AS3.0.80 Introduction

一、 软件的结构

Project explorer 窗口显示包含了三个项目的不同的开发视窗，分别为 Logical View，Configuration View 和 Physical View。可以通过 View-→Project Explorer/Logical View，Configuration View， Physical View 来加载不同的视窗。以下详细介绍不同的视窗的作用。

（1）Logical View

在 logical view 中，主要针对的是程序部分的组织和管理。每个 Package 包括了对于一个具体设备部件的软件程序、文档，而每个 Package 可以实现灵活的导入导出。可以实现多人合作。图 1 为基于 Demo 程序 Coffee Machine 项目的功能结构框架。将关联特定功能的组件封装组合，结构清晰。

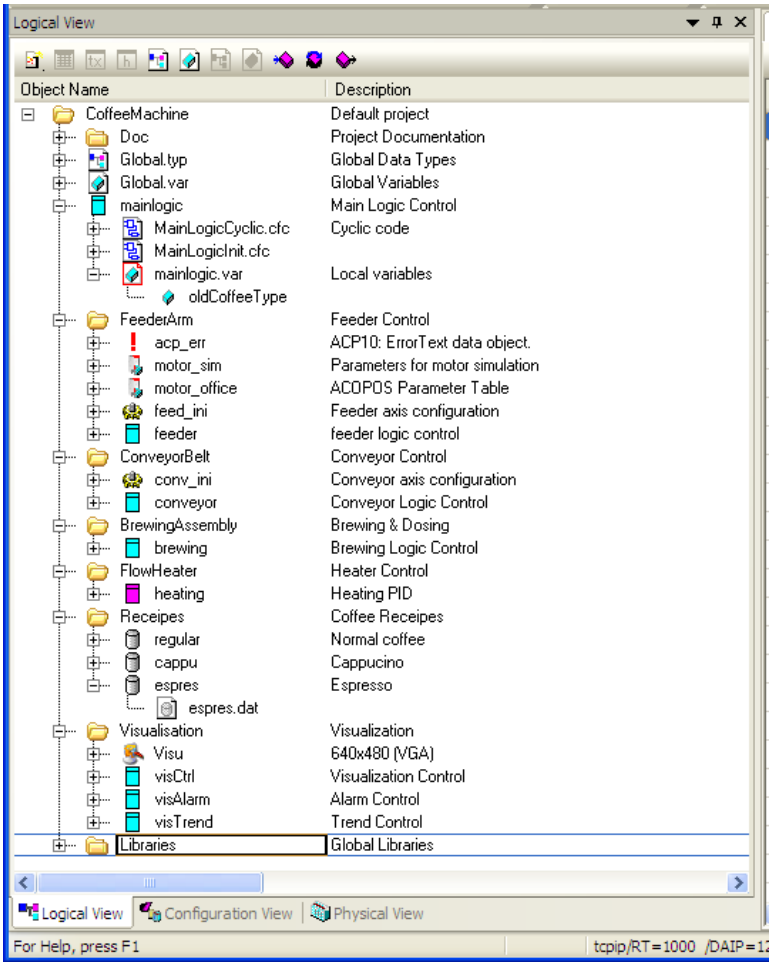

图 1 应用项目的 Logical View

（2）Configuration View

管理着不同的硬件配置。在一个项目中，可以添加多个硬件配置，例如对于一个较大的项目，有不同的配置方案。但同时只能激活一个硬件配置，激活的配置的硬件结构将会在 Physical View 中显示。在一般情况下，会添加一个额外的仿真器平台，作为测试功能。图 2 显示在 Configuration view 中添加了三个不同的系统配置。激活的硬件配置平台显示为黑色加粗。

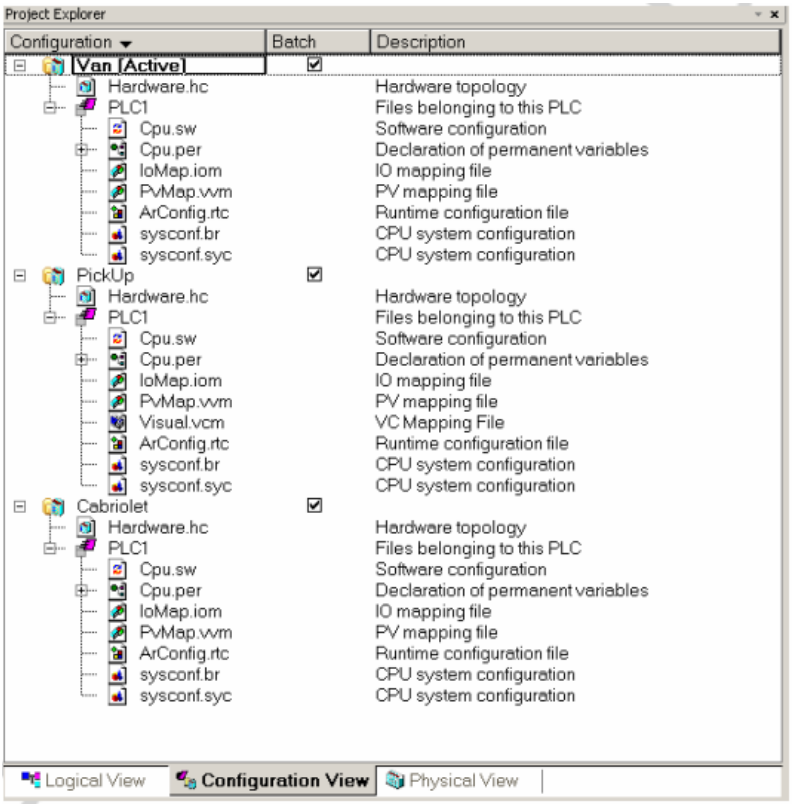

图 2 添加多个不同的系统配置

（3）Physical View

在Physical View中配置硬件树形结构。如图3所示的树形硬件结构。这里的树形结构类似于AS2.x中的显示。由于没有工作区的菜单，对 Physical View 的操作大多可以通过点击 CPU，右键打开各项功能。

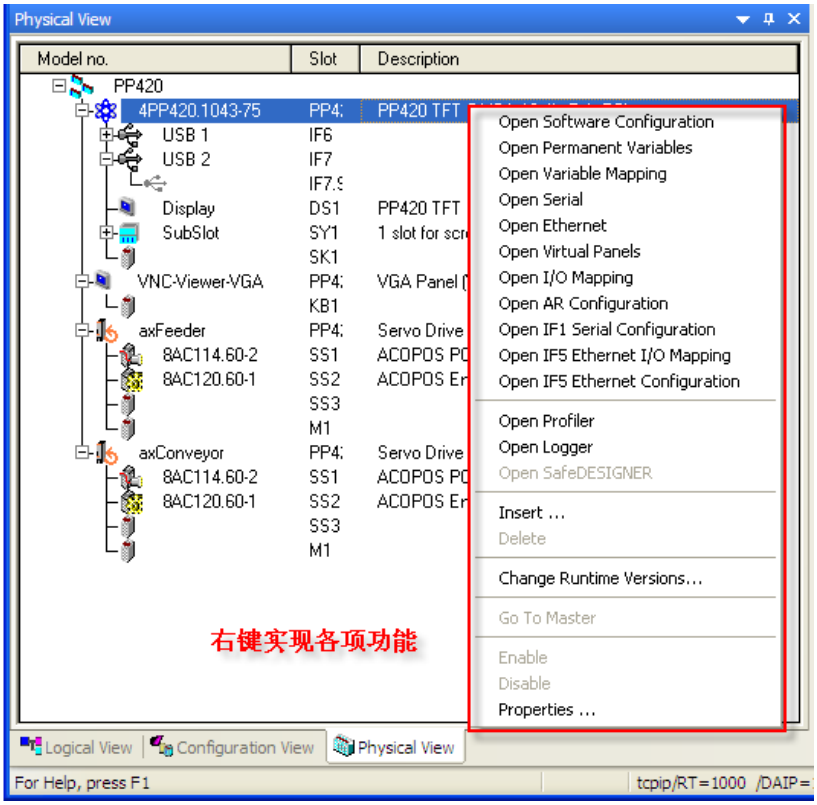

图 3 硬件树形结构

二、 创建项目流程

（1）新建项目

图 4 显示了新建项目的选项，选项 1 表示添加仿真器为系统默认的硬件配置，便于在项目过程中对功能的测试。选项 2 将该项目的操作系统保存在该项目目录中，避免了当项目拷贝到其他的电脑上，因操作系统的不匹配出现的问题。这两个为可选选项。Description of the project 建议写对于项目的描述。

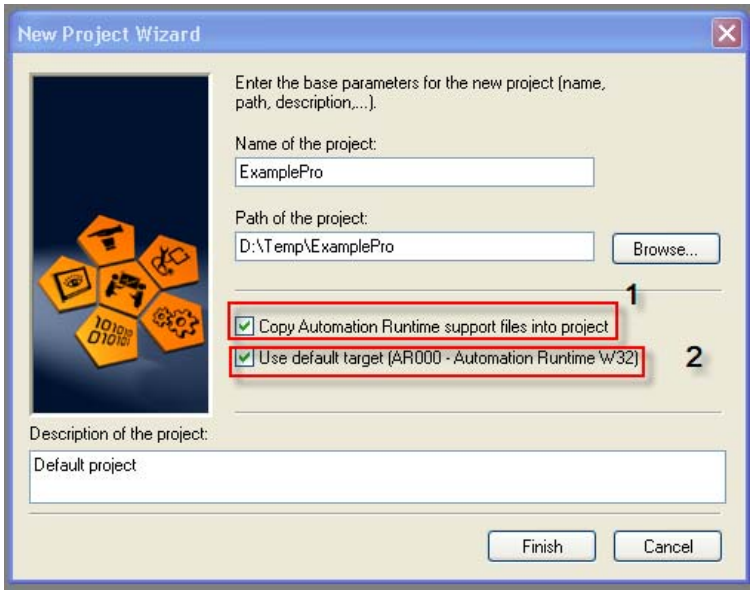

图 4 新建项目

（2）添加硬件配置

工程中可在 Configuration View 中添加新的硬件配置。点击已添加的配置名，右键 Add new Configuration，出现图 5 的添加硬件配置视图，添加完成后该配置显示为当前激活状态，在 Physical View 中显示该配置下的控制器，其他的硬件如 IO 等则需要在 Physical View 中逐一添加和配置。

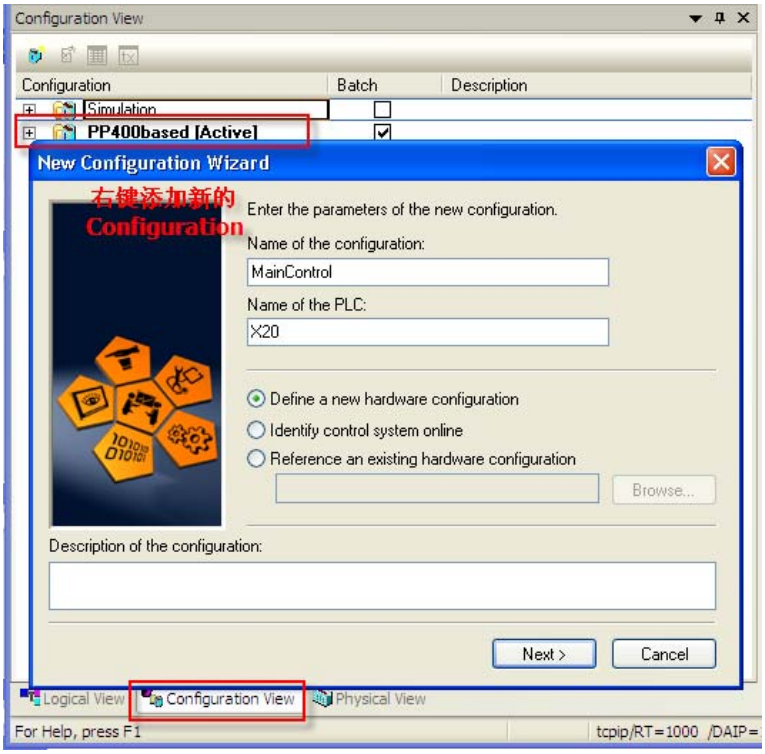

图 5 添加硬件配置

（3）添加硬件

在 Physical View 中，点击 CPU 右键，例如 Open X2X Link，将显示右边区域 2 的 IF6，在此可以添加 X20 的 IO 等硬件设备。如图 6 所示的添加了 X20 系列硬件。

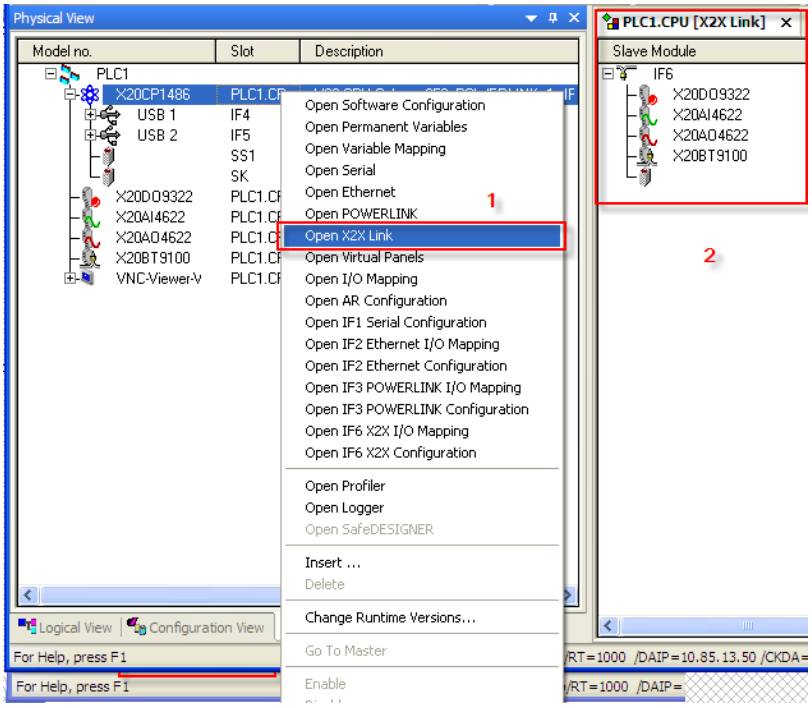

图 6 添加硬件

（4）添加项目组件

图 7 为项目中 Logical View 中可以添加的组件。

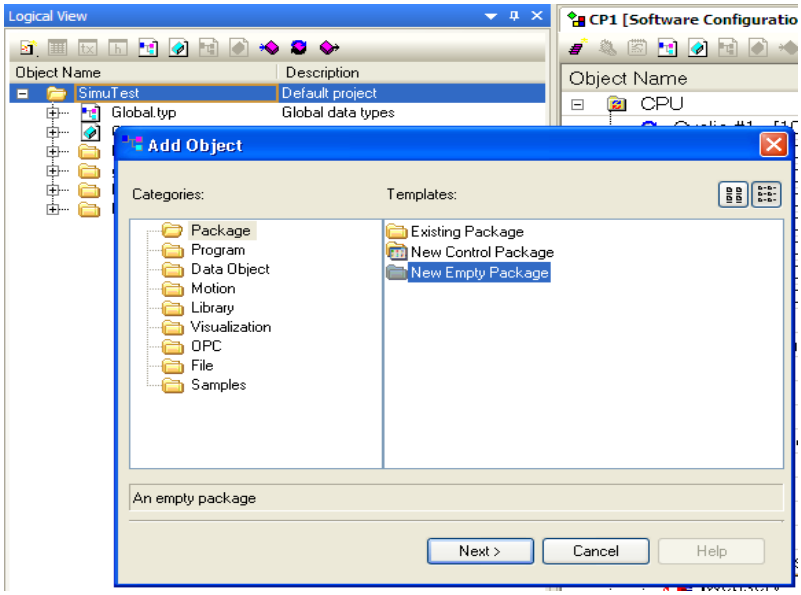

图 7 Logical View 中添加组件

- Package

在项目需要管理大量的项目组件，数据对象、变量、程序以及其他元素。这时可以用 Package 将这些元素组成一个整体的单元。

按照功能可以将项目中的元素归类和整合，在 Package 中的变量和变量类型只局限于在 Package 中使用，每个 Package 之间通过 Package 的全局变量进行关联。

在 Logical View 和 Configuration View 中都可定义 Package，但是这两个 View 中的 Package 并无直接的联系。图 8 为在 Logical View 和 Configuration View 中的 Package。

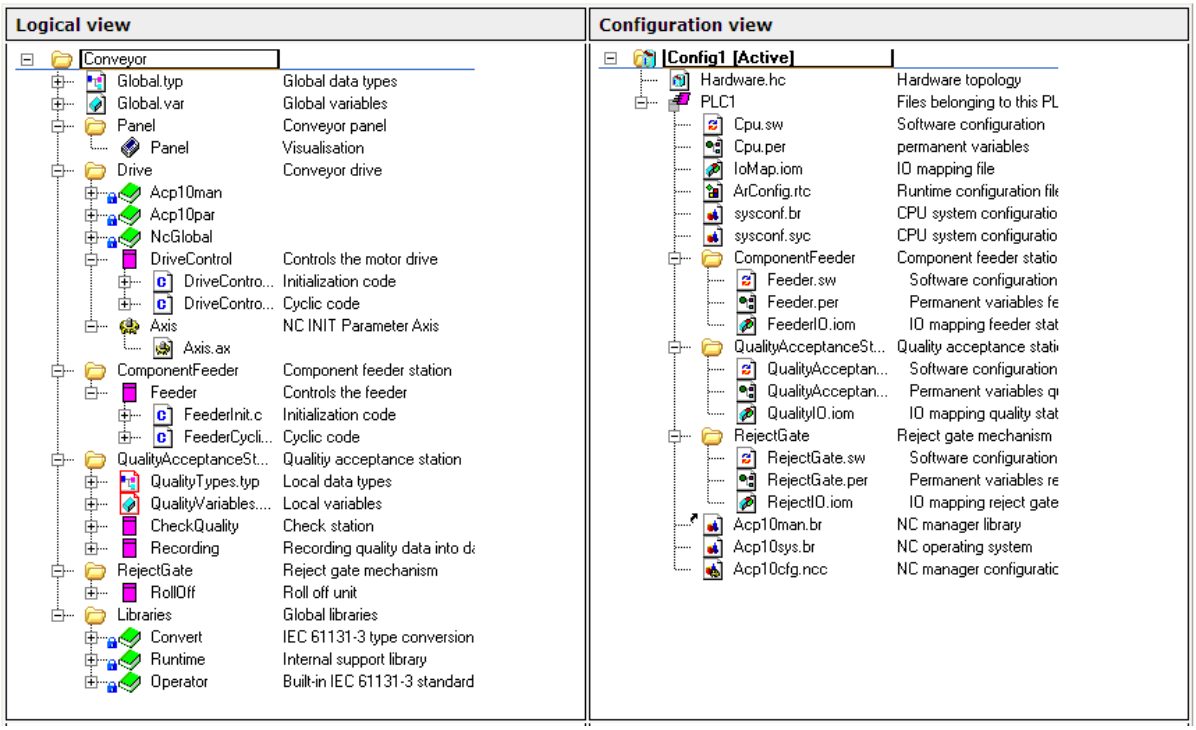

图 8 Logical View 和 Configuration View 中的 Package

- Program:

在 AS 中可以选用多种编程语言，选项 1 的下拉选项框中可以选择所需的编程语言，选项 2 表示是否将 Init program, Cyclic program, Exit program 生成一个文件。图 9 添加程序。

Init program:初始化部分程序。

Cyclic program:循环部分程序。

Exit program:当某个独立的任务卸载时，执行的退出程序。

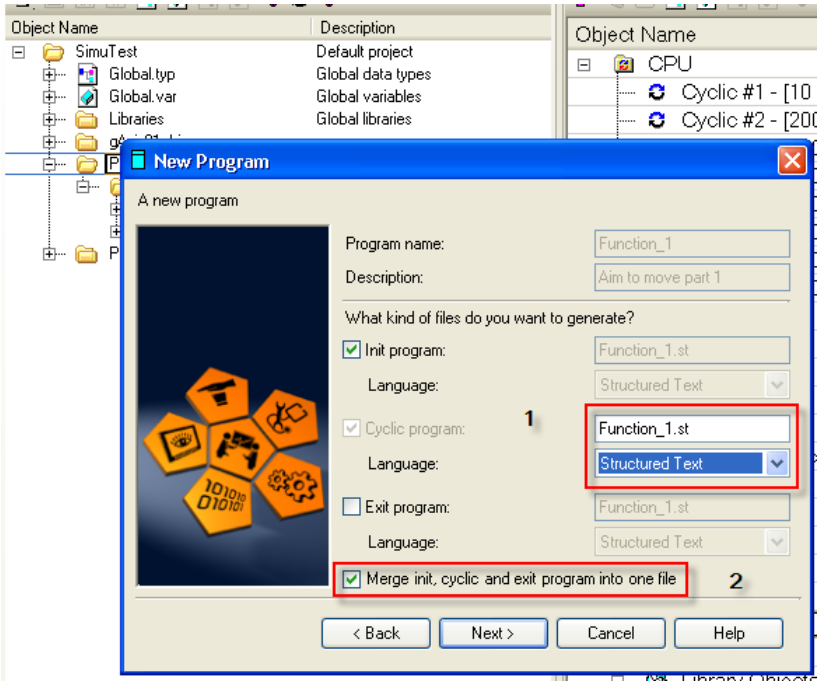

图 9 添加程序

当完成程序的创建后，可以选择是否添加到相应的硬件配置中。图 10 显示将软件程序添加到相应的硬件配置中。选项 1 表示只将此程序添加到当前激活的配置中，默认添加到 Cyclic#4 中；选项 2 表示将此程序添加到所有硬件配置的 Cyclic#4 中；选项 3 并不自动添加。对软件的配置在之后都可以灵活的配置（添加或者删除）。

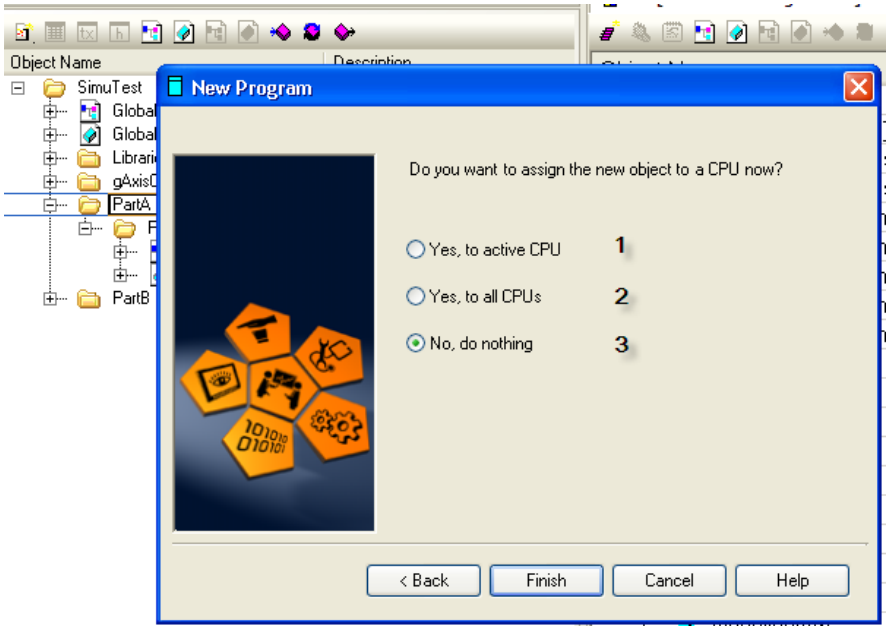

图 10 添加硬件配置的程序

- Data Object

B&R 格式的数据模块，可以使用 Dataobj 对数据模块进行操作。

- Motion

对于运动控制的组件，包括 Acopos 参数表、CNC 程序，CNC 表格、NC 凸轮仿形、NC 错误文本、NC 初始化轴参数、CNC 系统的 NC 初始化轴参数。图 11 显示了在 Logical View 中可以添加的 Motion 组件。

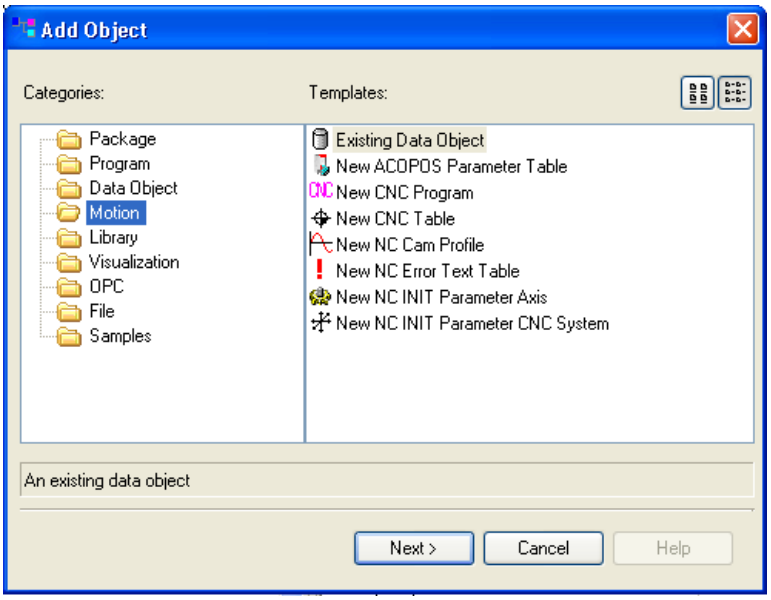

图 11 Motion 组件

- Library

B&R 的标准库函数以及用户自定义函数库。

- Visualization

VC3 和 VC4 组态画面。

- OPC

OPC 报警声明

OPC 特性声明

OPC Tag 声明

- File

添加文件，可以是对文件的说明或者为其他格式的文件。

- Sample

导入库函数的例子中。

（5）将程序加载到硬件系统中

在 Logical View 中的组件，可以拖到处于激活配置的工作区中。在 Configuration View 中双击

Cpu.sw 或者在 Physical View 中双击 CPU 可以打开右边的 Program workspace。

实现灵活组合和配置。图 12 将 Logical View 中的部件拖动加载到工作区中。

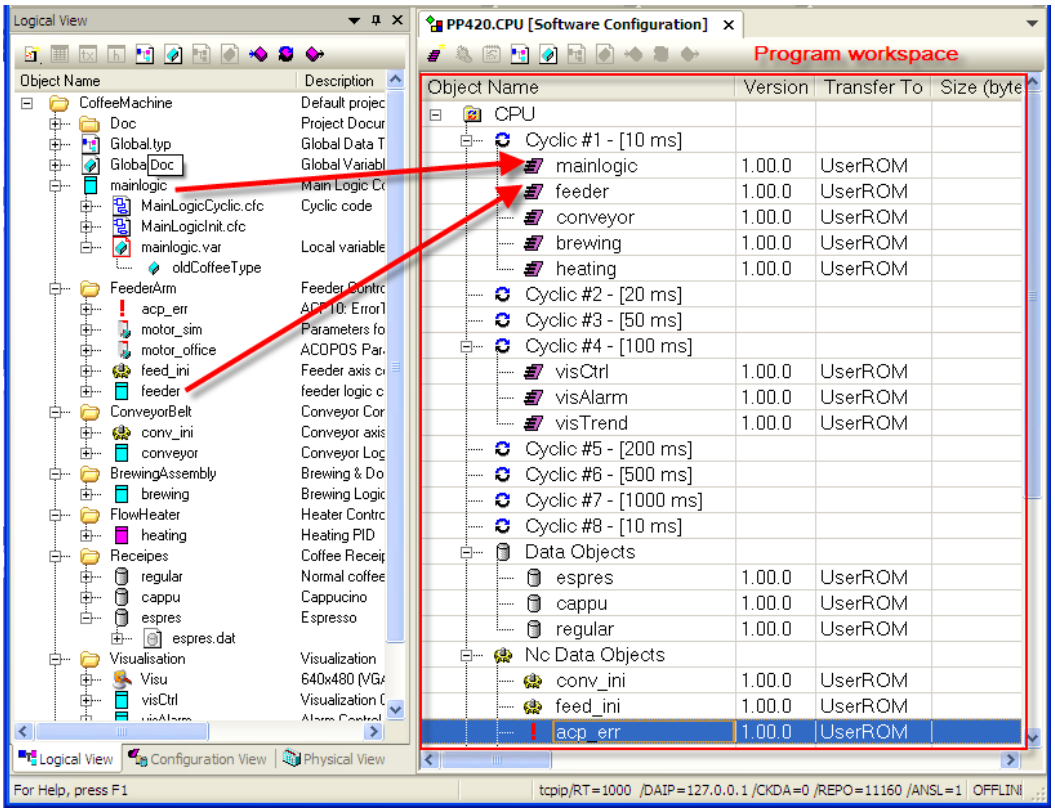

图 12 加载程序到激活的 Program workspace 中

（6）编译、下载

三、 灵活便捷的新功能

（1）智能编辑

自动变量声明：在单个程序中，可在变量声明表中首先添加新的变量，但如果使用 IEC 语言，使用了未定义的变量，将自动的跳出变量声明表格。首先打开 Tools--→Options--→Smart Edit，勾选上 Automatic declaration of new variables,如图 13 的设置此功能。也可以在每个程序的 variable 中添加。

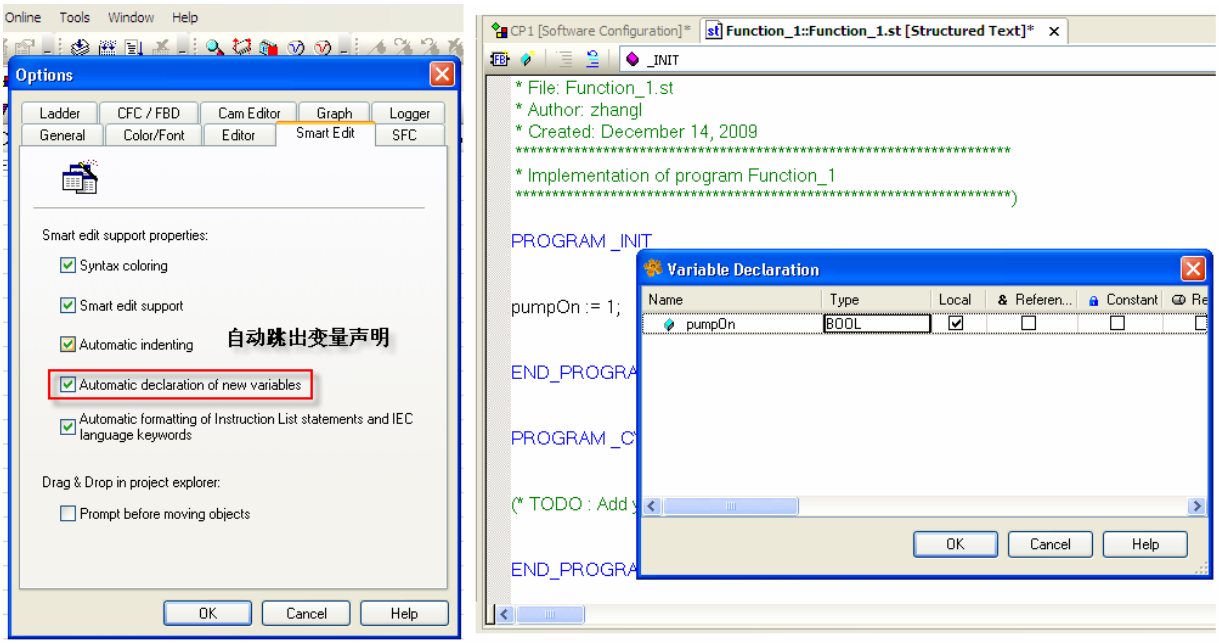

图 13 设置自动变量声明功能

结构体变量初始化：可以在变量声明表中对结构体变量赋初始化值。如图 14 所示。

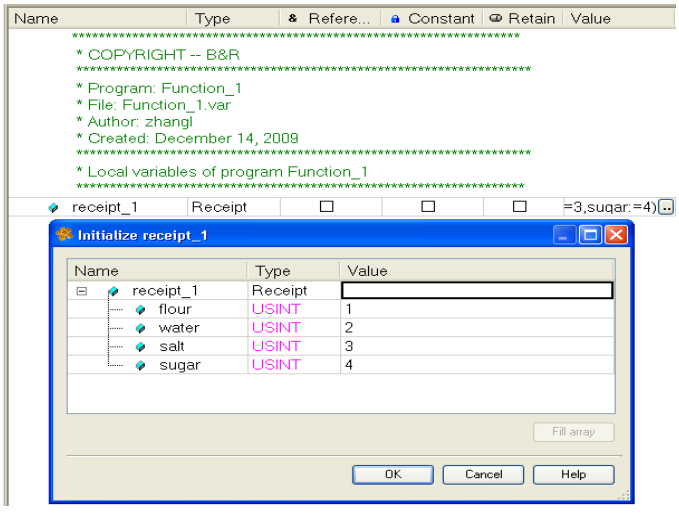

图 14 结构体变量赋初值

（2）网络中 IP 地址设定和识别

新建项目后，如图 15 所示网络设置。选项 1 为打开网络设置；选项 2 为默认激活网络配置；不同的网络配置可以使用选项 3 的下拉框中选择。

如不同网络配置修改，默认是以自动获得 IP 地址的的网络连接方式。

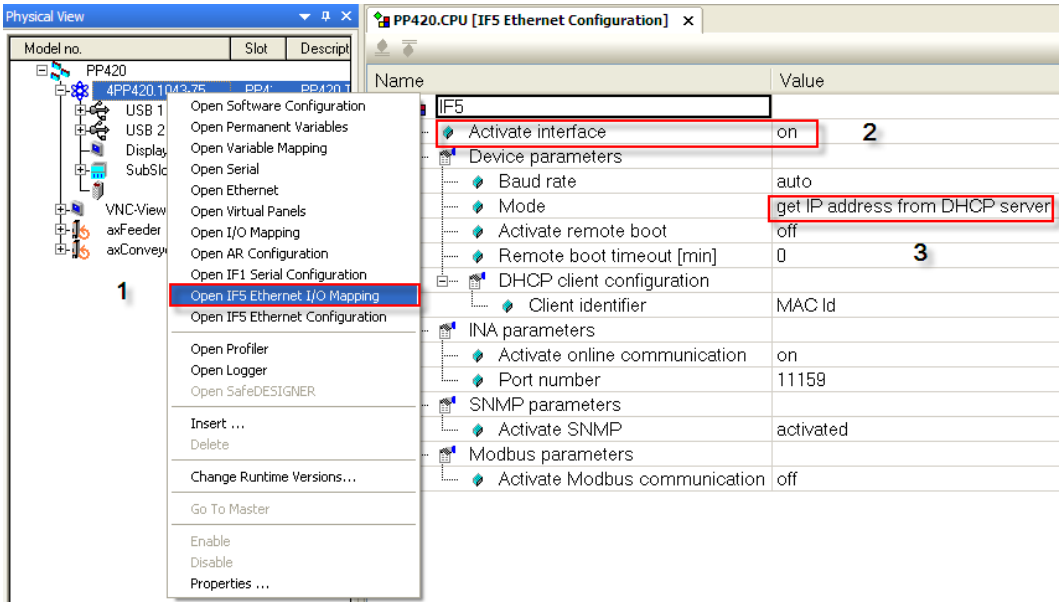

图 15 网络配置选项

当设备连接在局域网中，可以通过搜索功能，获得硬件的 IP 地址。打开 Online--→Setting，Browser 选项可以搜索在网络设备的 IP 地址。如图 16 所示。按键 1 可实现搜索网络中设备的 IP 地址，当查找到相应设备后，将其拖到连接选项框中，并勾选上“Use in active config”。可以用按键 2 建立连接。

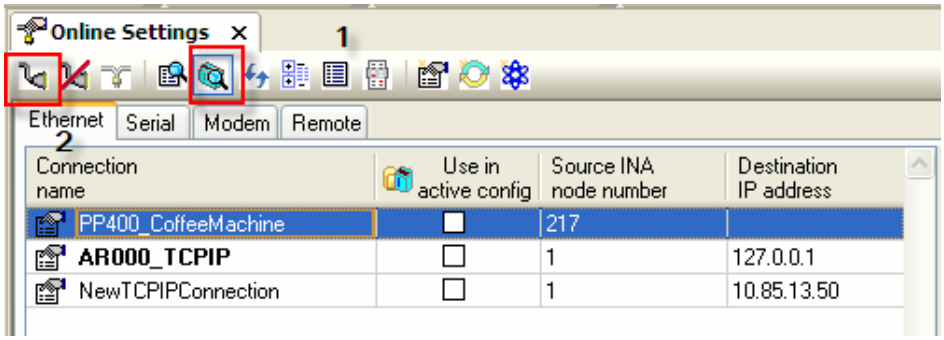

图 16 搜索网络中设备地址

（3）灵活导入现场总线设备

当使用现场总线时，灵活导入 GSP 或者 EDS 文件，进行配置。打开 Tools---→Import Fieldbus Device，添加 GSP 或者 EDS 文件。

（4）版本修改

打开 Project---→Change Runtime Version。可以修改 Runtime 版本、Visual Components 和 Motion Control 版本，如图 17 所示。

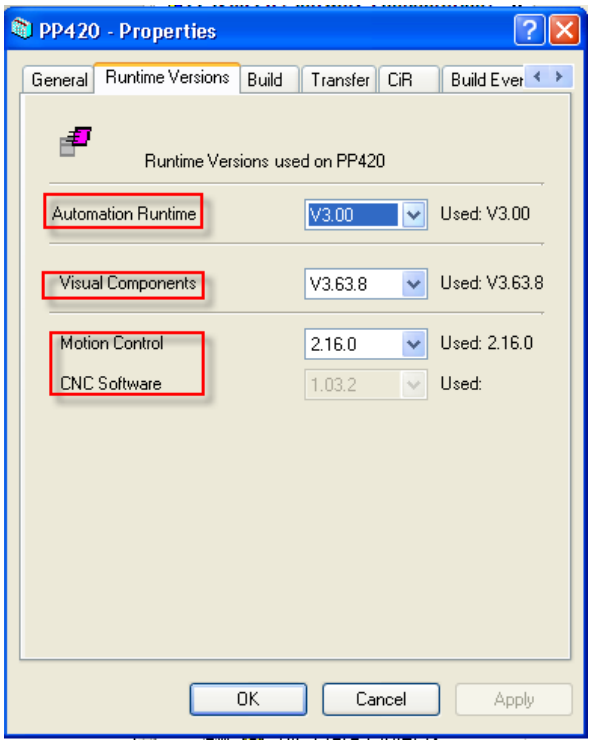

图 17 修改系统版本

（5）变量类型及定义

变量有三种类型：Global variables、Package-local variables、Local variables.

Global variables：基于整个控制器，在所有的 Task 中都可使用此变量。

Package-local variables：在 Package 中使用，对于它的子 Package 和它的程序有效。但是基于整个控制器而言，这些变量是全局有效的。

Local variables：只在一个程序可见，基于整个控制器，这些变量是局部的。

如图 18 所示，区域 1 定义 Global variables， 区域 2 定义 Package-local variables，区域 3 定义 Local variables。

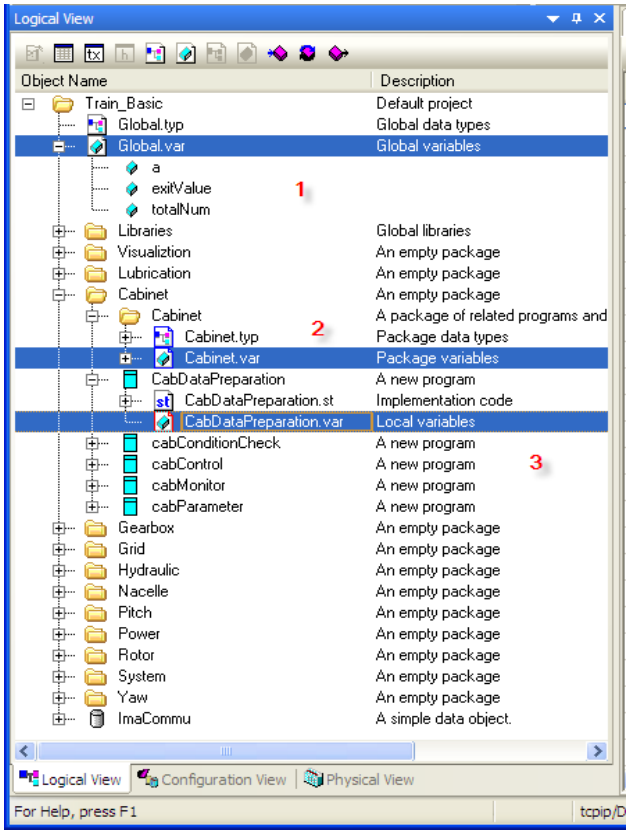

图 18 变量定义

对于 Global variables 和 Local variables 可在已有的程序或者项目中添加，对于 Package-local variables 点击 Package，右键 Add Object，选择 Package 项目中的 New control package。如图 19 所示。

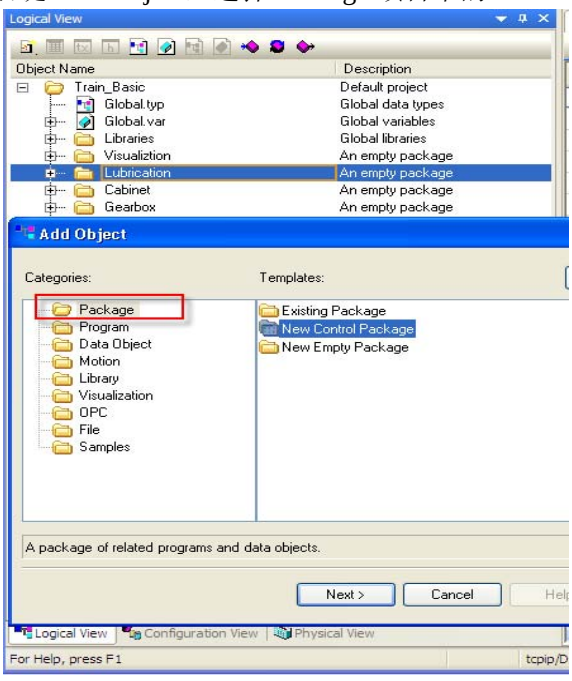

图 19 添加 Package variable

变量定义既可以使用 Text 或者 Table 形式编写。双击变量定义或者右键“Open as Text”，”Open as Table”可实现变量定义。变量的定义见图 20 所示：按键 1 为添加新变量；按键 2 为添加新 Comment；输入 3 为设定变量名称；输入 4 设定变量类型；选项框 5 为设定指针变量；选项框 6 是设定常量；选项框 7 为定义 Remenant 变量；输入框为定义变量初始值或者常量值。

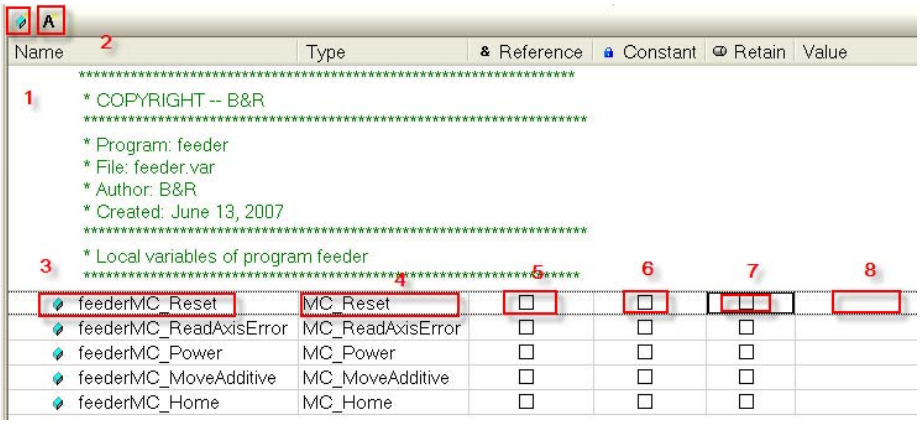

图 20 变量定义

（6）批编译处理（Batch）

可将各种的硬件配置的 Batch 勾选上，通过 Project----→Batch---→Build，对多个控制器的系统编译。图 21 显示了选中的控制器。

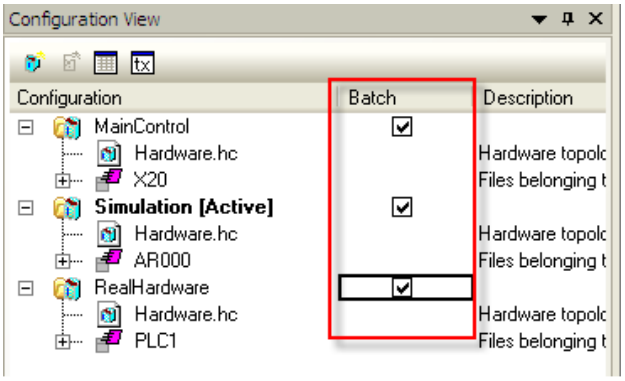

图 21 批编译控制器

（7）用户自定义库

当在 Library 中添加 New Library，可以按照添加用户定义的 Function 和 Function Block。用户库可以在不同的目标平台上编译，因此用户定义库之后可在不同的目标平台上使用。

首先要创建不同目标系统的配置，包括了 SG3 和 SG4 的平台，创建的用户添加到不同的系统平台中，并使用批编译（Batch build）使用户自定义库可以在不同的系统平台上使用。图 22 生成在不同目标平台上的用户自定义库。

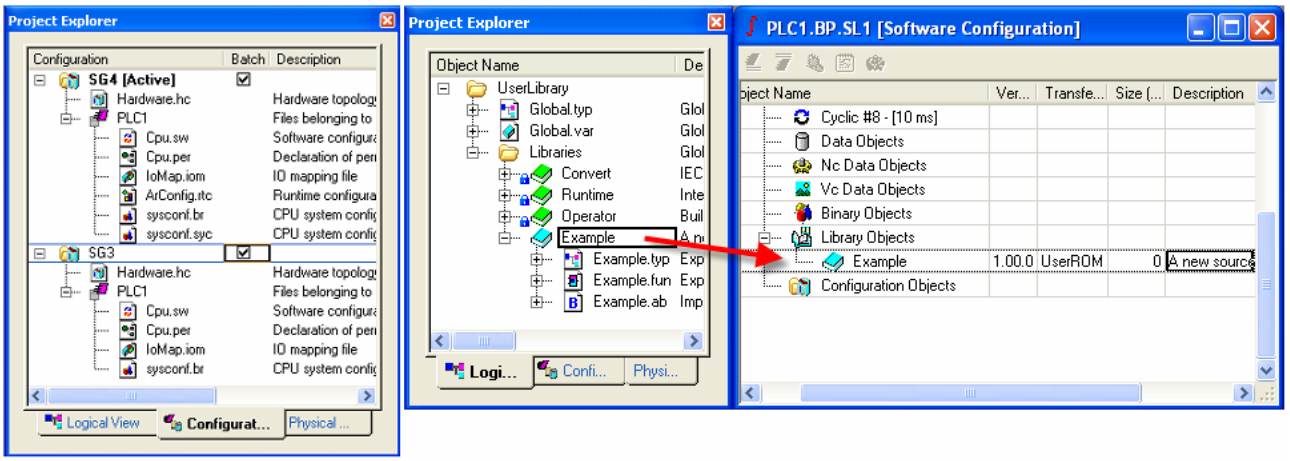

图 22 用户自定义库

用户库可以以二进制码的格式导出（即删除了源代码）；或者包括源码。

点击用户库，并通过 File---→Export Library 打开图 23 的导出用户库。默认选项框 1 为以二进制导出用户库，这时源代码不可见。否则不勾选选项框，如在其他程序中导入用户库将携带源代码。

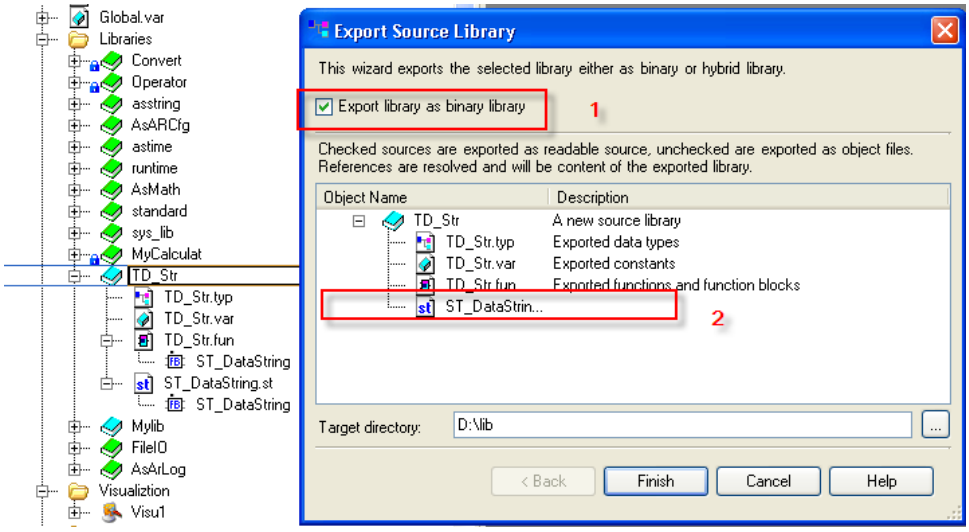

图 23 导出用户库

（8）伺服控制

驱动配置，在 Physical View 中打开 Powerlink interface，添加 Powerlink 通讯卡和编码器后，进入图 24 的画面，这时可以设定供电模式。

选项框 1 和 2 可以设定为 AC 或者 DC 的供电模式，如选择 AC 供电，可设定单相或者三相供电。选中选项 4 继续定义 Deployment Table，对轴进行设定。

如果采用灵活的 NC 配置，即软件配置，需要将所有的部件 Disable，所以无需过于在意此时的设置。

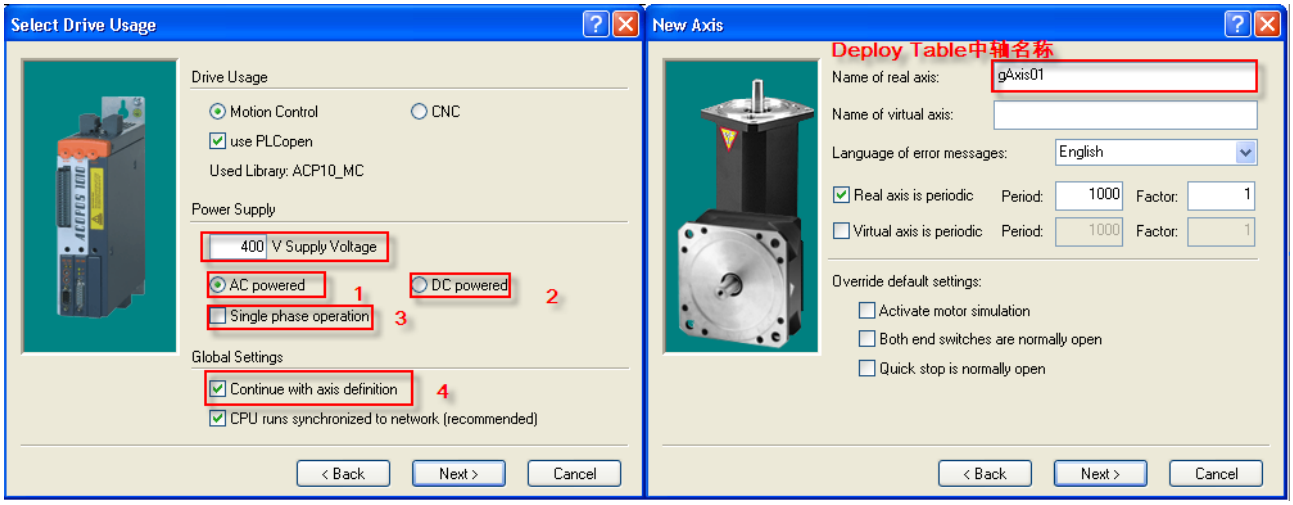

图 24 添加轴配置

添加完毕后，在不同的窗口会自动生成如下的组件，如图 25 所示，在 Physical View 区域 1 中显示相应的硬件，在 Configuration View 会生成名为 Motion 的 Package；在 Logical View 中会生成一个 ACP10AXIS_typ 类型的全局轴名称和区域 5 中的伺服的基本元素：初始化轴参数，参数表和错误文本。

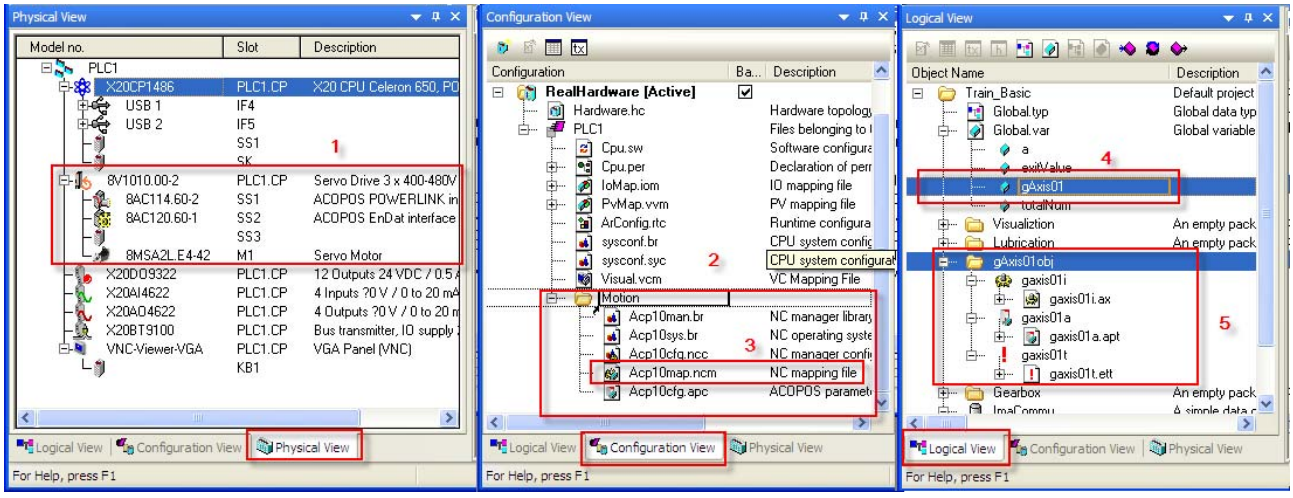

图 25 轴配置

在 Configuration View 中文件 3，Acp10map.ncm 为添加的轴的 Deployment Table，在这个文件下不能添加其他的轴的配置，如果有多轴的，可以有两种方法，一、再次添加硬件，会自动生成以上的所有的组件，如图 26 所示。这时在 Configuration View 中配置文件中将会显示两个轴的配置。

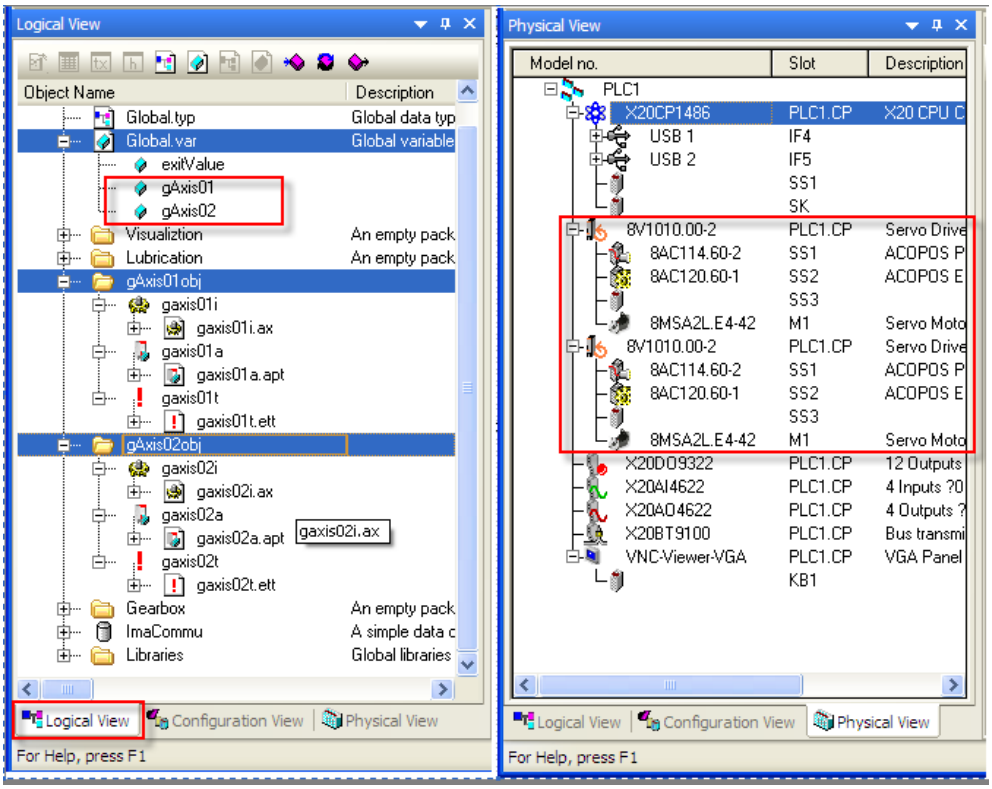

图 26 添加多个轴

方法二是使用灵活的 NC 配置，在 Physical View 中将硬件 Disable，由于在 Configuration View 中已生成的 Acp10map.ncm 同时也被 Disable，所以此时添加一个新的 acopos mapping table，如图 27 所示。其他的参数设置可以在 Logical View 中的参数表进行设定。

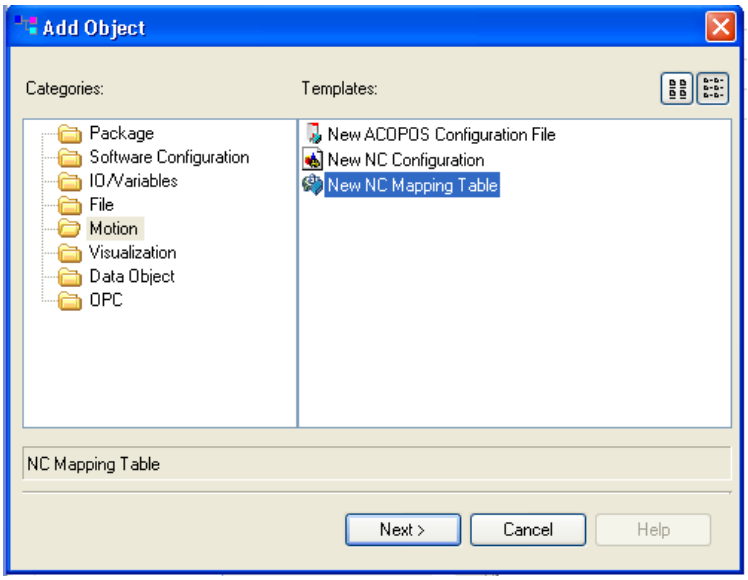

图 27 添加新的 NC Mapping Table

（9）交叉索引（Cross Reference）

交叉索引提供了变量、功能块、功能在程序中的使用的信息。他们在程序代码中的位置以及属性，例如是读操作或者写操作。

通过 View----→Output--→Cross Reference 打开交叉索引的功能窗口。

若需要生成 Cross Reference 列表，通过 Project---→Build Cross Reference 菜单实现此功能。如图 28 所示的 Cross Reference 输出框。区域 1 显示了变量的类型，如变量前有红色的叉号，代表这个变量在此配置下未使用（即程序未加载到相应的系统平台上），点击变量，将会显示右边的信息，此变量的使用情况，出现的位置以及使用的属性，进行了读或者写操作。图 28 显示了一个全局变量在两个程序中的使用情况，一个进行了读操作，另一个进行了写操作。点击区域 2 或者 3 将跳入程序，显示变量的操作位置。

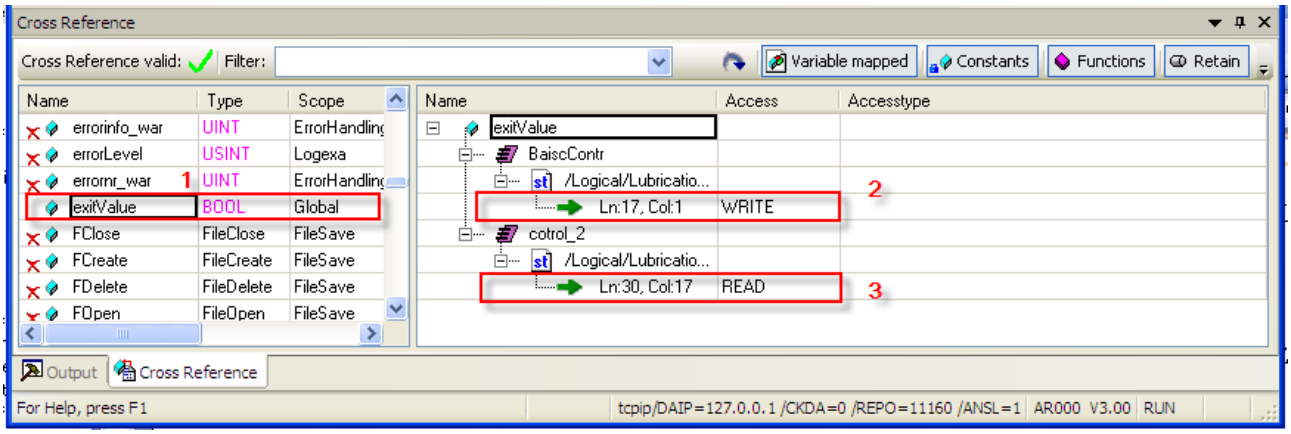

图 28 Cross Reference 输出框

可以设定条件显示筛选的条件，如图 29 所示的，但是这些条件是“或”的关系。如只选择“Global”，则显示 Global 的变量；若选择“Global”和“Function”，则显示所有 Global 的变量和所有程序中的 Function 变量。

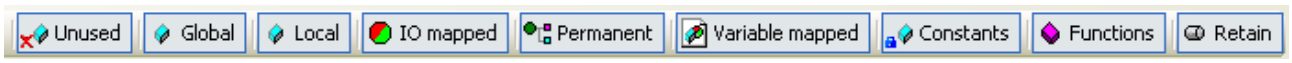

图 29 Cross Reference 条件筛选

（10）OPC

生成两个配置：1. Windows OPC server 的 OPC 配置；2. Automation Runtime OPCserver 的 OPC 配置。

1) 在 Logical View 中添加新的 OPC tag declaration。

如图 30 所示，可以添加一个 Package 后并在内添加 OPC tag declaration。

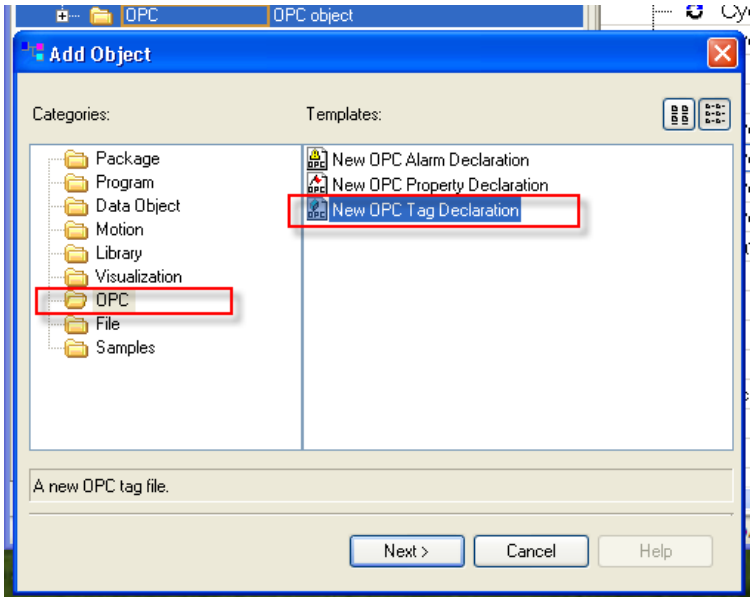

图 30 添加 OPC tag declaration

2) 编辑 OPC tag declaration

添加需要进行 OPC 通讯的变量，如图 31 所示。如操作 1 双击声明，并添加在程序中的变量，点击 OK 保存设置。

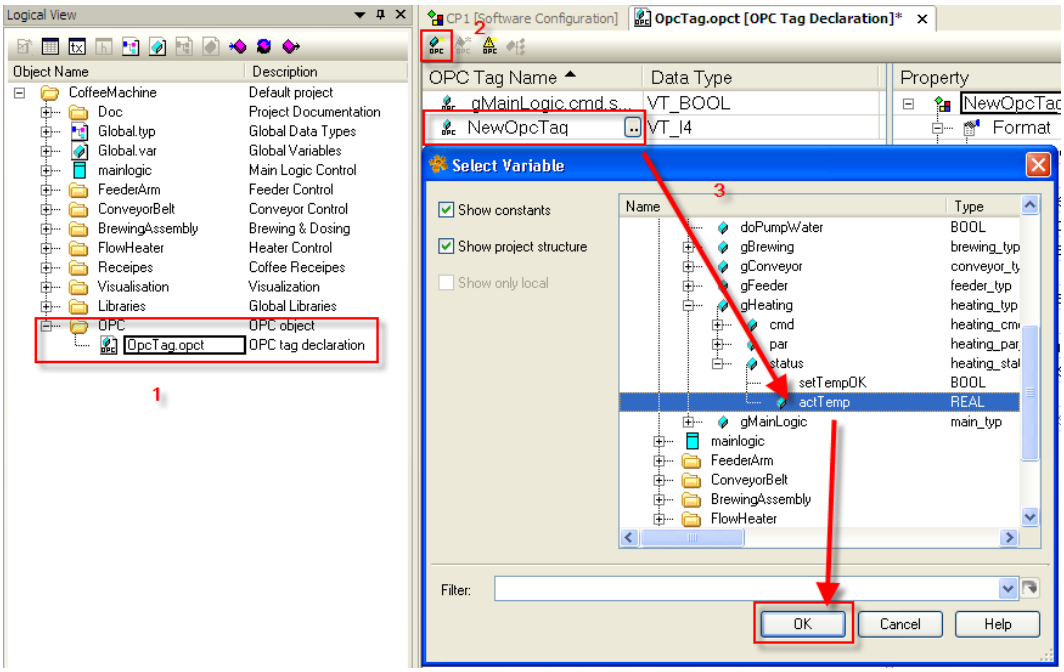

图 31 添加 OPC tag declaration 对象

3) 创建 OPC Mapping

在 Configuration View 中添加 OPC tags 和 Windows OPC server 的 Mapping。这个 Mapping 的作用是 OPC server 的属性。需要在 Configuration View 中添加 OPC Windows Server Mapping。见图 32.

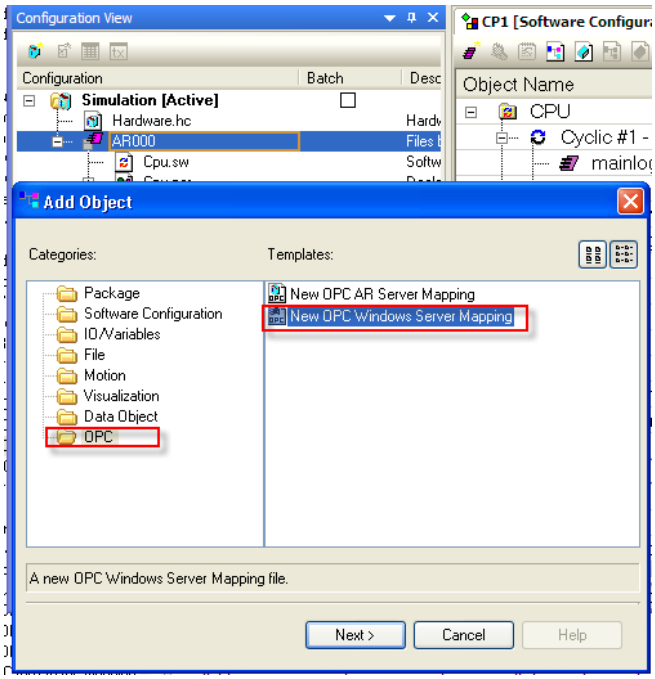

图 32 添加 OPC Mapping

4) 编辑 Mapping 对象，连接

打开 Configuration View 中之前新建的 OPC Windows Server Mapping，将 Logical View 中的 Opc tag declaration 拖至 Mapping 中，并修改 Communication 路径，因为此时是通过仿真器连接的，修改目标地址和端口号。见图 33 的设置和修改。

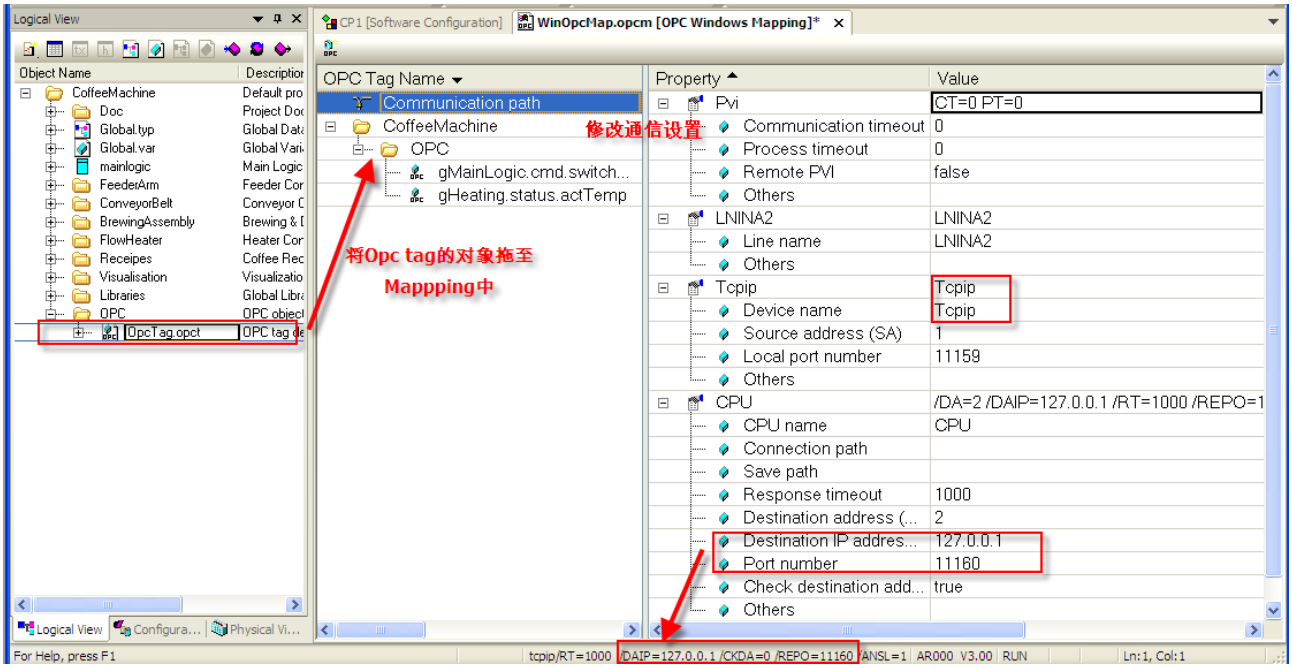

图 33 设置 OPC Mapping

5) 编译，在目标系统中进行 OPC 配置

编译之后，将生成的后缀名为.opcs 的 OPC server 的配置文件安装在电脑的 Windows OPC server 运行的路径下。

（项目生成的后缀名为.opcs 的文件路

X:\XXXX\CoffeeMachine\Temp\Objects\Simulation\AR000\winOpcMap.opcs

其中文件 WinOpcMapp.opcs 的名称即为在 Configuration View 中创建的 OPC Windows Server Mapping 名称一致。）

将这个文件拷贝到 AS 的 PVI 的 Bin 路径下，如 1 标出，并打开 2 处的文件 BR.OPC.Server，将 3 处修改为拷贝的 winOpcMap.opcs.

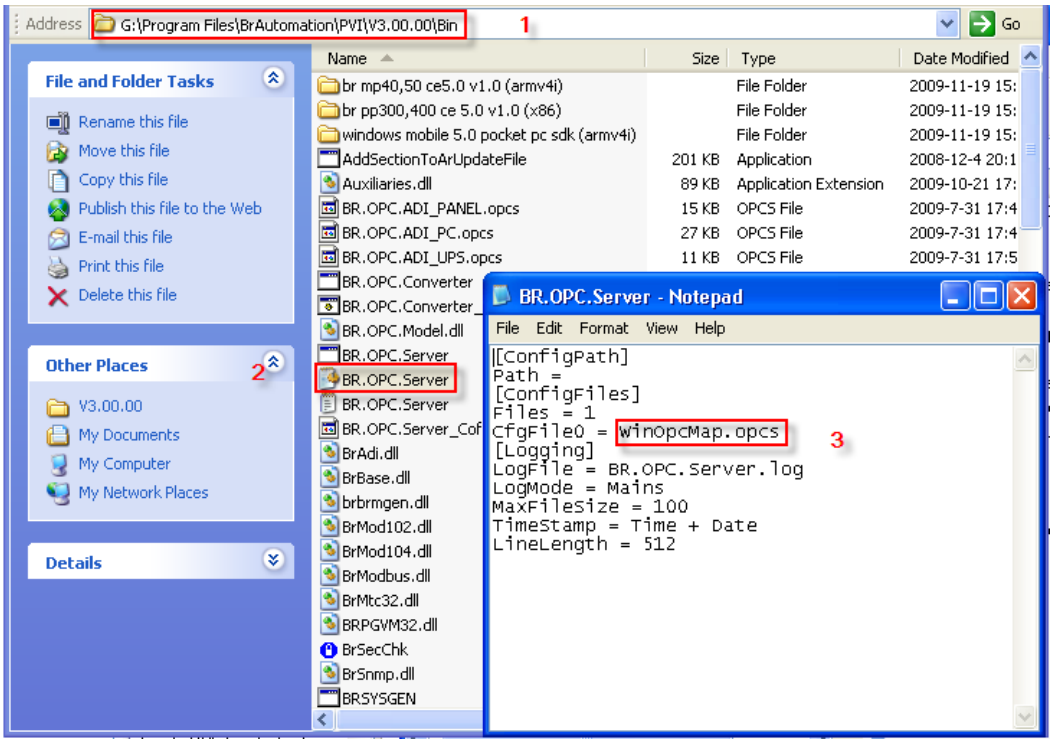

图 34 在目标系统中配置 OPC 文件

6) 测试 OPC

打开 OPC 测试软件路径，B&R Automation--→PVI3.0.0---→PVI Developer---→Dianostic Tools--→OPC Monitor。

如图 35 所示的操作，在空白处 1 右键 Add，出现 Add Server 的子窗口，通过 Search 功能查找 PVI Server，选择相应的链接。

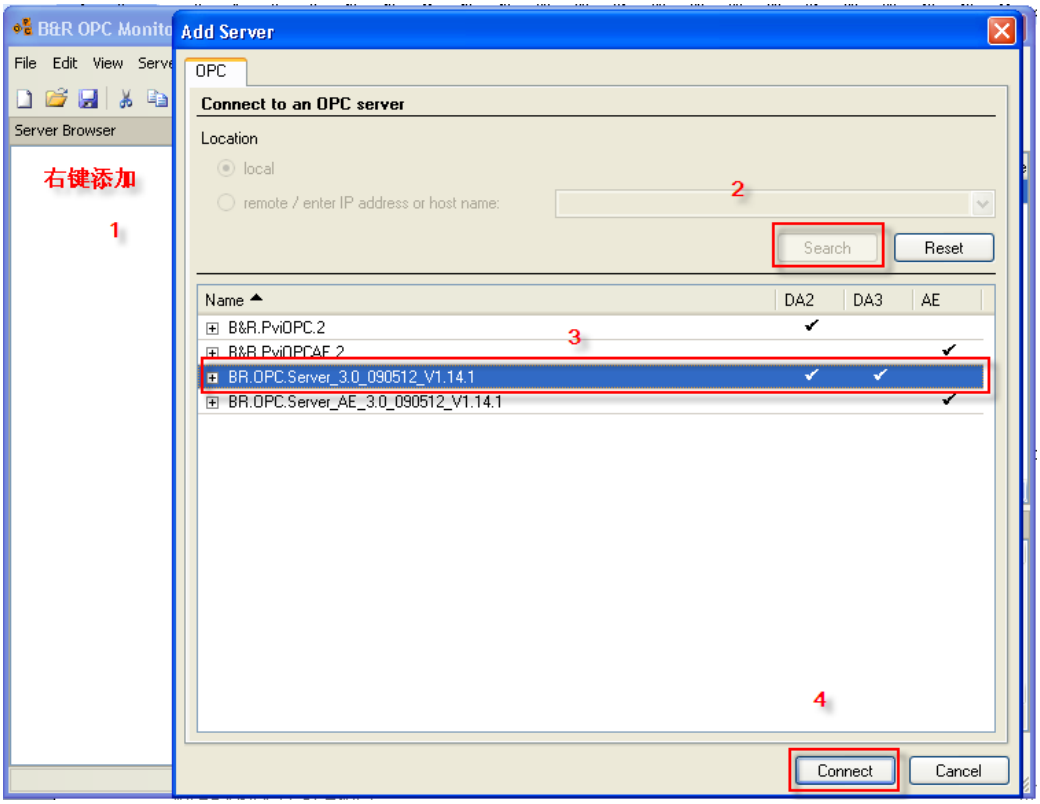

图 35 OPC 链接

将在 Items 中的内容拖至右边窗口，可对变量进行读写操作。如图 36 所示。

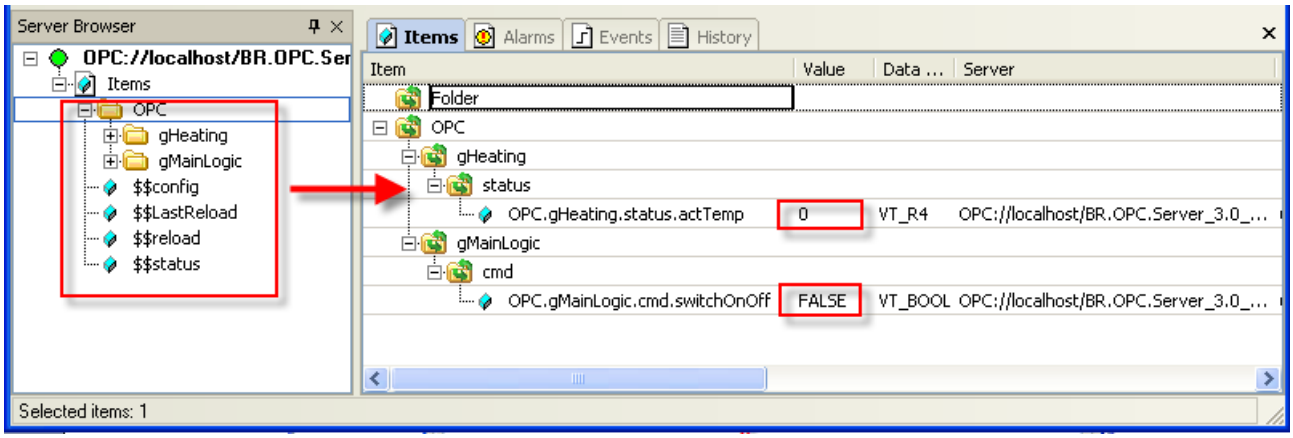

图 36 读写变量

四、 编程语言

图形化语言：LD、FBD、CFC

图形化+文本：SFC

文本化：IL、ST、AB、C

对于文本类型的语言，使用相同的编辑器。所以诊断工具的操作是相同的。可以方便使用。

在 AS2.X 中，使用 C 语言时，需要添加各种头文件，AS3.0 中将基本的头文件都包括了，不再需要添加.h 和.a 文件。如果使用其他的子函数的文件，依然需要添加编译路径。

使用 C 语言，依然可以将变量定义在文件中。在 Project---→Settiing 修改图 37 的设置。

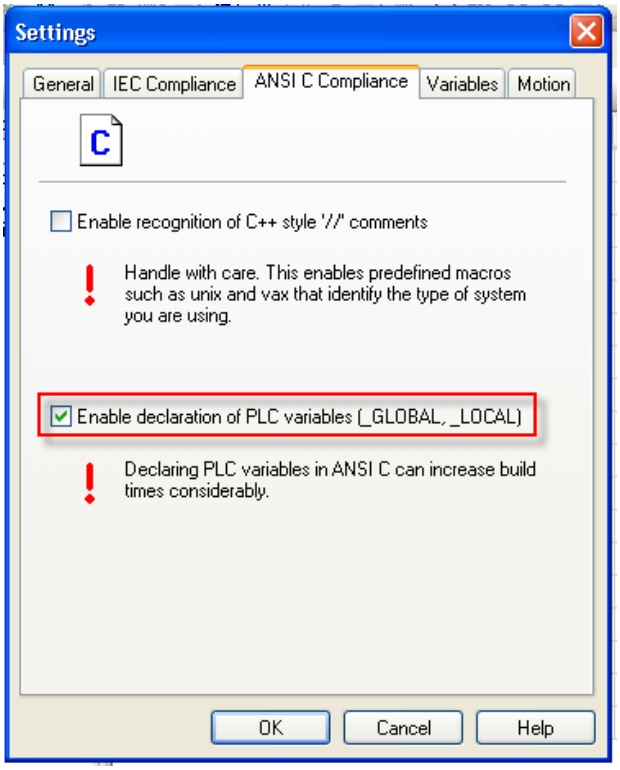

图 37 C 语言设置

五、 仿真器

（1）伺服仿真功能

使用仿真器 AR000，可以添加仿真的伺服和电机。图 38 显示了添加虚拟的伺服和电机。图中区域 1 为 M01 接口，右键添加 MotionIF 卡，之后在此接口上添加伺服和电机。

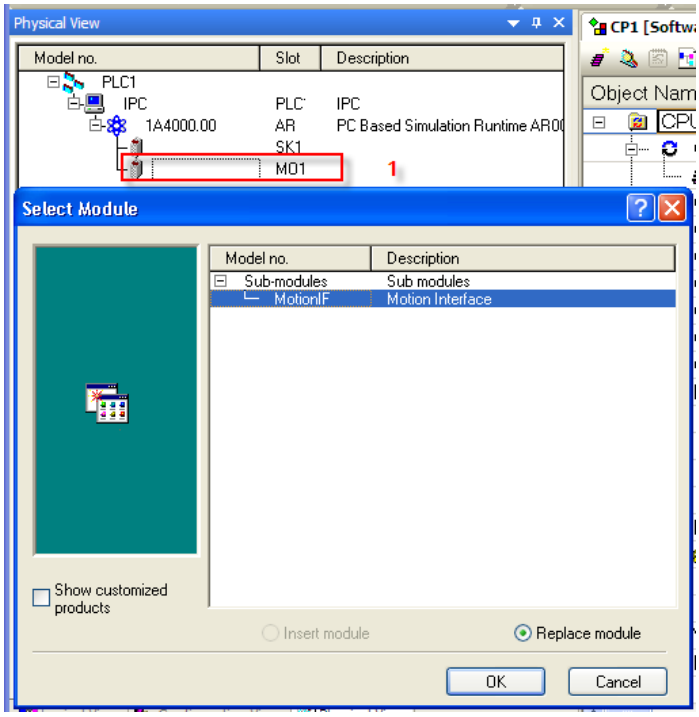

图 38 添加仿真伺服和电机

添加完毕后，将自动生成伺服所需的基本元素：初始化轴参数、参数表、错误文本以及相应的伺服配置文件。见图 39 自动添加的元素。Logical View 中在全局变量区域 1 自动添加了全局轴结构体变量；区域 2 的 Package 自动生成初始化轴参数表、参数表、错误文本；在 Configuration View 中生成关于伺服配置的各种配置文件。

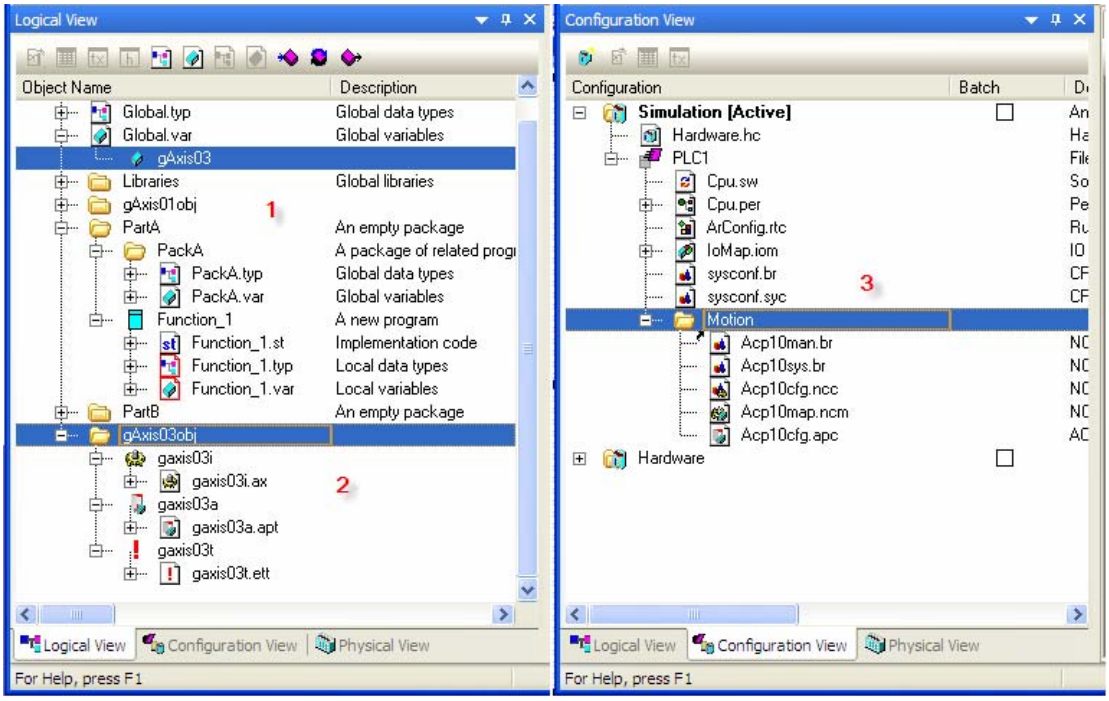

图 39 自动生成伺服组件

（2）可以同时打开多个 AR000