或者起着占位的作用。
3) OPC自动化包装DLL： 一般来说，为了达到数据传送的最高的性能，OPC服务器是用C++开发的，并只提供定制接口。与此相反，用VB或VBA（应用程序的Visual Basic）等语言开发的应用程序却要求OPC自动化接口。为了让VB或者VBA的客户应用程序可以使用OPC自动化接口，使用OPC自动化包装将OPC自动化接口变换成OPC定制接口，从而可以对OPC服务器进行访问。
4) OPC应用程序：对由OPC服务器提供的数据源进行访问，实现用户的特定目的而开发的应用程序。

利用VB的OPC服务器的配置形态，即可以是和OPC应用程序在同一台计算机内而在别的进程中运行的本地OPC服务器，也可以是通过网络在另外的计算机上运行的远程OPC服务器。图 1-7和图 1-8分别表示本地和远程OPC服务器的配置形态。

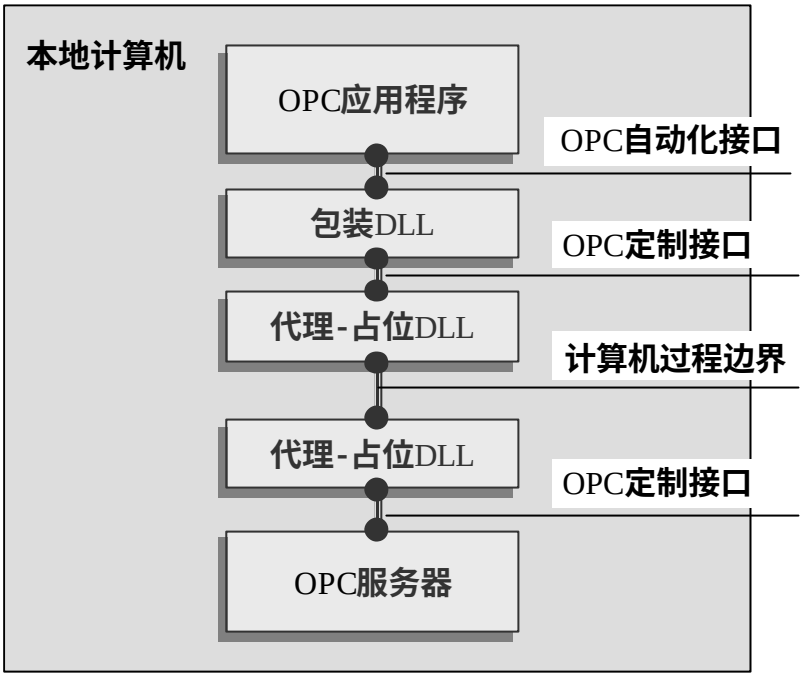


图 1-7 本地OPC服务器

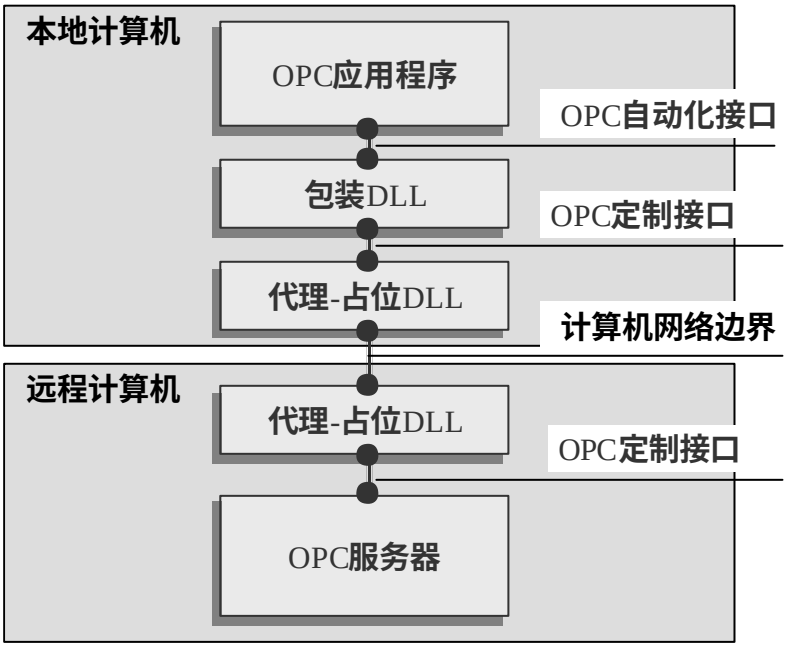


图 1-8 远程OPC服务器

OPC服务器，OPC代理－占位DLL以及OPC包装DLL应该由OPC服务器供应商（控制设备制造商或者独立软件供应商）提供的，并不在本书的说明范围之内。本书的目的主要是解说怎样利用VB或者VBA开发OPC的应用程序。

1.3.2 OPC的主要功能

OPC的数据访问方法主要有同步访问和异步访问两种。

对于如图 1-9所示的同步访问，OPC服务器把按照OPC应用程序的要求得到的数据访问结果作为方法的参数返回给OPC应用程序，OPC应用程序在结果被返回为止一直必须处于等待状态。

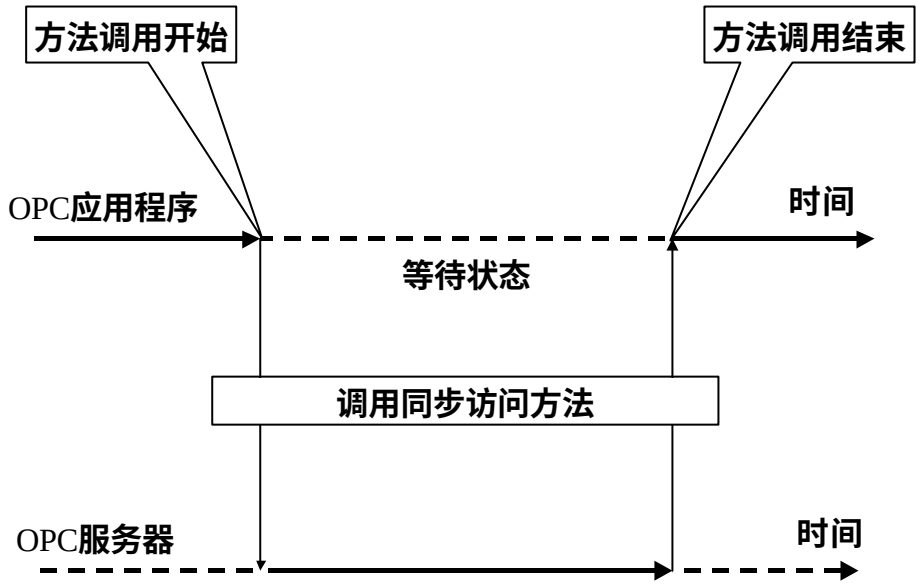


图 1-9 同步数据访问处理

与此相对，如图 1-10所示的异步访问，OPC服务器接到OPC应用程序的要求后，几乎立即将方法返回。OPC应用程序随后可以进行其他处理。当OPC服务器完成数据访问时，触发OPC应用程序的异步访问完成事件，将数据访问结果传送给OPC应用程序。OPC应用程序在VB的事件处理程序中接受从OPC服务器传送来的数据。

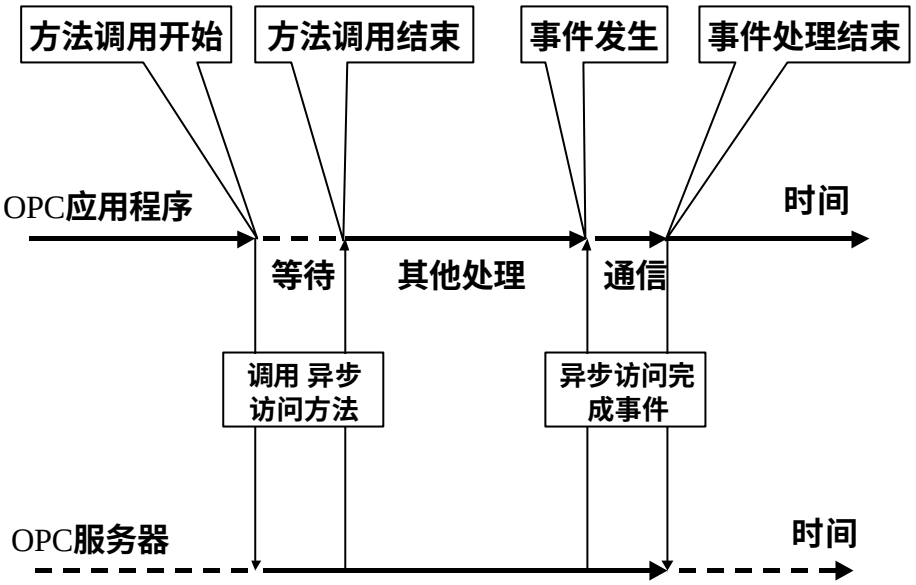


图 1-10 异步数据访问处理

除了上述的同步和异步数据访问以外，还有如图1-11**所示的并不需要**OPC**应用程序向**OPC**服务器要求，就可以自动接到从**OPC**服务器送来的变化通知的订阅方式数据采集（**Subscription**）。服务器按一定的更新周期（**UpdateRate**）更新**OPC**服务器的数据缓冲器的数值时，如果发现数值有变化时，就会以数据变化事件（**DataChange**）通知**OPC**应用程序。如果**OPC**服务器支持不敏感带（**DeadBand**），而且**OPC**标签的数据类型是模拟量的情况，只有现在值与前次值的差的绝对值超过一定限度时，才更新缓冲器数据并通知**OPC**应用程序。由此可以无视模拟值的微小变化，从而减轻**OPC**服务器和**OPC**应用程序的负荷。**

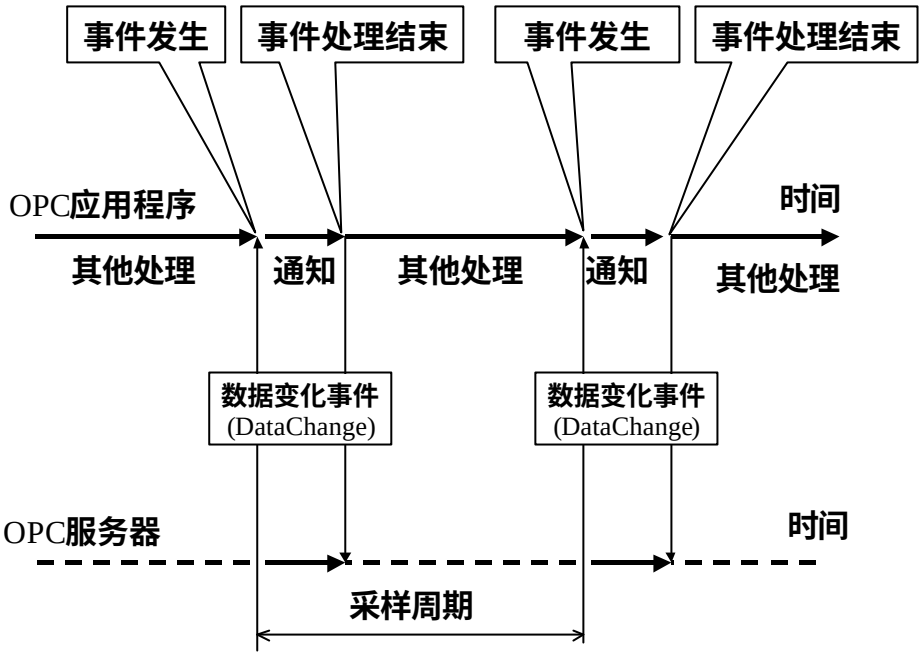


图 1-11 订阅方式数据采集

上述的OPC数据访问的功能可总结成表1-2。这些功能并非必须全部实现，其中有一部分功能是选用的。这些选用功能是否被支持将随供应厂商的具体的服务器类型而定。

表 1-2 OPC数据访问的功能

功能	方式	说明
过程数据读取 ※ 这是独立的互不影响的四种读取方式。	同步读取	读取指定OPC标签对应的过程数据。 应用程序一直等待到读取完成为止。
	异步读取	读取指定OPC标签对应的过程数据。 应用程序发出读取要求后立即返回，读取完成时发生读取完成事件，OPC应用程序被调出。
	刷新	读取所有活动的OPC标签对应的过程数据。 应用程序发出更新要求后立即返回，更新完成时发生数据变化事件，OPC应用程序被调出。
	订阅方式数据采集（Subscription）	服务器用一定的周期检查过程数据，如发现数据变化超过一定的幅度时，则更新数据缓冲器，并自动通知OPC应用程序。
过程数据写入	同步写入	写入指定OPC标签对应的过程数据。 应用程序一直等待到写入完成为止。

	异步写入	写入指定OPC标签对应的过程数据。应用程序发出写入要求后立即返回，写入完成时发生写入完成事件，OPC应用程序被调出。

同步和异步数据访问的特征可总结如表1-3。OPC应用程序的开发者可按照应用程序的用途和目的选择合适的数据访问方法。

表 1-3 OPC同步和异步数据访问的特征

特征	同步访问	异步访问
访问性能	因为在访问完成之前应用程序必须一直在等待，尤其大量数据的访问或直接向设备的访问对访问性能的影响很大。	因为在访问完成之前应用程序不必等待，可以并行处理，对访问性能的影响不大。
程序开发	处理程序比较简单，开发容易。	因为发出要求和访问完成事件处理是分别进行的，所以必须有事务（Transaction）识别功能，开发比较难。
远程连接的分布式COM设置	只要分布式COM启动权限和访问权限就可以运行，设置比较简单。	除了分布式COM启动权限和访问权限以外，还必须设置身份标志，设置比较复杂。

1.4 VB的对象

象前面说明过的那样，OPC应用程序的开发并不需要深奥的COM知识，只要理解Visual Basic使用方法就可以进行。这里先说明一下有关Visual Basic对象的基础知识和一种作为特殊的Visual Basic对象的叫做集合的概念。

1.4.1 Visual Basic对象

Visual Basic对象是可以作为一个单位处理的代码和数据的组合。Visual Basic对象支持分别叫做“属性”，“方法”，和“事件”的要素。在Visual Basic里，表示对象特征的数据（设置值） 叫做属性，对于对象可实施的各种各样的处理程序叫做方法。事件是对象可认识的外部变化，对应这样的变化事件，对象可以用代码记述对应外部变化的处理程序。

为了变更一个对象的特征，可以改变这个对象的属性的设置值。例如，电话的属性之一是电话号。比如用Visual Basic对象定义时，可以定义一个叫做“Phone”的对象（电话）具有一个叫做“TelNumber”的属性（电话号）。如果改变这个“TelNumber”属性的设置值，就意味着变更这个电话的号码（图1-12）。

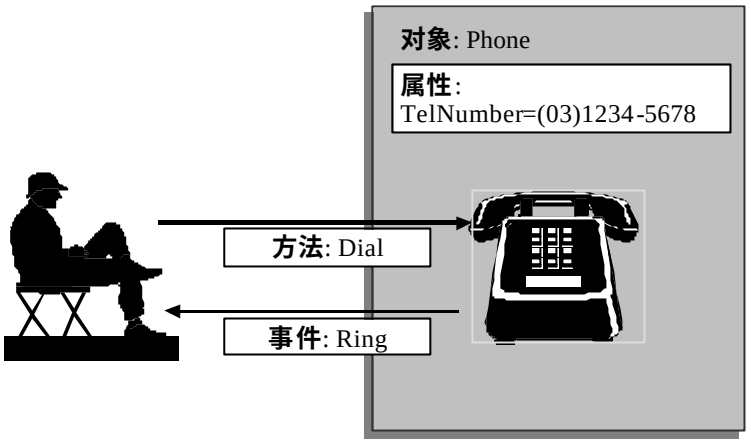


图 1-12 对象的属性，方法和事件

在使用Visual Basic**设置自动化接口对象的属性时，应使用**
<**对象变量**>.<**属性名**> = <**值**>
的方法记述。如果，从对象读取属性值时，应使用
<**保存变量**> = <**对象变量**>.<**属性名**>
的方法记述。
有了这样的记述方法，应用程序就可能简单地访问自动化接口对象。例如，用 Visual Basic**的语言描述刚才的电话对象时，可以在程序里用下列的代码记述。**

```
Dim MyPhone As Phone
MyPhone.TelNumber= "03-1234-5678"
```

在对象支持的要素中，除了属性之外还有方法。方法也是对象的一部分。一般来说，属性是用于设置和读取代表某种对象特征的设置值，而方法是用于进行某种操作。例如，打電話就是一种方法。如用 Visual Basic**定义时，可以是叫做“** Phone **”的对象（电话）具有一个叫做“**Dial**”的方法 （打电话）。**
在使用Visual Basic**调用自动化接口对象的方法时，应使用**
<**对象变量**>.<**方法名**>.（<**参数**>）
的方法记述。
例如，用 Visual Basic**进行打电话的操作，可以用下列的代码以通话对方的电话号“**045-5555-1111**”为参数，调用“**Dial**”方法。**

```
Dim MyPhone As Phone
MyPhone.Dial（"045-5555-1111"）
```

此外对象还具有一个叫做事件的要素。事件是由在对象对应发生的某种外部变化而启动的处理程序。例如，对于电话可以有一个叫做“Ring**”（来电话）的事件。为了使对象可以处理事件，首先必须在**Visual Basic**的对象变量声明文加上**WithEvents**。然后为了事件可以被对象处理，再应该准备好下列的事件处理代码。**

```
Dim WithEvents MyPhone As Phone
.....
Sub MyPhone_Ring()
‘ 这里记述接电话处理代码。
.....
```

End Sub

1.4.2 VB的集合对象

集合对象是由Visual Basic提供的一种特殊的对象。集合对象是一种可以集约多个的任意的要素的对象。与其他对象一样，也可以用New关键词生成。例如，把办公室里所有电话定义成电话的集约时，可以记述如下。

```
Dim OfficePhones As Phones
Set OfficePhones = New Phones
```

所有的集合对象都支持添加，清除和检索集合里的项目的方法。

属性：Count

Count 属性是返回集合的项目数。是只读的属性。例如，读取办公室电话集合里所有的电话的数目时，可记述如下。

```
Dim NumPhones As Long
Set NumPhones = OfficePhones.Count
```

方法：Add

Add 方法用于向集合里添加新项目。例如，向办公室电话集合里添加一个以“ Phone#1 ”为键的新电话MyPhone时，可记述如下。

```
Dim MyPhone As Phone
Set MyPhone = OfficePhones.Add("Phone#1")
```

方法：Remove

Remove 方法用于从集合里按照索引或者键清除指定的项目。例如，从办公室电话集合里清除“ Phone#1 ”的电话时，可记述如下。

```
OfficePhones. Remove("Phone#1")
```

方法：Item

Item 方法用于从集合里按照索引或键返回的指定项目。 例如，取得键为“ Phone#1 ”的电话时，可记述如下。

```
Dim MyPhone As Phone
Set MyPhone = OfficePhones.Item("Phone#1")
```

取得索引为1的电话时，可记述如下。

```
Dim MyPhone As Phone
Set MyPhone = OfficePhones.Item(1)
```

1.5 OPC的对象

OPC应用程序应该首先生成OPC服务器支持的OPC对象，然后就可以使用OPC对象支持的属性和方法，对其进行简单的操作。这种结构使得应用程序可以象使用自己支持的数据和功能一样，去使用服务器对象支持的数据和功能。例如，如图 1-13所示，OPC应用程序可以取得OPC服务器支持属性的ServerState即服务器执行状态，还可以调用OPC服务器支持的方法Connect()以和服务器连接。

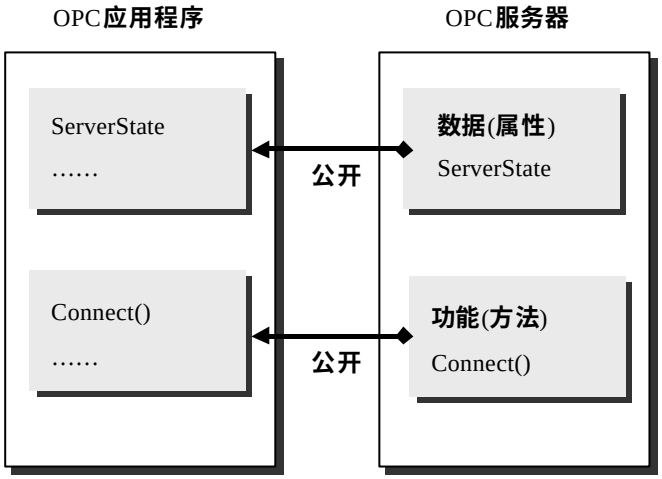


图 1-13 OPC应用程序和OPC服务器

注意一个OPC应用程序可以与多个的服务器同时连接，同时一个OPC服务器也可以同时被多个的OPC应用程序连接。

1.5.1 OPC对象的分层结构

OPC数据访问提供从数据源读取和写入特定数据的手段。OPC数据访问对象是由象图 1-14所示的分层结构构成。即一个OPC服务器对象（OPCServer）具有一个作为子对象的OPC组集合对象（OPCGroups）。在这个OPC组集合对象里可以添加多个的OPC组对象（OPCGroup）。各个OPC组对象具有一个作为子对象的OPC标签集合对象（OPCItems）。在这个OPC标签集合对象里可以添加多个的OPC标签对象（OPCItem）。此外，作为选用功能，OPC服务器对象还可以包含一个OPC浏览器对象（OPCBrowser）。

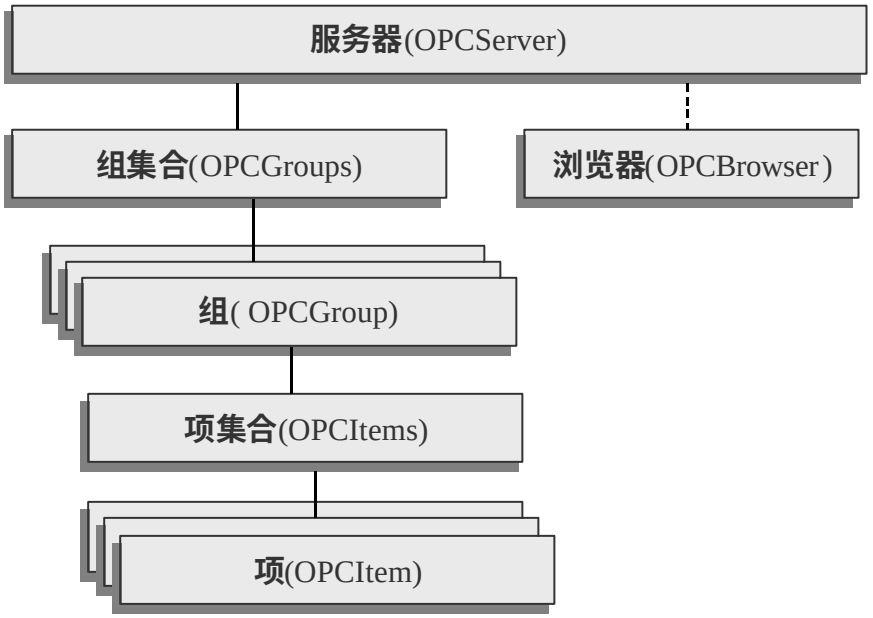


图 1-14 OPC数据访问对象的分层结构

OPC对象中的最上层的对象是OPC服务器。一个OPC服务器里可以设置一个以上的OPC组。OPC服务器经常对应于某种特定的控制设备。例如，某种DCS控制系统，或者某种PLC控制装置。

OPC组是可以进行某种目的数据访问的多个的OPC标签的集合，例如某监视画面里所有需要更新的位号变量。正因为有了OPC组，OPC应用程序就可以以同时需要的数据为一批的进行数据访问，也可以以OPC组为单位启动或停止数据访问。此外OPC组还提供组内任何OPC标签的数值变化时向OPC应用程序通知的数据变化事件（表1-4）。

表 1-4 OPC数据访问对象模型

名称	对象名	说明
OPC服务器	OPCServer	OPC服务器对象在使用其他OPC对象前必须生成。OPC服务器自动含有一个OPC组集合对象，并可在其基础上生成一个OPC浏览器对象。
OPC组集合	OPCGroups	OPC服务器中添加的所有的OPC组的集合。
OPC组	OPCGroup	OPC组对象是用于组的状态管理以及利用项集合为单位的数据访问。
OPC标签集合	OPCItems	在对应OPC组中添加的所有的OPC标签的集合。
OPC标签	OPCItem	含有OPC标签的定义，现在值，状态，以及最后更新时间等信息的对象。
OPC浏览器	OPCBrowser	用于浏览OPC服务器的名称空间的对象。

1.5.2 OPC标签

OPC对象里最基本的对象是OPC标签。OPC标签是OPC服务器可认识的数据定义，通常相当于位号的单一变量（调整点或过程数据），并和数据提供源（控制设备）相连接。OPC标签具有多个属性，但是其中最重要的属性是OPC标签标识符。项标识符是在控制系统中可识别OPC标签的字符串，例如，

DCS的例：”TIC100.PV"

PLC的例："COM1.STATION:42.REG:40001;0,4095,-100,+1234.0"

1.5.3 服务器句柄

一旦OPC组或者OPC标签在OPC服务器里添加成功，OPC服务器将赋予被添加的各个OPC组或者各个OPC标签一个独特的标识符。这个标识符叫做服务器句柄。被赋予的服务器句柄将返回给OPC应用程序。

OPC应用程序应该将由OPC服务器返回的OPC组或者OPC标签的服务器句柄好好保管。因为随后对添加的OPC组或者OPC标签进行操作时，只有使用这些服务器句柄才可以唯一地识别特定的OPC组或者OPC标签。

1.5.4 OPC服务器对象

因为OPC服务器对象OPCServer提供连接数据源（OPC定值接口服务器）以及数据访问（读取· 写入）的方法，所以在建立OPC组和OPC标签以前必须建立OPC服务器对象，然后使用OPC数据访问自动化接口的“ Connect ”方法和数据源连接。本节说明一些经常使用的OPC服务器的属性和方法。本节的示例程序需要表1-5的示例基础。

表 1-5 OPC服务器的示例基础

示例基础	Dim WithEvents AnOPCServer As OPCServer Set AnOPCServer = New OPCServer

属性：ServerState

说明	**只读的属性，返回服务器的运行状态（**OPCServerStatus**）。**
文法	ServerState As Long **下列的**OPCServerStatus**值是可能的：** OPCRunning：OPC**服务器正在正常运转。** OPCFailed：OPC**服务器由于异常而停止。** OPCNoconfig：OPC**服务器正在运转，但没有被设置。** OPCSuspended：OPC**服务器正处于暂时停止状态。** OPCTest：OPC**服务器正在实验模式下运转。** OPCDisconnected：**服务器对象没有连接任何实际的**OPC**服务器。**
示例	Dim ServerState As Long ServerState = AnOPCServer.ServerState

属性：OPCGroups

说明	**只读的属性，**OPC**组的集合。这是**OPCServer**的默认属性。**
文法	OPCGroups As OPCGroups
示例	Dim MyGroups As OPCGroups **‘ 使用显然的属性声明** Set MyGroups = AnOPCServer.OPCGroups **‘ 或使用默认的属性声明** Set Mygroups = AnOPCServer

方法：Connect

说明	**连接**OPC**数据访问服务器。**
文法	Connect (ProgID As String, Optional Node As Variant) ProgID: **程序标识符是可以识别特定**OPC**服务器的注册字符串。** Node: **选用参数，是利用分布式**COM**进行远程连接的计算机**UNC**名称（例如，**"Server"**）或者**DNS**名称（例如，**www.vendor.com**或者**"180.151.19.75"**）。被省略时，将连接本地**OPC**服务器。**
示例	Dim ProgID As String, NodeName As String ProgID = "VendorX.DataAccessCustomServer" NodeName = "SomeComputerNodeName" AnOPCServer.Connect (ProgID, NodeName)

方法：Disconnect

说明	**断开和**OPC**服务器的连接。**OPC**应用程序在断开和**OPC**服务器的连接前，建议显式的清除所有添加的**OPC**组和**OPC**标签的程序，虽然调用本方法也可以黯然地清除所有的**OPC**组，并释放所有的引用。**
文法	Disconnect ()

示例	AnOPCServer.Disconnect

事件：ServerShutDown

说明	这个事件在服务器即将关机前发生，OPC服务器以此事件通知OPC应用程序预告即将关机。OPC应用程序应该在接到此事件通知后，立即清除所有的OPC组并断开与OPC服务器连接。
文法	ServerShutDown (Reason As String) ServerReason是选用的参数，是说明服务器关机理由的字符串。
示例	Dim WithEvents AnOPCServer As OPCServer Dim ProgID As String, NodeName As String ProgID = "VendorX.DataAccessCustomServer" NodeName = "SomeComputerNodeName" AnOPCServer.Connect (ProgID, NodeName) Private Sub AnOPCServer_ServerShutDown(ByRef aServerReason As String) ‘ 在这儿记述服务器关机处理代码 Debug.Print aServerReason AnOPCServer. OPCGroups.RemoveAll AnOPCServer.Disconnect End Sub

所有OPC服务器支持的属性,方法和事件的一览如表1-6，表1-7和表1-8所示。详细资料请参照6.5参考资料的“OPC数据访问自动化接口标准”。

表 1-6 OPC服务器的属性

属性名	属性	说明
StartTime	只读	OPC服务器的启动时间（国际标准时间）。
CurrentTime	只读	OPC服务器的现在时间（国际标准时间）。
LastUpdateTime	只读	对于本OPC应用程序的最后数据更新时间（国际标准时间）。
MajorVersion	只读	OPC服务器软件的主版本号。
MinorVersion	只读	OPC服务器软件的次版本号。
BuildNumber	只读	OPC服务器软件的版本生成号。
VendorInfo	只读	开发者提供的有关OPC服务器软件版本信息的文字串。（建议含有公司名以及所支持的设备类型）
ServerState	只读	返回OPC服务器的运行状态（OPCServerStatus）。
LocaleID	读写	返回区域标识符（语言标识符）。
BandWidth	只读	OPC服务器的特有值。返回OPC服务器使用可能的不敏感带的百分比（%）。
OPCGroups	只读	OPC的组集合。服务器的默认属性。
PublicGroupNames	只读	返回OPC服务器的OPC公用组名称。
ServerName	只读	返回OPC服务器的名称。
ServerNode	只读	返回OPC服务器的计算机名。
ClientName	读写	OPC应用程序在OPC服务器中的注册名。用于调试。

表 1-7 OPC服务器的方法

方法名	说明
GetOPCServers	返回注册的OPC服务器的程序标识符（ProgID）。
Connect	连接OPC数据访问服务器。
Disconnect	断开和OPC服务器的连接。
CreateBrowser	生成OPC浏览器对象。
GetErrorString	得到按区域标识符（LocaleID）指定的错误码文字说明。
QueryAvailableLocaleIDs	询问在服务器和应用程序之间使用可能的区域标识符（LocaleID）。
QueryAvailableProperties	对于指定的项属性，询问可能取得的项属性的标识符和文字说明。
GetItemProperties	按照指定的OPC标签和属性标识符，取得项属性的现在值。
LookUpItemIDs	按照指定的OPC标签和属性标识符，取得与项属性值对应的OPC标签标识符。

表 1-8 OPC服务器的事件

事件	说明
ServerShutDown	这个事件在服务器即将关机前发生，OPC服务器以此事件通知OPC应用程序预告即将关机。OPC应用程序应该在接到此事件通知后，立即清除所有的OPC组并断开与服务器连接。

1.5.5 OPC组集合对象

OPC组集合对象OPCGroups是OPC组的集合，这个对象的用途是添加，清除和管理OPC组。本节说明一些经常使用的OPC组集合的方法。本节的示例程序需要表1-9的示例基础。

表 1-9 OPC组集合的示例基础

示例基础	Dim WithEvents AnOPCServer As OPCServer Dim ProgID As String Dim NodeName As String Dim MyGroups As OPCGroups Dim OneGroup As OPCGroup Set AnOPCServer = New OPCServer ProgID = "VendorX.DataAccessCustomServer" NodeName = "SomeComputerNodeName" AnOPCServer.Connect(ProgID, NodeName) Set MyGroups = AnOPCServer.OPCGroups

方法：Add

说明	在OPC组集合中建立新的OPC组。
文法	Add(Optional Name As Variant) As OPCGroup Name是独特的OPC组名。这是选用参数，没有被定义时，由OPC服务器自动产生组名。
示例	Set OneGroup = MyGroups.Add("AnOPCGroupName")

方法：Remove

说明	清除指定的OPC组。
文法	Remove(ItemSpecifier As Variant) ItemSpecifier是要清除的OPC组的服务器句柄或者OPC组名。
示例	Set OneGroup = MyGroups.Add("AnOPCGroupName") ‘ 使用OPC组名的处理代码 MyGroups.Remove("AnOPCGroupName") ‘ 或者使用组服务器句柄的处理代码 MyGroups.Remove(OneGroup.ServerHandle)

方法：RemoveAll

说明	为服务器关机作准备，清除所有的OPC组和OPC标签。
文法	RemoveAll()
示例	Set OneGroup = MyGroups.Add("AnOPCGroupName") Set OneGroup = MyGroups.Add("AnOPCGroupName1") Set OneGroup = MyGroups.Add("AnOPCGroupName2") MyGroups.RemoveAll

所有OPC组集合支持的属性，方法和事件的一览如表1-10，表1-11和表1-12所示。详细资料请参照6.5参考资料的“ OPC数据访问自动化接口标准 ”。

表 1-10 OPC组集合的属性

属性名	属性	说明
Parent	只读	返回所属OPC服务器对象。
DefaultGroupIsActive	读写	新添加的OPC组的活动状态的默认值。初期值是活动状态。
DefaultGroupUpdateRate	读写	新添加的OPC组的更新周期的默认值（毫秒）。初期值是1000毫秒。
DefaultGroupDeadBand	读写	新添加的OPC组的不敏感带的默认值（%）。初期值是0%。
DefaultGroupLocaleID	读写	新添加的OPC组区域标识符的默认值。
DefaultGroupTimeBias	读写	新添加的OPC组的时间偏差的默认值（分钟）。初期值是0分。
Count	只读	集合对象的固有属性。包含的组数。

表 1-11 OPC组集合的方法

方法名	说明
Item	OPC组集合的默认方法。返回由集合索引（ItemSpecifier）指定的OPC组对象。
Add	在OPC组集合中添加新的OPC组。
GetOPCGroup	返回指定的OPC组对象。
Remove	清除指定的OPC组。
RemoveAll	为服务器关机作准备，清除所有的OPC组和OPC标签。
ConnectPublicGroup	连接OPC公用组。
RemovePublicGroup	清除OPC公用组。

表 1-12 OPC组集合的事件

事件	说明
AllGroupsDataChange	由多个OPC组的数据变化而引发的事件

1.5.6 OPC组对象

OPC组对象OPCGroup提供满足OPC应用程序要求的数据访问手段。本节说明一些经常使用的OPC服务器的属性，方法和事件。本节的示例程序需要表1-13的示例基础。

表 1-13 OPC组的示例基础

示例基础

```
Dim WithEvents AnOPCServer As  OPCServer
Dim ProgID As String
Dim NodeName As String
Dim MyGroups As OPCGroups
Dim WithEvents OneGroup As OPCGroup
Dim AnOPCItemCollection As OPCItems
Dim AnOPCItem As OPCItem
Dim ClientHandles(10) As Long
Dim AnOPCItemIDs(10) As String
Dim AnOPCItemServerHandles() As Long
Dim AnOPCItemServerErrors() As Long
Dim AddItemCount As Long
Dim I As Long

Set AnOPCServer = New OPCServer
ProgID = "VendorX.DataAccessCustomServer"
NodeName = "SomeComputerNodeName"
AnOPCServer.Connect(ProgID, NodeName)

Set MyGroups = AnOPCServer.OPCGroups
Set OneGroup = MyGroups.Add("AnOPCGroupName" )

Set AnOPCItemCollection = OneGroup.OPCItems
AddItemCount = 10
For  I = 1 To AddItemCount
          ClientHandles(I) = I + 1
          AnOPCItemID(I) = "Item_"  &  I
Next
‘ 添加OPC标签
AnOPCItemCollection.AddItems AddItemCount, AnOPCItemIDs, _
    ClientHandles, AnOPCItemServerHandles, AnOPCItemServerErrors
```

属性：IsActive

说明	可设置的属性，用以控制OPC组的活动状态。只有处于活动状态的OPC组才进行定期的数据更新。非活动状态的OPC组，除了在接到显然的数据读写要求外，并不收集任何数据。
文法	IsActive As Boolean
示例	Dim CurrentValue As Boolean ‘ 设置OPC组为活动状态

	OneGroup.IsActive = True **‘ 读取OPC组的活动状态** CurrentValue = OneGroup.IsActive

属性：IsSubscribed

说明	**可设置的属性，用以控制OPC组的订阅状态。进行订阅的OPC组可以自动收到从服务器送来的数据变化通知。**
文法	IsSubscribed As Boolean
示例	Dim CurrentValue As Boolean **‘ 设置OPC组为订阅状态** OneGroup.IsSubscribed = True **‘ 取得OPC组的订阅状态** CurrentValue = OneGroup.IsSubscribed

属性：ServerHandle

说明	**只读的属性，服务器句柄是由OPC服务器指定的，用于识别指定的OPC组的一个独特的长整型数。OPC应用程序可以利用这个服务器句柄，向OPC服务器要求对指定的OPC组进行操作，例如清除指定的OPC组。**
文法	ServerHandle As Long
示例	Dim CurrentValue As Long **‘ 读取OPC组的服务器句柄** CurrentValue = OneGroup.ServerHandle

属性：UpdateRate

说明	**可设置的属性，以毫秒为单位的数据更新周期。**
文法	UpdateRate As Long
示例	Dim CurrentValue As Long **‘ 设置OPC组的更新周期为 5 秒** OneGroup.UpdateRate = 5000 **‘ 取得OPC组的更新周期** CurrentValue = OneGroup.UpdateRate

属性：OPCItems

说明	**只读的OPC组的默认属性，OPC标签集合对象。**
文法	OPCItems As OPCItems
示例	**‘ 取得OPC标签集合对象** Set AnOPCItemCollection = OneGroup.OPCItems

方法：SyncRead

说明	**同步读取OPC组内单个或者多个OPC标签的数据值，质量标志和采样时间。**
文法	SyncRead(Source As Integer, NumItems As Long, _ ServerHandles() As Long, ByRef Values() As Variant, _ ByRef Errors() As Long, Optional ByRef Qualities As Variant, _ Optional ByRef, TimeStamps As Variant)

	Source **数据源。可以指定为**OPCCache**（缓冲器）或者**OPCDevice**（设备）。** NumItems **要读取的**OPC**标签的数目。** ServerHandles **要读取的**OPC**标签的服务器句柄的数组。** Values **返回的读取的数值的数组。** Errors **返回的与读取项对应的错误码的数组。** Qualities **选用参数，读取数值的质量标志的数组。** TimeStamps **选用参数，读取数据的采样时间的数组。**
示例	Dim NumItems As Long Dim ServerHandles(10) As Long Dim Values() As Variant Dim Errors() As Long Dim Qualities As Variant Dim TimeStamps As Variant Dim I As Long NumItems = 10 **‘ 设置要读取的**OPC**标签的服务器句柄** For I = 1 to NumItems ServerHandles(I) = AnOPCItemServerHandles(I) Next **‘ 同步读取** OneGroup.SyncRead OPCDevice, NumItems, _ ServerHandles, Values, Errors, Qualities, TimeStamps **‘ 表示读取**OPC**标签的数值** For I = 1 to NumItems Debug.Print I;Values(I) Next

方法：SyncWrite

说明	**同步写入**OPC**组内单个或者多个**OPC**标签的数据值。因为数据被直接同步地写入到设备中，所以只有等数据被设备接受或拒绝后，这个方法的调用才会结束。**
文法	SyncWrite(NumItems As Long, ServerHandles() As Long, _ Values() As Variant, ByRef Errors() As Long) NumItems **要写入的**OPC**标签的数目。** ServerHandles **要写入的**OPC**标签的服务器句柄的数组。** Values **要写入的数值的数组。** Errors **返回的与写入项对应的错误码的数组。**
示例	Dim NumItems As Long Dim ServerHandles() As Long Dim Values() As Variant Dim Errors() As Long Dim I As Long NumItems = 10 **‘ 设置要写入的**OPC**标签的服务器句柄** For I = 1 to NumItems ServerHandles(I) = AnOPCItemServerHandles(I)

	Values(I) = I * 2 Next ‘ 同步写入 OneGroup.SyncWrite NumItems, ServerHandles, Values, Errors ‘ 表示写入OPC标签的错误码 For I = 1 to NumItems Debug.Print I; Errors(I) Next

方法：AsyncRead

说明	**异步读取OPC组内单个或者多个OPC标签的数据值，质量标志和采样时间。利用异步数据访问时，必须将OPC组声明为可响应事件的对象变量（**Dim WithEvents xxx As OPCGroup**）。读取结果是由**AsyncReadComplete**事件返回。请注意因为本方法的数据是直接从设备中读取的，所以并不受到OPC组的活动状态的影响。**
文法	AsyncRead(NumItems As Long, ServerHandles() As Long, _ ByRef Errors() As Long, TransactionID As Long, ByRef CancelID As Long) NumItems **读取的OPC标签的数目。** ServerHandles **要读取的OPC标签的服务器句柄的数组。** Errors **返回的与读取项对应的错误码的数组。** TransactionID **由OPC应用程序发行的事务标识符。当数据访问完成后事件发生时，OPC应用程序可以利用这个事务标识符识别所完成的异步数据访问。** CancelID **由服务器发行的取消标识符。OPC应用程序使用这个标识符，可以取消正在进行中的异步数据访问。**
示例	Dim NumItems As Long Dim ServerHandles(10) As Long Dim Errors() As Long Dim ClientTransactionID As Long Dim ServerTransactionID As Long Dim I As Long NumItems = 10 ClientTransactionID = 1975 ‘ 设置要读取的OPC标签的服务器句柄 For I = 1 to NumItems ServerHandles(I) = AnOPCItemServerHandles(I) Next ‘ 异步读取 OneGroup.AsyncRead NumItems, ServerHandles, Errors, _ ClientTransactionID, ServerTransactionID

方法：AsyncWrite

说明	**异步写入OPC组内单个或者多个OPC标签的数据值。利用异步数据访问时，必须将OPC组声明为可响应事件的对象变量（**Dim WithEvents xxx As OPCGroup**）。写入结果是由**AsyncWriteComplete**事件返回。**
文法	AsyncWrite(NumItems As Long, ServerHandles() As Long, _ Values() As Variant, ByRef Errors() As Long, TransactionID As Long, _ ByRef CancelID As Long)

	NumItems 要写入的OPC标签的数目。 ServerHandles 要写入的OPC标签的服务器句柄的数组。 Values 要写入的数值的数组。 Errors 返回的与写入项对应的错误码的数组。 TransactionID 由OPC应用程序发行的事务标识符。当数据访问完成后事件发生时，OPC应用程序可以利用这个事务标识符识别所完成的异步数据访问。 CancelID 由服务器发行的取消标识符。OPC应用程序使用这个标识符，可以取消正在进行中的异步数据访问。
示例	Dim NumItems As Long Dim ServerHandles(10) As Long Dim Values() As Variant Dim Errors() As Long Dim ClientTransactionID As Long Dim ServerTransactionID As Long Dim I As Long NumItems = 10 ClientTransactionID = 1957 ‘ 设置要写入的OPC标签的服务器句柄 For I = 1 to NumItems ServerHandles(I) = AnOPCItemServerHandles(I) Values(I) = I * 2 Next ‘ 异步写入 OneGroup.AsyncWrite NumItems, ServerHandles, Values, Errors, _ ClientTransactionID , ServerTransactionID

事件：DataChange

说明	在OPC组内任何OPC标签的数据值或者质量标志变化时触发的事件。但不会在下次OPC组的更新周期（UpDateRate）以前发生。注意订阅方式数据采集（Subscription）和异步的数据刷新（AsyncRefresh）都可以触发这个事件，但是不同的是由订阅方式数据采集触发的事件返回的事务标识符为零（TransactionID = 0），而由异步数据刷新触发的事件返回的事务标识符非零（TransactionID ? 0）。
文法	DataChange (TransactionID As Long, NumItems As Long, _ ClientHandles() As Long, Values() As Variant, Qualities() As Long, _ TimeStamps() As Date) TransactionID 由OPC应用程序发行的事务标识符。事务标识符为零的是订阅方式数据采集（Subscription）的返回结果，而事务标识符非零的是异步的数据刷新（AsyncRefresh）的返回结果。 NumItems 读取的OPC标签的数目。 ClientHandles 读取的OPC标签的客户句柄的数组。 Values 返回的读取的数值的数组。 Qualities 读取的质量标志的数组。 TimeStamps 读取的采样时间的数组。
示例	Private Sub AnOPCGroup_DataChange (TransactionID As Long, _ NumItems As Long, ClientHandles() As Long, Values() As Variant, _ Qualities() As Long, TimeStamps() As Date) Dim I As Long

	‘ 表示读取OPC标签的客户句柄和数值 For I = 1 to NumItems Debug.Print I; ClientHandles(I); Values(I) Next End Sub

事件：AsyncReadComplete

说明	**在异步读取（**AsyncRead**）完成时发生的事件。**
文法	AsyncReadComplete (TransactionID As Long, NumItems As Long, _ ClientHandles() As Long, Values() As Variant, Qualities() As Long, _ TimeStamps() As Date, Errors() As Long) TransactionID **由**OPC**应用程序发行的事务标识符。** NumItems **读取的**OPC**标签的数目。** ClientHandles **读取的**OPC**标签的客户句柄的数组。** Values **返回的读取的数值的数组。** Qualities **读取的质量标志的数组。** TimeStamps **读取的采样时间的数组。**
示例	Private Sub AnOPCGroup_AsyncReadComplete (TransactionID As Long, _ NumItems As Long, ClientHandles() As Long, ItemValues() As Variant, _ Qualities() As Long, TimeStamps() As Date) Dim I As Long ‘ 表示读取OPC标签的客户句柄和数值 For I = 1 to NumItems Debug.Print I; ClientHandles(I); Values(I) Next End Sub

事件：AsyncWriteComplete

说明	**在异步写入（**AsyncWrite**）完成时发生的事件。**
文法	AsyncWriteComplete (TransactionID As Long, NumItems As Long, _ ClientHandles() As Long, Errors() As Long) TransactionID OPC**应用程序发行的事务标识符。** NumItems **写入的**OPC**标签数。** ClientHandles **写入的**OPC**标签的客户句柄の的数组。** Errors **返回的与写入项对应的错误码的数组。**
示例	Private Sub AnOPCGroup_AsyncWriteComplete (TransactionID As Long, NumItems As Long, ClientHandles() As Long, ItemValues() As Variant, Qualities() As Long, TimeStamps() As Date) Dim I As Long ‘ 表示写入错误码 For I = 1 to NumItems Debug.Print I; Errors(I) Next End Sub

所有OPC**组支持的属性，方法和事件的一览如表**1-14**，表**1-15**和表**1-16**所示。详细资料请参照**6.5**参考资料的“**OPC**数据访问自动化接口标准**”。

表 1-14 OPC组的属性

属性名	属性	说明
Parent	只读	返回所属OPC服务器对象。
Name	读写	OPC组的名称。
IsPublic	只读	OPC组是否是公用组的真伪值。
IsActive	读写	用以控制OPC组的活动状态。只有活动状态的OPC组才进行定期的数据更新。
IsSubscribed	读写	用以控制OPC组的订阅状态。
ClientHandle	读写	客户句柄是由应用程序指定的用于识别某个OPC组的长整型数。当进行数据访问或询问OPC组状态时，服务器将这个数值和结果一起返回给OPC应用程序。
ServerHandle	只读	服务器句柄是由OPC服务器指定的用于识别某个OPC组的一个独特的长整型数。
LocaleID	读写	区域标识符。
TimeBias	读写	数据采样时间的时间偏差值，用于调整设备时间和OPC服务器时间之间的偏差。
DeadBand	读写	不敏感带（全量程的百分比；合法值从0到100）。只有数据变化超过此不敏感带时，服务器才触发数据变化事件发生。
UpdateRate	读写	数据更新周期（毫秒）。
OPCItems	只读	OPC组的默认属性，OPC标签集合对象。

表 1-15 OPC组的方法

方法名	说明
SyncRead	同步读取OPC组内单个或者多个OPC标签的数据值，质量标志和采样时间。
SyncWrite	同步写入OPC组内单个或者多个OPC标签的数据值
AsyncRead	异步读取OPC组内单个或者多个OPC标签的数据值，质量标志和采样时间。
AsyncWrite	异步写入OPC组内单个或者多个OPC标签的数据值。
AsyncRefresh	触发数据变化事件发生，刷新OPC组内所有活动的OPC标签的数据。结果由数据变化（DataChange）事件返回。
AsyncCancel	取消尚未完成的异步数据访问事务。处理结果由异步取消完成（AsyncCancelComplete）事件返回。

表 1-16 OPC组的事件

事件名	说明
DataChange	在OPC组内任何OPC标签的数据值或者质量标志变化时触发的事件。
AsyncReadComplete	在异步读取（AsyncRead）完成时发生的事件。
AsyncWriteComplete	在异步写入（AsyncWrite）完成时发生的事件。
AsyncCancelComplete	在取消异步访问（AsyncCancel）完成时发生的事件。

1.5.7 OPC标签集合对象

OPC标签集合对象OPCItems具有OPC标签的默认属性，当添加新的OPC标签时，下述的DefaltXXX属性将是新添加的OPC标签的默认属性值。本节说明一些经常使用的OPC标签集合的属性和方法。本节的示例程序需要表1-17的示例基础。

表 1-17 OPC标签集合的示例基础

示例基础	Dim WithEvents AnOPCServer As OPCServer Dim ProgID As String Dim NodeName As String Dim MyGroups As OPCGroups Dim WithEvents OneGroup As OPCGroup Dim AnOPCItemCollection As OPCItems Dim AnOPCItem As OPCItem Dim AnOPCItemServerHandles() As Long Dim AnOPCItemServerErrors() As Long Set AnOPCServer = New OPCServer ProgID = "VendorX.DataAccessCustomServer" NodeName = "SomeComputerNodeName" AnOPCServer.Connect(ProgID, NodeName) Set MyGroups = AnOPCServer.OPCGroups Set OneGroup = MyGroups.Add("AnOPCGroupName") Set AnOPCItemCollection = OneGroup.OPCItems

属性：Count

说明	**只读的属性，返回OPC标签集合中的项数。**
文法	Count As Long
示例	Dim CurrentValue As Long **‘ 读取OPC标签的数目** CurrentValue = AnOPCItemCollection.Count

方法：AddItems

说明	**在OPC标签集合中添加新的OPC标签。其初期属性取决与OPC标签集合的默认值。**
文法	AddItems (Count As Long, ItemIDs() As String, ClientHandles() As Long, _ ByRef ServerHandles() As Long, ByRef Errors() As Long, _ Optional RequestedDataTypes As Variant, _ Optional AccessPaths As Variant) Count **添加的OPC标签的数目。** ItemIDs **要添加的OPC标签的标识符的数组。** ClientHandles **要添加的OPC标签的客户句柄的数组。** ServerHandles **返回的对应添加的OPC标签的服务器句柄的数组。** Errors **返回的对应添加的OPC标签的错误码的数组。** RequestedDataTypes **选用参数，是要添加的OPC标签的要求的数据类型的数组。** AccessPaths **选用参数，要添加的OPC标签路径的数组。**
示例	Dim ClientHandles(100) As Long Dim AnOPCItemIDs(100) As String Dim AddItemCount As Long

	Dim I As Long AddItemCount = 10 ‘ 设置要添加的OPC标签的客户句柄和项标识符 For I = 1 To AddItemCount ClientHandles(I) = I + 1 AnOPCItemID(I) = "Item_" & I Next ‘ OPC标签的添加 AnOPCItemCollection.AddItems AddItemCount, AnOPCItemIDs, _ ClientHandles, AnOPCItemServerHandles, AnOPCItemServerErrors

方法：Remove

说明	**清除指定的**OPC**标签。**
文法	Remove (Count As Long, ServerHandles() As Long, _ ByRef Errors() As Long) Count **要清除的**OPC**标签的数目。** ServerHandles **要清除的**OPC**标签的服务器句柄的数组。** Errors **返回的对应被清除**OPC**标签的错误码的数组。**
示例	‘ 清除OPC标签 AnOPCItemCollection.Remove (AnOPCItemServerHandles, _ AnOPCItemServerErrors)

所有OPC**标签集合支持的属性和方法的一览如表**1-18**和表**1-19**所示。详细资料请参照**6.5**参考资料的“**OPC**数据访问自动化接口标准**”。

表 1-18 OPC标签集合的属性

属性名	**属性**	**说明**
Parent	**只读**	**返回所属的**OPC**组对象。**
DefaultRequestedDataType	**读写**	**在添加**OPC**标签时，默认的要求数据类型。初期值是**VT_Empty**（＝控制设备的固有数据类型）。**
DefaltAccessPath	**读写**	**在添加**OPC**标签时，默认的数据访问路径。初期值是**""**（＝无路径）。**
DefaultIsActive	**读写**	**在添加**OPC**标签时，默认的活动状态。初期值是**True**（真＝活动）。**
Count	**只读**	**集合对象的固有属性。**OPC**标签集合中的**OPC**标签数。**

表 1-19 OPC标签集合的方法

方法名	**说明**
Item	**返回**OPC**标签集合中由集合索引（**ItemSpecifier**）指定的**OPC**标签。**
GetOPCItem	**返回**OPC**标签集合中由服务器句柄指定的**OPC**标签。**
AddItem	**在**OPC**标签集合中添加新的**OPC**标签。**
Remove	**清除指定的**OPC**标签。**
Validate	**检查被添加的**OPC**标签。**

SetActive	分别设置OPC标签为活动状态或非活动状态。
SetClientHandles	设置OPC标签的客户句柄。
SetDataTypes	设置OPC标签的要求的数据类型。

1.5.8 OPC标签对象

OPC标签对象OPCItem表示与OPC服务器内某个数据的连接。各个OPC标签由数据值，质量标志以及采样时间构成。所有OPC标签支持的属性和方法的一览如表1-20和表1-21所示。详细资料请参照6.5参考资料的“OPC数据访问自动化接口标准”。

表 1-20 OPC标签的属性

属性名	属性	说明
Parent	只读	返回所属的OPC组对象。
ClientHandle	读写	客户句柄是由应用程序指定的用于识别某个OPC标签的长整型数。当OPC组事件发生时，服务器将这个客户句柄和结果一起返回给OPC应用程序。
ServerHandle	只读	服务器句柄是由OPC服务器设置的用于识别某个OPC标签的一个独特长整型数。
AccessPath	只读	返回OPC应用程序指定的访问路径。
AccessRight	只读	返回OPC标签的访问权限。
ItemID	只读	返回识别这个OPC标签的标识符。
IsActive	读写	用以控制OPC标签的活动状态。只有活动状态的OPC标签才进行定期的数据更新。
RequestedDataType	读写	要求的数据类型。
Value	只读	返回从OPC服务器读取的最新数据值。
Quality	只读	返回从OPC服务器读取的最新数据的质量标志。
TimeStamp	只读	返回从OPC服务器读取的最新数据的采样时间。
CanonicalDataType	只读	返回OPC服务器内固有的数据类型。
EUType	只读	返回工程单位（Engineering Unit）的数据类型。
EUInfo	只读	返回表示工程单位（Engineering Unit）的Variant型数值。

表 1-21 OPC标签的方法

方法名	说明
Read	从OPC服务器中读取OPC标签的数据。
Write	向OPC服务器中写入OPC标签的数据。

1.5.9 OPC浏览器对象

OPC浏览器对象OPCBrowser是OPC服务器名称空间的枝和叶（项）的集合。浏览功能是选用功能，OPC服务器不支持浏览的时候，即使执行CreateBrowser也不生成这个对象。所有OPC浏览器支持的属性和方法的一览如表1-22和表1-23所示。详细资料请参照6.5参考资料的“OPC数据访问自动化接口标准”。

表 1-22 OPC浏览器的属性

属性名	属性	说明
Organization	只读	OPC服务器的名称空间的类型（平层型或分层

		型）。
Filter	读写	使用ShowBranches或ShowLeafs方法时的浏览对象过滤器。使用这个过滤器可以缩小被浏览的名称范围。
DataType	读写	使用ShowLeafs方法时，希望浏览的项的数据类型。
AccessRights	读写	使用ShowLeafs方法时，希望浏览的项的访问权限。
CurrentPosition	只读	返回在名称空间中的现在位置。
Count	只读	浏览结果中的浏览项数。

表 1-23 OPC浏览器方法

方法名	说明
Item	返回浏览结果中按集合索引（ItemSpecifier）指定的对象。
ShowBranches	将现在位置下的所有符合过滤条件的枝（Branch）加入到浏览结果中去。
ShowLeafs	将现在位置下的所有符合过滤条件的叶（Leav）加入到浏览结果中去。
MoveUp	向现在位置的上一层移动。
MoveToRoot	向名称空间的最上层移动。
MoveDown	向现在位置的下一层移动。
MoveTo	向浏览器的绝对位置移动。
GetItemID	由浏览项的名称返回OPC标签的标识符。
GetAccessPath	返回OPC标签标识符的路径字符串。

使用Visual Basic开发OPC应用程序

本章介绍怎样利用微软的Visual Basic开发OPC自动化接口的客户应用程序。这里介绍的应用程序使用OPC数据访问自动化接口（版本2.0），进行同步方式和异步方式的数据读取和写入。

请参照第６章的［5.4示范源程序的使用方法］，可以从配套光盘找到本章介绍的示范程序。所介绍的源代码可以在Visual Basic Version 5.0/6.0 专业/企业版上运行。

2.1 建立一个Visual Basic工程

利用Visual Basic开发OPC应用程序时，实现OPC自动化接口的OPC包装DLL是必须的。这个OPC包装DLL一般应该是由OPC服务器的供应商提供的。这个OPC包装DLL的名称可能随供应商有所不同，具体信息请向你的OPC服务器供应商查询。

本书使用为OPC基金会成员制作的OPC包装DLL进行说明。有关OPC包装DLL的注册方法请参照第６章的［5.4.1 复制和注册示范源程序］。

1.5.10 启动Visual Basic

首先启动Visual Basic，新建一个Visual Basic的工程。请选择[标准 EXE]作为新建工程的类型（图 2-1）。

图 2-1 Visual Basic工程的建立

1.5.11 设置OPC包装DLL

因为在新建的Visual Basic工程里OPC包装DLL还没有被注册，必须用下述方法对OPC包装DLL进行注册。

＜设置方法＞

1) 从Visual Basic菜单里选择[工程(P)]-[引用(N)]。
2) 在[可用的引用(A)]的一览表示中，请选择对应OPC包装DLL的文件名。这里我们选择[OPC Automation 2.0]（图 2-2）。

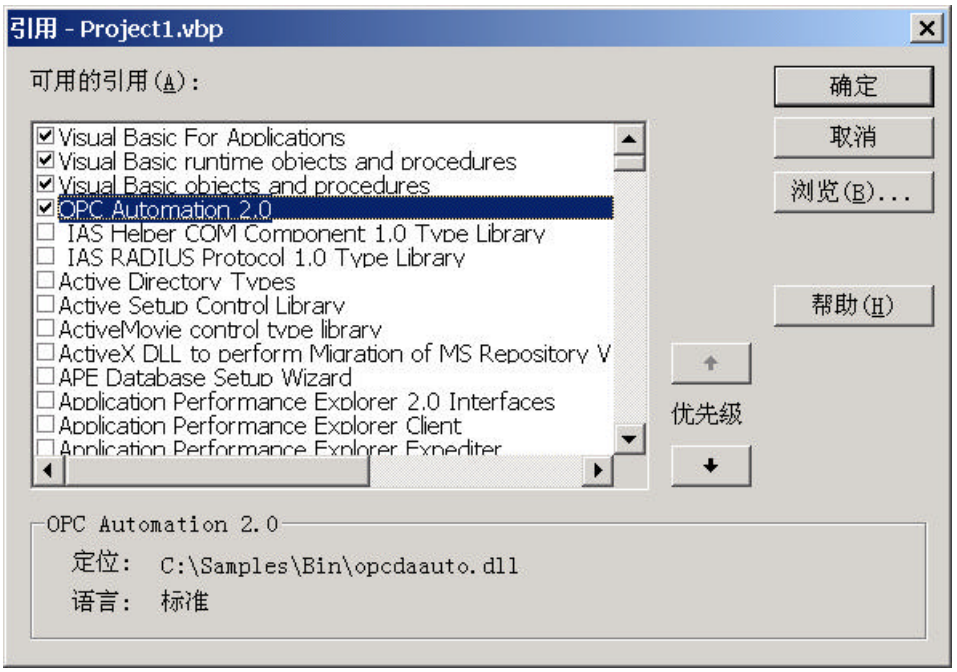


图 2-2 引用的设置

设置了对OPC**包装**DLL**的引用，我们就可以使用**OPC**自动化对象了。如果希望查看在工程内可以使用的**COM**组件的时候，可以使用对象浏览器。选择**Visual Basic**菜单的[视图(V)]-[对象浏览器(O)], 可以显示对象浏览器的视窗。如果已经设置了对**OPC**包装的引用，那么对象浏览器左上角的列表框中应该已经包括了**[OPC Automation]**的选项（图 2-3）。**

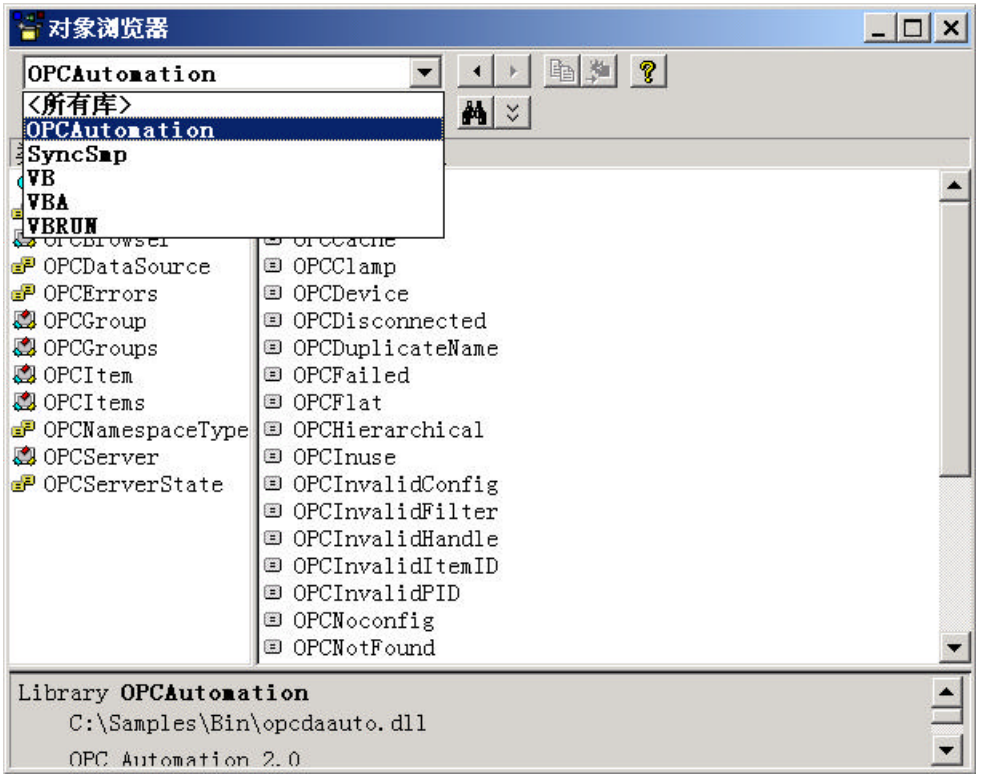


图 2-3 对象浏览器

2.2 建立一个OPC对象

在Visual Basic里，是以对象的单位对OPC服务器进行访问。OPC自动化接口是由以下四种对象所定义。

- OPC服务器
- OPC组（OPC组集合）
- OPC标签（OPC标签集合）
- OPC浏览器

这里只说明OPC服务器，OPC组以及OPC标签对象的使用方法。具体的来说，先连接一个特定的OPC服务器，然后建立OPC组，最后添加OPC标签。

2.2.1 变量声明

参考图 2-4所示的示范代码，先对OPC对象变量进行声明。变量的数据类型应该指定为对象型。这些对象变量最好在窗体代码的（通用）部分声明。因为在（通用）部分声明的变量，可以在窗体的任何方法的代码内引用。

除了变量的声明以外，我们还推荐指定下述的VB选项。

1) 和C以及C++等编程语言相比，Visual Basic可以很简单地处理变量，即使不进行变量声明也可以使用变量，但是这也正是可能产生程序缺陷的原因。再有，如果进行了变量的声明，由于可以在程序编译时对对象进行事前结合，有助于提高运转性能。所以我们建议在模块代码的最初的部分，加上“Option Explicit”语句，表示模块里的所有变量需要显式的声明。这样进行程序编译时将对所有变量名进行检查，如果出现未声明的变量名将会以编译错误作出提示。
2) 因为再有OPC自动化接口的数组的索引要求必须从１开始開始，为了避免错误建议在代码的最初加上“Option Base 1”语句。

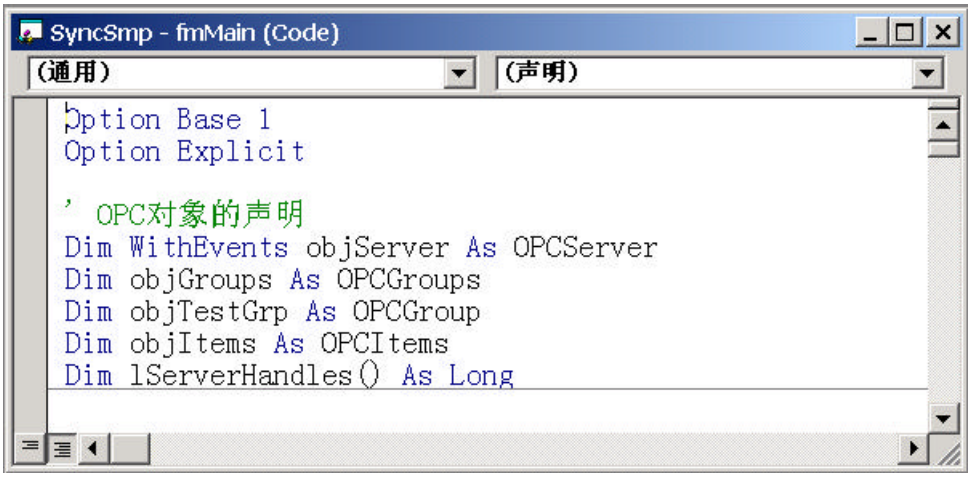


图 2-4 OPC对象变量的声明

有关定义OPC对象的声明及其说明列于表2-1。

表2-1 OPC对象变量的声明

变量名	说明
objServer	OPCServer对象，用于连接OPC服务器。
objGroups	OPCGroups 对象，用于添加OPC组的OPC组集合。
objTestGrp	OPCGroup对象，演示用的OPC组。

objItems	OPCItems对象，用于添加OPC标签的OPC标签集合。
lServerHandles()	长整型的数组，用于保存OPC标签的服务器句柄。

2.2.2 连接OPC服务器和建立OPC组

下面说明怎样连接OPC服务器和建立OPC组。考虑到代码的可反复使用性，这里采用子程序进行编程。

这里用“New”关键词生成OPC服务器的对象，然后调用OPC服务器对象的“Connect”方法，和OPC服务器连接。在连接远程服务器的时候，需要指定作为选用参数的远程计算机名（表2-2）。

表2-2 Connect子程序

```
Sub Connect(strProgID As String, Optional strNode As String)

    If objServer Is Nothing Then
        ' 创建一个OPC服务器对象
        Set objServer = New OPCServer
    End If

    If objServer.ServerState = OPCDisconnected Then
        ' 连接OPC服务器
        objServer.Connect strProgID, strNode
    End If

    If objGroups Is Nothing Then
        ' 建立一个OPC组集合
        Set objGroups = objServer.OPCGroups
    End If

    If objTestGrp Is Nothing Then
        ' 添加一个OPC组
        Set objTestGrp = objGroups.Add("TestGrp")
End If

End Sub
```

2.2.3 添加OPC标签

对OPC服务器进行访问前，必须先在OPC组里添加要访问的OPC标签。

这里添加OPC标签的标识符和数目是固定的，但是实际的OPC应用程序往往要按照用户的指定或读取组态文件取得和处理需要添加的OPC标签的一览（表2-3）。

表2-3 AddItem子程序

```
Sub AddItem()
    Dim strItemIDs(8) As String
    Dim lClientHandles(8) As Long
    Dim lErrors() As Long
    Dim I As Integer

    If objTestGrp Is Nothing Then
        Exit Sub
```

```
    End If

    If Not objItems Is Nothing Then
        If objItems.Count > 0 Then
            Exit Sub
        End If
    End If

    ' 设置组活动状态
    objTestGrp.IsActive = True
    ' 取消组异步通知
    objTestGrp.IsSubscribed = False

    ' 建立OPC标签集合
    Set objItems = objTestGrp.OPCItems

    ' 生成从TAG1到TAG8的项标识符
    For I = 1 To 8
        strItemIDs(I) = "TAG" & I
        lClientHandles(I) = I
    Next
    ' 添加OPC标签
    Call objItems.AddItems(8, strItemIDs, _
        lClientHandles, lServerHandles, lErrors)

End Sub
```

2.2.4 断开OPC服务器

连接着OPC服务器的OPC应用程序，在退出前必须断开和OPC服务器的连接（表2-4）。因为OPC服务器并不知道OPC应用程序的退出，如果不先断开连接，那么OPC服务器使用的计算机资源就不会被释放。如果这样的问题反复发生，久而久之，连续运转的自动控制系统可能会使计算机资源渐渐枯竭从而发生严重问题。

表2-4 Disconnect子程序

```
Sub Disconnect()
Dim lErrors() As Long

    If Not objItems Is Nothing Then
        If objItems.Count > 0 Then
            ' 清除OPC标签
            objItems.Remove 8, lServerHandles, lErrors
        End If
        Set objItems = Nothing
    End If

    If Not objTestGrp Is Nothing Then
        ' 清除OPC组
        objGroups.Remove "TestGrp"
        Set objTestGrp = Nothing
    End If

    If Not objGroups Is Nothing Then
        Set objGroups = Nothing
```

```
    End If

    If Not objServer Is Nothing Then
        If objServer.ServerState <> OPCDisconnected Then
            ' 断开OPC服务器
            objServer.Disconnect
        End If

        Set objServer = Nothing
    End If

End Sub
```

2.3 同步数据读写

到此为止，我们已经基本说明了OPC对象。现在，让我们制作一个实际的OPC数据访问应用程序。首先让我们先说明同步方式的数据访问。

2.3.1 窗体设计

让我们制作具有如图 2-5所示窗体的OPC应用程序。

这个程序读取８点的数据，并用棒图表示读取的数据。棒图的更新周期为１秒，使用定时器以１秒的周期对OPC服务器进行同步数据读取。

当文字框内按下[Enter]键时，对OPC服务器进行同步数据写入。

再有，我们使用的演示用OPC服务器，OPC标签一旦被写入后就停止数据仿真，数据被固定在写入值不再变化。

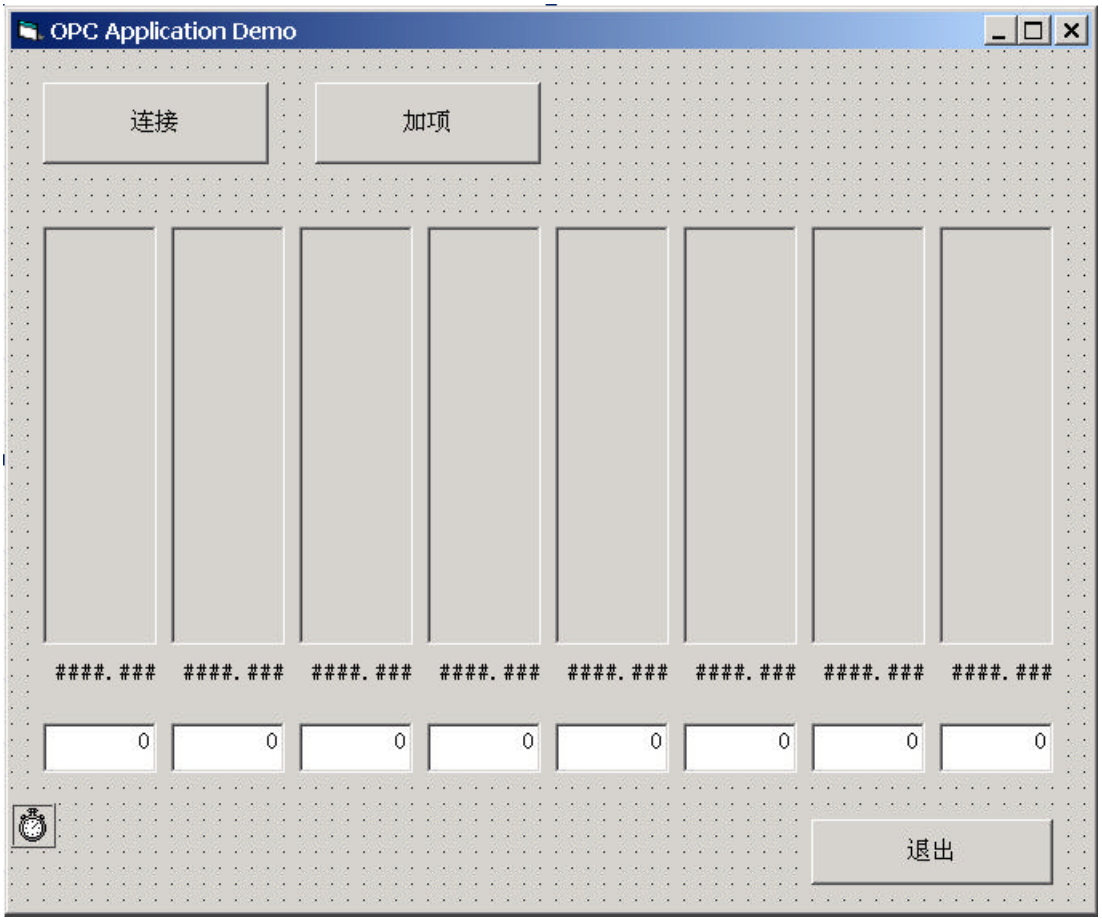


图 2-5 窗体设计

窗体里所使用的控件种类如表2-5所示。

表2-5 fmMain的控件

控件	名称
窗体（Form）	FmMain
命令按钮（CommandButton）	BtnConnect
命令按钮（CommandButton）	BtnAddItem
命令按钮（CommandButton）	BtnQuit
定时器（Timer）	TmUpdate
图象（PictureBox）	picBar（作为数组使用。从左开始1～8）
标签（Label）	lbBar（作为数组使用。从左开始1～8）
文字框（TextBox）	txbBar（作为数组使用。从左开始1～8）

2.3.2 命令按钮的事件处理

当按下在窗体上的命令按钮的事件发生时，通过调用在2.2节说明的各种方法对发生的事件进行处理（表2-6，表2-7，表2-8）。

表2-6 btnConnect命令按钮的单击事件处理

```
Private Sub btnConnect_Click()
    ' 调用Connect子程序
```

```
    Call Connect("OPCJ.SampleServer.1")

End Sub
```

表2-7 btnAddItem命令按钮的单击事件处理

```
Private Sub btnAddItem_Click()
    ' 调用AddItem子程序
    Call AddItem

    If Not objTestGrp Is Nothing Then
        If objTestGrp.OPCItems.Count > 0 Then
            ' 启动定时器
            tmUpdate.Enabled = True
        End If
    End If

End Sub
```

表2-8 btnQuit命令按钮的单击事件处理

```
Private Sub btnQuit_Click()
    ' 卸载窗体
    Unload fmMain

End Sub
```

为了在中止应用程序时断开和OPC服务器的连接，在窗体的Unload事件处理中调用了断开OPC服务器连接的子程序（表2-9）。

表2-9 fmMain窗体的卸载事件处理

```
Private Sub Form_Unload(Cancel As Integer)
    ' 调用Disconnect子程序
    Call Disconnect

End Sub
```

2.3.3 同步数据读取

在本示范程序中是利用定时器的定时器事件进行更新数据的显示。这时所有的数据是采用同步方式对OPC服务器进行读取的。表2-10是定时器的事件处理以及同步读取处理的代码。

表2-10 tmUpdate定时器的定时事件处理

```
Private Sub tmUpdate_Timer()
    Dim vtItemValues() As Variant
    Dim lErrors() As Long
    Dim strBuf As String
    Dim nWidth As Integer
    Dim nHeight As Integer
    Dim nDrawHeight As Integer
    Dim sglScale As Single
```

```
    Dim I As Integer

    ' 同步读取
    SyncRead OPCCache, vtItemValues, lErrors

    ' 棒图的表示
    For I = 1 To 8
        ' 数据的格式化
        strBuf = Format(vtItemValues(I), "###.000")
        ' 表示数据字符串
        lbBar(I).Caption = strBuf
        ' 计算棒的宽和高
        nWidth = picBar(I).ScaleWidth
        nHeight = picBar(I).ScaleHeight
        sglScale = vtItemValues(I) / 100
        nDrawHeight = CInt(nHeight * sglScale)
        ' 清除现棒图
        picBar(I).Cls
        ' 绘制棒图
        picBar(I).Line (0, nHeight - nDrawHeight)-(nWidth, nHeight), _
            RGB(255, 0, 0), BF
    Next

End Sub
```

定时器事件处理内调用的“ SyncRead **”子程序如表**2-11**所示。再有在读取前为了避免错误发生，对**OPC**组和**OPC**标签数进行检查。**

表*2-11 SyncRead****子程序***

```
Sub SyncRead(nSource As Integer, ByRef vtItemValues() As Variant, _
    ByRef lErrors() As Long)

    If objTestGrp Is Nothing Then
        Exit Sub
    End If

    If objTestGrp.OPCItems.Count > 0 Then
        ' 同步读取
        objTestGrp.SyncRead nSource, 8, lServerHandles, _
            vtItemValues, lErrors
    End If

End Sub
```

2.3.4 同步数据写入

在本示范程序中是利用文字框的按键事件对OPC**服务器进行写入的。当某文字框内回车键按下时，则对其对应的**OPC**标签进行写入。在文字框的按键事件处理中，首先先对按下的键进行判别，如果是回车键则进行同步写入。表**2-12**是按键事件处理以及同步写入处理的代码。**

表*2-12 txtBar****文字框的按键事件处理***

```
Private Sub txbBar_KeyPress(Index As Integer, KeyAscii As Integer)
    Dim strData As String
    Dim vtItemData(1) As Variant
    Dim lError() As Long

    ' 是回车键?
    If KeyAscii = Asc(vbCr) Then
        ' 得到输入的字符串
        strData = txbBar(Index).Text
        ' 转换成单精度浮点数
        vtItemData(1) = CSng(strData)
        ' 同步写入
        SyncWrite Index, vtItemData, lError
    End If

End Sub
```

按键事件处理内调用的“SyncWrite”子程序如表2-13所示。

表2-13 SyncWrite子程序

```
Sub SyncWrite(nIndex As Integer, ByRef vtItemValues() As Variant, _
    ByRef lErrors() As Long)
Dim lHandle(1) As Long

    If objTestGrp Is Nothing Then
        Exit Sub
    End If

    If objTestGrp.OPCItems.Count > 0 Then
        lHandle(1) = lServerHandles(nIndex)

        ' 同步写入
        objTestGrp.SyncWrite 1, lHandle(), _
            vtItemValues, lErrors
    End If

End Sub
```

2.3.5 运行结果

本节中只记述了有关同步方式的读取和写入部分的源代码。你可以参考第6章的［5.4 示范源程序的使用方法］得到完整的示范程序代码。本示范程序的OPC连接处理和OPC标签添加处理，只对应既定的OPC服务器和既定的OPC标签。在制作实际的OPC应用程序时，往往需要建立另外的视窗让用户可以选择OPC服务器与OPC标签。

再有，对于动作几乎相同的图象，标签以及文字框是作为控件数组处理的，为了使记述更为简单。实际的运行结果如图 2-6所示。

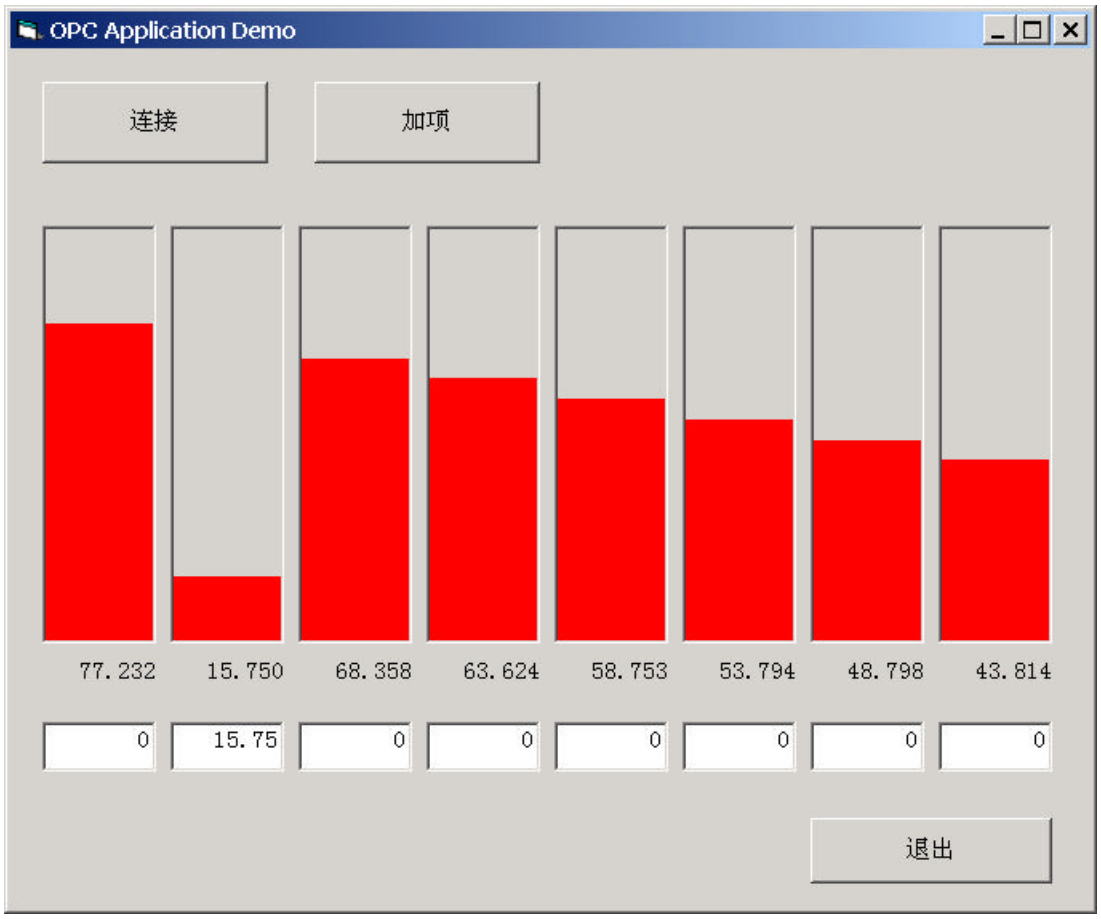


图 2-6 运行结果

2.4 异步数据读写

现在以刚才说明的同步方式的示范程序为基础，说明异步方式的数据读取和写入的编程。异步方式与同步方式的区别在于，在调用数据访问方法时，OPC应用程序必须等待OPC服务器的动作完成。也就是说同步方式的数据访问在要求的动作没有完成前不能从调用的方法中返回到主处理程序中去。所以从调用方法到返回到主处理程序之间，不可能执行任何其他OPC应用程序侧的处理，导致对用户操作也不能作出及时反应。本章示范程序的OPC标签数比较少可能不会发生上述问题，但是要求的数据比较多时或者直接对设备进行访问时，这样的问题就会变得更加明显了。

在这种情况下，异步方式的数据访问就显得更为有效。异步方式的访问对OPC服务器提出数据访问要求后，立即返回到OPC应用程序侧的主处理程序中去。OPC服务器完成数据访问时通知OPC应用程序，OPC应用程序从而得到数据访问的结果。

有关异步方式的数据访问编程，我们只说明怎样变更前述的同步方式示范程序，使之成为异步方式的示范程序。

2.4.1 OPC对象声明的改变

首先为了使对象可以处理事件，必须将objTestGrp对象的声明中加上“ WithEvents ”语句，表示声明的对象可以响应事件（图2-7）。

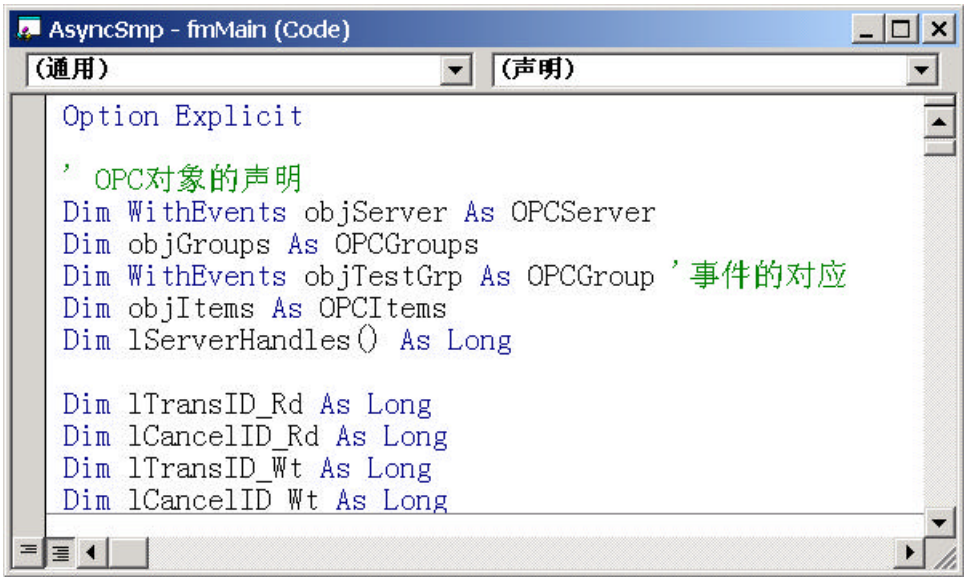


图 2-7 异步OPC对象变量的声明

异步方式访问使用的新定义的变量及其说明如表2-14所示。

表2-14 用于异步方式访问新定义的变量

变量名	说明
lTransID_**	异步用的事务标识符。用于读取和写入。
LCancelID_**	异步用的取消标识符。用于读取和写入。

2.4.2 OPC组对象属性的改变

在进行异步访问前，必须建立异步用的通信通路。使用这个通信通路，OPC服务器可以反调OPC应用程序的事件处理程序，以通知数据访问的結果。通信通路的建立方法是改变OPC组的属性，把OPC组的“IsSubscribed”属性设置为True（表2-15）。

表2-15 OPC标签添加子程序

```
Sub AddItem()
－省略－

' 启动组异步通知
objTestGrp.IsSubscribed = True

－省略－
End Sub
```

2.4.3 异步读取代码的改变

对于前述的同步方式的数据访问示范程序，我们作出以下变更，使之可以进行异步方式的数据访问。定时器事件处理内只进行异步读取要求，而在异步读取完成事件处理时接受数据访问的结果（表2-16）。

表2-16 tmUpdate定时器的定时事件处理

```
Private Sub tmUpdate_Timer()
```

```
    ' 异步读取
    Call AsyncRead

End Sub
```

定时器事件处理内调用的“ AsyncRead **”子程序如表**2-17**所示。**

表*2-17* ***异步读取子程序***

```
Sub AsyncRead()
    Dim lErrors() As Long

    If objTestGrp Is Nothing Then
        Exit Sub
    End If

    If objTestGrp.OPCItems.Count > 0 Then
        ' 异步读取
        lTransID_Rd = lTransID_Rd + 1
        objTestGrp.AsyncRead 8, lServerHandles, _
            lErrors, lTransID_Rd, lCancelID_Rd
    End If

End Sub
```

在前述的同步方式读取示范程序，在定时器事件处理内进行更新棒图的显示，而对于异步方式即使调用方法后并不得到结果数据，所以更新棒图的显示要在异步完成事件处理中进行。

在OPC**对象声明中带有“** WithEvents **”语句的**objTestGrp**对象，会出现在代码编辑器左上角的**[**对象**]**列表框的项目中。如果这个对象已经定义了事件的话，那么事件处理过程会在代码编辑器右上角的**[**过程**]**列表框的项目中出现。这里让我们选择[AsyncReadComplete]**，记述异步读取的完成事件处理代码（表**2-18**）。**

表*2-18 objTestGrp****组的异步读取完成事件处理***

```
Private Sub objTestGrp_AsyncReadComplete( _
    ByVal TransactionID As Long, ByVal NumItems As Long, _
    ClientHandles() As Long, ItemValues() As Variant, _
    Qualities() As Long, TimeStamps() As Date, Errors() As Long)
    Dim strBuf As String
    Dim nWidth As Integer
    Dim nHeight As Integer
    Dim nDrawHeight As Integer
    Dim sglScale As Single
    Dim I As Integer
    Dim index As Integer

    ' 棒图的表示
    For I = 1 To NumItems
        ' 数据的格式化
        strBuf = Format(ItemValues(I), "###.000")
        ' 得到客户标识符
        index = ClientHandles(I)
        ' 表示数据字符串
```

```
        lbBar(index).Caption = strBuf
        ' 计算棒的宽和高
        nWidth = picBar(index).ScaleWidth
        nHeight = picBar(index).ScaleHeight
        sglScale = ItemValues(I) / 100
        nDrawHeight = CInt(nHeight * sglScale)
        ' 清除现棒图
        picBar(index).Cls
        ' 绘制棒图
        picBar(index).Line (0, nHeight - nDrawHeight)-(nWidth, nHeight), _
            RGB(255, 0, 0), BF
    Next

End Sub
```

2.4.4 异步写入的改变

写入处理示范程序也和读取处理一样变更为异步方式（表2-19**）。**

表2-19 txbBar文字框的按键事件处理

```
Private Sub txbBar_KeyPress(index As Integer, KeyAscii As Integer)
    Dim strData As String
    Dim vtItemData(1) As Variant
    Dim lError() As Long

    ' 是回车键?
    If KeyAscii = Asc(vbCr) Then
        ' 得到输入的字符串
        strData = txbBar(index).Text
        ' 转换成单精度浮点数
        vtItemData(1) = CSng(strData)

        ' 异步写入
        Call AsyncWrite(index, vtItemData, lError)
    End If

End Sub
```

按键事件处理内调用的“ AsyncWrite **”子程序如表**2-20**所示。**

表2-20 异步写入子程序

```
Sub AsyncWrite(nIndex As Integer, ByRef vtItemValues() As Variant, _
    ByRef lErrors() As Long)
    Dim lHandle(1) As Long

    If objTestGrp Is Nothing Then
        Exit Sub
    End If

    If objTestGrp.OPCItems.Count > 0 Then
        lHandle(1) = lServerHandles(nIndex)
```

```
        ' 异步写入
        lTransID_Wt = lTransID_Wt + 1
        objTestGrp.AsyncWrite 1, lHandle(), vtItemValues, _
              lErrors, lTransID_Wt, lCancelID_Wt
    End If

End Sub
```

检查异步写入处理结果也是在异步完成事件处理中进行的。请选择ObjTestGrp对象的[AsyncWriteComplete]过程，然后记入异步写入完成事件的处理代码。因为本示范程序省略了错误检查，所以没有任何代码记入。

2.5 订阅方式的数据采取

订阅方式的数据采取也属于异步读取方式的一种，但是OPC应用程序侧并不需要发出读取要求，OPC服务器在定期更新数据缓冲器的数据时，如果发现数据有一定变化则自动向OPC应用程序侧通知和传送变化的数据。

为了使订阅方式的数据采取有效，必须将OPC组以及OPC标签的“IsActive”属性设置为True。有了这样的设置，OPC组就可以接受数据变化事件的通知了。

在刚才的异步方式的示范程序中，加上如表2-21所示的订阅方式的数据采取的处理代码。

表2-21 objTestGrp组的数据变化事件处理

```
Private Sub objTestGrp_DataChange(ByVal TransactionID As Long, _
    ByVal NumItems As Long, ClientHandles() As Long, ItemValues() As Variant, _
    Qualities() As Long, TimeStamps() As Date)
    Dim strBuf As String
    Dim nWidth As Integer
    Dim nHeight As Integer
    Dim nDrawHeight As Integer
    Dim sglScale As Single
    Dim I As Integer
    Dim index As Integer

    ' 棒图的表示
    For I = 1 To NumItems
        ' 数据的格式化
        strBuf = Format(ItemValues(I), "###.000")
        ' 得到客户标识符
        index = ClientHandles(I)
        ' 表示数据字符串
        lbBar(index).Caption = strBuf
        ' 计算棒的宽和高
        nWidth = picBar(index).ScaleWidth
        nHeight = picBar(index).ScaleHeight
        sglScale = ItemValues(I) / 100
        nDrawHeight = CInt(nHeight * sglScale)
        ' 清除现棒图
        picBar(index).Cls
        ' 绘制棒图
        picBar(index).Line (0, nHeight - nDrawHeight)-(nWidth, nHeight), _
            RGB(255, 0, 0), BF
    Next
```

```
End Sub
```

2 使用Visual Basic开发OPC ActiveX控件

本章说明怎样使用微软的Visual Basic（以下简称为VB），开发一个可以收集和显示控制测量数据的ActiveX控件。本章的示范程序是以由日本OPC协会开发的演示用OPC ActiveX控件以及OPC服务器为例，说明怎样利用VB开发一个OPC ActiveX控件的方法。基本方法是先建立一个开发OPC ActiveX控件的VB工程，然后与另一个用于测试作成的OPC ActiveX控件的标准EXE工程组成一个VB工程组，分别用于开发和调试OPC ActiveX控件的动作。

请参照第６章的［5.4示范源程序的使用方法］，可以从配套光盘找到本章的介绍的示范程序的源代码。所介绍的源代码可以在Visual Basic Version 5.0/6.0 专业/企业版上运行。

用VB开发连接一个现场设备进行控制测量的计算机程序时，往往需要制作一个专用的窗体。于是在需要连接多个现场设备进行控制测量的计算机程序时，也往往需要制作分别与其对应的窗体。这样的窗体往往需要设置不同的属性，所以使用不同的源代码和分别进行调试工作。如果窗体的内容十分互不相同而进行这样的工作当然是比不可免的，但是如果共同的部分很多的时候，有可以反复使用的窗体的话则更为有效。这时应该采用可以反复使用的软件组件ActiveX控件。

本章介绍的ActiveX控件是从指定OPC服务器读取数据并进行图形表示的用户控件。这个图形以横轴（X轴）为时间，以纵轴（Y轴）为变量幅度，表示读取数值的趋势图。这个趋势图是从图形表示领域的最左端开始，一直描绘到最右端为止，然后周而复始清除整个画面再次从最左端开始描绘新的趋势线。这个ActiveX控件的默认设置值，是连接OPC协会开发的演示用OPC模拟服务器，并以一定的周期（10毫秒）读取OPC标签标识符为“SV1”变量的数值。

2.1 建立一个ActiveX控件

在使用VB开发OPC ActiveX控件时，首先应该新建一个具有ActiveX控件雏形的VB工程。从VB的菜单选择[文件(F)]-[新建工程(N)]，则显示图 3-1的画面。

图 3-1 工程的建立

请在[新建工程]对话框里选择[ActiveX控件]图标，然后按下[确定]，于是生成了如图 3-2的画面所示的一个新的ActiveX控件雏形。这样已经自动生成了一个有UserControl1的ActiveX控件。从VB的菜单选择[运行(R)]-[启动(S)]，可以用Internet Explorer（以下简

称为IE）显示这个新建的ActiveX控件。但是此时这个控件仅仅是雏形而已，并无任何实际显示内容。

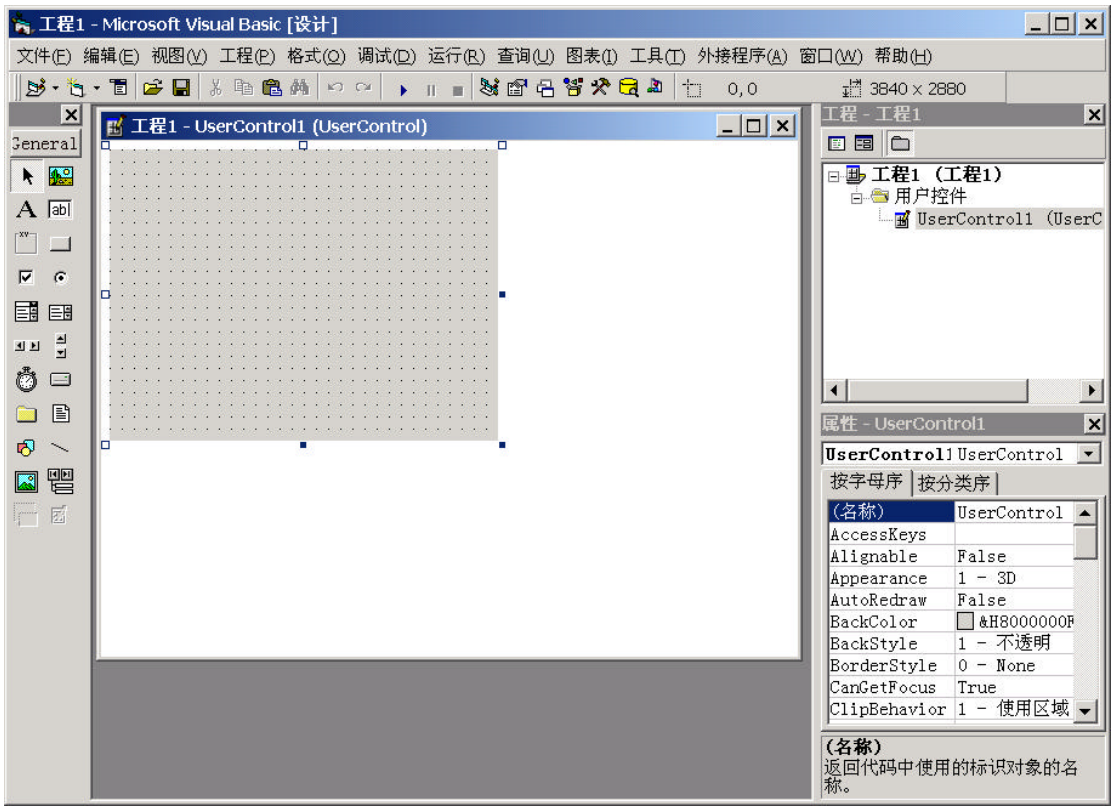


图 3-2 工程设计画面

因为新建的ActiveX控件工程的默认名称是“ Project1 ”，下面把这个名称改变为“ OPCTrend ”。从VB菜单选择[工程(P)]-[Project1属性(E)...]。请在表示的[Project1 – 工程属性]的对话框里改变以下的项目（图3-3）。

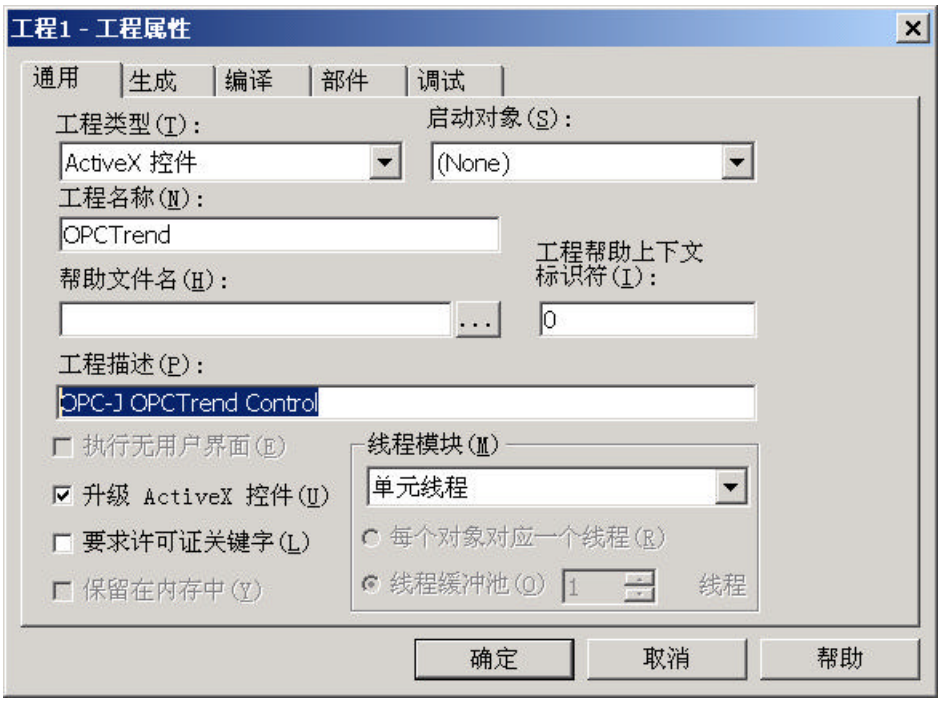


图 3-3 工程属性

[**工程名称**(N):] → “OPCTrend”

[**工程描述**(P):] → “OPC-J OPCTrend Control”

※[**工程描述**]**的定义将成为以后在**[**工程**(P)]-[**部件**(O)]**的一览中表示的部件名称。**

下面先将用户控件的属性视窗的[(**名称**)]**属性改变为“** TrendGraph **”（图**3-4**）。这个名称将成为建成**OPC ActiveX**控件的类名。例如用建成的**OPC ActiveX**控件配置在标准** EXE**文件的窗体时，配置的**OPC ActiveX**控件的默认名称将是“** TrendGraphX **”。这里** X**是从**1，2，3...**的建成顺序号，在应用程序中可使用这个名称引用配置的**OPC ActiveX**控件对象。**

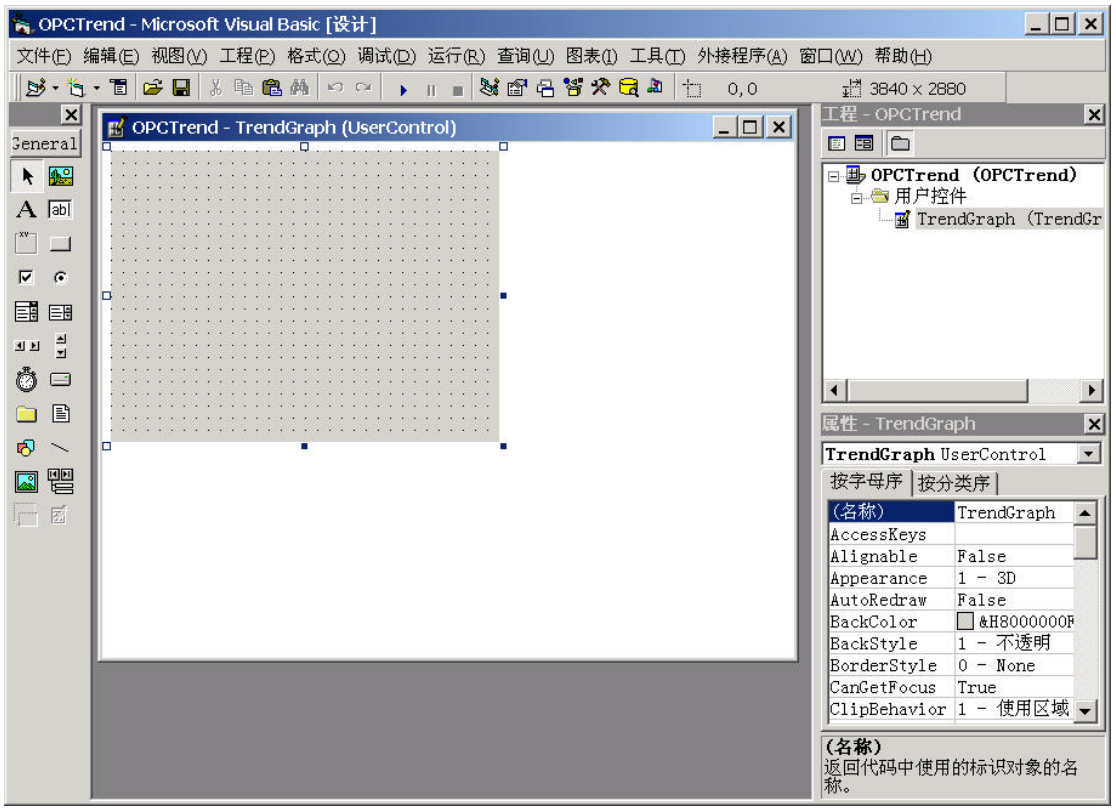


图 3-4 OPCTrend设计画面之一

接着把用户控件的[BackColor]**属性设置为“** &H00000000& **”（黑）。再在这个用户控件上添加一个线（**Line**）控件对象，并将线的**[(名称)]**属性设置为“** linHorizon **”，并将其**[Border Color]**属性设置为“** &HFFFFFFFF& **”（白）。于是这个用户控件就变成图** 3-5**的样子。**

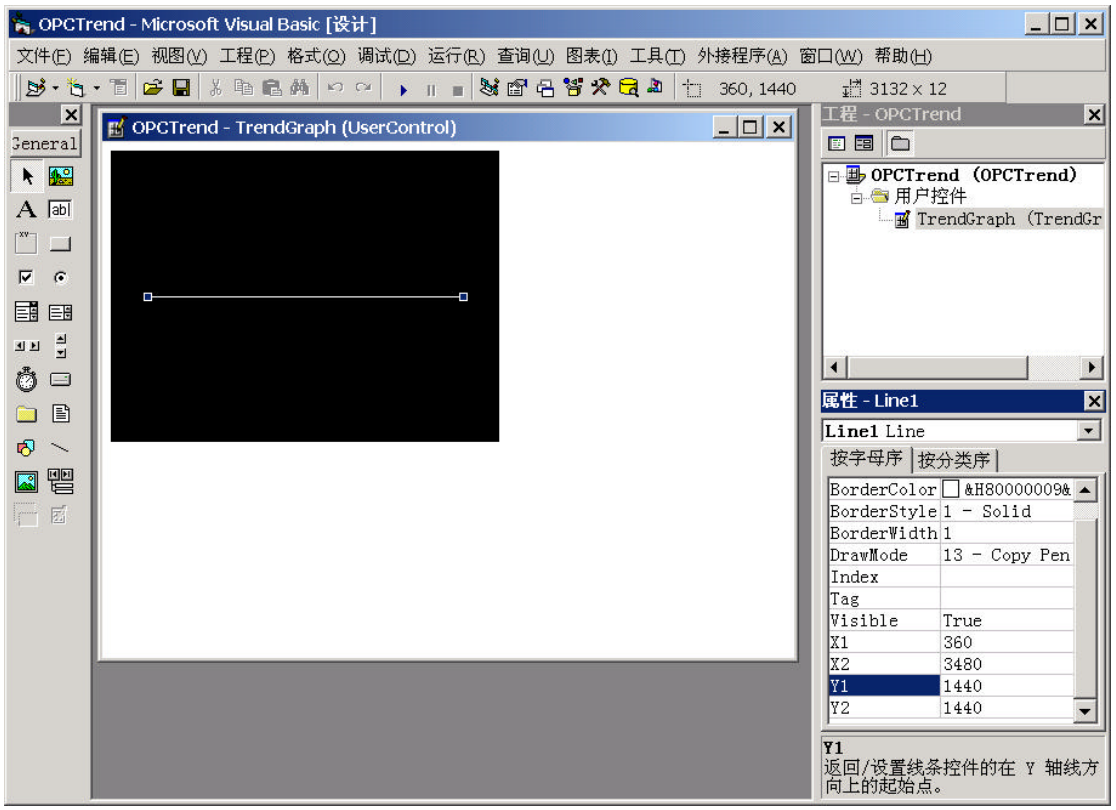


图 3-5 OPCTrend设计画面之二

到此为止，有关我们的用户控件的外观设置已经全部完成。

2.1.1 必需的引用和声明

我们制作的OPC ActiveX**控件需要使用**OPC**自动化包装**DLL**，但是新建的**VB**工程并没有注册这个包装**DLL**。首先需要在**VB**的引用设置里选择**OPC**自动化包装**DLL**，并在用户控件的代码部分里对**OPC**自动化对象进行声明。

2.1.2 OPC自动化包装的引用

本章的OPC ActiveX**示范程序使用日本**OPC**协会开发的演示用的包装**DLL**。从**VB**菜单选择**[工程(P)]-[引用(N)...]，然后选择[OPC Automation 2.0]（图3-6）。

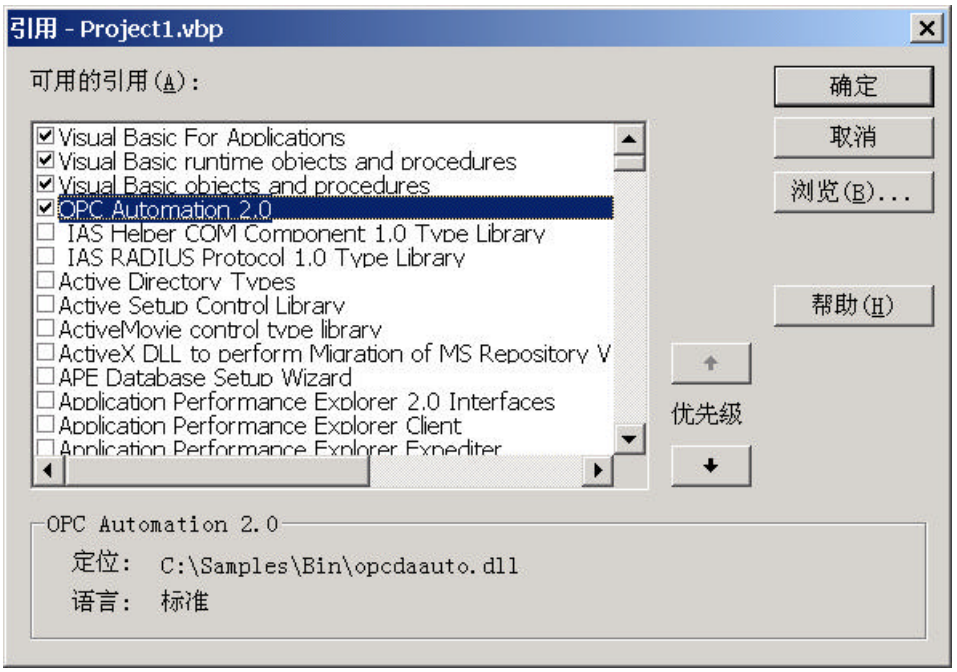


图 3-6 引用的设置

2.1.3 OPC对象和Windows API的声明

在OPC ActiveX**控件的代码框内最初部分需要进行**OPC**对象的声明和**Windows API**的声明（表**3-1**），由此在**OPC ActiveX**控件内就可以引用**OPC**自动化对象，并且在图形上表示数据时可以调用**Windows**的定时器**API**关数。**

表3-1 OPC对象和WindowsAPI函数的声明

```
Private Declare Function timeGetTime Lib "winmm.dll" () As Long
Private Declare Function timeBeginPeriod Lib "winmm.dll" (ByVal uPeriod As Long) As Long
Private Declare Function timeEndPeriod Lib "winmm.dll" (ByVal uPeriod As Long) As Long

' OPC对象
Dim WithEvents objMyOpcServer As OPCServer
Dim WithEvents objMyOpcGroup As OPCGroup
Dim objMyOpcGroups As OPCGroups
```

表3-2 OPC的处理代码之一

```
Private Sub objMyOpcServer_ServerShutDown(ByVal Reason As String)
    ' OPC服务器关机要求的处理
    Call Disconnect

    MsgBox Title:=Extender.Name, _
        Prompt:="OPC服务器关机。" & vbCr & """" & Reason & """", _
        Buttons:=vbInformation
End Sub

Private Sub objMyOpcGroup_DataChange(ByVal TransactionID As Long, ByVal NumItems As
Long, _
```

```
        ClientHandles() As Long, ItemValues() As Variant, Qualities() As Long, TimeStamps() As
Date)
    ' 数据变化事件的处理
    ' 记录时间和次数
    g_lOldTime = g_lNowTime
    g_lNowTime = timeGetTime
    g_dPf = g_dPf + 1

    ' 描绘趋势图
    Plot g_lNowTime, ItemValues(LBound(ItemValues))
End Sub
```

表3-3　OPC的处理代码之二

```
Private Sub OpcConnect(ByVal strProgID As String, ByVal strItemID As String)
    Dim myOpcServer As OPCServer
    Dim myOpcItems As OPCItems
    Dim myOpcGroups As OPCGroups
    Dim myOpcGroup As OPCGroup
    Dim strNode As String
    Dim lLength As Long

    ' 连接OPC服务器
    If g_bConnect Then Exit Sub
    Set myOpcServer = New OPCServer

    If Left(strProgID, 2) = "\\" Then
        ' 连接远程OPC服务器
        lLength = InStr(3, strProgID, "\")
        If lLength = 0 Then
            MsgBox Title:=Extender.Name, _
                Prompt:="程序标识符不正确。" & vbCr & "(" & strProgID & ")", _
                Buttons:=vbExclamation
            Set myOpcServer = Nothing
            Exit Sub
        End If
        strNode = Left(strProgID, lLength - 1)
        strProgID = Right(strProgID, Len(strProgID) - lLength)

        On Error GoTo ConnectError
        myOpcServer.Connect strProgID, strNode
        On Error GoTo 0
    Else
        ' 连接本地OPC服务器
        On Error GoTo ConnectError
        myOpcServer.Connect strProgID
        On Error GoTo 0
    End If

    ' OPC组的添加
    Set myOpcGroups = myOpcServer.OPCGroups
    Set myOpcGroup = myOpcGroups.Add("MyGroup")
    myOpcGroup.UpdateRate = 10          ' 设置更新周期为10毫秒。
```

```
    myOpcGroup.IsSubscribed = True      ' 使数据变化事件有效。
    Set myOpcItems = myOpcGroup.OPCItems

    Dim ItemServerHandles() As Long
    Dim ItemServerErrors() As Long
    Dim RequestedDataTypes(1) As Integer
    Dim AccessPaths As Variant
    Dim ClientHandles(1) As Long
    Dim OPCItemIDs(1) As String

    ' OPC标签的添加
    OPCItemIDs(1) = strItemID
    ClientHandles(1) = 1
    RequestedDataTypes(1) = vbDouble
    myOpcItems.AddItems 1, OPCItemIDs, ClientHandles, ItemServerHandles, _
        ItemServerErrors, RequestedDataTypes, AccessPaths

    Set objMyOpcServer = myOpcServer
    Set objMyOpcGroups = myOpcGroups
    Set objMyOpcGroup = myOpcGroup

    g_bConnect = True
    Exit Sub
ConnectError:
    ' OPC服务器连接错误
    Set myOpcServer = Nothing

    MsgBox Title:=Extender.Name, _
        Prompt:="程序标识符错误。" & vbCr & "(" & ProgID & ")", _
        Buttons:=vbExclamation
End Sub

Private Sub OpcDisConnect()
    ' 断开OPC服务器
    If Not g_bConnect Then Exit Sub

    objMyOpcGroups.RemoveAll
    objMyOpcServer.Disconnect

    Set objMyOpcGroup = Nothing
    Set objMyOpcGroups = Nothing
    Set objMyOpcServer = Nothing

    g_bConnect = False
End Sub
```

2.2 添加控件的属性，方法和事件

这里说明怎样在VB工程里添加OPC ActiveX控件可以使用的过程（属性，方法和事件）。所有添加的过程列于表3-4，在表的后面分别有说明方法和属性的添加示例。

表3-4　OPC ActiveX控件的方法和属性

过程名称	过程类型	数据类型	备注
Connect	**方法**	Empty	
Disconnect	**方法**	Empty	
Plot	**方法**	Empty	**参数：**ByVal dX As Double, ByVal dY As Double
RangeX	**属性**	Double	**默认值：**10000
RangeY	**属性**	Double	**默认值：**1
PointColor	**属性**	Double	**默认值：**&HFFFFFFFF
LineColor	**属性**	OLE_COLOR	**默认值：**&HFFFFFFFF
BackColor	**属性**	OLE_COLOR	**默认值**
PointSize	**属性**	Long	**默认值：**1
AutoLink	**属性**	Boolean	**默认值：**False
ProgID	**属性**	String	**默认值：**"OPCJ.DADemoServer.1"
ItemID	**属性**	String	**默认值：**"SV1"
Time	**属性**	Long	
Pf	**属性**	Double	

（一）方法的添加示例

在Active**控件中添加表**3-4**中**Connect**方法时（表**3-5**），**

表**3-5　OPC ActiveX**控件方法的添加示例

过程名称	过程类型	数据类型	备注
Connect	**方法**	Empty	

在代码视窗打开的状态下选择[**工具**(T)]-[**添加过程**(P)]**，则显示**[**添加过程**] **视窗（图**3-7**）。**

图**3-7　Connect**子程序过程的添加

单击[**确认**]**按钮，则生成如图** 3-8**所示的方法过程的雏形。**

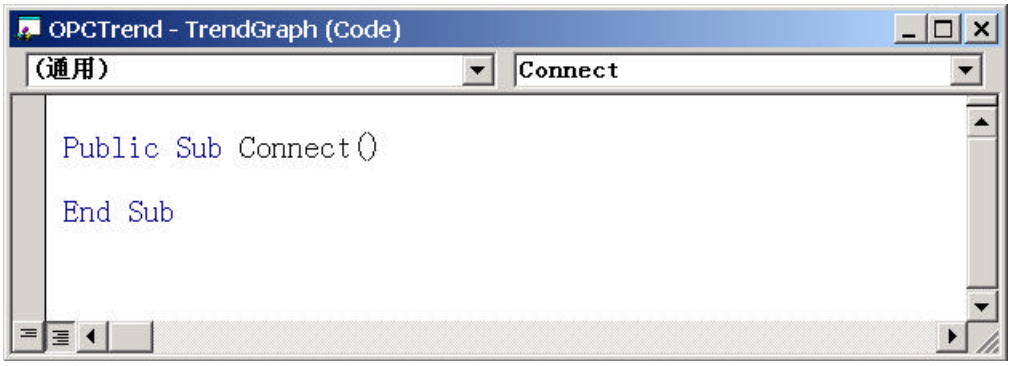


图 3-8 Connect子程序

在这里应记述执行这个方法的处理程序，记述的处理代码请参考表3-6**。**

表3-6 Connect方法的处理代码

```
Public Sub Connect()
   ' 连接OPC服务器
   If Ambient.UserMode Then
      ' 连接OPC服务器仅限于运行模式
      Call OpcConnect(strProgID, strItemID)
   End If
End Sub
```

（二）属性的添加示例

在Active**控件中添加表**3-4**中**RangeX**属性时（表**3-7**），**

表3-7 OPC ActiveX控件属性的添加示例

过程名称	过程类型	数据类型	备注
RangeX	属性	Double	默认值：10000

在代码视窗打开的状态下选择[工具(T)]-[添加过程(P)...]，则显示[添加过程] 视窗（图3-9**）。**

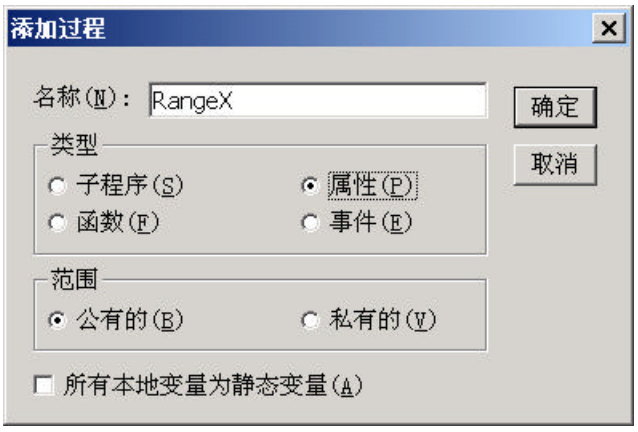


图 3-9 RangeX属性过程的添加

单击[确认]按钮，则生成如图3-10**所示的属性过程的雏形。**

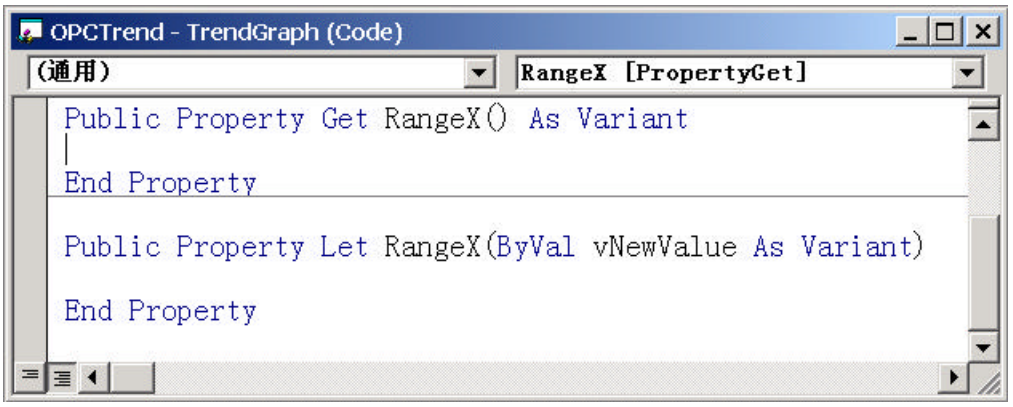


图 3-10 RangeX 属性

上面的“ Get RangeX() **”属性过程是读取属性值用的，而下面的“** Let RangeX **”属性过程则是设置属性值用的。新生成的过程的数据类型被自动声明为“** As Variant **”，但是如果考虑到处理速度，请指定具体的数据类型而不使用万能的“** Variant **”类型。这里按照表**3-7**将数据类型变成“** Double **”型。**

RengeX**的过程处理代码列于表**3-8**，而所有各个过程的处理内容总结于表**3-9**，表**3-10**和表**3-11**。**

表3-8 RengeX 属性的过程处理

```
' 表示X轴的范围
Public Property Get RangeX() As Double
    RangeX = dRangeX
End Property

Public Property Let RangeX(ByVal dNewValue As Double)
    dRangeX = dNewValue
    PropertyChanged "RangeX"
End Property
```

表3-9 属性的声明代码

```
Private Declare Function timeGetTime Lib "winmm.dll" () As Long
Private Declare Function timeBeginPeriod Lib "winmm.dll" (ByVal uPeriod As Long) As Long
Private Declare Function timeEndPeriod Lib "winmm.dll" (ByVal uPeriod As Long) As Long

' OPC对象
Dim WithEvents objMyOpcServer As OPCServer
Dim WithEvents objMyOpcGroup As OPCGroup
Dim objMyOpcGroups As OPCGroups

' 属性
Dim dRangeX As Double
Dim dRangeY As Double
Dim ocPointColor As OLE_COLOR
Dim bAutoLink As Boolean
Dim strProgID As String
Dim strItemID As String
```

```
' 内部工作变量
Dim g_bConnect As Boolean   ' 连接标志
Dim g_lNowTime As Long      ' 本次事件的发生时间
Dim g_lOldTime As Long      ' 上次事件的发生时间
Dim g_dPf As Double         ' 事件数
Dim g_dQuotient As Double
Dim g_bFirstPaint As Boolean

' 属性的初期值
Const conDefaultRangeX As Double = 10000
Const conDefaultRangeY As Double = 1
Const conDefaultPointColor As Long = &HFFFFFF
Const conDefaultLineColor As Long = &HFFFFFF
Const conDefaultBackColor As Long = &H0
Const conDefaultPointSize As Long = 1
Const conDefaultAutoLink As Boolean = False
Const conDefaultProgID As String = "OPCJ.DADemoServer.1"
Const conDefaultItemID As String = "SV1"
```

表3-10　属性的处理代码

```
Public Sub Connect()
   ' 连接OPC服务器
   If Ambient.UserMode Then
      ' 连接OPC服务器仅限于运行模式
      Call OpcConnect(strProgID, strItemID)
   End If
End Sub

Public Sub Disconnect()
   ' 断开OPC服务器
   Call OpcDisConnect
End Sub

Public Sub Plot(ByVal dX As Double, ByVal dY As Double)
   ' 描绘趋势图
   Dim dNewQuotient As Double
   Dim dAbsolute As Double

   dNewQuotient = dX \ RangeX
   dAbsolute = Abs(dY)
   If dNewQuotient <> g_dQuotient Then
      ' 改变显示范围
      Cls
      g_dQuotient = dNewQuotient
      g_bFirstPaint = True
   End If

   If g_bFirstPaint Then
      ' 描绘趋势图的第一点
      PSet (ScaleWidth * ((dX Mod RangeX) / RangeX), (ScaleHeight / 2) - ((ScaleHeight / 2)
* (dAbsolute / RangeY) * Sgn(dY))), PointColor
      g_bFirstPaint = False
```

```
    Else
        Line -(ScaleWidth * ((dX Mod RangeX) / RangeX), (ScaleHeight / 2) - ((ScaleHeight / 2)
* (dAbsolute / RangeY) * Sgn(dY))), PointColor
    End If
End Sub

' 表示X轴的范围
Public Property Get RangeX() As Double
    RangeX = dRangeX
End Property

Public Property Let RangeX(ByVal dNewValue As Double)
    dRangeX = dNewValue
    PropertyChanged "RangeX"
End Property

' 表示Y轴的范围
Public Property Get RangeY() As Double
    RangeY = dRangeY
End Property

Public Property Let RangeY(ByVal dNewValue As Double)
    dRangeY = dNewValue
    PropertyChanged "RangeY"
End Property

' 图的颜色
Public Property Get PointColor() As OLE_COLOR
    PointColor = ocPointColor
End Property

Public Property Let PointColor(ByVal ocNewValue As OLE_COLOR)
    ocPointColor = ocNewValue
    PropertyChanged "PointColor"
End Property

' 中心线的颜色
Public Property Get LineColor() As OLE_COLOR
    LineColor = linHorizon.BorderColor
End Property

Public Property Let LineColor(ByVal ocNewValue As OLE_COLOR)
    linHorizon.BorderColor = ocNewValue
    PropertyChanged "LineColor"
End Property

' 背景的颜色
Public Property Get BackColor() As OLE_COLOR
    BackColor = UserControl.BackColor
End Property

Public Property Let BackColor(ByVal ocNewValue As OLE_COLOR)
    UserControl.BackColor = ocNewValue
    PropertyChanged "BackColor"
```

```
End Property

' 图的宽
Public Property Get PointSize() As Long
    PointSize = UserControl.DrawWidth
End Property

Public Property Let PointSize(ByVal lNewValue As Long)
    UserControl.DrawWidth = lNewValue
    PropertyChanged "PointSize"
End Property

' 是否自动连接?
Public Property Get AutoLink() As Boolean
    AutoLink = bAutoLink
End Property

Public Property Let AutoLink(ByVal bNewValue As Boolean)
    bAutoLink = bNewValue
    PropertyChanged "AutoLink"
End Property

' OPC的设置
Public Property Get ProgID() As String
    ProgID = strProgID
End Property

Public Property Let ProgID(ByVal strNewValue As String)
    strProgID = strNewValue
    PropertyChanged "ProgID"
End Property

Public Property Get ItemID() As String
    ItemID = strItemID
End Property

Public Property Let ItemID(ByVal strNewValue As String)
    strItemID = strNewValue
    PropertyChanged "ItemID"
End Property

' 读取数据的时间间隔（仅用于运行模式）
Public Property Get Time() As Long
    Time = Abs(g_lNowTime - g_lOldTime)
End Property

Public Property Let Time(ByVal lNewValue As Long)
    ' 错误，只读用属性
    Err.Raise Number:=383
End Property

' 读取数据的次数（仅用于运行模式）
Public Property Get Pf() As Double
    Pf = g_dPf
```

```
End Property

Public Property Let Pf(ByVal dNewValue As Double)
    ' 错误，只读用属性
    Err.Raise Number:=383
End Property

Public Sub ShowAboutBox()
    ' 版本信息
    dlgAbout.Show vbModal
    Unload dlgAbout
    Set dlgAbout = Nothing
End Sub
```

表3-11　用户控件初始化和终止处理

```
Private Sub UserControl_Initialize()
    ' 内部工作变量的初始化
    g_bConnect = False
    g_lNowTime = 0
    g_lOldTime = 0
    g_dPf = 0
    g_dQuotient = 0
    g_bFirstPaint = True

    ' 启动定时器
    timeBeginPeriod 1
End Sub

Private Sub UserControl_InitProperties()
    ' 属性的初始化
    dRangeX = conDefaultRangeX
    dRangeY  = conDefaultRangeY
    ocPointColor = conDefaultPointColor

    linHorizon.BorderColor = conDefaultLineColor
    UserControl.BackColor = conDefaultBackColor
    UserControl.DrawWidth = conDefaultPointSize

    bAutoLink = conDefaultAutoLink
    strProgID = conDefaultProgID
    strItemID = conDefaultItemID

    If AutoLink Then
        ' 自动连接
        Call Connect
    End If
End Sub

Private Sub UserControl_ReadProperties(PropBag As PropertyBag)
    ' 保存属性的读取
    dRangeX = PropBag.ReadProperty("RangeX", conDefaultRangeX)
    dRangeY = PropBag.ReadProperty("RangeY", conDefaultRangeY)
```

```
    ocPointColor = PropBag.ReadProperty("PointColor", conDefaultPointColor)

    linHorizon.BorderColor = PropBag.ReadProperty("LineColor", conDefaultLineColor)
    UserControl.BackColor = PropBag.ReadProperty("BackColor", conDefaultBackColor)
    UserControl.DrawWidth = PropBag.ReadProperty("PointSize", conDefaultPointSize)

    AutoLink = PropBag.ReadProperty("AutoLink", conDefaultAutoLink)
    ProgID = PropBag.ReadProperty("ProgID", conDefaultProgID)
    ItemID = PropBag.ReadProperty("ItemID", conDefaultItemID)

    If AutoLink Then
        ' 自动连接
        Call Connect
    End If
End Sub

Private Sub UserControl_WriteProperties(PropBag As PropertyBag)
    ' 属性的保存
    PropBag.WriteProperty "RangeX", dRangeX, conDefaultRangeX
    PropBag.WriteProperty "RangeY", dRangeY, conDefaultRangeY
    PropBag.WriteProperty "PointColor", ocPointColor, conDefaultPointColor

    PropBag.WriteProperty "LineColor", linHorizon.BorderColor, conDefaultLineColor
    PropBag.WriteProperty "BackColor", UserControl.BackColor, conDefaultBackColor
    PropBag.WriteProperty "PointSize", UserControl.DrawWidth, conDefaultPointSize

    PropBag.WriteProperty "AutoLink", AutoLink, conDefaultAutoLink
    PropBag.WriteProperty "ProgID", strProgID, conDefaultProgID
    PropBag.WriteProperty "ItemID", strItemID, conDefaultItemID
End Sub

Private Sub UserControl_Terminate()
    ' 断开OPC服务器
    Call Disconnect

    ' 停止定时器
    timeEndPeriod 1
End Sub

Private Sub UserControl_Resize()
    ' 控件大小变化的处理
    linHorizon.X1 = 0
    linHorizon.Y1 = ScaleHeight / 2
    linHorizon.X2 = ScaleWidth
    linHorizon.Y2 = ScaleHeight / 2
End Sub
```

2.3 建立属性页

下面我们要制作OPC ActivX**控件的属性页，用以设置以上我们添加的各种属性。请从**VB**菜单选择**[**工程**(P)]-[**添加属性页**(P)]，并选择**[**属性页**]的图标，再按下**[**打开**(O)]的命令按钮（图**3-11）。

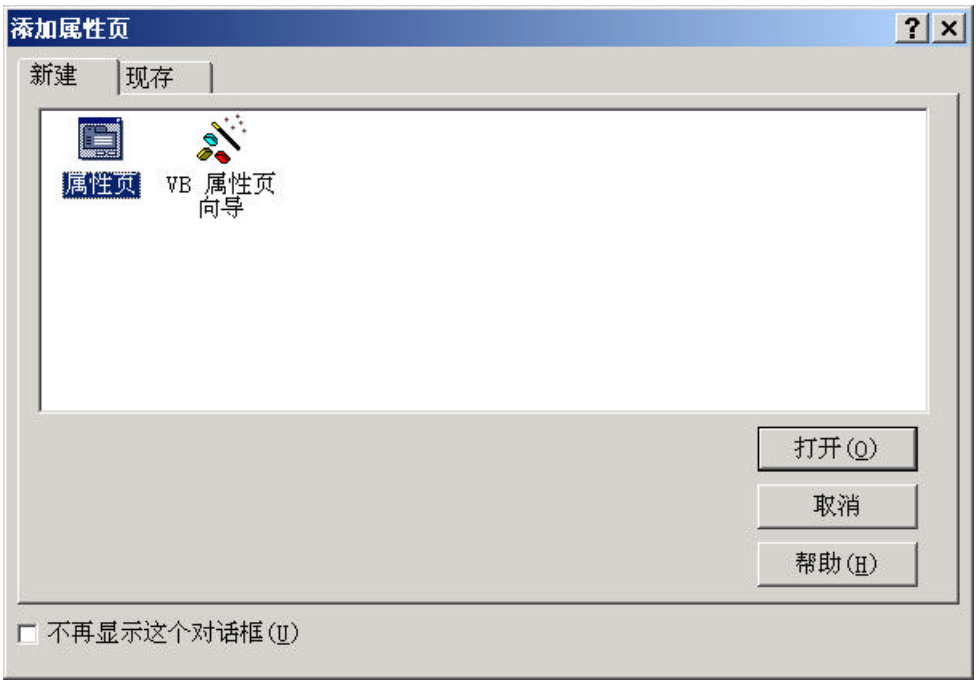


图 3-11 属性页的添加

在新建的属性页，请配置三个标签和三个文件框（图3-12）。

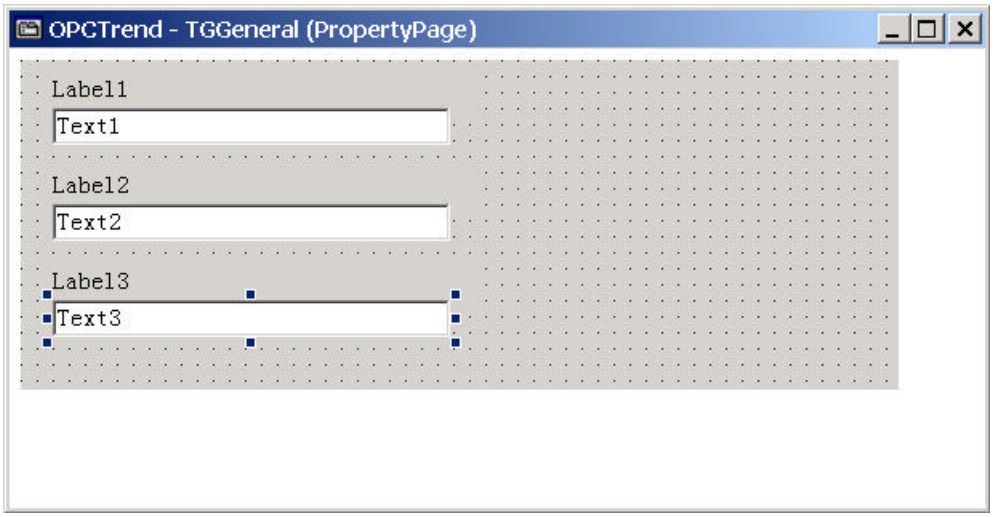


图 3-12 OPCTrend的属性页

然后设置属性页内控件的属性值（表3-12）。

表3-12 [通用]属性页的控件

控件	属性	值
属性页	(名称)	TGGeneral
	Caption	通用
标签	Caption	X轴范围(X)：
文字框	Caption	Y轴范围(Y)：

文字框	Caption	**刷新点大小(S)：**
文字框	**(名称)**	TxtRangeX
	Text	
文字框	**(名称)**	TxtRangeY
	Text	
文字框	**(名称)**	TxtPointSize
	Text	

这个属性页的画面成为图 3-13的样子。

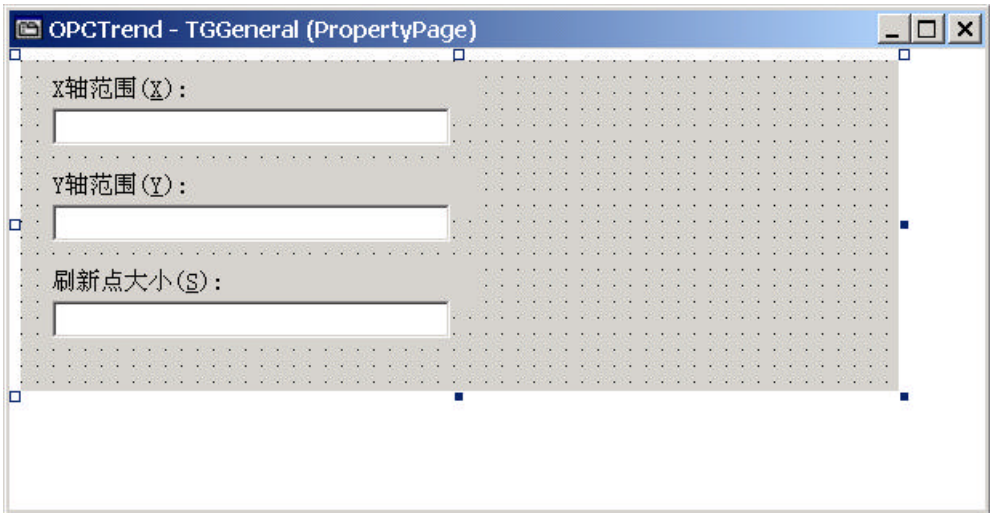


图 3-13 [通用]属性页

再在作成的通用属性页内，添加表3-13的处理代码。

表3-13 [通用]属性处理代码

```
Private Sub PropertyPage_ApplyChanges()
    Dim objControl As Object
    For Each objControl In SelectedControls
        objControl.RangeX = txtRangeX.Text
        objControl.RangeY = txtRangeY.Text
        objControl.PointSize = txtPointSize.Text
    Next
End Sub

Private Sub PropertyPage_SelectionChanged()
    ' 初始化
    txtRangeX.Text = SelectedControls(0).RangeX
    txtRangeY.Text = SelectedControls(0).RangeY
    txtPointSize.Text = SelectedControls(0).PointSize
End Sub

Private Sub txtRangeX_Change()
    Changed = True
End Sub
```

```
Private Sub txtRangeY_Change()
    Changed = True
End Sub

Private Sub txtPointSize_Change()
    Changed = True
End Sub
```

请使用上述同样的方法，再新建一页内容如表3-14所示的属性页。

表3-14 [OPC设置]属性页的控件

控件	属性	值
属性页	(名称)	OPCGeneral
	Caption	OPC设置
标签	Caption	程序标识符(P)：
标签	Caption	项标识符(I)：
标签	(名称)	txtProgID
	Text	
文字框	(名称)	TxtItemID
	Text	
复选框	(名称)	ChkAutoLink
	Caption	自动连接(A)：

这个属性页的显示成为图 3-14的样子。

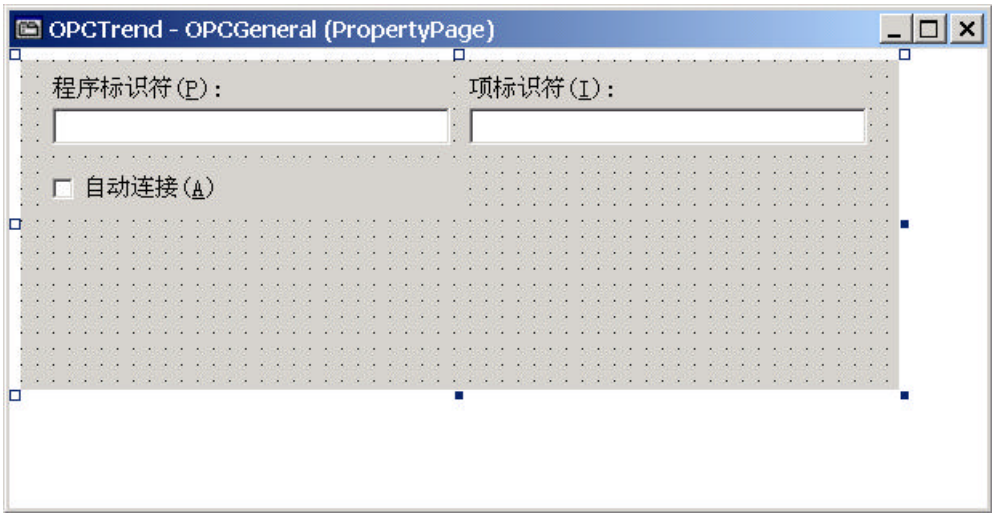


图 3-14 [OPC设置]属性页

请在作成的OPC设置属性页内，添加表3-15的处理代码

表3-15 [OPC设置]属性页处理代码

```
Private Sub PropertyPage_ApplyChanges()
    Dim objControl As Object
```

```
    For Each objControl In SelectedControls
        objControl.ProgID = txtProgID.Text
        objControl.ItemID = txtItemID.Text
        If chkAutoLink.Value = 0 Then
            objControl.AutoLink = False
        Else
            objControl.AutoLink = True
        End If
    Next
End Sub

Private Sub PropertyPage_SelectionChanged()
    ' 初始化
    txtProgID.Text = SelectedControls(0).ProgID
    txtItemID.Text = SelectedControls(0).ItemID
    If SelectedControls(0).AutoLink Then
        chkAutoLink.Value = 1
    Else
        chkAutoLink.Value = 0
    End If
End Sub

Private Sub txtProgID_Change()
    Changed = True
End Sub

Private Sub txtItemID_Change()
    Changed = True
End Sub

Private Sub chkAutoLink_Click()
    Changed = True
End Sub
```

在用户控件TrendGraph**的属性框内选择**[PropertyPages]**属性，可以确认属性页的连接。选择的属性页和顺序应该如图** 3-15**所示。**

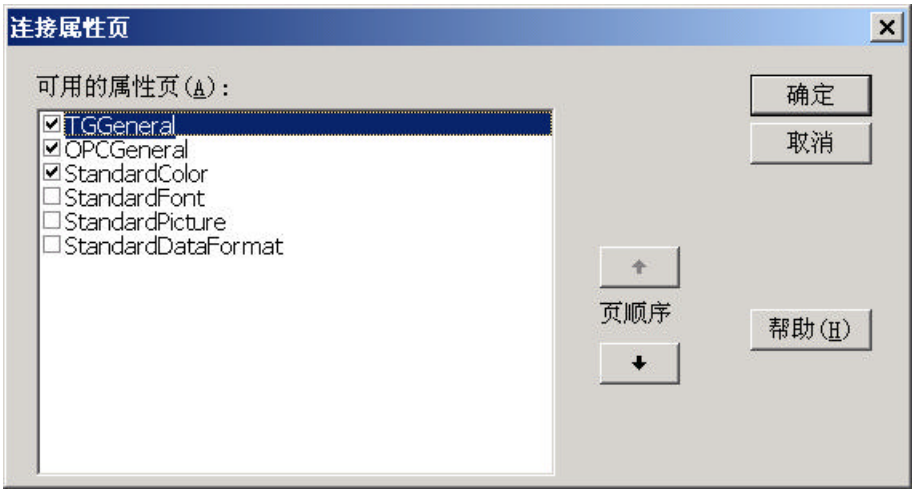


图** 3-15 OPCTrend**属性页的连接

确认属性页动作的方法是将这个TrendGraph控件配置在VB的标准EXE窗体上，将鼠标移至这个配置的TrendGraph1控件上并按下鼠标右按钮，在显示的菜单中选择[属性(R)]项目，可以显示作成的属性页。另外在配置的TrendGraph1控件的属性框内，双击[(名称)]栏下面的[(自定义)]也可以显示作成的属性页。图 3-16是到此为止作成的属性页的画面。

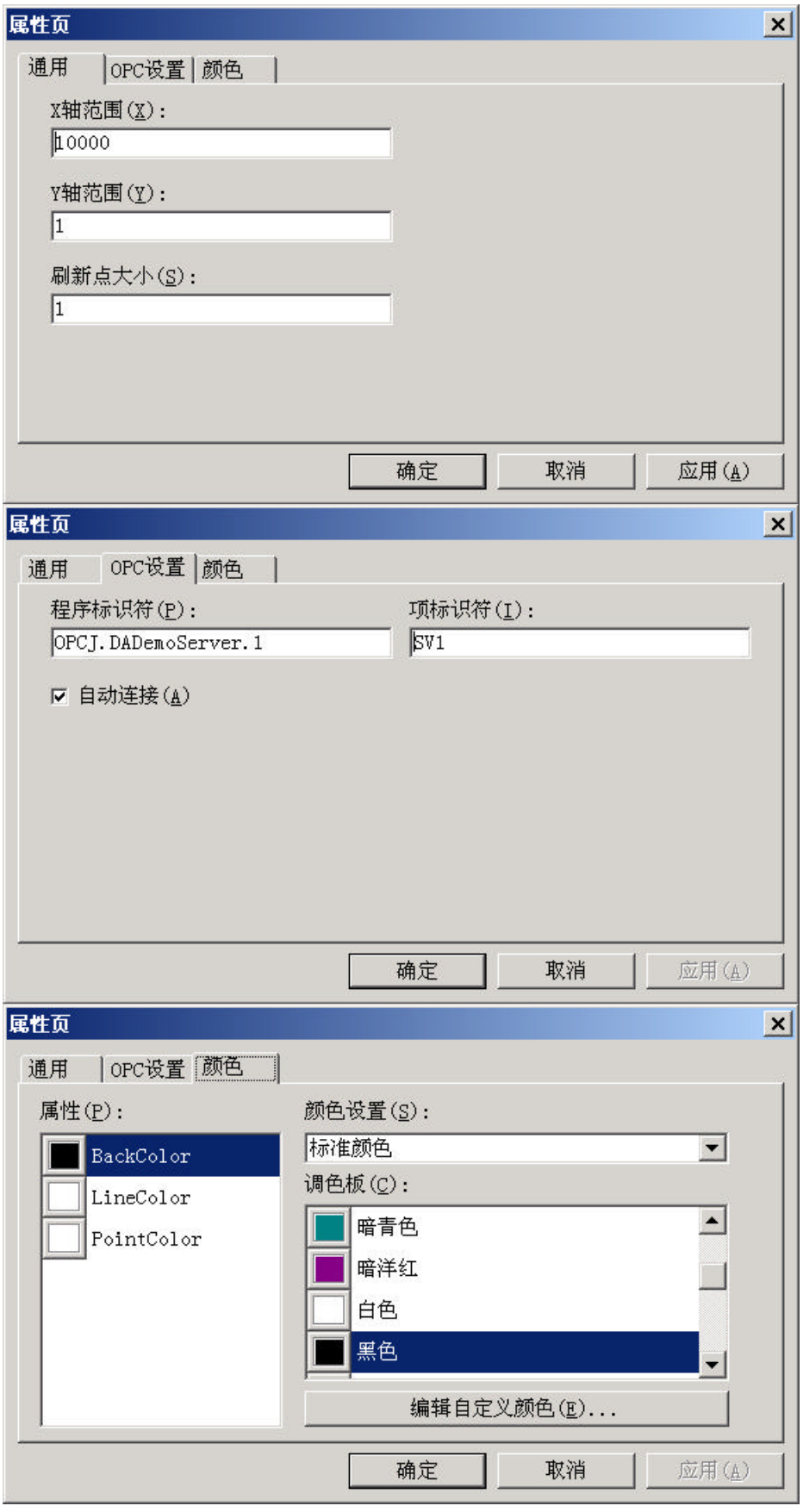


图 3-16 完成的OPCTrend属性页画面

2.4 版本信息窗体

版本信息窗体是表示ActiveX控件的版本和供应商的信息的对话框。在控件内添加版本信息窗体，需要新建一个显示版本信息窗体的过程，再把这个过程的标识符设置于版本信息窗体。本节的示例是怎样使用自定义窗体作为控件的版本信息窗体，其实利用对话框函数（MsgBox）也可以作成简单的版本信息窗体。

从菜单选择[工程(P)]-[添加窗体(F)]（图3-17）。

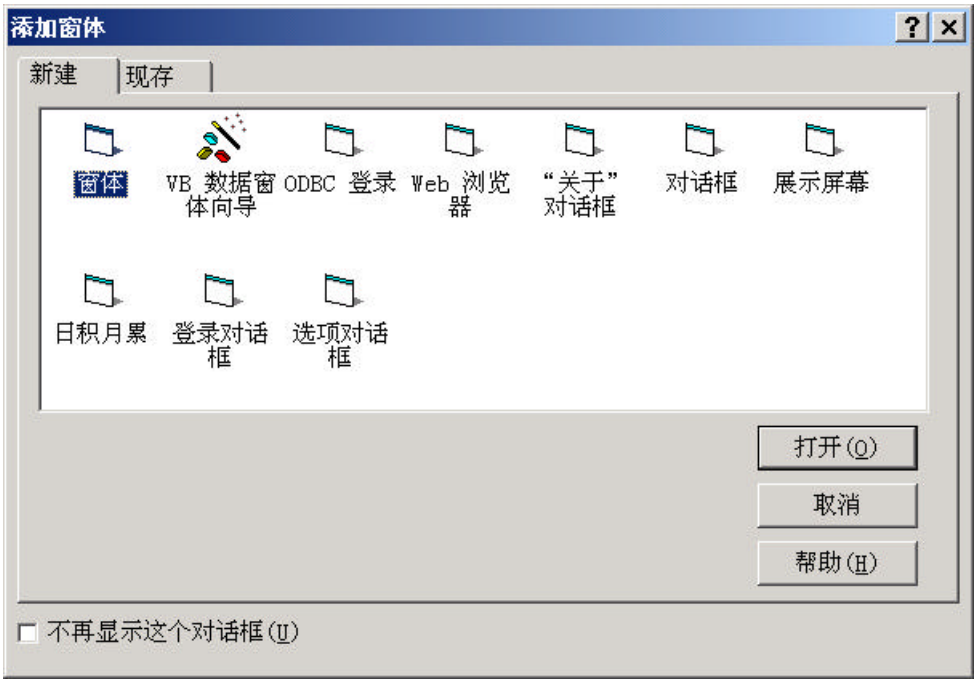


图 3-17添加窗体

选择[窗体]图标，再按下[打开(O)]按钮（图3-18）。

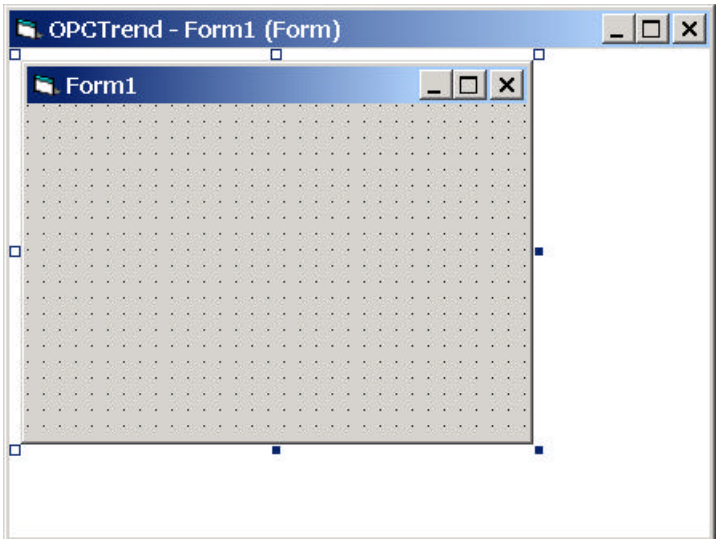


图 3-18 OPCTrend的窗体

请参考列于表3-16的属性设置值，制作版本信息窗体的模块。

表3-16 版本信息窗体的控件

控件	属性	值
窗体	(名称)	DlgAbout
	Caption	有关TrendGraph控件...
标签	(名称)	LblVersion
	Caption	版本
标签	(名称)	LblCopyright
	Caption	版权
命令按钮	(名称)	CmdOK
	Caption	确认
图形框	(名称)	(Metafile)

在作成的版本信息窗体内记述表3-17的代码。在控件代码窗体表示的情况下，从菜单选择[工具(T)]-[添加过程(P)...]。然后在表示的[在添加过程]对话框内，将[名称(N)]定义为“ ShowAboutBox ”，选择[类型]为[子程序(S)]，再按下[确定]按钮，则可以生成子程序的雏形（图3-19）。

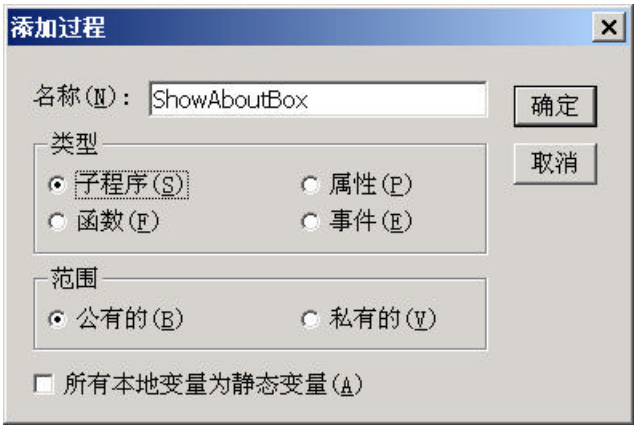


图 3-19 子程序过程的添加

表3-17　版本信息窗体的处理代码

```
Private Sub Form_Load()
   lblVersion.Caption = App.Title & " 控件版本" & App.Major & "." & App.Minor & _
       "Rev" & App.Revision
   lblCopyright.Caption = App.LegalCopyright
End Sub

Private Sub cmdOK_Click()
   Unload Me
End Sub
```

接着从菜单选择[工具(T)]-[过程属性(A)...]，在表示的[过程属性]对话框内，将[名称(N)]设置为“ ShowAboutBox ”，再按下[高级(V) >>]按钮，将[过程标识符(I):]选择为[AboutBox]。最后按下[确定]（图3-20）。

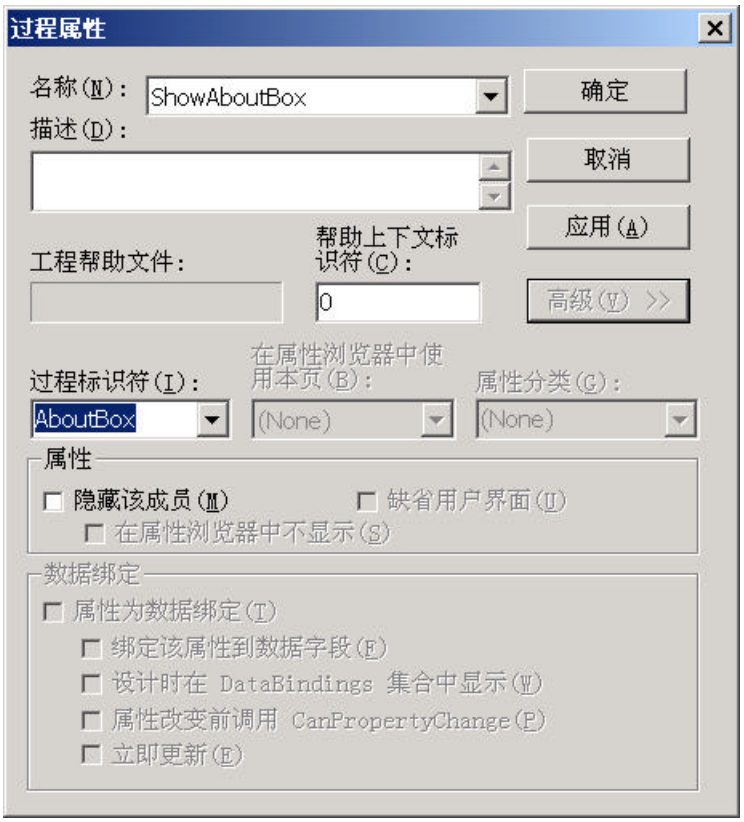


图 3-20 过程属性

将添加了版本信息窗体的ActiveX**控件配置于测试的窗体，按下在属性框内[(名称)]栏上面的[(关于)]栏的[...]命令按钮，则可以显示作成的版本信息窗体（图**3-21**）。**

图 3-21 版本信息

下面说明工具箱图标的添加方法。如果想设置自定义的工具箱图标的话，只要在用户控件 TrendGraph**的属性框的[**ToolBoxBitmap**]属性栏设置要使用的位图文件名就可以了。这个文件应该是一个**16×15**像素的**16**色的位图（图**3-22**）。**

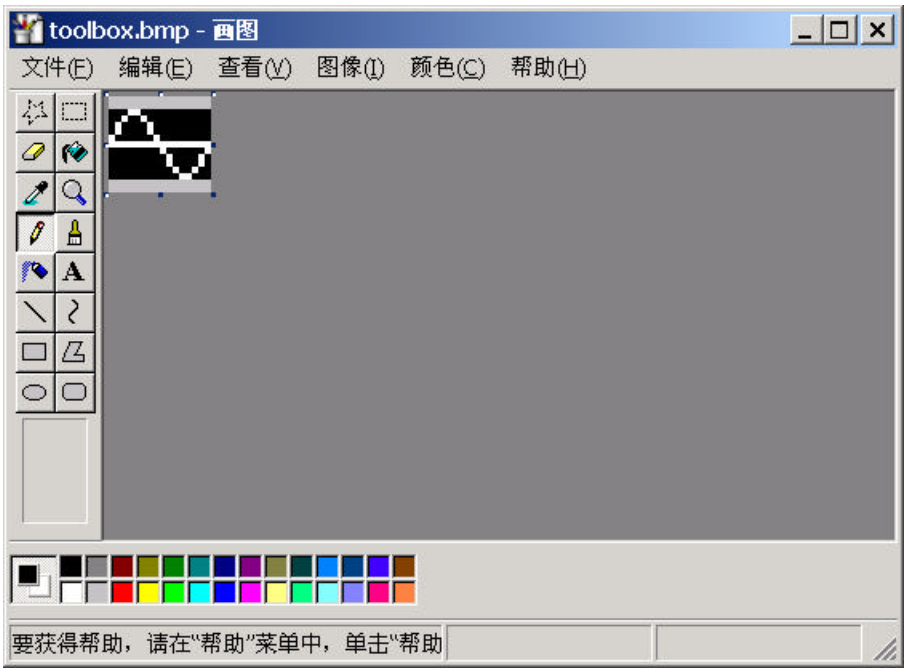


图 3-22 工具箱图标位图

在[ToolBoxBitmap]**里指定的位图将成为在控件工具箱上表示的图标（图** 3-23**）。**

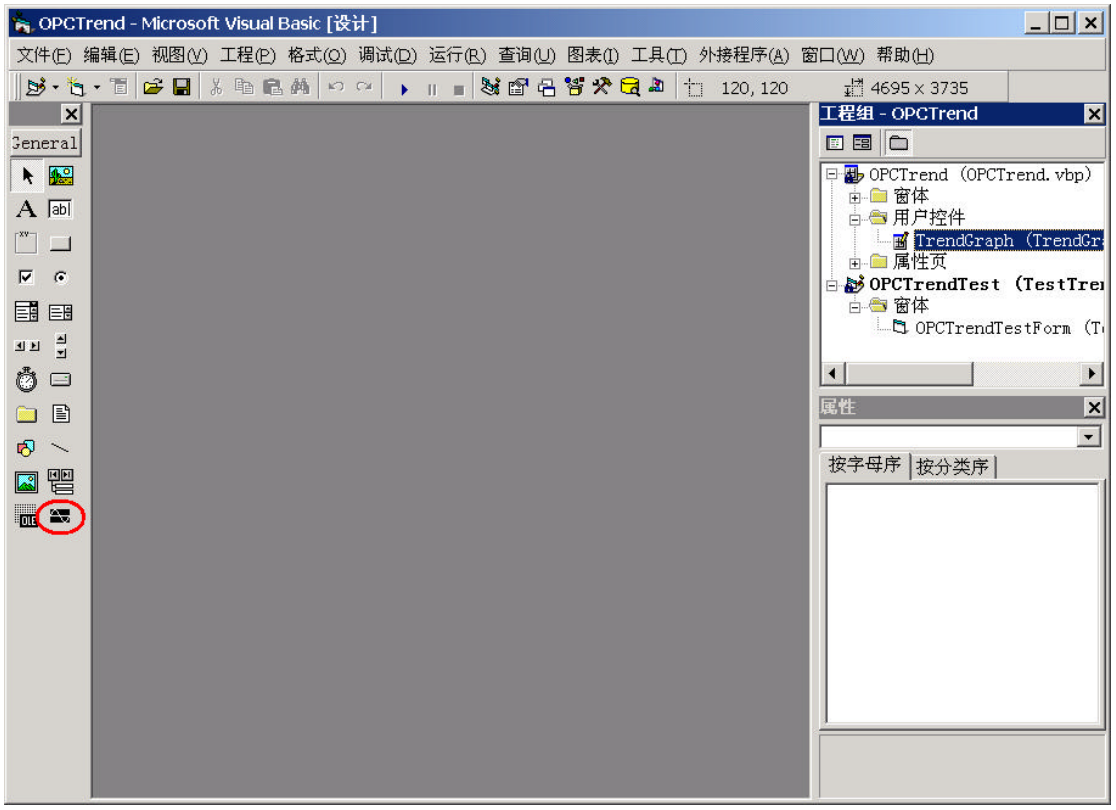


图 3-23 工具箱的表示

到此为止，OPC ActiveX**控件的所有设置已经完成。从菜单选择**[**文件(**F**)**]-[**保存工程(**V**)...]**保存初步完成的工程文件。**

2.5 调试ActiveX控件

一般来说，有两种方法调试刚作成的OPC ActiveX控件。其一是使用互联网浏览器（例如，3.0或更高版本的Internet Explorer）的方法，其二是使用VB的调试方法。本节所介绍的是使用VB的调试方法。

2.5.1 建立一个Visual Basic工程

在上节作成的OPCTrend的VB工程之上，再添加一个调试用的VB工程，组成一个VB工程组，用于调试已初步完成的OPCTrend ActiveX控件。

从VB菜单选择[文件(F)]-[添加工程(D)...]，并在[新建]框内选择[标准EXE]工程类型。

新建的VB工程的默认名称为“Project1”，让我们把这个名称设置为“OPCTrendTest”。工程名称的设置，应该从VB菜单选择[工程(P)]-[Project1 属性(E)...]。然后在表示的[Project1 – 工程属性]对话框内，改变以下的项目（图3-24）。

[工程名称(N):] → “OPCTrendTest”

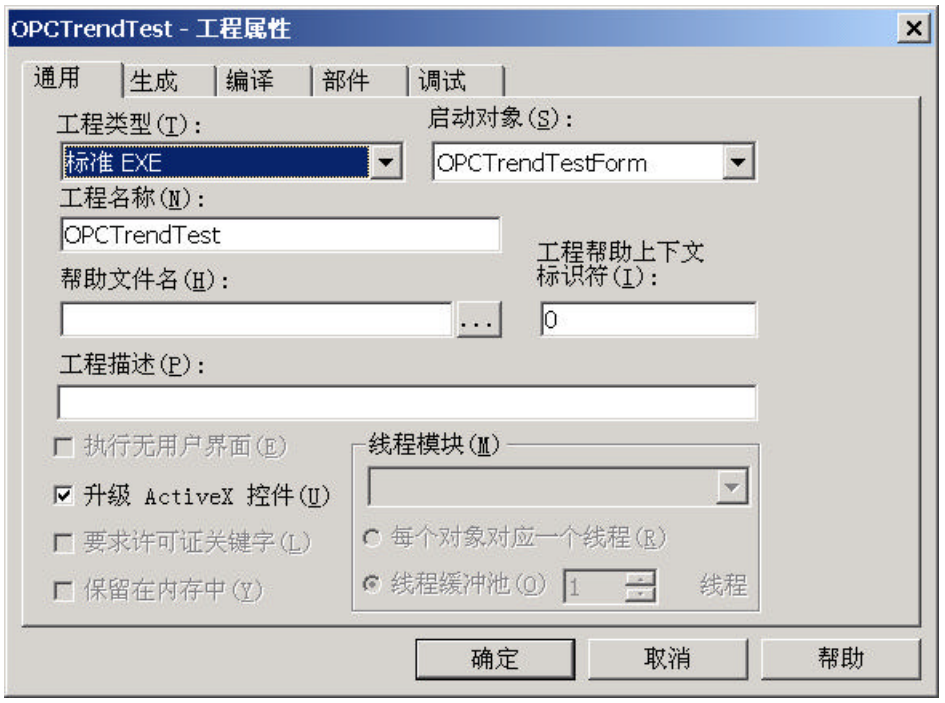


图 3-24 工程属性

下面在表示的窗体上配置上面作成OPC ActiveX控件。如果OPC ActiveX控件（OPCTrend.vbp）的窗体或者代码框在打开的状态的话，请将其关闭。OPC ActiveX控件的图标应该表示在左侧控件工具箱的最下面（图3-25）。

图 3-25 工具箱

选择这个注册了的ActivX**控件图标，然后指定在窗体上的配置领域，即可完成**ActiveX**的配置（图**3-26**）。**

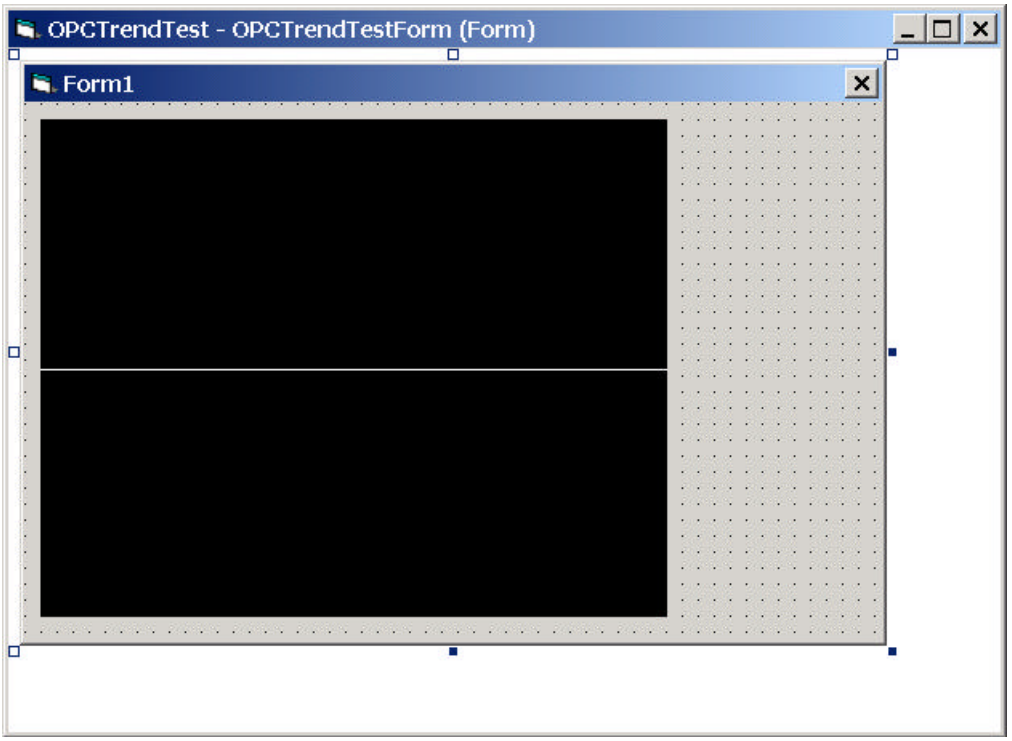


图 3-26 OPC ActiveX 控件的添加

再将列于表3-18**的控件配置于调试窗体，并设置其属性值。**

表3-18 OPCTrendTestFrom的控件

控件	属性	值
窗体	(名称)	OPCTrendTestFrom
	Caption	实验
文字框	Caption	横轴(X)：
标签	Caption	纵轴(Y)：
文字框	(名称)	TxtRangeX
	Text	
文字框	(名称)	TxtRangeY
	Text	
命令按钮	(名称)	CmdRange
	Caption	应用 (A)：

完成的调试用窗体如图 3-27**所示。**

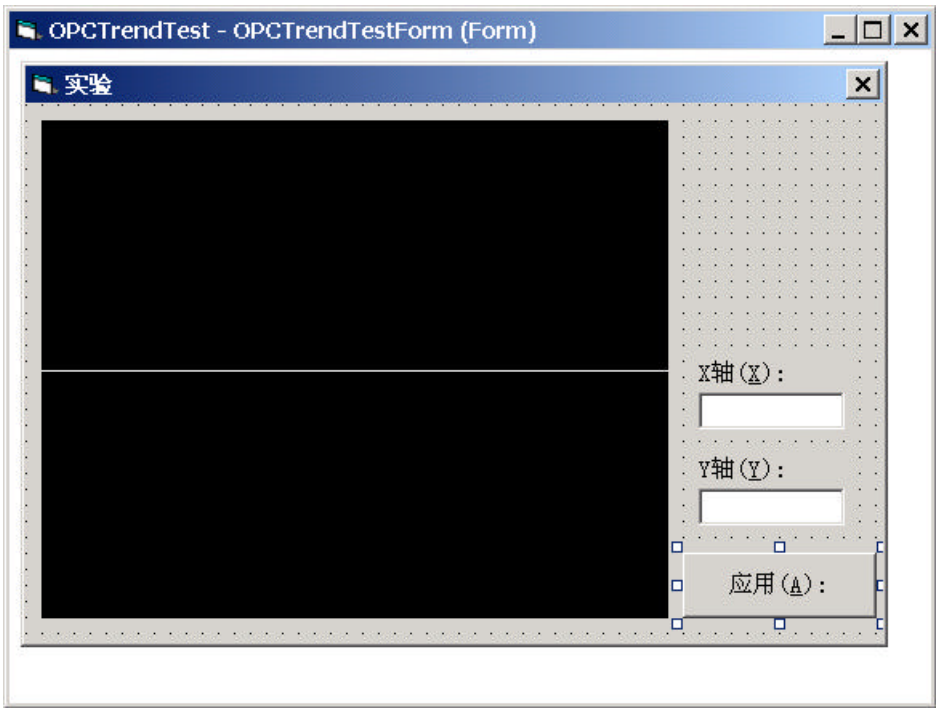


图 *3-27 OPC ActiveX* ***的实验窗体***

这个调试用窗体内记述的代码如表3-19**所示。**

表*3-19 OPCTrendTestFrom****的处理代码***

```
Private Sub cmdRange_Click()
    TrendGraph1.RangeX = CDbl(txtRangeX.Text)
    TrendGraph1.RangeY = CDbl(txtRangeY.Text)
End Sub

Private Sub Form_Load()
    txtRangeX.Text = CStr(TrendGraph1.RangeX)
    txtRangeY.Text = CStr(TrendGraph1.RangeY)
End Sub
```

2.5.2 调试ActiveX控件

首先需要将调试用VB工程设置为启动工程。用鼠标选择OPCTrendTest工程（TestTrend.vbp）并按下鼠标右按钮，在显示的弹出式菜单上选择[设置为启动(U)]（图3-28）。

图 3-28 作为启动工程的设置

在OPCTrendTest工程内，选择配置的TrendGraph1控件并按下鼠标右按钮，在显示的菜单中选择[属性(R)]项目，可以显示作成的属性页。显示[OPC设置]属性页，并选择[自动连接(A)]复选框（图3-29）。

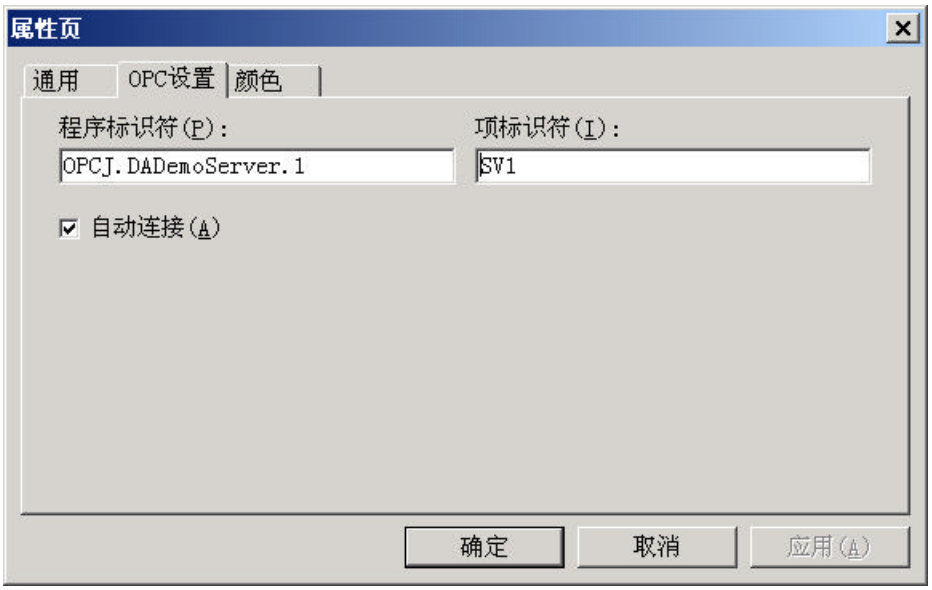


图 3-29 [OPC设置]属性页

从VB菜单选择[运行(R)]-[启动(S)]，可以启动OPCTrendTest工程。图 3-30是运行的画面显示。

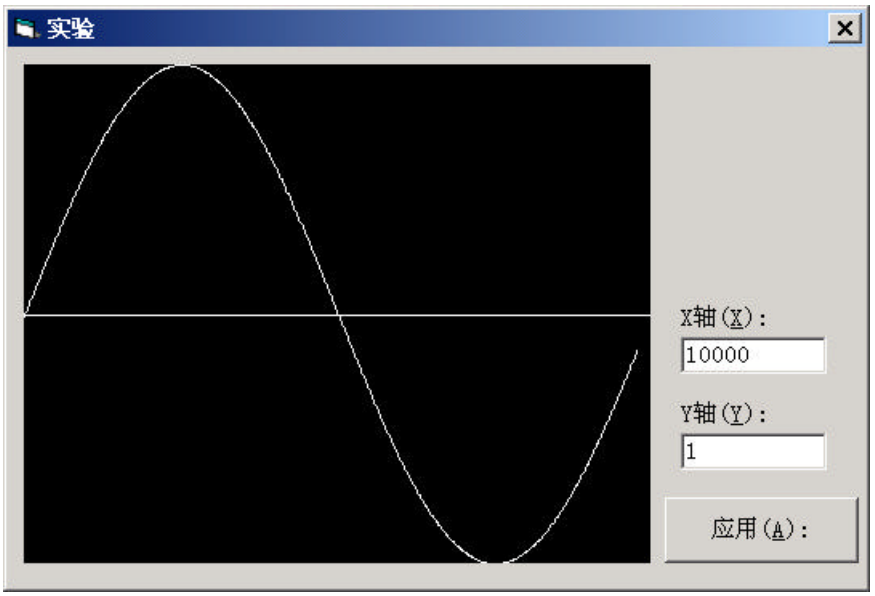


图 3-30 运行画面

X轴范围的默认值是“ 10000（毫秒） ”，而Y轴的范围是“ 1 ”。运行时选择X轴和Y轴对应文字框并改变设置值，可以改变图形的表示范围。在设置值变更后，需按下[应用 (A)]命令按钮使设置值生效。

2.6 生成ActiveX控件

在作成的ActiveX控件充分调试之后，应该生成一个可执行文件。请选择OPCTrend工程，再从VB菜单选择[文件(F)]-[生成OPCTrend.ocx(K)...]，在显示的[生成工程]对话框内按下[确定]命令按钮，就可生成可执行文件。（图3-31）。

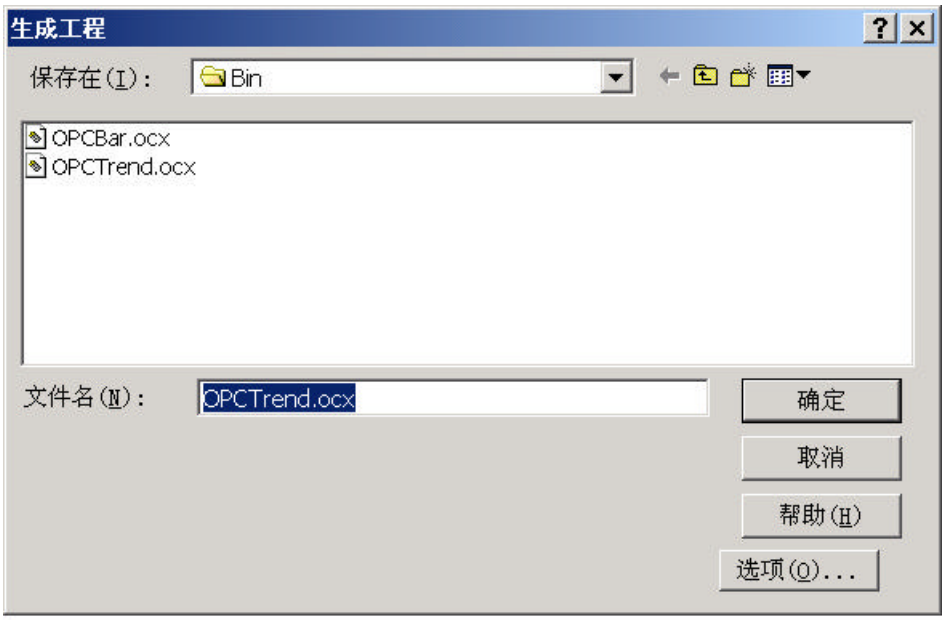


图 3-31 运行可能文件的生成

但是在第２次生成这样的可执行文件前，为了维持与以前生成的ActiveX控件的版本兼容性，需要设置版本的兼容性选项。如果不进行这样的兼容性设置，使用以前生成的OPCTrend.ocx控件的现存VB应用程序，就不再能正常动作了。

请从VB菜单选择[工程(P)]-[Project1 属性(E)...]，在表示的[Project1 – 工程属性]对话框内选择[部件]选项卡，并进行以下兼容性选项设置。

选择[版本兼容]的[二进制兼容(B)]选项按钮。

此外，按下[...]命令按钮，显示[可兼容的 ActiveX部件]对话框，将以前生成的OPCTrend.ocx设置为需要与其维持兼容性的文件（图3-32）。

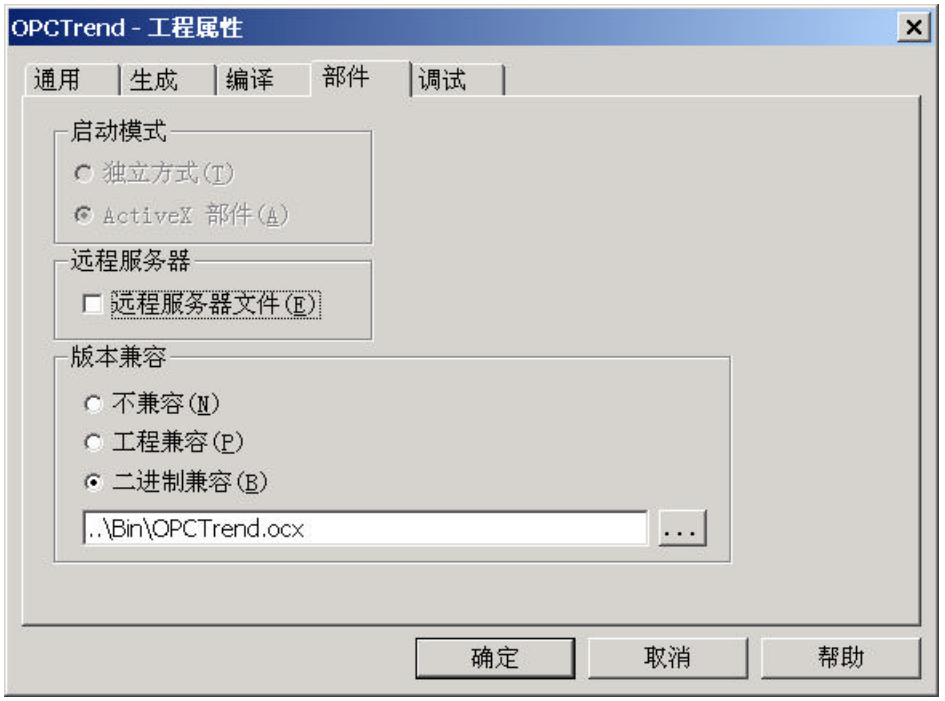


图 3-32 工程属性页

因为版本兼容设置为[二进制兼容]后生成的可执行的ActiveX控件可以维持与以前生成的部件的版本兼容性，所以使用以前生成的ActiveX控件的VB应用程序还可以象以前一样继续正常动作。

3 使用Excel开发OPC应用程序

本章是介绍两种使用微软Excel的VBA（应用程序的Visual Basic）的OPC应用程序的开发方法。一种是直接使用OPC自动化接口的Excel应用程序的方法，而另一种是使用OPC ActiveX控件的Excel应用程序的开发方法。本章介绍的使用VBA的OPC应用程序的开发，是使用OPC数据访问（版本2.0）接口，进行同步方式的数据读取。有关怎样开发具有OPC读写功能的ActiveX控件，可以参考第３章的内容。本章使用OPC ActiveX控件是由日本OPC协会开发的演示用软件组件，可以在本书配套光盘中找到它的源程序和可执行文件。

请参考第６章的［5.4 示范源程序的使用方法］，可以从配套光盘找到本章的介绍的示范程序。所介绍的源代码可以在Excel 97/ Excel 2000/ Excel XP上运行。

第２章和第３章说明了使用Visual Basic进行访问OPC数据访问服务器的方法。使用Visual Basic不仅可以开发可单独执行的EXE形式的文件，而且还可以开发供其他程序使用的ActiveX控件。有了这样的ActiveX控件，使得测量控制系统的构筑变得更为简单易行。

但是在制造现场仅仅是进行设备控制和数据收集往往还是不够的，有时还需要进行图形分析以及报表打印。当然使用Visual Basic也可以开发支持图形分析以及报表打印功能的应用程序，但是使用Excel往往可以更简单地实现这些功能。

3.1 使用Excel和VBA的OPC应用程序

使用VBA开发OPC的应用程序时，也和使用Visual Basic一样，需要OPC自动化接口包装DLL。因为这个OPC包装DLL一般应该是由OPC服务器供应商提供的，所以OPC包装DLL的名称等信息也是随供应商有所不同，有关具体情况请询问你的OPC服务器供应商。

本书使用为OPC基金会成员制作的OPC包装DLL进行说明。有关本书的OPC包装DLL的注册方法请参照第６章的［5.4.1 复制和注册示范源程序］。

3.1.1 定义Excel宏

1) 启动Excel，新建一个工作簿文件。
2) 从Excel的菜单选择[工具(T)]-[宏(M)]-[Visual Basic 编辑器(V)]，显示Visual Basic编辑器（图4-1）。

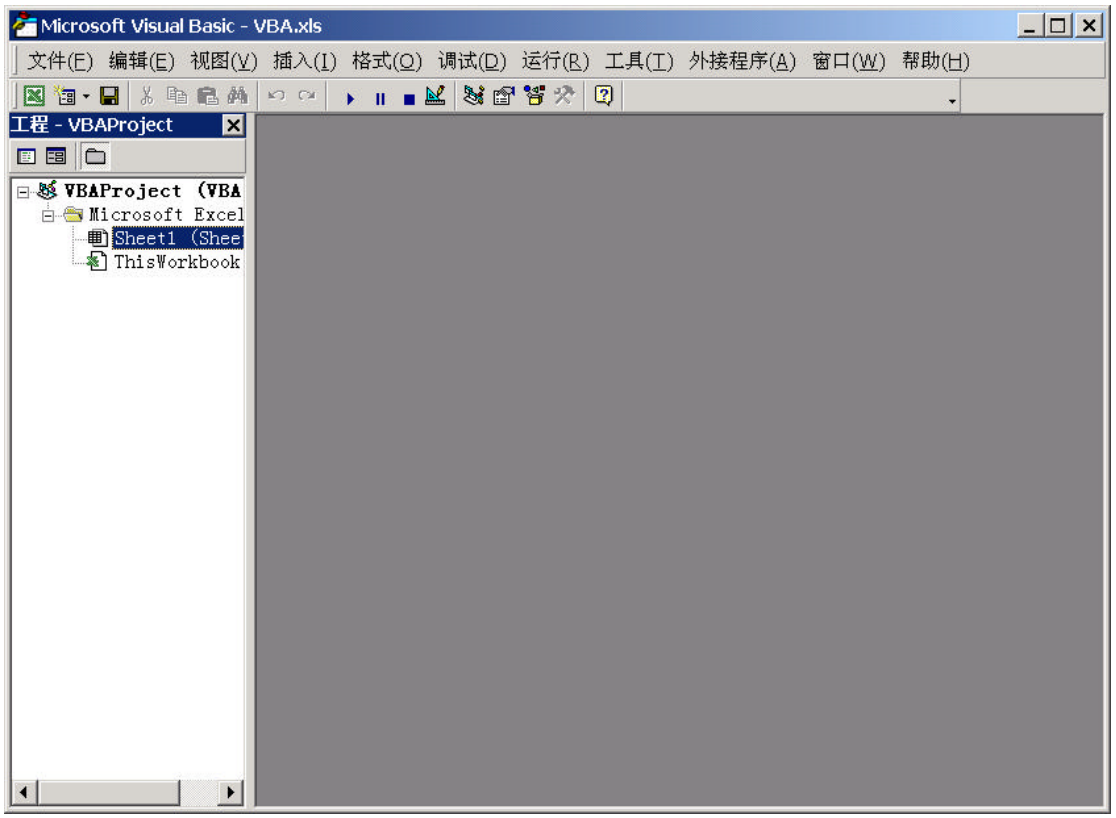


图 4-1 Visual Basic 编辑器

3) **从**Visual Basic**编辑器的菜单选择[工具(T)]-[引用(R)…]。在[可使用的引用(A)]的一览表示中，选择对应**OPC**包装**DLL**的文件（图**4-2**）。**

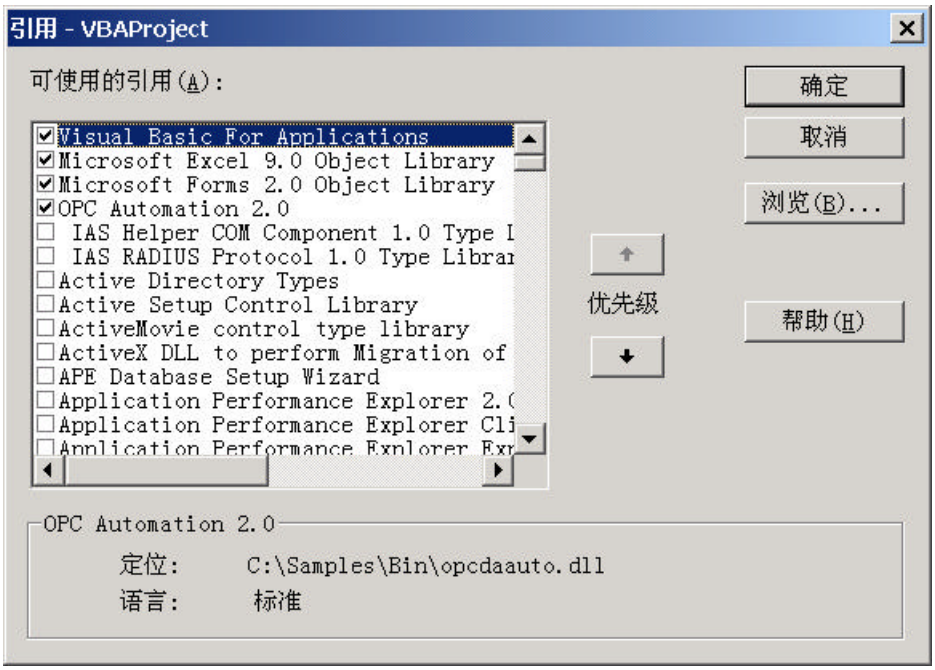


图 4-2 引用的设置

4) **从Visual Basic编辑器的菜单选择[插入(I)]-[模块(M)]后，在左侧的工程资源管理器内会自动添加一个[模块]文件夹，并在它的下面添加了一个名为[Module1]的项目（图4-3）。**

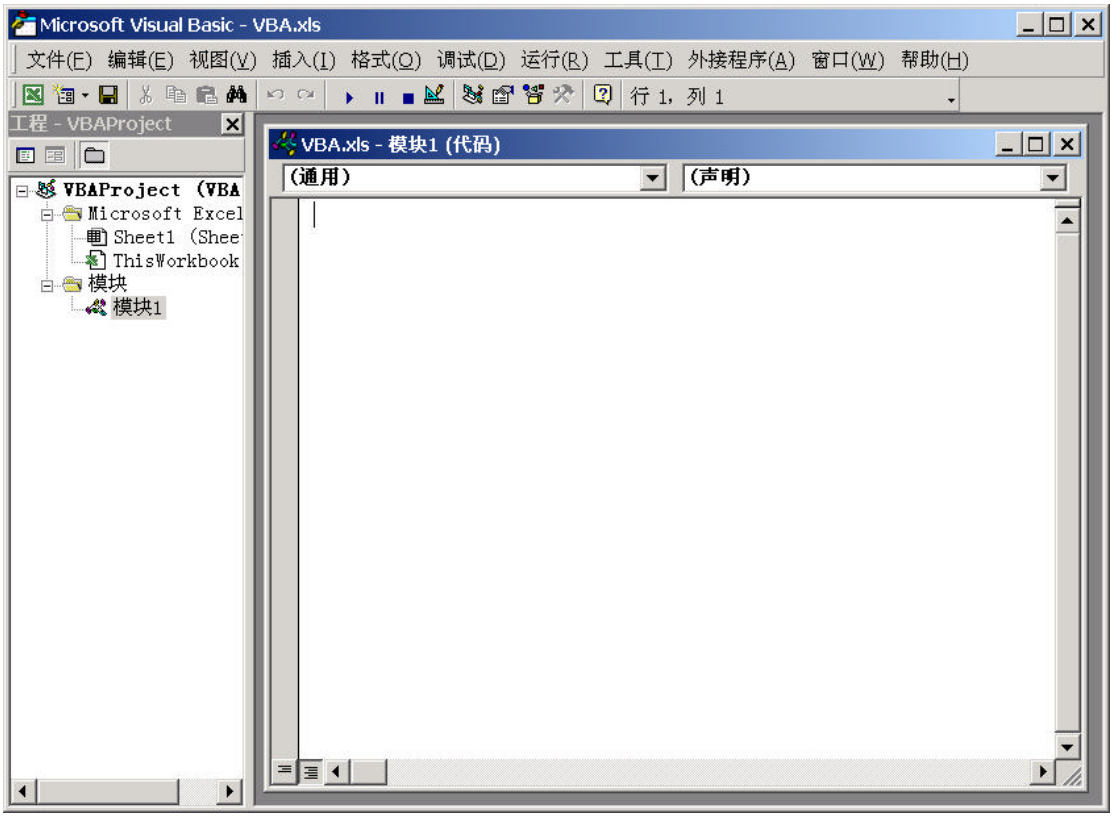


图 4-3 模块的插入

5) **如图4-4所示，请先在代码视窗里记述变量的声明文。所声明的OPC对象的变量名和说明列于表4-1。**

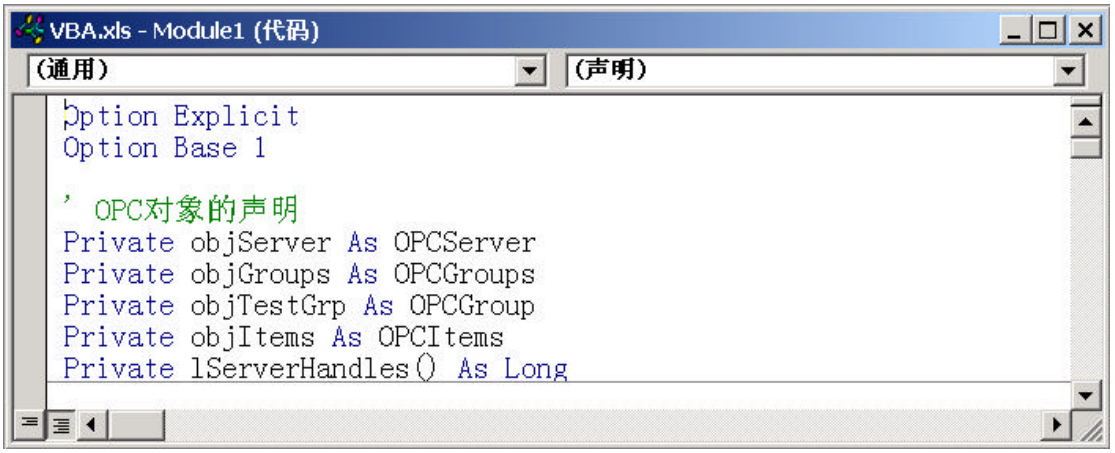


图 4-4 OPC对象的声明

表4-1 OPC对象变量的声明

变量名	说明

objServer	OPCServer对象，用于连接OPC服务器。
objGroups	OPCGroups 对象，用于添加OPC组的OPC组集合。
objTestGrp	OPCGroup对象，演示用的OPC组。
objItems	OPCItems对象，用于添加OPC标签的OPC标签集合。
lServerHandles()	长整型的数组，用于保存OPC标签的服务器句柄。

6) 从Visual Basic编辑器的菜单选择[插入(I)]-[过程(P)...]。在表示的[添加过程]对话框上的[名称(N)]文字框内输入“ OPC_Open ”，再按下[确定]命令按钮（图4-5）。

图 4-5 子程序过程的插入

7) 公共的（Public）OPC_Open子程序的雏形自动地被添加在代码视窗中，。

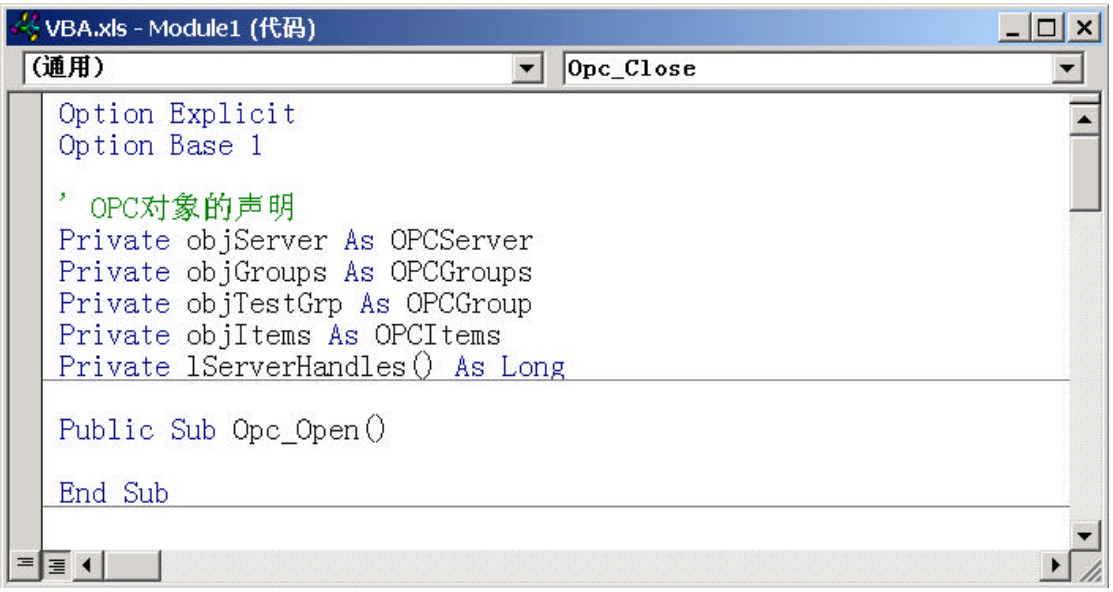


图 4-6 子程序过程的插入结果

8) **在**OPC_Open**过程里记述表**4-2**的代码。**

表*4-2 Opc_Open****子程序***

```
Public Sub Opc_Open()
   Dim strItemIDs(8) As String
   Dim lClientHandles(8) As Long
   Dim lErrors() As Long
   Dim I As Integer

   If objServer Is Nothing Then
      ' 建立一个OPC服务器对象
      Set objServer = New OPCServer
   End If

   If objServer.ServerState = OPCDisconnected Then
      ' 连接OPC服务器
      objServer.Connect ("OPCJ.SampleServer.1")
   End If

   If objGroups Is Nothing Then
      ' 建立一个OPC组集合
      Set objGroups = objServer.OPCGroups
   End If

   If objTestGrp Is Nothing Then
      ' 添加一个OPC组
      Set objTestGrp = objGroups.Add("Test")
   End If

   If objItems Is Nothing Then
      Set objItems = objTestGrp.OPCItems

      With Worksheets("Sheet1")
         For I = 1 To 8
         ' 从工作表中得到TAG1到TAG8的项标识符
            strItemIDs(I) = .Cells(1, I).Text
            lClientHandles(I) = I
         Next I
         .Range("A2:J2").ClearContents
      End With

      ' 添加OPC标签
      Call objItems.AddItems(8, strItemIDs, lClientHandles, _
         lServerHandles, lErrors)
   End If
End Sub
```

9) **请用相同的方法添加**OPC_Close**子程序和**OPC_Read**子程序。代码分别列于表**4-3**和表**4-4**。**

表*4-3 Opc_Close****子程序***

```
Public Sub Opc_Close()
   Dim lErrors() As Long
```

```
    If objServer Is Nothing Then
        Exit Sub
    End If

    If Not objItems Is Nothing Then
        If objItems.Count > 0 Then
            ' 清除OPC标签
            Call objItems.Remove(8, lServerHandles, lErrors)
        End If
        Set objItems = Nothing
    End If

    If Not objTestGrp Is Nothing Then
        ' 清除OPC组
        objGroups.Remove ("Test")
        Set objTestGrp = Nothing
    End If

    If Not objGroups Is Nothing Then
        Set objGroups = Nothing
    End If

    If objServer.ServerState <> OPCDisconnected Then
        ' 断开OPC服务器.
        Call objServer.Disconnect
        Set objServer = Nothing
    End If
End Sub
```

表4-4　Opc_Read子程序

```
Public Sub Opc_Read()
    Dim ItemVal() As Variant
    Dim lErrors() As Long
    Dim I As Integer

    If objServer Is Nothing Then
        Exit Sub
    End If

    If objServer.ServerState = OPCRunning Then
        ' 同步读取
        Call objTestGrp.SyncRead(OPCCache, 8, lServerHandles, _
            ItemVal, lErrors)

        With Worksheets("Sheet1")
            For I = 1 To 8
                ' 在工作表上表示数据字符串
                .Cells(2, I).Value = ItemVal(I)
            Next I
        End With
    End If
```

```
End Sub
```

10) **从**Visual Basic**编辑器的菜单选择**[**调试**(D)]-[**编译**VBAProject(L)]**，可以检查代码内是否有语法上的错误。**

3.1.2 编辑Excel工作表

（一）OPC**标签标识符的定义**

1) **请在**Excel**工作表的从**A1**开始到**H1**的单元格内,写入**OPC_Open**子程序所要访问的８个**OPC**标签标识符（图**4-7**）。**

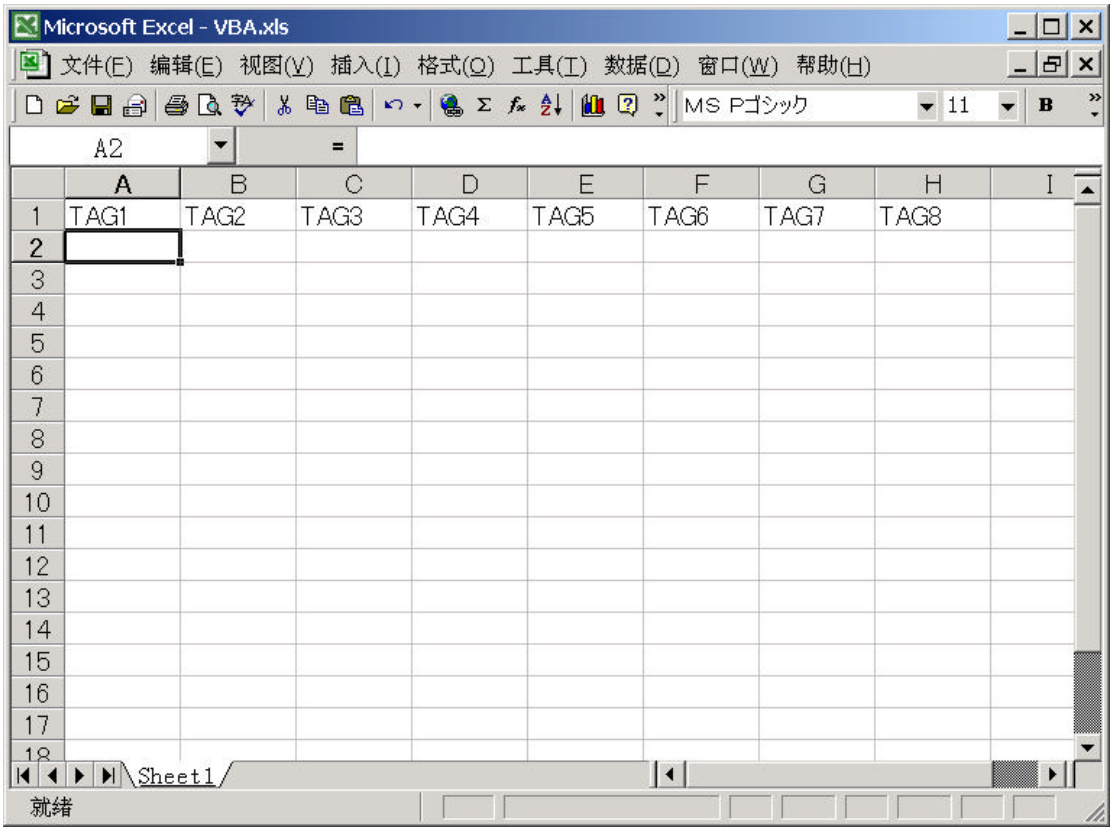


图 4-7 项名的设置

（二）图表的定义

1) **请从**Excel**的菜单选择**[**插入**(I)]-[**图表**(H)…]。**在表示的**[**图表向导－４步骤之１－图表类型**(C)]**内，选择**[**饼图**]（图 4-8），再按下**[**下一步 >**]**命令按钮。**

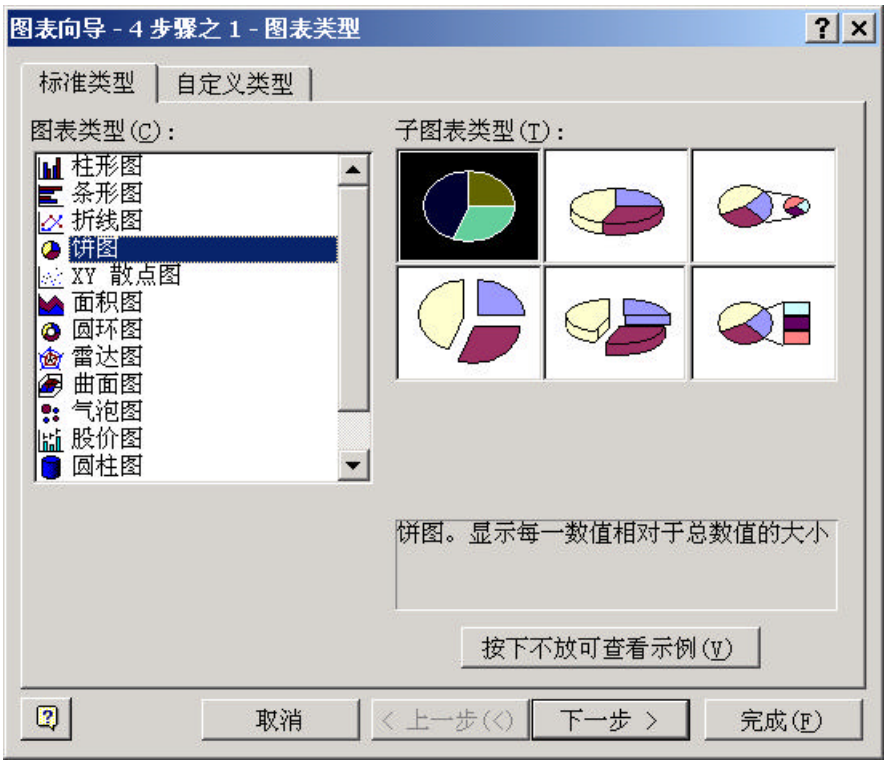


图 4-8 图表向导－4 步骤之 1 －图表类型

2) **在下一步显示的[图表源数据]对话框里，用鼠标指定[Sheet1]工作表的A1到H2的单元格，设置[数据区域(D)]文字框的内容（图4-9）。再按下[下一步 >]命令按钮。**

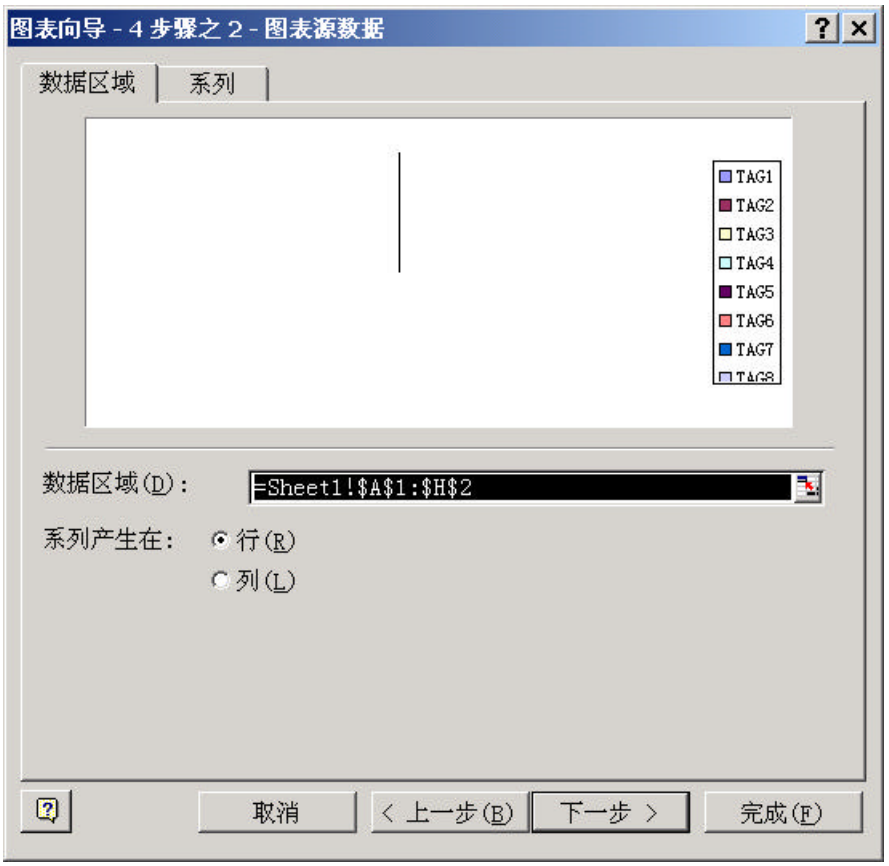


图 4-9 图表向导－4 步骤之 2 －图表源数据

3) **在下一步显示的[图表向导－4 步骤之 3 －图表选项]对话框里，在[图表标题(T)]的文字框内输入“ 比率 ”（图4-10），再按下[下一步 >]命令按钮。**

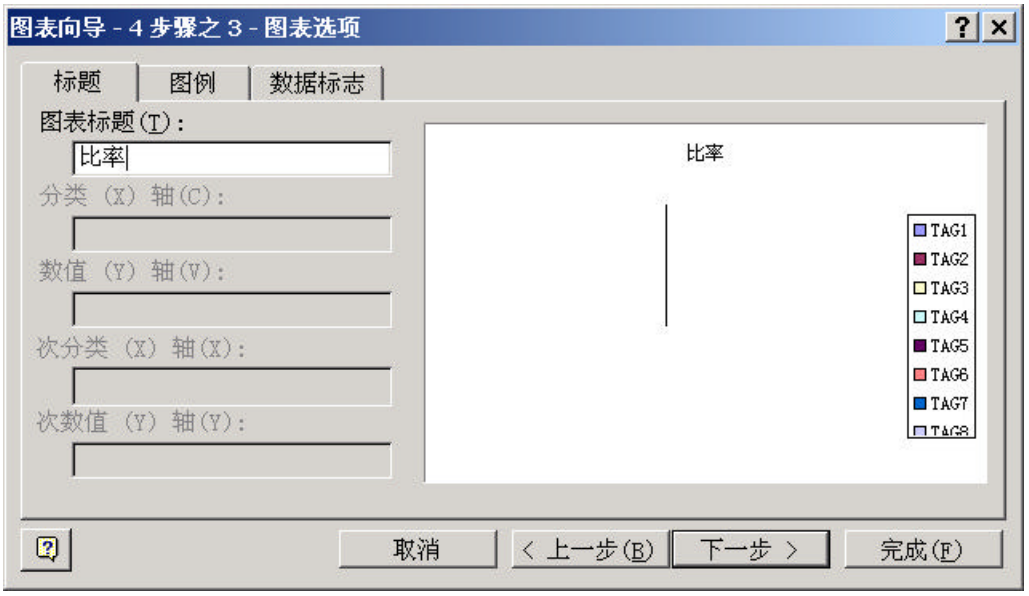


图 4-10 图表向导－4 步骤之 3 －图表选项

4) **在下一步显示的[图表向导－４步骤之４－图表位置]对话框里，请从[作为其中的对象插入(O)]的列表框内选择[Sheet1]（图4-11），最后按下[完成(F)]命令按钮。**

图 4-11　图表向导－４步骤之４－图表位置

（三）命令按钮的生成

1) **从Excel的菜单选择[视图(V)]-[工具栏(T)]-[控件工具箱]，则可以显示控件工具箱（图4-12）。**

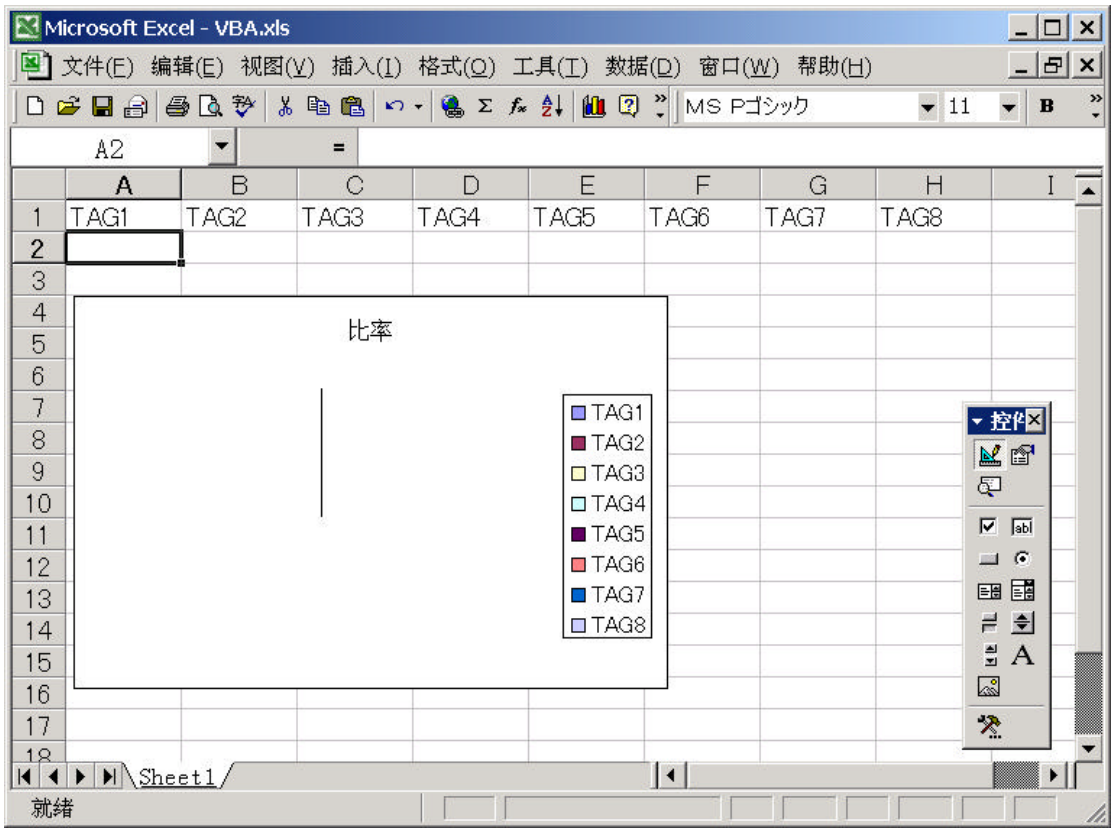


图 4-12　控件工具箱的表示

2) **请选择控件工具箱内的[命令按钮]控件，并配置在工作表上（图4-13）。**

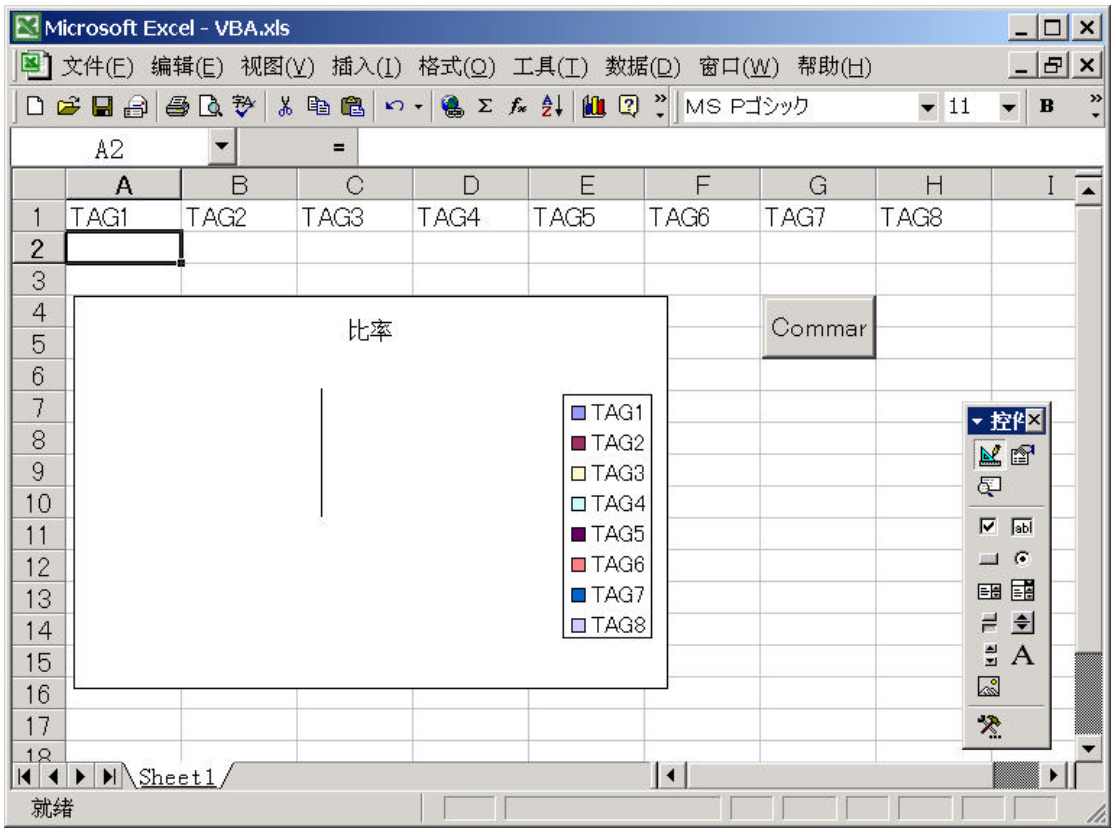


图 4-13 命令按钮的配置

3) **请用鼠标右按钮选择在**2)**的步骤配置的命令按钮，在显示的弹出式菜单内选择[属性(P)]项目，显示[属性]对话框。把**[Caption]**属性值改变为“连接”（图** 4-14）**。**

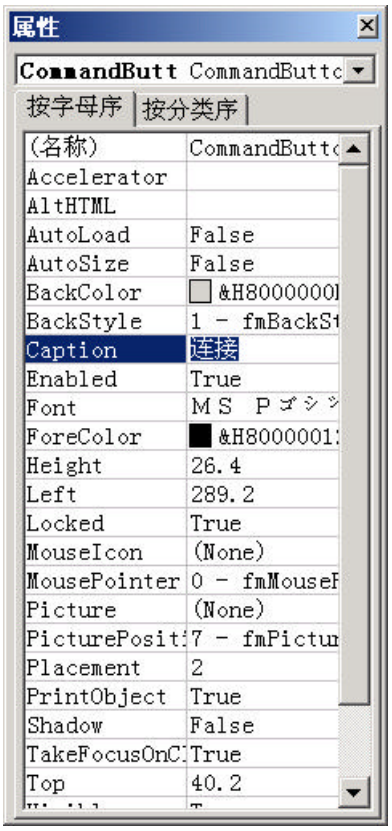


图 4-14命令按钮的属性

4) **请用鼠标右按钮选择在**2)**的步骤配置的命令按钮，在显示的弹出式菜单内选择[查看代码]项目。**Visual Basic**编辑器将被自动启动，随之单击命令按钮时过程**CommandButton1_Click**的雏形将自动生成。请在这个过程雏形内记述**OPC_Open**的过程处理代码（图**4-15**）。**

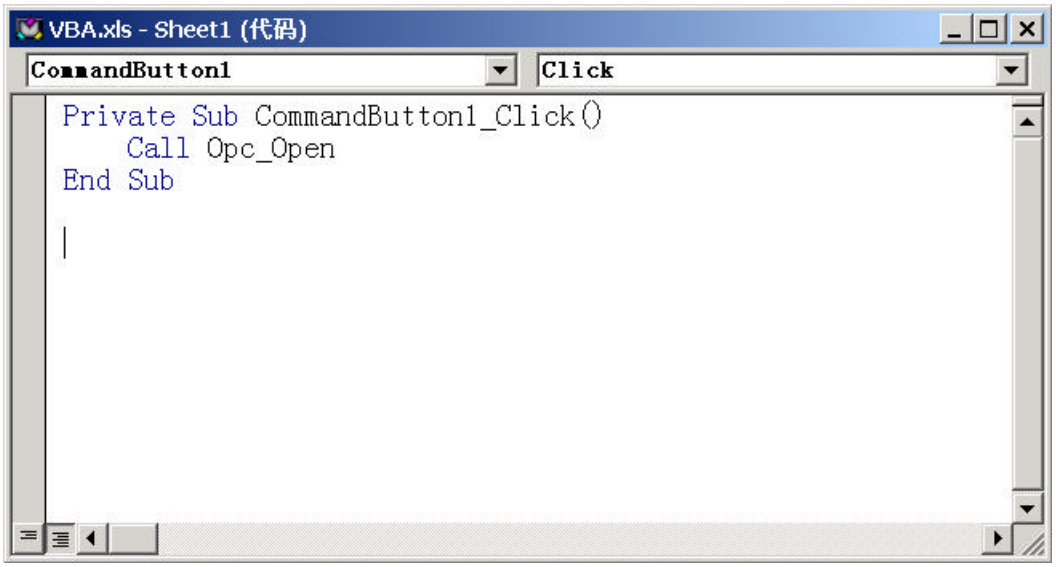


图 4-15 连接的调用子程序过程

5) **用上述相同的方法，在工作表上配置[断开]命令按钮和[读取]命令按钮，并分别记述**OPC_Close**过程处理代码和**OPC_Read**过程处理代码（图**4-16**，图** 4-17**）。**

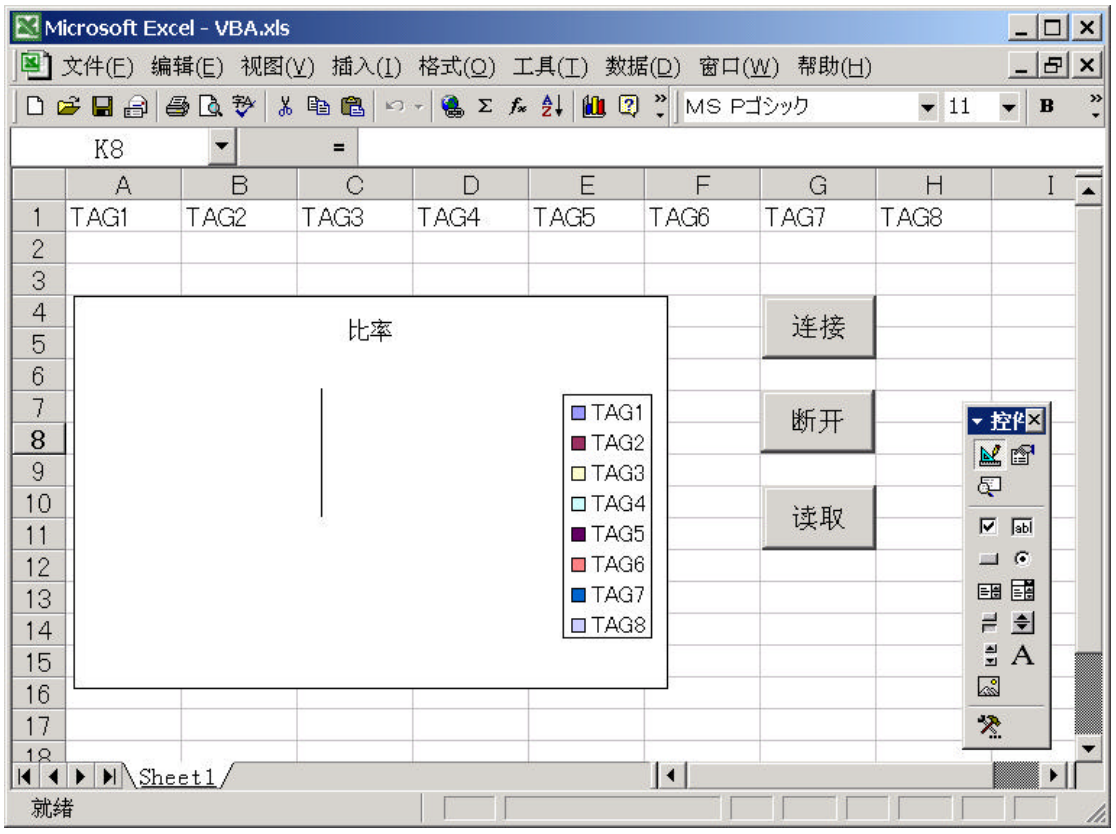


图 4-16 断开和读取命令按钮的配置

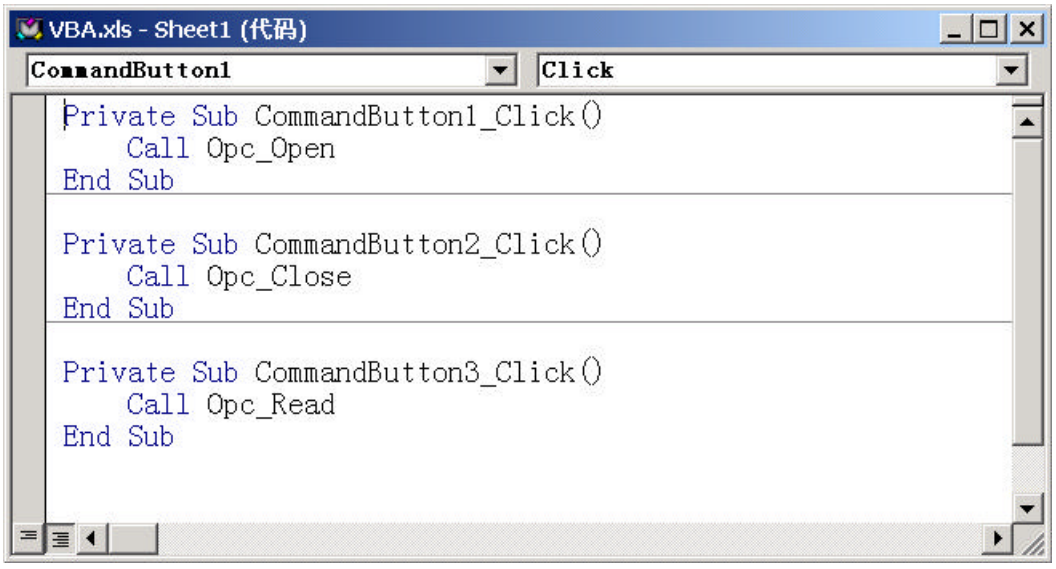


图 4-17 断开和读取的调用子程序过程

6) **释放控件工具箱左上角的图标按钮，退出设计模式。（图4-18）。**

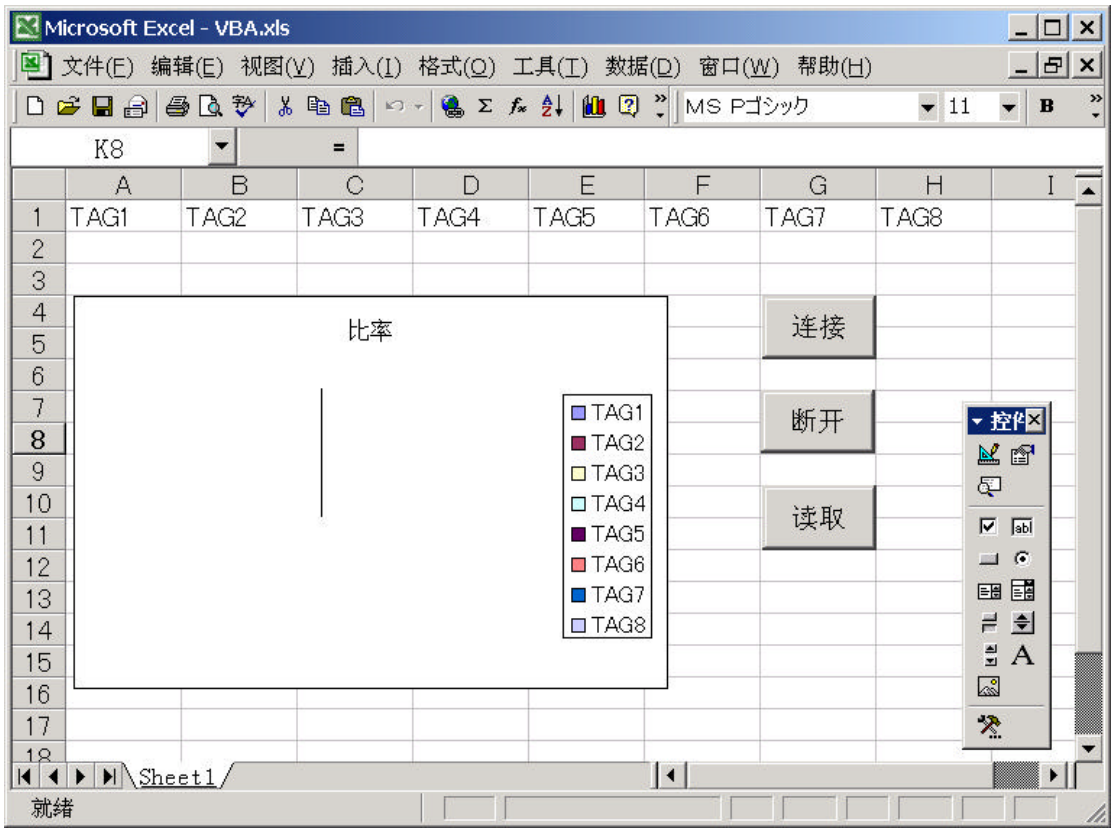


图 4-18 设计模式的结束状态

3.1.3 试运行

1) **按下[连接]命令按钮，OPC服务器被启动，同时单元格A2到H2的数值被清除。**
2) **按下[读取]命令按钮，从OPC服务器读取的各个OPC标签的数值表示在单元格A2到H2，同时以这些单元格的数据为数据源的饼图也被更新（图4-19）。**

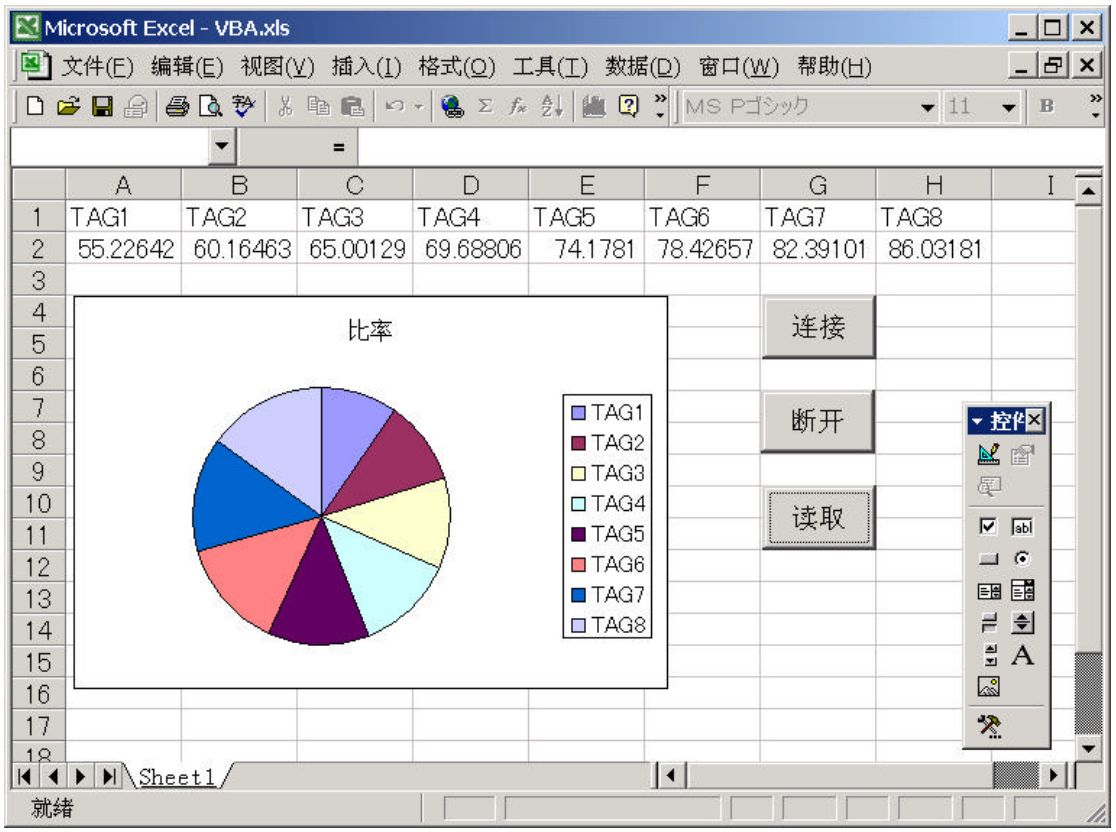


图 4-19 运行画面

3) **按下[断开]命令按钮，OPC服务器被关机。**

3.2 使用ActiveX控件的OPC应用程序

ActiveX控件的特点之一是不单是可以被Visual Basic调用也可以被其他容器程序所使用。例如也可以在微软的Internet Explorer上表示和使用ActiveX控件。

3.2.1 在Excel中使用ActiveX控件

下面让我们在微软的Excel上使用ActiveX控件，试着制作一个简单的测量控制程序。使用本节介绍的方法，你就可以在Excel的工作表上开发象图 4-20表示的那样的数据收集和显示演示文件。这个演示文件可以在本书的示范程序的文件夹中找到，路径是“ \Samples\Chap4\Demo.xls ”。

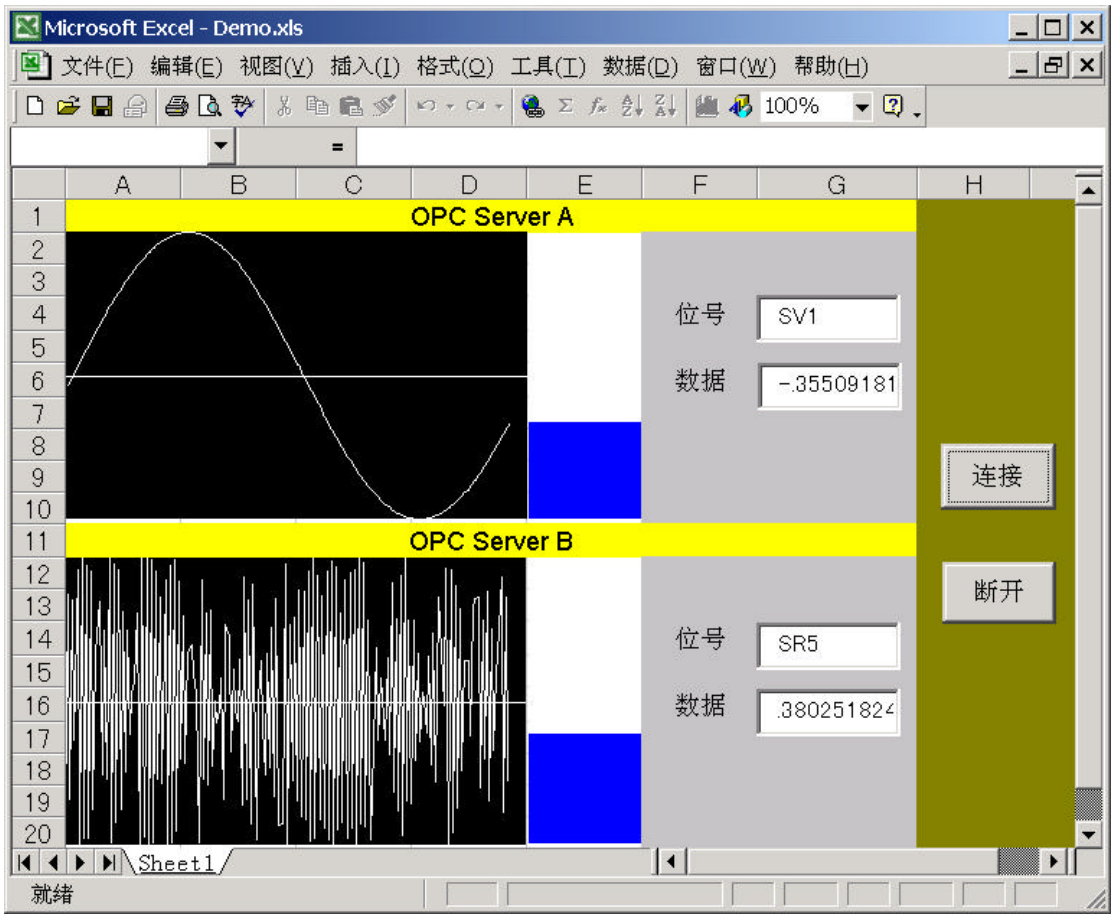


图 4-20 数据收集和显示演示文件

首先从Excel**菜单选择[视图(V)]-[工具栏(T)]-[控件工具箱]（图** 4-21）。

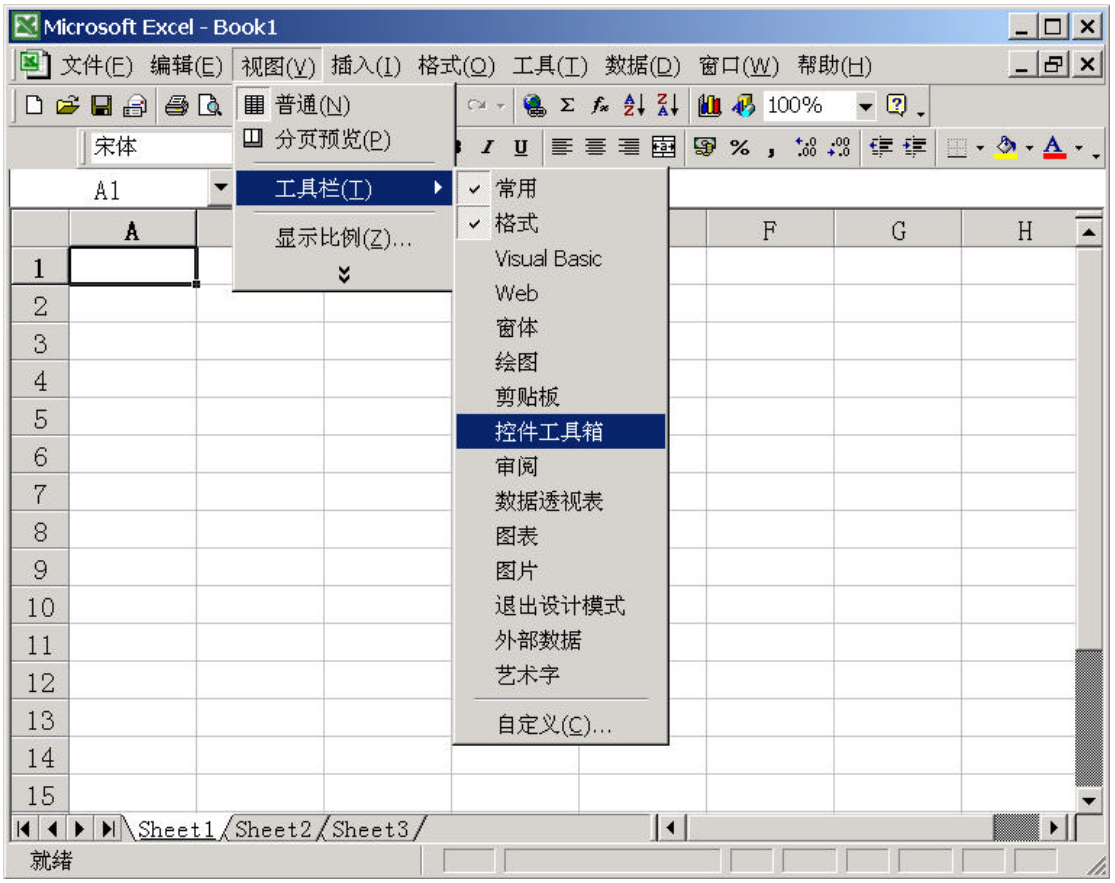


图 4-21 控件工具箱的表示

在表示的[控件工具箱]里选择[其他控件]命令按钮。如图 4-22所示，当按下命令按钮后会显示可使用的控件的一览。请从中选择名为[OPCBar.BarMeter]的OPC棒图ActiveX控件。

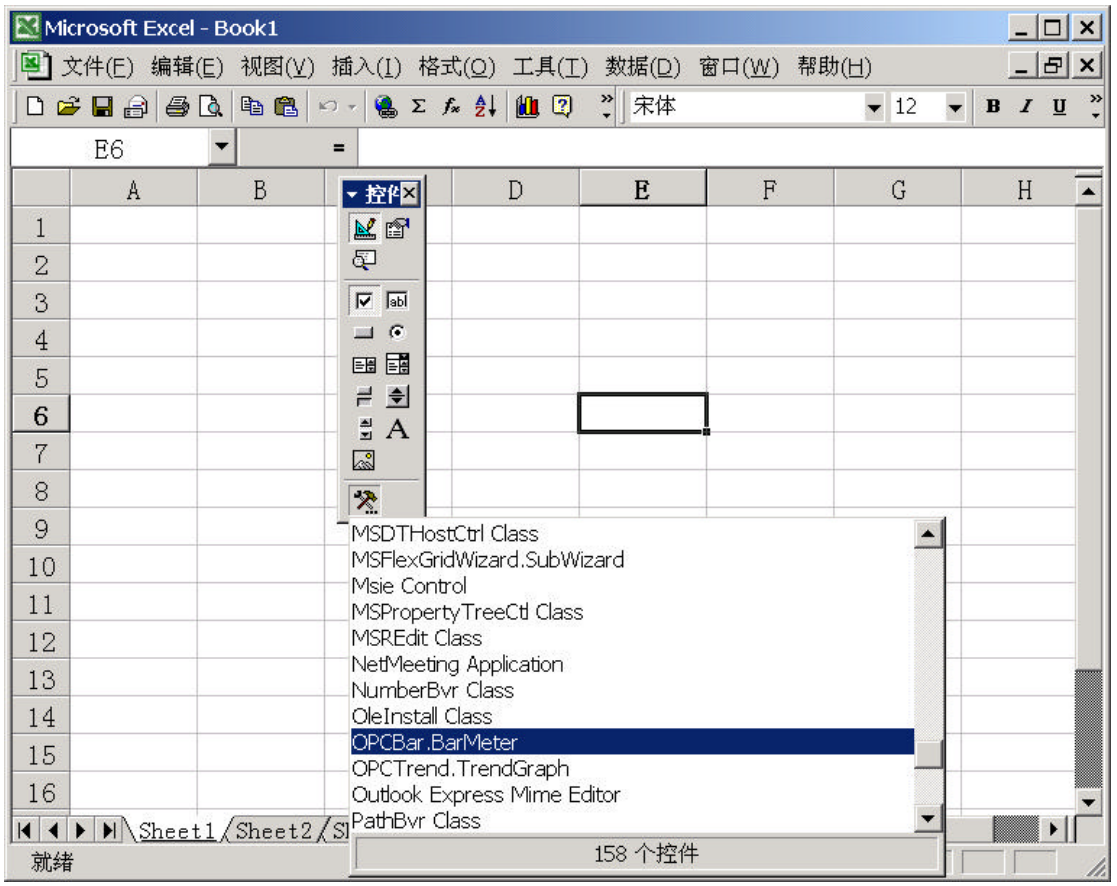


图 4-22 控件清单的表示

注意，因为这里所表示的一览是已存在的并且已注册的 ActiveX**控件，所以在进入本节之前，请参照第 6 章的［**5.4.1 **复制和注册示范源程序］，将生成的**ActiveX**控件复制到使用的计算机上并加以注册。**

于是在工作表上已经配置了一个OPC**棒图**ActiveX**控件。与在**Visual Bacic **的窗体上配置控件一样，需要象图 **4-23**那样在工作表上指定**ActiveX**控件的配置领域。**

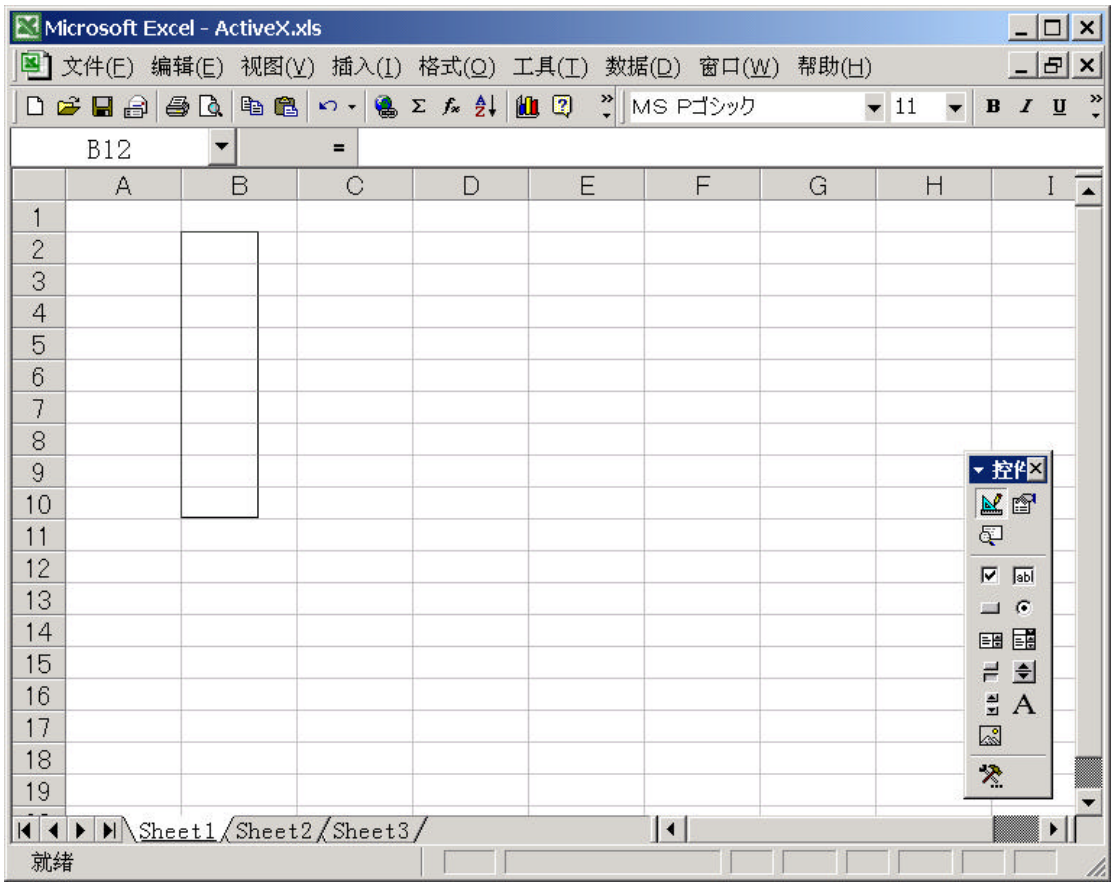


图 4-23 控件配置领域的指定

这时显示的[控件工具箱]左上角的[退出设计模式]按钮被按下着，即表示正处于设计模式状态。如果进行试运行的话，需要退出设计模式。释开左上角的[退出设计模式]按钮，刚才配置的ActiveX控件就会自动进入运行状态。

但是因为这时的OPC棒图控件没有连接任何OPC服务器，所以显示不会有任何变化（图4-24）。

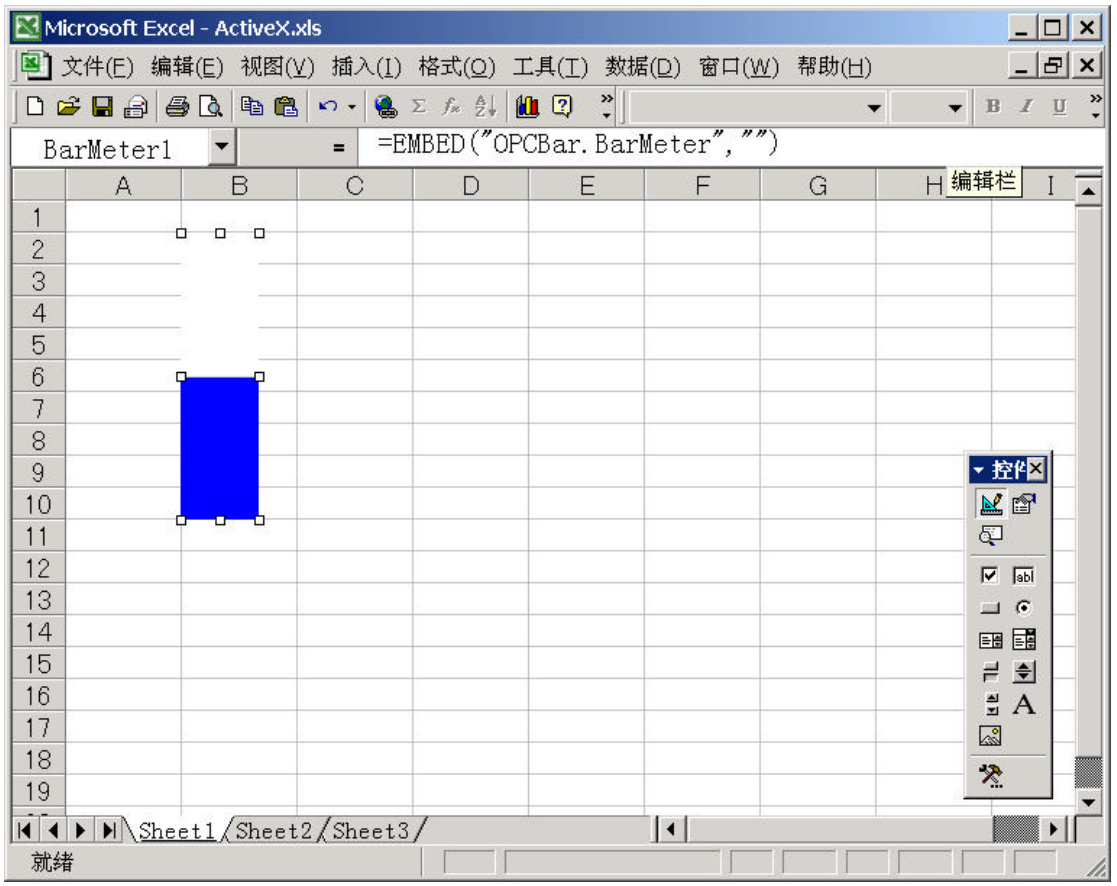


图 4-24 ActiveX 控件的表示

于是让我们再配置一个可以使OPC ActiveX**控件与**OPC**服务器连接的命令按钮。首先让我们检查一下刚才在工作表上配置的**OPC**棒图的属性。检查控件的属性时，需要进入设计模式，然后选择配置的棒图，再按下鼠标右命令按钮并选择**[属性(P)]**的项目（图4-25）。

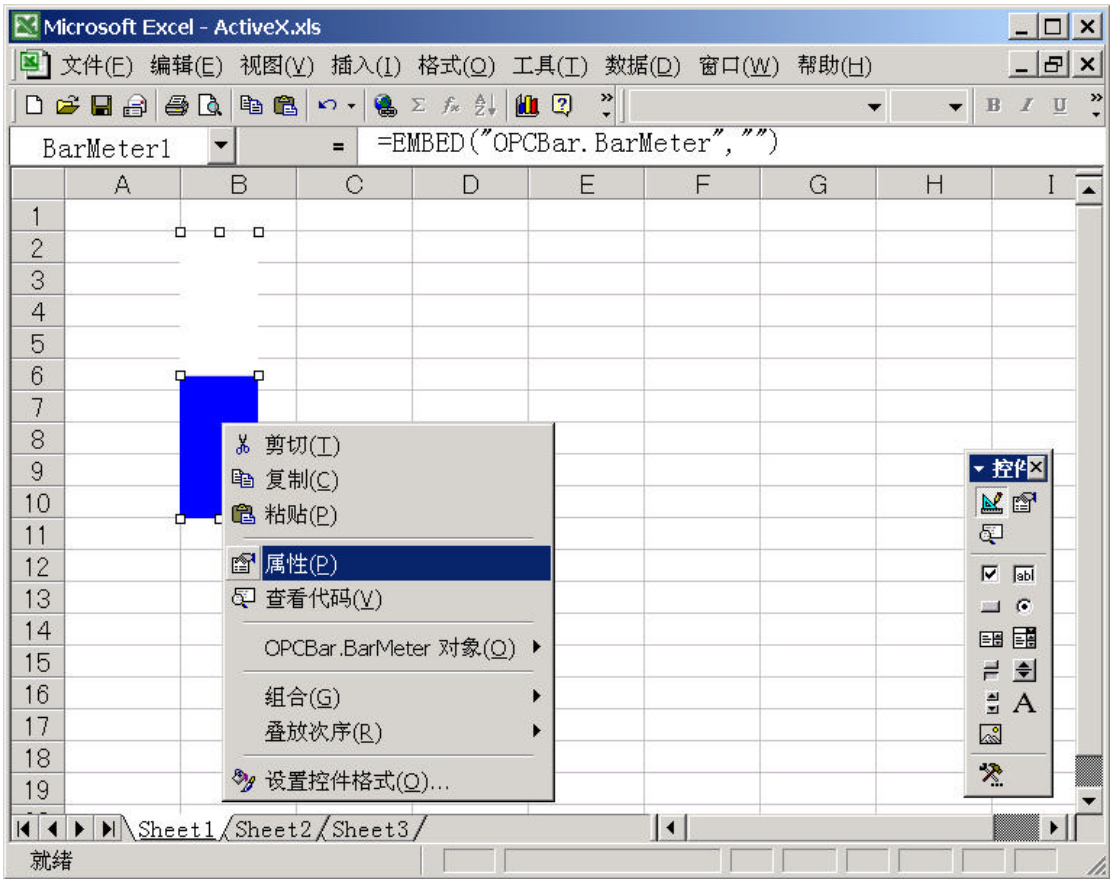


图 4-25 属性菜单的选择

在属性对话框里，检查一下[(名称)]是不是为“ BarMeter1 **”，[ItemID]属性是不是为“** SV1 **”。这两个属性分别表示配置的控件对象的名称为“** BarMerter1 **”，而从**OPC**服务器要读取项标识符为“** SV1 **”的数据（图**4-26**）。**

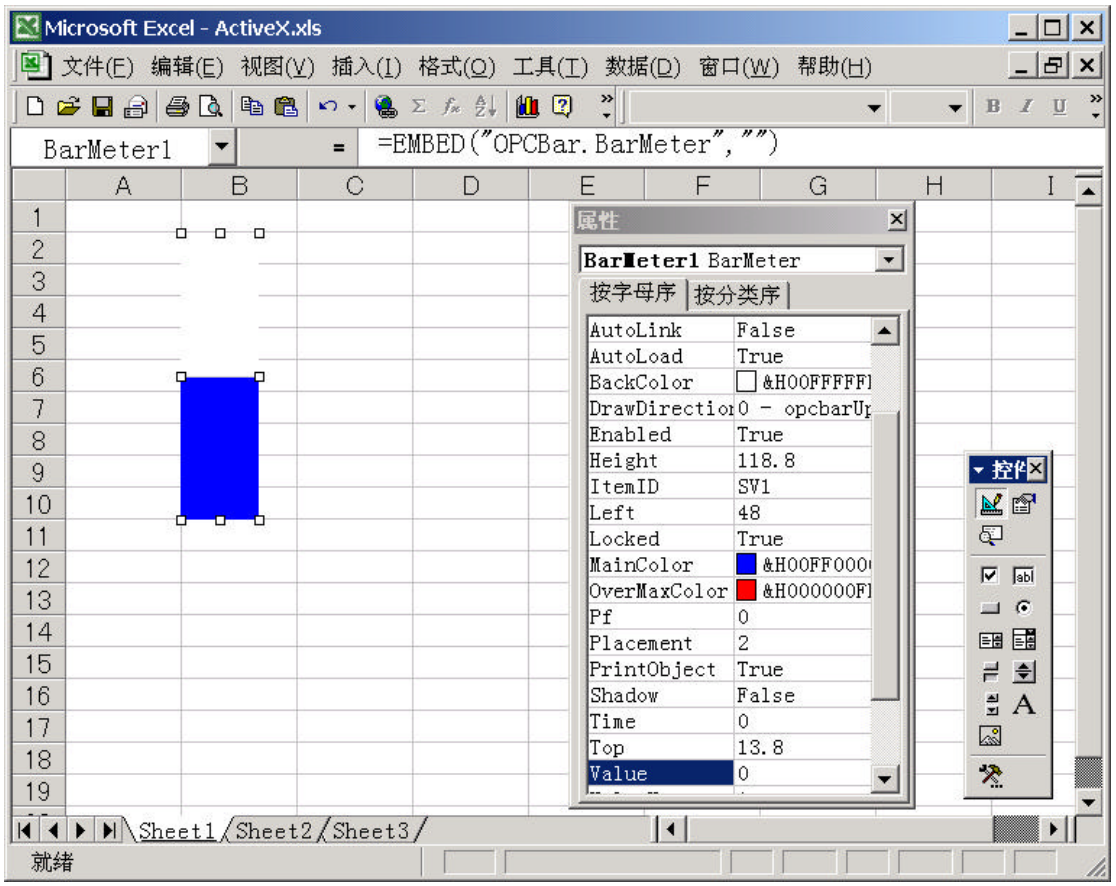


图 4-26 属性的表示

用刚才同样的办法，在控件工具箱里选择[命令按钮]，并在工作表上配置一个命令按钮（图4-27）。

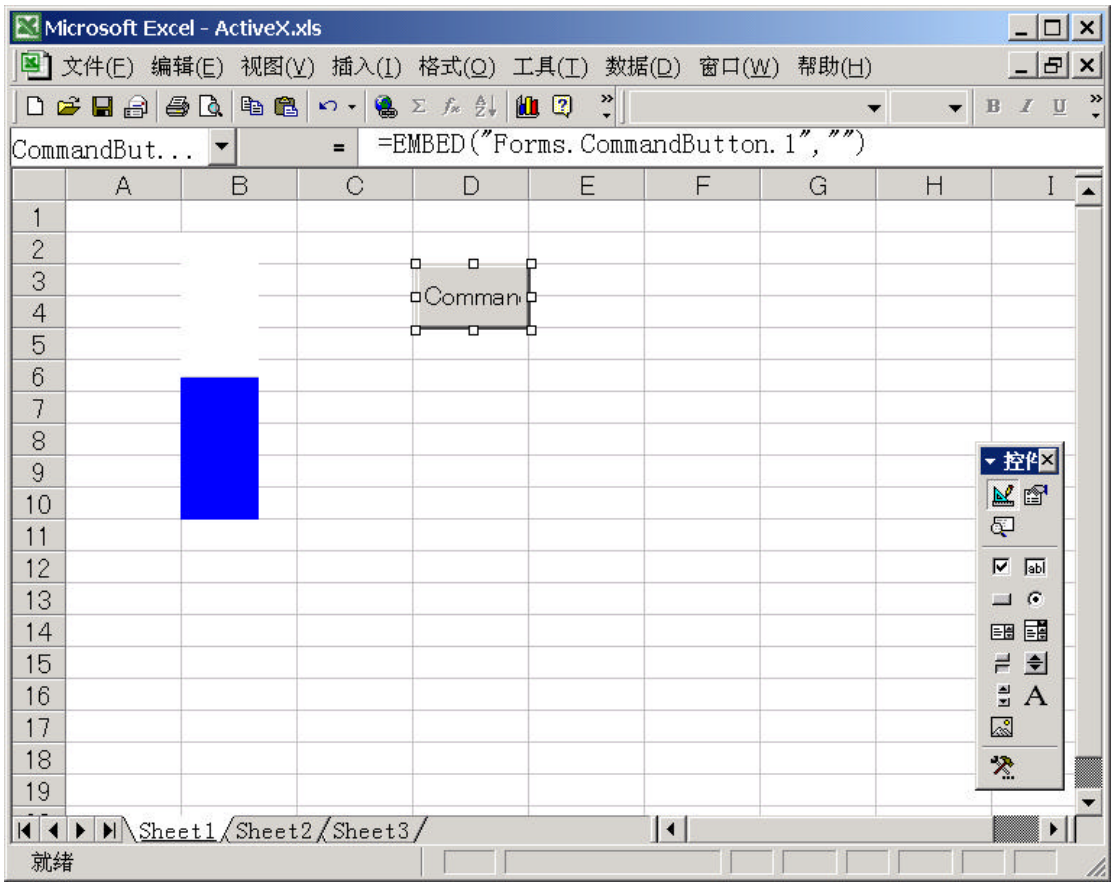


图 4-27 命令按钮的添加

配置后为了表明是一个用于OPC**服务器连接的命令按钮，在控件属性框内将**[Caption]**属性改变为“连接”（图**4-28**）。**

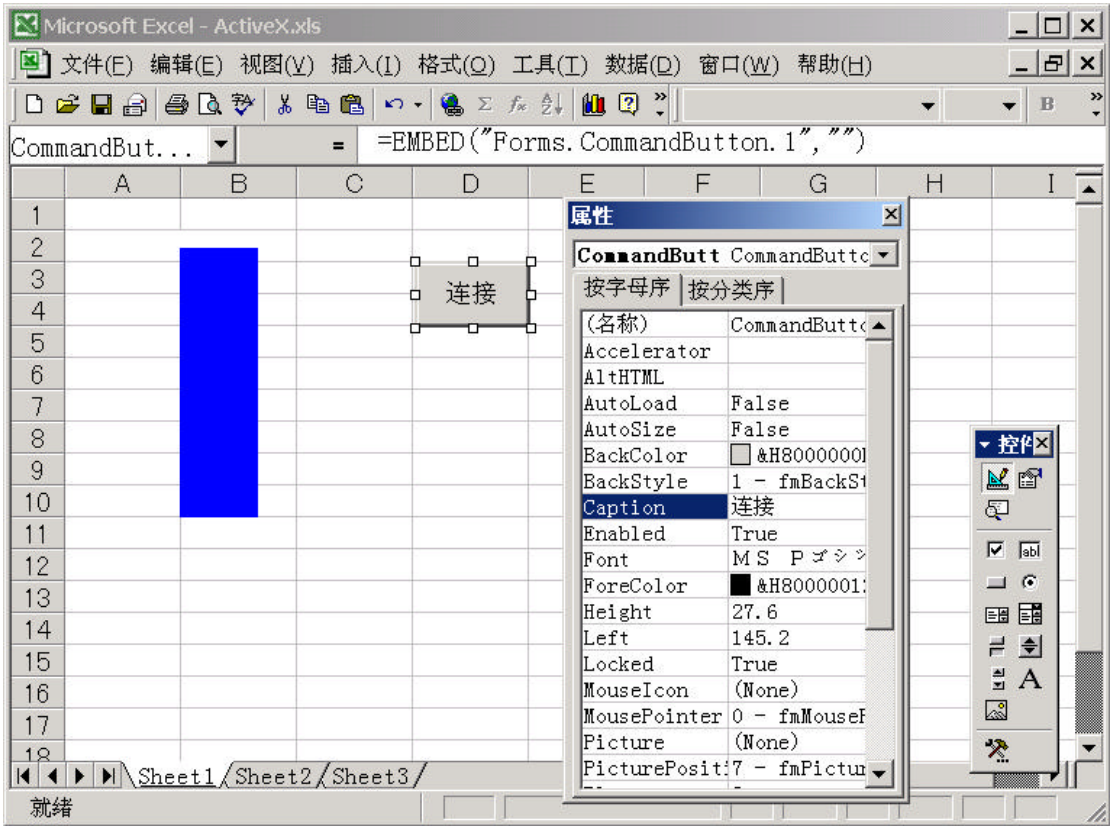


图 4-28 连接按钮的添加

用同样的方法，再配置一个断开OPC服务器连接的命令按钮，并将[Caption]**定义为“断开”（图**4-29**）。**

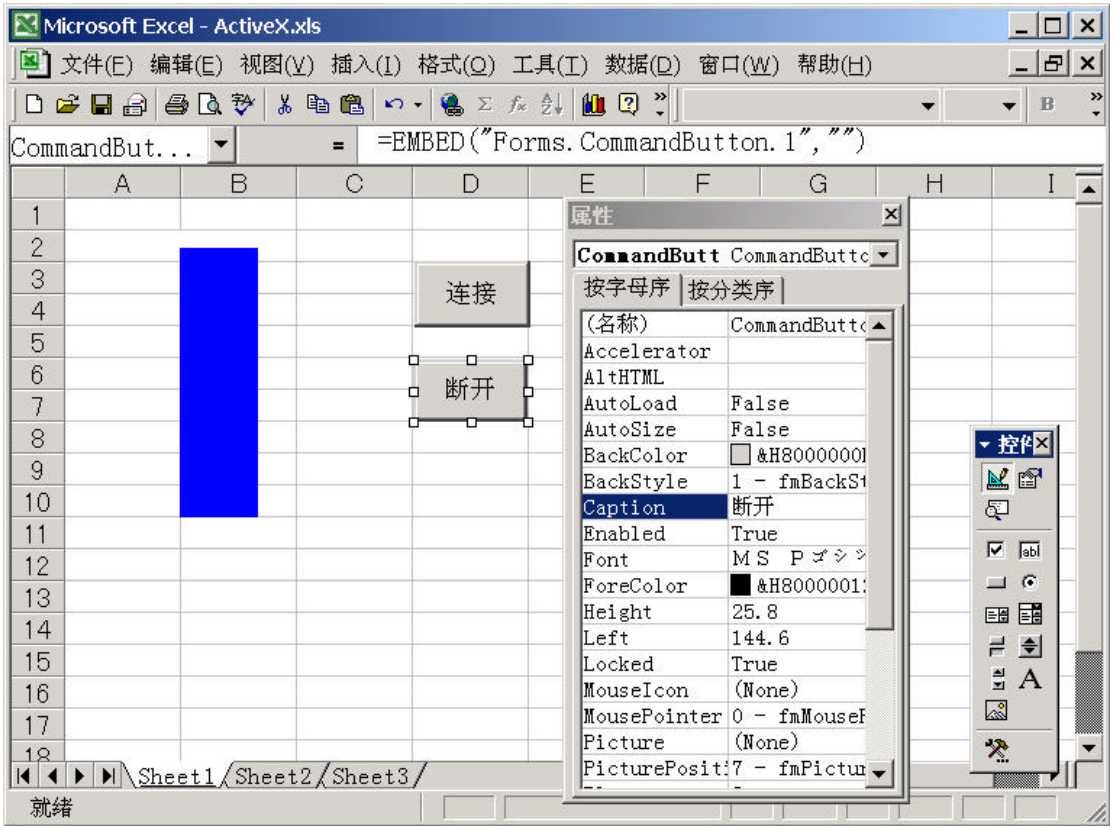


图 4-29 断开按钮的添加

到此为止，一个棒图和用于连接与断开OPC服务器的两个命令按钮已经准备完毕。使用命令按钮将会启动OPC服务器显示棒图或者退出运行。

3.2.2 使用VBA建立OPC服务器数据访问程序

下面让我们使用OPC**棒图**ActiveX**控件的**Connect**和**DisConnect**方法，记述使棒图与**OPC**服务器连接和断开的代码。**

在进入[设计模式]的状态下，双击刚才配置的 CommandButon1**命令按钮，启动**Visual Basic**编辑器。**

这时单击CommandoButton1**按钮时发生时处理程序的雏形会自动被自动生成并表示出来，这个程序的名称为**CommandButton1_Click**（图**4-30**）。**

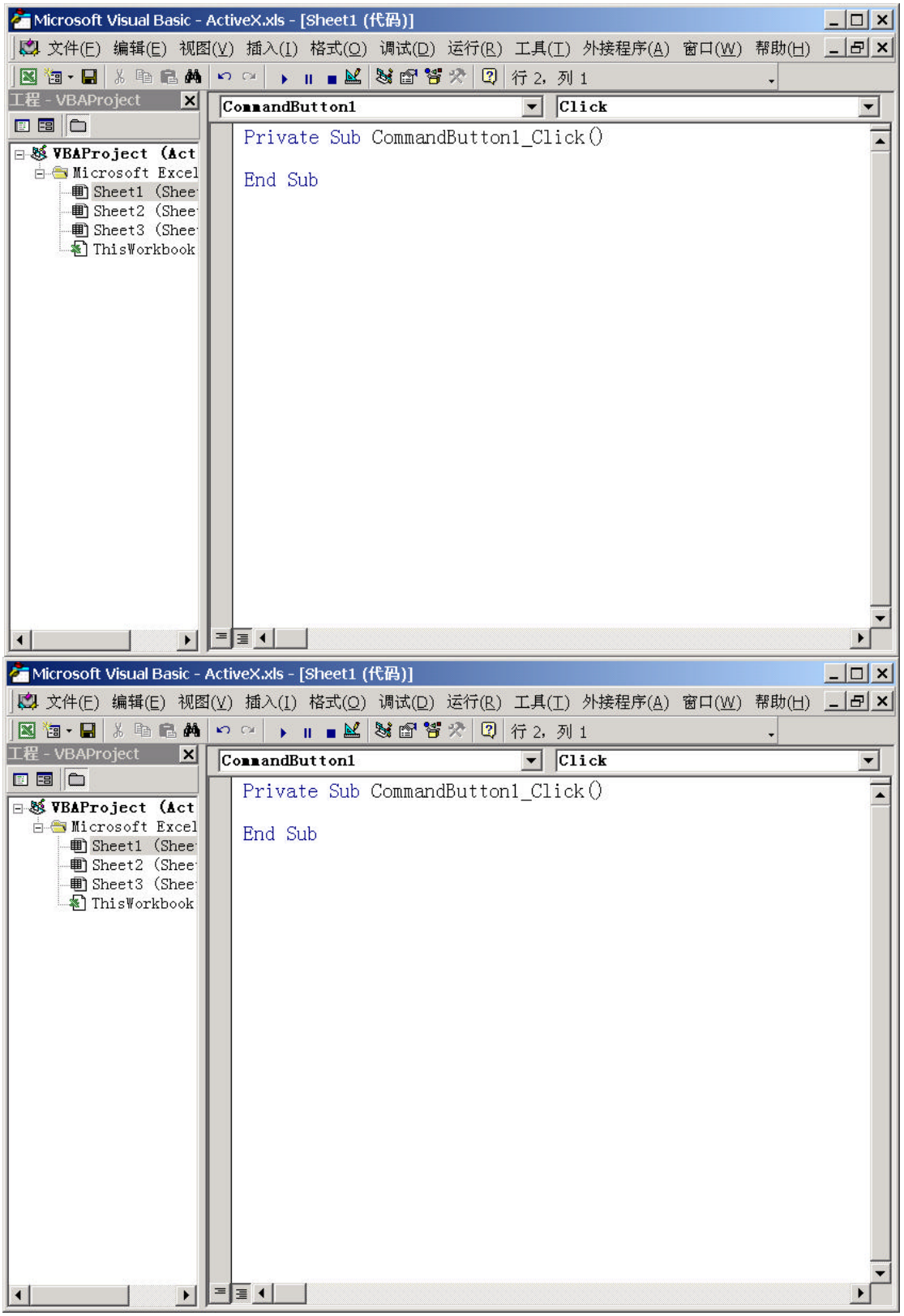


图 4-30 Visual Basic 编辑器

于是作为按钮的单击事件处理，让我们在这里记述使棒图要求与OPC服务器连接的代码。

如图 4-31所示，在CommandButtn1_Click的程序中，当写入要与OPC服务器连接的OPC棒图控件对象名称“ BarMeter1. ”后，OPC棒图控件可用的属性和方法的对话框会自动表示出来。请直接从中选择“ Connect ”作为处理代码（表4-5）。

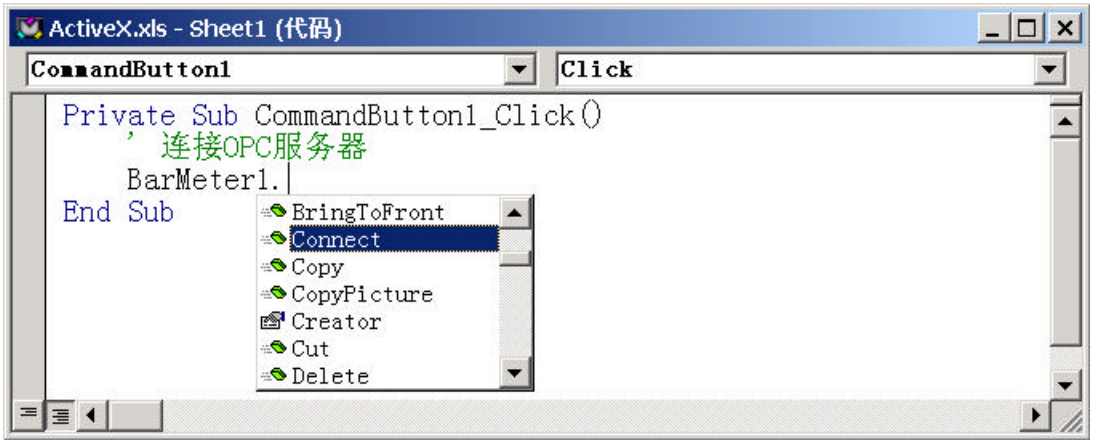


图 4-31 代码的记述

表4-5 命令按钮的处理代码

```
Private Sub CommandButton1_Click()
   ' 连接OPC服务器
   BarMeter1.Connect
End Sub

Private Sub CommandButton2_Click()
   ' 断开OPC服务器.
   BarMeter1.Disconnect
End Sub
```

有了上述的代码，就完成了棒图控件与OPC服务器连接的要求处理。

用同样的方法，在CommandButton2的按键处理程序CommandButton2_Click中，写入断开与OPC服务器连接的处理代码。

到此为止，按下[连接]命令按钮就使棒图控件连接OPC服务器并自动地进行数据采集，按下[断开]命令按钮就断开与OPC服务器的工作表已经作成了。

下面让我们实验一下这个作成的程序。关闭Visual Basic编辑器，或者单击工具栏上的Excel图标，就可以返回到Excel工作表。

释开控件工具栏的[设计模式]图标，退出设计模式，并同时启动配置的工作表上的控件对象。

按下[连接]命令按钮，从棒图控件发出与OPC服务器连接的要求，使OPC服务器被启动，并开始定期采集以标识符为“ SV1 ”的OPC标签的数值，并用棒图显示采集到的数据。按下[断开]命令按钮，断开与OPC服务器的连接并停止数据采集（图4-32）。

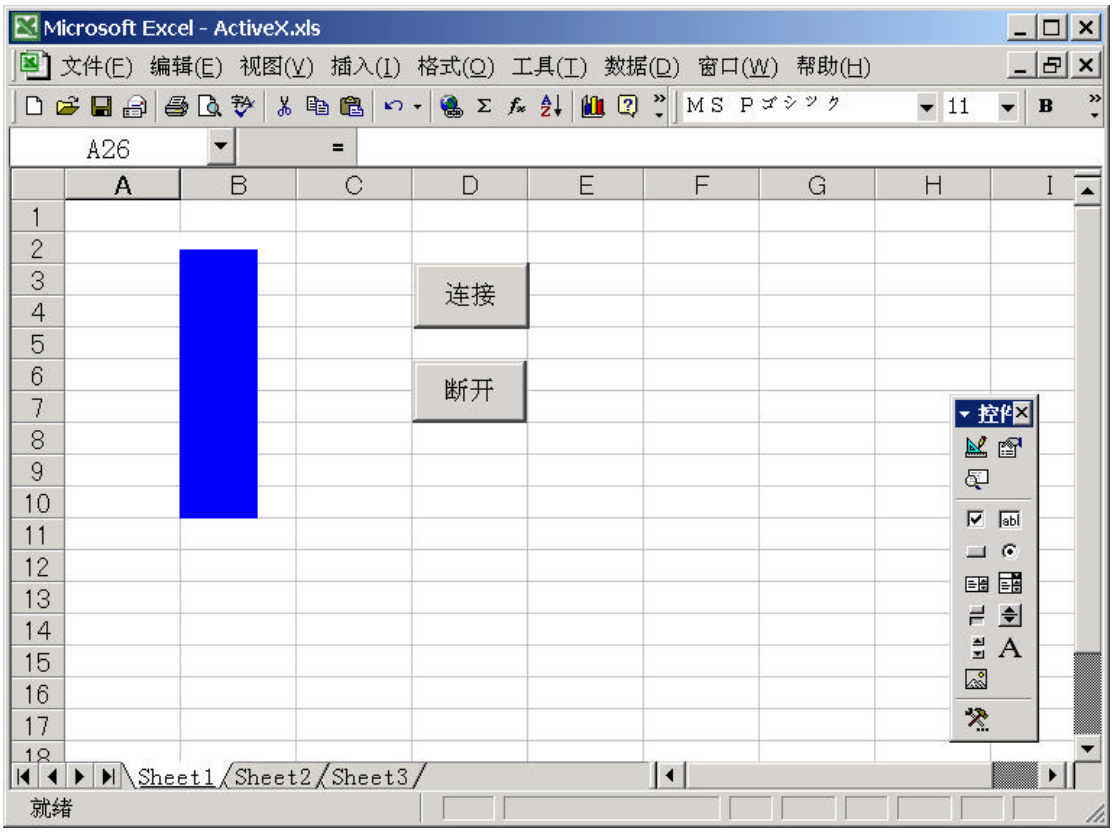


图 4-32 运行画面之一

最后，让我们设法在Excel工作表上表示棒图控件从OPC服务器采集到的“ SV1 ”数值。这个演示用的OPC BarMeter控件提供了将被采集数据的变化自动通知Excel等应用程序的事件。先进入[设计模式]，双击BarMeter1棒图控件，名为“ BarMeter1_DataChange2 ”的事件处理程序的雏形会自动表示出来。

这个DataChange2事件，是在棒图表示的数值发生变化时由OPC服务器触发的事件，应用程序可以在事件处理程序中接受OPC服务器的通知。我们可以在这个事件的处理程序中将“ SV1 ”的最新数值表示在Excel的工作表的单元格内。在单元格内数值表示可以使用如表4-6所示的代码实现，即将“ BarMeter1 ”的Value代入 Cells(行号，列号).Value中。这里是将棒图的数值表示在第12行，第2列的单元格（图4-33）。

表4-6 DataChange2事件的处理代码

```
Private Sub BarMeter1_DataChange2()
    ' 在工作表上表示数据字符串
    Cells(12, 2).Value = BarMeter1.Value
End Sub
```

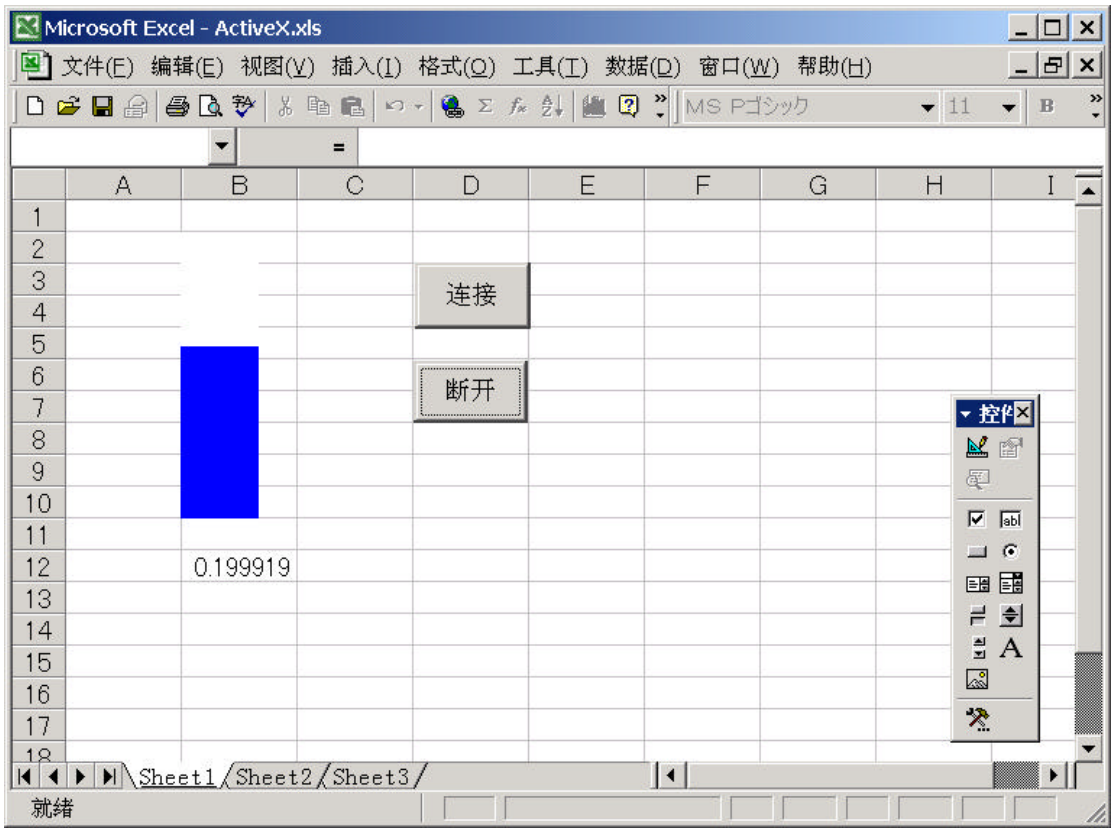


图 4-33 运行画面之二

本节说明了使用本书提供的OPC ActiveX控件，在Excel工作表上制作OPC应用程序的基本方法，当然也可以用同样的方法开发象图 4-20那样的比较复杂的OPC演示应用程序。

4 运行环境的设置

本章的内容是介绍怎样设置从第２章到第４章开发的VB或者VBA应用程序的计算机运行环境。主要是就OPC服务器和OPC客户应用程序分别在不同的计算机上运行的远程连接形式，说明OPC运行环境的设置方法。OPC服务器和OPC客户应用程序在相同的计算机上运行的本地连接形式形態，几乎使用分布式COM的默认设置就可以运行，或者参考下面介绍的远程连接的设置方法，将OPC服务器的设置和OPC客户应用程序的设置在同一台计算机上进行也可以解决问题。

4.1 远程连接所需的软件

当OPC服务器用作为远程服务器使用时，请参考表 5-1，随使用的操作系统不同可能需要附加必须的软件。

表 5-1 远程连接所必须的软件

操作系统	必须的软件
Windows 2000	无
Windows NT 4.0	Service Pack 3或更高的版本

4.2 添加一个OPC专用用户

与OPC服务器连接时，需要由OPC服务器的计算机对OPC客户应用程序的计算机进行身份验证。如果双方的计算机同时隶属于计算机域时，因为身份验证可以由域控制器进行，可以不需要OPC的专用用户。在这种情况下，你可以越过本节，直接去读下一节。

但是现状是运转在自动控制系统的控制计算机大多数并没有隶属于计算机域，而是运转在工作组中。如果没有OPC专用用户，因为OPC服务器计算机没有方法识别客户应用程序计算机的身份，所以OPC服务器也可能无法正常运转。

所以在以下两种情况下，OPC的专用用户将是必须的。

一种情况是OPC服务器的计算机或OPC应用程序计算机不隶属于计算机域，但是自动控制系统需要安全性机制。这时的OPC客户应用程序计算机的注册用户应该使用我们在这里介绍的OPC专用用户。也就是说，以OPC专用用户的名义，注册OPC客户应用程序计算机后，然后连接和访问OPC服务器。

另一种情况是作为OPC服务器的身份标识的指定用户。也就是说，在5.4.2的[分布式COM安全机制的设置]里使用的用户。特别是希望OPC服务器计算机不需任何用户注册就可以在背后运转的时候，这种OPC服务器的身份标识用户更是必不可少的。

如果OPC应用程序的计算机运转在工作组，但是控制系统并不要求安全性机制，而且OPC服务器计算机又总是运转在被用户注册的状态，OPC专用用户则是可有可无的。你也可以越过本节，直接去读下一节。

以上两种目的用户，可以使用同一OPC专用用户，也可以设置分别的OPC专用用户。

按照Windows的安全性机制规范，对于不隶属于计算机域的另外的计算机上用户，一般无法进行身份验证。但是唯一的例外是，如果在二台计算机上存在用户名相同并且密码也相同的用户，可以相互进行身份验证。所以为了可以进行这样的相互身份验证，必须事先在工作组运转的OPC服务器计算机和OPC客户应用程序计算机上，添加OPC专用用户。比如，分别在OPC服务器计算机上和OPC客户应用程序计算机上添加用户名为“ opc_user ”而密码为“ password ”的用户。

4.3 推荐的分布式COM安全机制的设置

这里说明对于OPC服务器计算机和OPC客户应用程序计算机的分布式COM的推荐设置方法。请读者可以以此为基础，再进一步参考OPC服务器供应商提供的技术文件，决定必要的设置方法。

4.3.1 没有分布式COM安全机制的设置

多数的自动控制系统的计算机运转在独立的计算机网络环境，也就是物理上于外界的互联网并不相互连接着的网络环境。这样的自动控制系统，对于安全性机制的要求并不是必不可少的。但是问题是由于引入了分布式COM的安全性机制却没有进行适当的设置，往往会在现场遇到不能正常连接的故障。因此，本节介绍使分布式COM的安全性机制无效的设置方法（表 5-2）。对于要求安全性机制的系统，请参考下一节5.3.2的设置方法。

表 5-2 不要求安全性机制时的分布式COM的推荐设置值

计算机	分布式COM设置	推荐值
OPC 服务器 计算机	[身份认证级别]	[无]
	[启动权限]	[Everyone]
	[访问权限]	[Everyone]
	[身份标志]	[交互式用户] 但是，需要不注册OPC服务器计算机也可以运行的场合，请选用[指定用户]。
OPC 客户 计算机	[默认身份认证级别]	[无]

在不隶属于任何计算机域只属于工作组的自动控制系统里，在[身份标志]指定为[指定用户]时，请参考5.2节，事先添加一个OPC专用用户，并将此用户用作[指定用户]。

4.3.2 具有分布式COM安全机制的设置

本节介绍，怎样设置要求分布式COM的安全性机制的OPC服务器计算机和OPC客户计算机（表 5-3）。

表 5-3 要求安全性机制时的分布式COM的推荐设置值

计算机	分布式COM设置	推荐值
OPC 服务器 计算机	[身份认证级别]	[连接]
	[启动权限]	特定的用户，例如，“ opc_user ”
	[访问权限]	和启动权限相同的用户，例如，“ opc_user ”
	[身份标志]	[交互式用户] 但是，需要不注册OPC服务器计算机也可以运行的场合，请选用[指定用户]。
OPC 客户 计算机	[默认身份认证级别]	[无]

在不隶属于任何计算机域只属于工作组的自动控制系统里，当给予特定的用户访问权限时，或者当[身份标志]指定为[指定用户]时，请参考5.2节，事先添加一个OPC专用用户，并将此用户用作[指定用户]。

4.4 OPC服务器计算机的设置

4.4.1 安装OPC服务器

一般来说，OPC服务器供应商应该提供OPC服务器的安装程序。在OPC服务器计算机上运行OPC服务器的安装程序，应该可以进行复制和注册OPC服务器程序，OPC代理－占位DLL，以及注册OPC的类标识符。使用本书的示范代码时，请参考第６章的［5.4 示范源程序的使用方法］。

4.4.2 分布式COM安全机制的设置

对于OPC远程服务器的设置，除了本地服务器所需要的类标识符的注册表注册以外还需要进行分布式COM的设置。如果只使用分布式COM的默认值，会发生违反访问权限（0x80070005L）的错误，OPC远程服务器也不会正常运行。

应该设置的项目包括[身份验证级别]，[访问权限]，[启动权限]以及[身份标识]。这些设置应该在执行OPC服务器的安装程序之后进行，也就是说应该在注册了OPC服务器的类标识符注册表信息之后进行。

（一）启动分布式COM配置属性的实用程序

使用分布式COM配置属性的实用程序，注册用户应该具有本地计算机管理者的权限或者域管理者的权限。用下述的方法启动这个实用程序。

1) 使用具有管理者权限的用户（例如，“ Administrator ”），注册OPC服务器计算机。
2) 从菜单选择[开始]-[运行(R)…]，显示[运行]对话框。
3) 在[打开(O)]文字框里输入“ dcomcnfg ”（图5-1）。
4) 按下[确定]命令按钮，启动COM配置属性的实用程序COMCNFG.EXE。

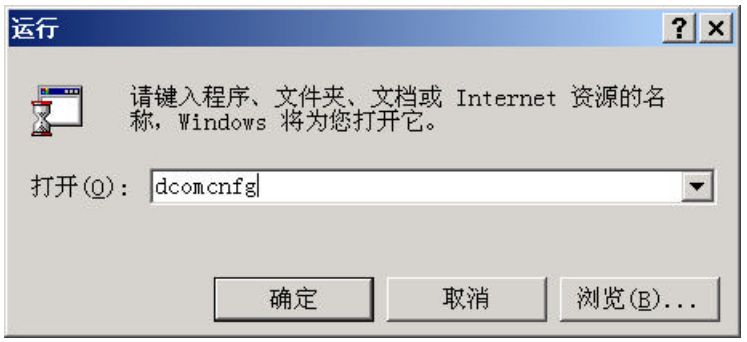


图 5-1 指定文件名的运行

（二）身份验证级别的设置

5) 选择[应用程序]选项卡，可以表示已注册的COM组件的一览。从中选择要设置的OPC服务器。例如，对于本書示范程序的OPC模拟服务器，请选择[OPC-J Sample Server]项目。
6) 按下[属性(P)...]命令按钮，显示[OPC-J Sample Server属性]（图5-2）。
7) 选择[常规] 选项卡。
8) 按照系统对于安全性机制的要求，请参考5.3介绍的推荐值进行适当的设置。

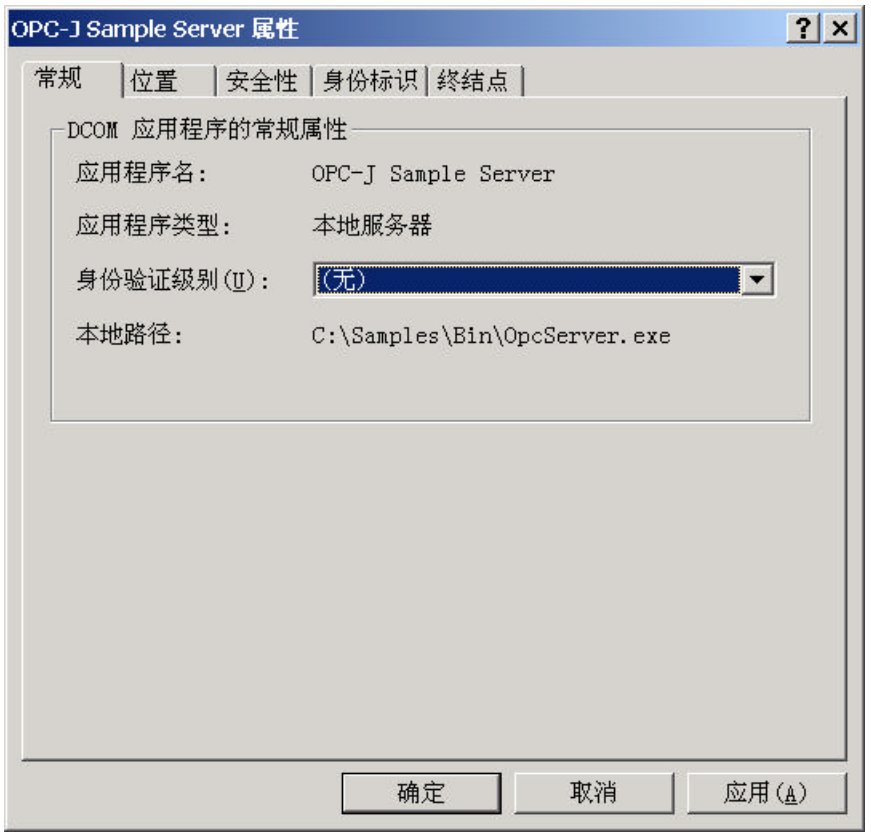


图** 5-2 OPC-J Sample Server**的属性（常规）

（三）启动权限的设置

9) **选择[安全性]选项卡，显示图** 5-3**的画面。选择[使用自定义访问权限(S)]后，按下[编辑(D)...]命令按钮，显示[注册表值的访问权限]对话框。**

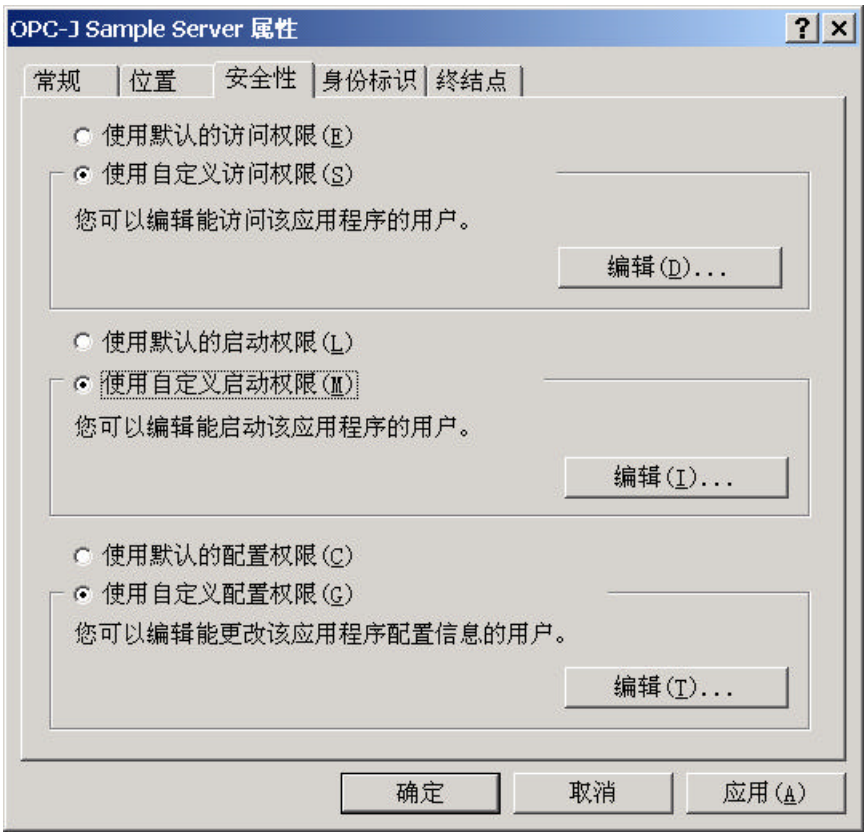


图 5-3 OPC-J Sample Server的属性（安全性）

10) **因为默认值没有设置任何远程用户的启动权限，为了使远程OPC服务器正常动作，按下[添加(A)...]命令按钮，从[添加用户及组]的对话框里，并给予至少一个用户以上访问权限。按照系统对于安全性机制的要求，请参考5.3介绍的推荐值进行适当的设置。表 5-4是经常使用的设置值以及授予访问权限的范围。**

表 5-4 访问用户以及访问权限

设置值	**授予访问权限的用户**
[Everyone]	**所有的用户**
[INTERACTIVE]	**在本地进行数据访问的用户**
[NETWORK]	**在远程进行数据访问的用户**
[SYSTEM]	**操作系统（例如，作为服务的应用程序）**
特定的[域组名]	**特定域组里的所有的用户**
特定的[用户名]	**特定用户**

（四）访问权限的设置

11) **和设置启动权限一样，选择[安全性]选项卡，选择[使用自定义访问权限(M)]后，按下[編集(I)...]命令按钮，显示[注册表值的访问权限]对话框（图5-4）。**

图 5-4 注册表值的权限

12) **请给予访问权限与刚才授予起動权限的相同的用户。**

（五）身份标识的设置

13) **选择[身份标识]选项卡，显示图 5-5的画面。OPC服务器以这个用户的权限进行运行，以便被OPC客户程序进行身份验证。特别使当服务器触发事件时，会由OPC服务器反调OPC客户的事件处理程序。如果OPC服务器不使用被OPC客户应用程序所承认的用户，OPC客户应用程序将拒绝OPC服务器的调用。**

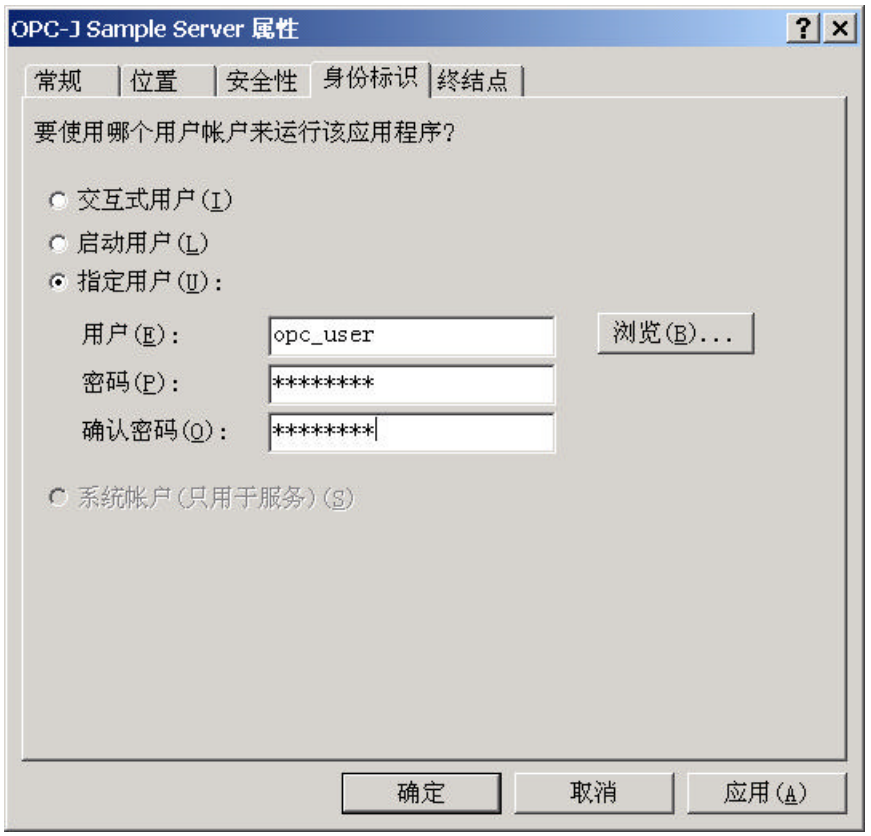


图 5-5 OPC-J Sample Server的属性（身份标志）

14) **因为使用默认值的[启动用户(L)]，OPC服务器不会正常运行，需要按照表** 5-5 **进行必要的设置。**

表 5-5 身份标志

服务器的运行状态	身份标志用户	用户名
OPC服务器必须经常在被注册的状态下运行	[交互式用户(I)]	
即使OPC服务器没有被注册也可以在背后运行	[指定用户(U)]	在参加域的情况下：请指定一个域用户。 Windows 95/98/ME用户或者不能参加域或者属于工作组的情况下，请指定一个按照5.2节的方法建立的OPC专用用户。

4.5 OPC客户程序计算机的设置

4.5.1 安装OPC客户程序

一般来说，OPC客户应用程序供应商应该提供OPC客户应用程序的安装程序。在OPC客户应用程序计算机上运行OPC客户应用程序的安装程序，应该可以进行复制和注册OPC代理－占位DLL和OPC包装DLL。使用本书的示范代码时，请参考第６章的［5.4 示范源程序的使用方法］。

4.5.2 分布式COM安全机制的设置

对于OPC客户应用程序计算机，只需要设置分布式COM的默认身份验证级别。

（一）启动分布式COM配置属性的实用程序

使用分布式COM配置属性的实用程序，注册用户应该具有本地计算机管理者的权限或者域管理者的权限。用下述的方法启动这个实用程序。

1) 使用具有管理者权限的用户（例如，“ Administrator ”），注册OPC服务器计算机。
2) 从菜单选择[开始]-[运行(R)...]，显示[运行]对话框。
3) 在[打开(O)]文字框里输入“ dcomcnfg ”（图5-1）。
4) 按下[确定]命令按钮，启动COM配置属性的实用程序COMCNFG.EXE。

（二）默认身份验证级别的设置

5) 选择[默认属性]选项卡，显示[分布式COM的配置属性]对话框。
6) 设置[默认身份验证级别(U)]为[(无)]。

5 附录

5.1 OPC符号

5.1.1 OPC名称空间符号

表 6-1 OPC名称空间符号

符号	说明
OPCHierarchical	树型的名称空间
OPCFlat	平面型的名称空间

5.1.2 OPC数据源符号

表 6-2 OPC数据源符号

符号	说明
OPCCache	数据缓冲器的数据源
OPCDevice	控制设备的数据源

5.1.3 OPC访问权限符号

表 6-3 OPC访问权限符号

符号	说明
OPCReadable	可读取的访问权限
OPCWritable	可写入的访问权限

5.1.4 OPC服务器状态符号

表 6-4 OPC服务器状态符号

符号	说明
OPCRunning	OPC服务器正在正常运转。
OPCFailed	OPC服务器由于异常而停止。
OPCNoconfig	OPC服务器正在运转，但没有被设置。
OPCSuspended	OPC服务器正处于暂时停止状态。
OPCTest	OPC服务器正在实验模式下运转。
OPCDisconnected	服务器对象没有连接任何实际的OPC服务器

5.2 OPC错误码

表 6-5 OPC错误码

错误码	值	说明
OPCInvalidHandle	0xC0040001L	句柄值不正确。注意: OPC应用程序并没有向OPC服务器提供有效的句柄。这种错误的原因是OPC应用程序的编

		程错误或者是OPC服务器的处理错误。
OPCBadType	0xC0040004L	OPC服务器无法在指定的数据格式或者数据类型与固有的数据类型之间变换。
OPCPublic	0xC0040005L	所要求的操作无法对OPC公用组执行。
OPCBadRights	0xC0040006L	由于OPC标签的访问权限所限，操作被拒绝执行。
OPCUnknownItemID	0xC0040007L	在OPC服务器的名称空间中，要求的OPC标签标识符不存在。
OPCInvalidItemID	0xC0040008L	要求的OPC标签标识符不符合OPC服务器的语法规定。
OPCInvalidFilter	0xC0040009L	OPC浏览器过滤器的字符串无效。
OPCUnknownPath	0xC004000AL	OPC服务器不认识OPC标签的访问路径。
OPCRange	0xC004000BL	数值超出范围。
OPCDuplicateName	0xC004000CL	不允许的重复名称。
OPCUnsupportedRate	0x0004000DL	OPC服务器不支持要求的数据更新周期，但使用最相近的周期。
OPCClamp	0x0004000EL	写入值虽被接受，但输出受到箝位限制。
OPCInuse	0x0004000FL	因为对象正在被使用，要求的操作无法执行。
OPCInvalidConfig	0xC0040010L	OPC服务器组态文件的格式不正确。
OPCNotFound	0xC0040011L	找不到所要求的对象（例如，公用组）。
OPCInvalidPID	0xC00402.03L	对于指定的OPC标签，要求的属性标识符无效。

5.3 OPC数据型

5.3.1 经常使用的OPC数据型

随OPC服务器的供应商不同，所支持的数据类型可能有所不同。表 6-6列出了经常使用的OPC数据类型。

表 6-6 OPC的数据类型

数据类型	字节数	对应VB数据类型	说明
VT_I2	2	Integer	带符号２字节整数
VT_I4	4	Long	带符号４字节整数
VT_R4	4	Single	４字节实数
VT_R8	8	Double	８字节实数
VT_CY	8	Currency	货币
VT_DATE	8	Date	从1899年12月30日开始计算的日期

VT_BSTR	可变长	String	字符串
VT_BOOL	1	Boolean	0：FALSE，-1:TRUE

5.3.2 定制数据型和自动化数据型

因为自动化接口的OPC应用程序只支持定制接口OPC服务器的数据类型的一部分，所以会按照如表 6-7所示的对应关系自动地进行数据类型的变换。

表 6-7 定制数据类型和自动化数据类型

定制数据类型	自动化数据类型	对应VB数据类型
VT_I1	VT_I2	Integer
VT_UI2	VT_I4	Long
VT_UI4	VT_R8（或VT_CY）	Double （或Currency）

5.4 示范源程序的使用方法

在本书配套光盘里收录了第２，３，４章作成的示范程序。

5.4.1 复制和注册示范源程序

在使用各章的示范程序前，需要按照以下的方法复制和注册OPC模拟服务器，有关的DLL以及ActiveX控件。

1) 将本书配套光盘插入光盘驱动器，使用浏览器打开光盘驱动器。
2) 选择存放示范代码的文件夹“ \Samples ”，然后把它拖放至硬盘上要放入复制文件的位置（比如，“ C:\ ”），完成示范代码的复制。
3) 从菜单选择[开始]-[运行(R)...]，然后按下[浏览(B)...]按钮，选择存放复制文件位置内的OPC注册批处理文件“ \Samples\Bin\RegOPC.BAT ”（图6-1）。

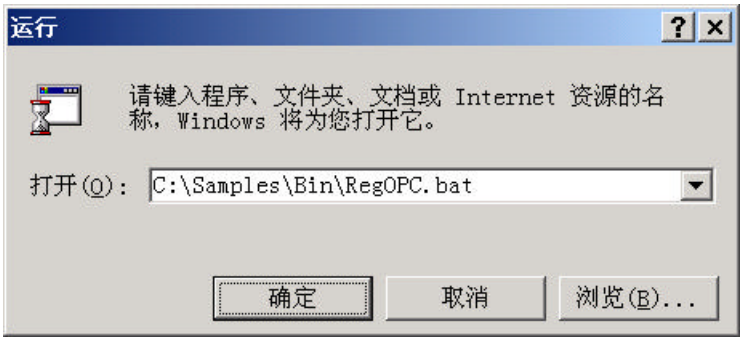


图 6-1 示范源程序的注册

4) 按下[确定]命令按钮，在注册表里注册OPC模拟服务器，OPC代理－占位DLL以及ActiveX控件。
5) 因为从光盘复制的文件是只读属性的，不可以对其进行变更。需要变更时，从浏览器选择要进行变更的文件，按下鼠标右按钮并选择[属性(R)]项目。清除[常规]选项卡里的[只读(R)]复选框的选择，就可以解除文件的只读属性（图6-2）。

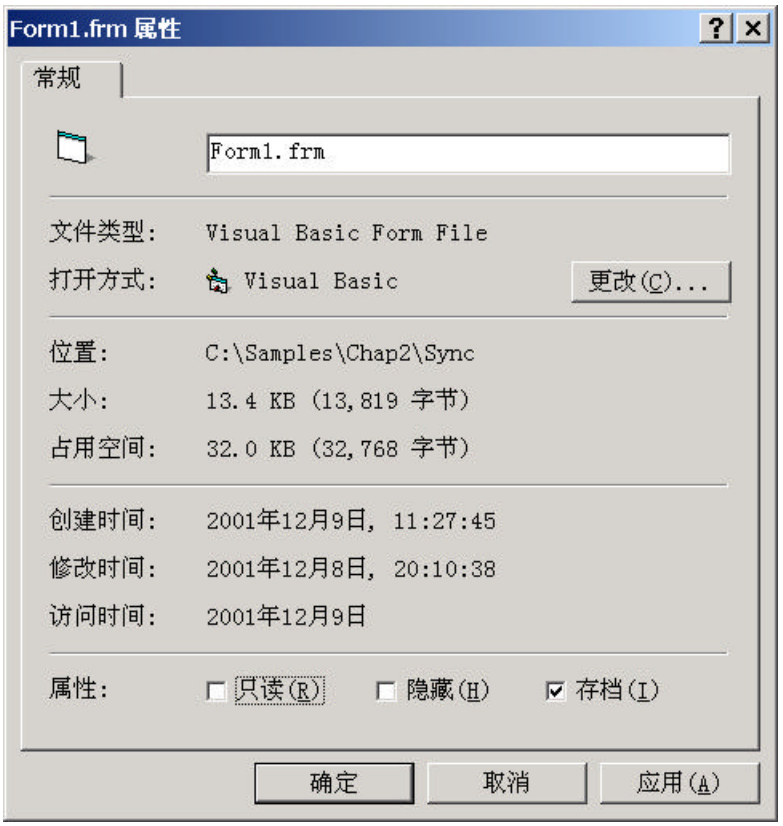


图 6-2 文件只读属性的取消

5.4.2 运行示范源程序

本书的示范程序的启动文件如表 6-8**所示：**

表 6-8 OPC的示范代码

章	示范程序	启动文件
第２章	OPC同步数据访问VB示范程序	\Samples\Chap2\Sync\Project1.vbp
第２章	OPC异步数据访问VB示范程序	\Samples\Chap2\Async\Project1.vbp
第３章	OPC ActiveX VB示范程序	\Samples\Chap3\OPCTrend.vbg
第４章	OPC VBA Excel示范程序	\Samples\Chap4\VBA.xls
第４章	OPC ActiveX Excel示范程序	\Samples\Chap4\ActiveX.xls

5.4.3 示范源程序的运行环境

本书的示范程序要求的运行环境如下所示：

操作系统: WindowsNT 4.0 Workstation（SP3**或更高**）/Windows 2000 Professional

Visul Basic: Visual Basic 5.0/6.0 **专业/企业版（第２章和第３章）**

Microsoft Excel: Excel 97/ Excel 2000/ Excel XP **（第４章）**

5.5 参考资料

在本书配套光盘里收录了原文的OPC**数据访问标准。内容如表** 6-9**所示。**

表 6-9 有关 OPC的参考资料

文件名	语言	内容
\Documents\opcda20_auto.doc	英语	OPC数据访问自动化接口标准（版本2.02）
\Documents\opcda204_cust.doc	英语	OPC数据访问定制接口标准（版本2.05）

5.6 有关OPC的互联网站

表 6-9 有关OPC的WEB网站

名称	互联网址
OPC基金会	http://opcfoundation.org/
日本OPC协会	http://www.microsoft.com/japan/business/industry/mfg/opchome.asp
欧洲OPC协会	http://www.opceurope.org/index.htm
中国OPC促进会	http://opcchina.org/
OPC编程者联络网	http://ds.dial.pipex.com/town/estate/on50/index.shtml

5.7 OPC专用名词中英对照表

英文	中文
Asynchronous Access	异步数据访问
Automation Interface	自动化接口
Branch (Name Space)	枝（名称空间）
Cache	缓冲器
Client Handler	客户句柄
Custom Interface	定制接口
Dead Band	不敏感带
Flat Name Space	平面型名称空间
Hierarchical Name Space	树形型名称空间
Leav (Name Space)	叶（名称空间）
Locale ID	区域标识符
OPC Browser	OPC浏览器
OPC Client	OPC客户应用程序
OPC Group	OPC组
OPC Group Collection	OPC组集合
OPC Item	OPC标签
OPC Item Collection	OPC标签集合
OPC Item ID	OPC标签标识符
OPC Item Path	OPC标签路径
OPC Server	OPC服务器
OPC Wrapper DLL	OPC包装DLL
Program ID	程序标识符
Proxy-Stub DLL	代理－占位DLL

Public Group	**公用组**
Quality	**质量**
Refresh	**刷新**
Server Handler	**服务器句柄**
Subscription	**订阅方式数据采集**
Synchronous Access	**同步数据访问**
Time Bias	**时间偏差**
Time Stamp	**采样时间**
Transaction ID	**事务标识符**