SIEMENS

SIMOTION 运动控制器
基础应用

2010 / 03 版

Motion Control

Application

前　言

针对运动控制与伺服驱动器的基础应用，西门子（中国）运动控制应用中心编写了全套共 3 本手册。分别为：

- 应用基础手册 1 —— 驱动与控制基础
- 应用基础手册 2 —— SINAMICS S120 驱动器基础应用
- 应用基础手册 3 —— SIMOTION 运动控制器基础应用

该手册为全套手册的第三部分，介绍了西门子 SIMOTION 运动控制器的基本组态、调试、编程与常用运动控制命令。

本系列手册仅适用于西门子内部工程师交流学习及相关客户培训之用。由于编者水平有限，手册中难免出现纰漏之处，恳请广大同事能够不吝赐教。

SLC I DT MC APC

Solution Team

2010-03

目 录

一、SIMOTION 概述

1.1 应用背景

现代科学技术的不断发展给机械制造行业带来了机遇的同时也带来了更多的挑战。即使最先进的机器也必须不端满足更高的要求，必须应对诸如高产品质量、循环率不断提高的最高程度的生产能力和最低寿命周期成本的挑战。因此电子元件正在逐步取代机械部件，不仅如此，控制系统还必须承担更多复杂的处理任务，控制更多的轴，必须应对更短的创新周期，跟上快速变化的市场需求步伐，另外在满足高效率、高质量的同时还必须尽量降低成本、控制机器价格。

应对以上所有挑战，西门子给出了一个统一的解决方案，即新一代的运动控制系统 SIMOTION。

SIMOTION 作为一个单一的系统，集运动控制、逻辑控制与工艺控制功能于一身，能够最大程度简化工程系统的开发与调试时间，同时还能保证较高的循环率和最高的产品质量。模块化的设计顺应了模块化机器概念的趋势，使用 PROFIBUS 和 PROFINET 实现模块之间的通信，使 SIMOTION 运动控制系统具有更大的灵活性。目前，SIMOTION 已广泛应用于印刷、包装、纺织、连续物料加工、金属成型等行业。

1.2 SIMOTION 的系统组成及功能

SIMOTION 运动控制系统由硬件平台、工程开发系统（SCOUT）以及运行时软件模块组成。如图 1.1 所示。

SCOUT 工程系统为 SIMOTION 提供了系统组态、一体化的编程、参数设定、调试及故障诊断工具；运行时软件模块使逻辑控制、运动控制与工艺控制功能融为一体，以工艺对象的形式封装诸如轴、凸轮等具有典型功能的对象，用户可以直接使用；不同的硬件平台使用相同的工程开发系统和运行时软件模块，能够满足用户灵活配置的需要。

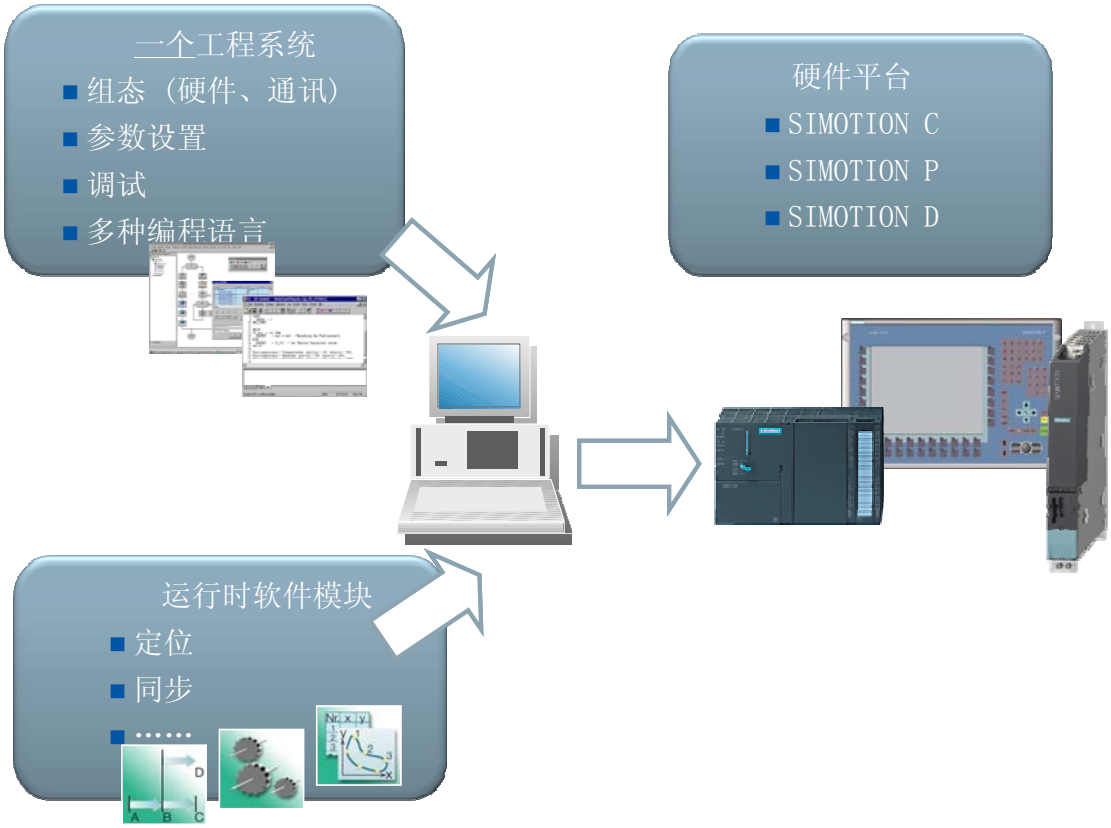


图 1.1 SIMOTION 系统组成

SIMOTION 的主要包括三大功能：逻辑控制功能（PLC）、运动控制功能以及其他工艺功能。SIMOTION 强大的功能性使得对于运动控制要求比较高的场合中，PLC 不再成为必要，它自带的梯形图编程工具极大地方便了用户从使用 PLC 到 SIMOTION 的过渡。对于各种运动控制，SIMOTION 提供可扩展的工艺包，使用户能够根据自己的需要进行选择，从而节省了工程成本。在某些需要温度控制等工艺控制的场合，SIMOTION 也能发挥它的特色，使用 SIMOTION 的工艺功能可以极大地减少用户编写程序的繁琐。

1.3 硬件平台

SIMOTION 有三种硬件平台，即基于控制器的 SIMOTION-C、基于 PC 的 SIMOTION-P 和基于驱动的 SIMOTION-D。如图 1.2 所示。

图 1.2 SIMOTION 硬件平台

SIMOTION 的全部系统功能都可以在每个平台上实现，所有的平台使用相同的开发系统。

1. SIMOTION-D

SIMOTION-D 内部集成 SINAMICS S120 驱动控制单元，特别适合紧凑和模块化的设计。有多种型号可供选择（D410、D425、D435、D445 和 D445-1），最多可带 64 根轴，最小用户任务周期为 0.5ms，最小总线循环时间为 0.5ms。支持 PROFIBUS、PROFINET 及工业以太网等通讯方式。

2. SIMOTION-C

SIMOTION-C 集成数字两输入/输出、模拟量、编码器接口以及步进电机和液压伺服的接口，可以使用 S7-300 的输入输出模块进行扩展，特别适合于生产机械改造。SIMOTION-C 有 C230-2 和 C240 两种规格可供选择，最多可以控制 32 根轴，最小用户任务周期为 1ms，最小总线循环时间为 1ms，支持 PROFIBUS、PROFINET 及工业以太网等通讯方式。

3. SIMOTION-P

SIMOTION-P 带有 Windows XP 和 SIMOTION 实时扩展，特别适合有大量数据处理和 PC 机的应用。SIMOTION-P 支持 IT 通讯，带有 USB、并口、串口以及显示器接口等，最多能带 64 根轴，最小用户任务周期为 0.25ms，最小总线循环时间为 0.25ms，支持 PROFIBUS、PROFINET 及工业以太网等通讯方式。

1.4 SCOUT 工程开发平台

SCOUT 具有友好的人机界面，为用户开发 SIMOTION 工程项目提供一体化的工具，且无缝集成于 SIMATIC 环境中（TIA），目前最新的版本为 V4.1.4 sp1。

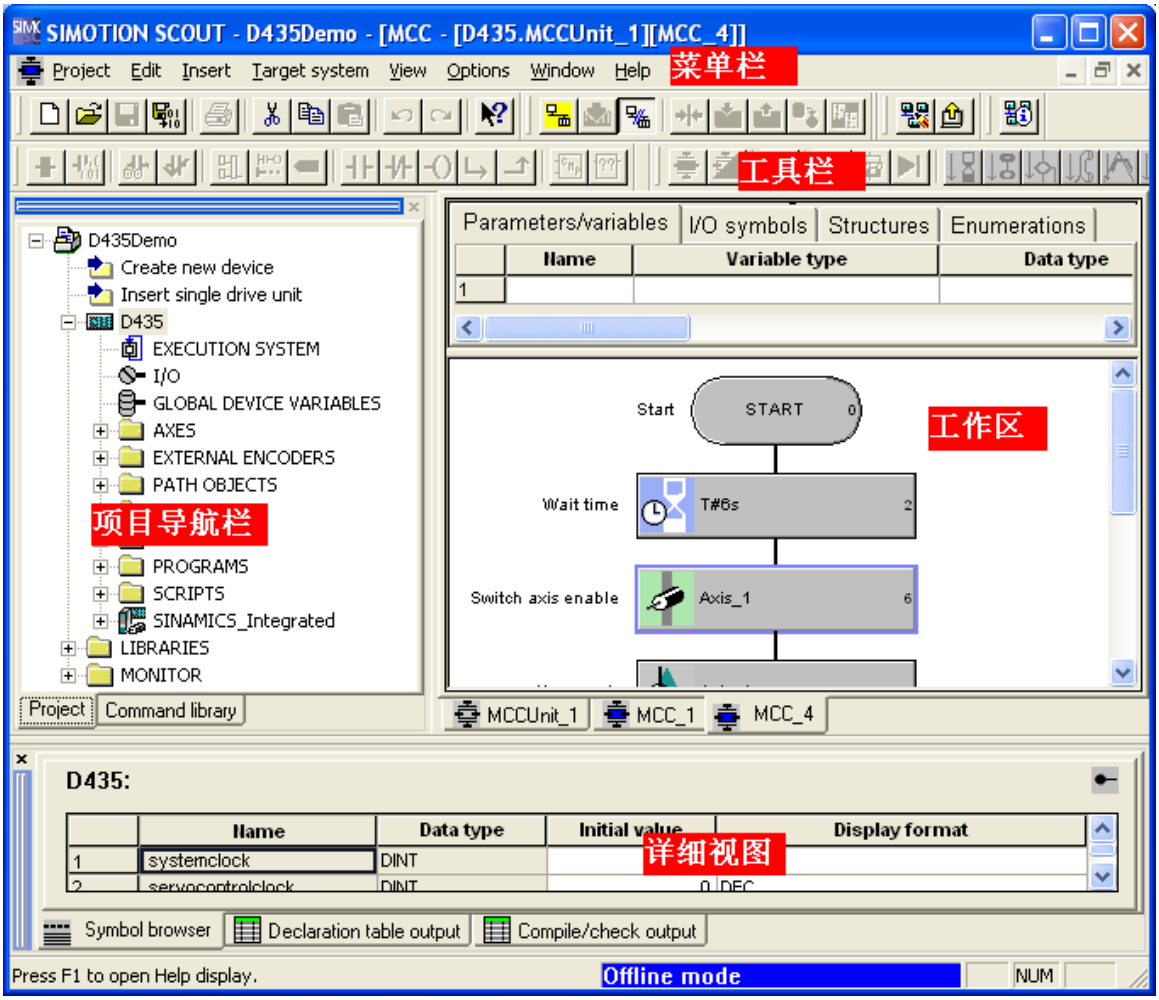


图 1.3 SCOUT 软件界面

使用 SCOUT 开发为自动化生产机械开发 SIMOTION 项目的步骤：

（1）新建一个项目；

（2）在项目下新建设备（SIMOTION D，SIMOTION C，SIMOTION P）；

（3）确定附加组件，如 I/O 等；

（4）系统组态，包括硬件组态和网络组态；

（5）根据向导组态工艺对象；

（6）编程（MCC，ST，LAD/FBDDCC）；

（7）下载项目导 SIMOTION 设备；

（8）测试。

SIMOTION 用工艺对象来描述具有典型功能性的集合，如轴、凸轮等。在 SCOUT 的项目导航栏中工艺对象以“文件夹”图标列出在树形结构中，用户可以通过“文件夹”下的向导组态工艺对象。关于 SCOUT 的使用，在以后章节中将有更多的描述。

二、系统组态

本章将基于图 2.1 所示的系统结构介绍 SIMOTION-D 运动控制系统的基本组态与调试。

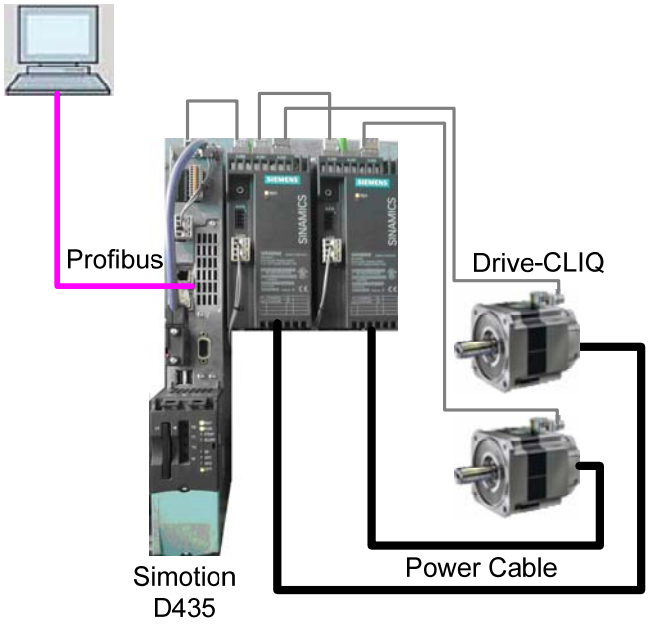


图 2.1 系统结构图

2.1 新建项目

启动 Scout 软件，选择 Project->New 创建一个新的项目：“D435Demo”。如图 2.2 所示。

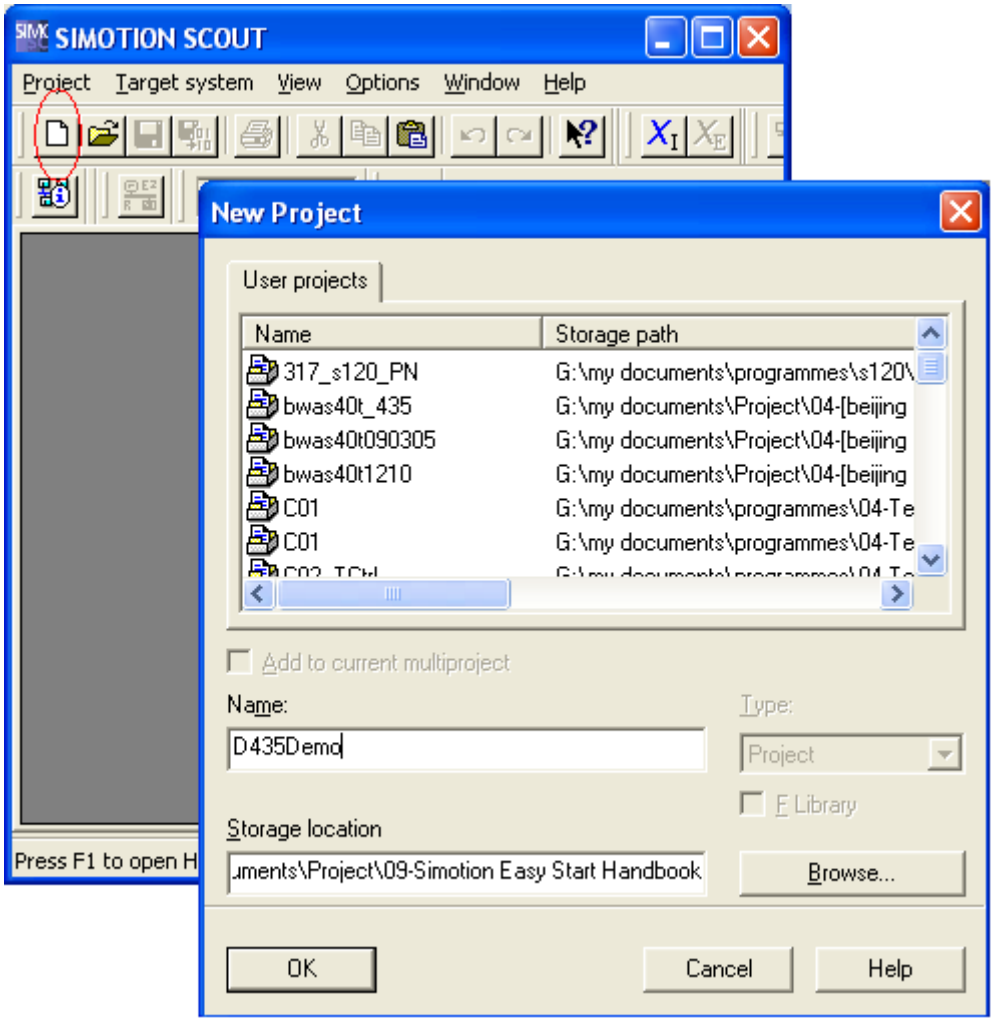


图 2.2 新建项目

2.2 插入设备

插入一个新的设备，选择 D435，如图 2.3 所示。

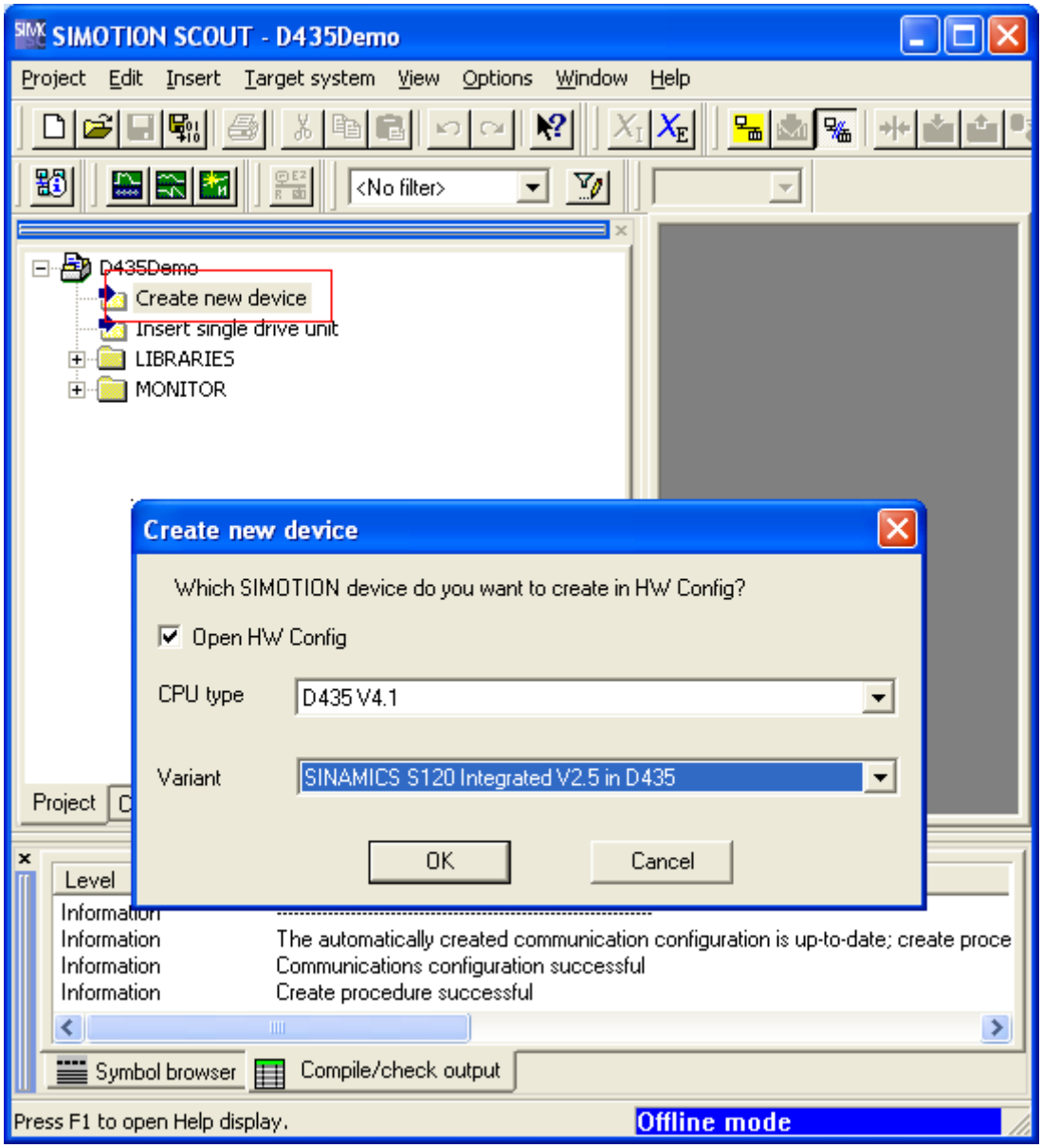


图 2.3 插入 D435

2.3 建立连接

2.3.1 设置通讯接口

选择 Profibus DP2/MPI (X136)，再根据 PG/PC 所用通讯接口选择 CP5512 或者 PC Adapter。如图 2.4。

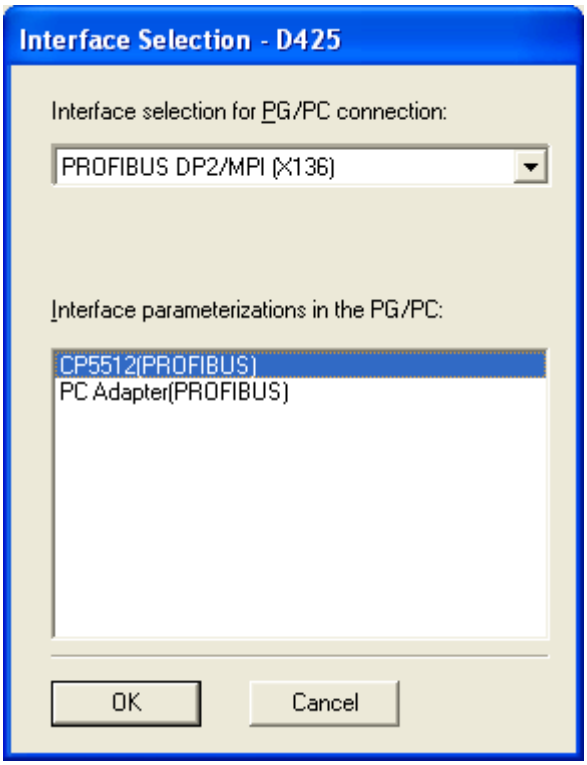


图 2.4 设置通讯接口

2.3.2 配置 SIMOTION 上 DP 接口参数

自动打开硬件组态界面，配置 DP2/MPI 通讯接口，这里设置 DP 地址为 2。如图 2.5。接口的具体参数如波特率、最高站地址等可以通过点击“Properties..”按钮进行设置。缺省方式下，波特率为 1.5Mbps。

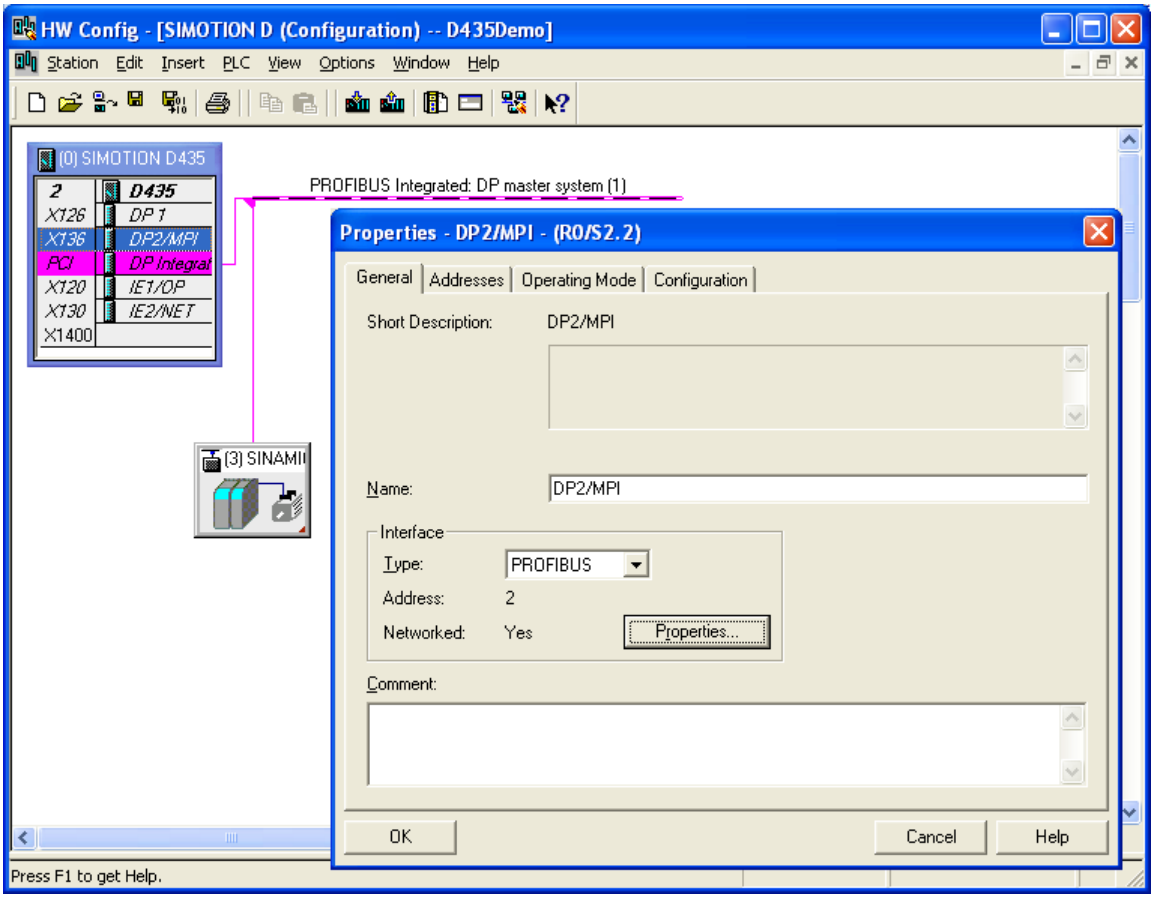


图 2.5 配置 DP 接口

2.3.3 配置 PG/PC 的通讯方式

进入 PC 控制面板，双击打开 Set PG/PC interface。配置 PG/PC 通讯方式为 CP5512(Profibus)。双击列表中的 CP5512(Profibus)打开其属性画面，在这里可以设置 PG/PC 的 DP 地址、波特率等，其配置须与 SIMOTION 中的 DP2/MPI 接口配置参数一致。如图 2.6 所示。

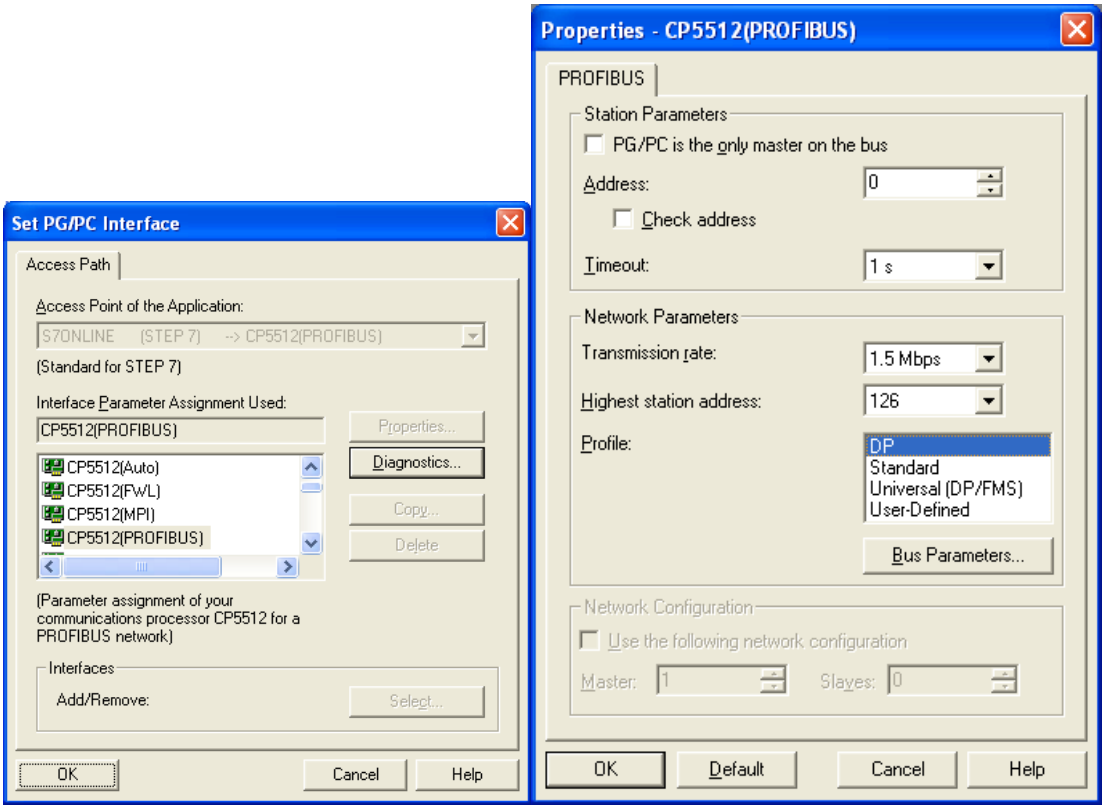


图 2.6 配置 PG/PC 通讯接口

2.3.4 激活至 SINAMICS_Integrated 的路由

打开网络组态，若此时 PG/PC 与网络的连线为紫色，则 Scout 不能对 SINAMICS_Integrated 进行在线，因此需要开启 PG/PC 至 SINAMICS_Integrated 的路由。

双击 PG/PC，在 Assignment 属性页中，点击 Assigned 列表中的 PROFIBUS Interface，然后将右边 Acitive 边上的方框打上钩，点击 OK 确认即可(如图 2.7)。此时可以看到网络组态中 PG/PC 与网络的连线变为黄色，表示至 SINAMICS_Integrated 的路由已经开启。

设置完成后，编译保存项目并关闭网络组态。

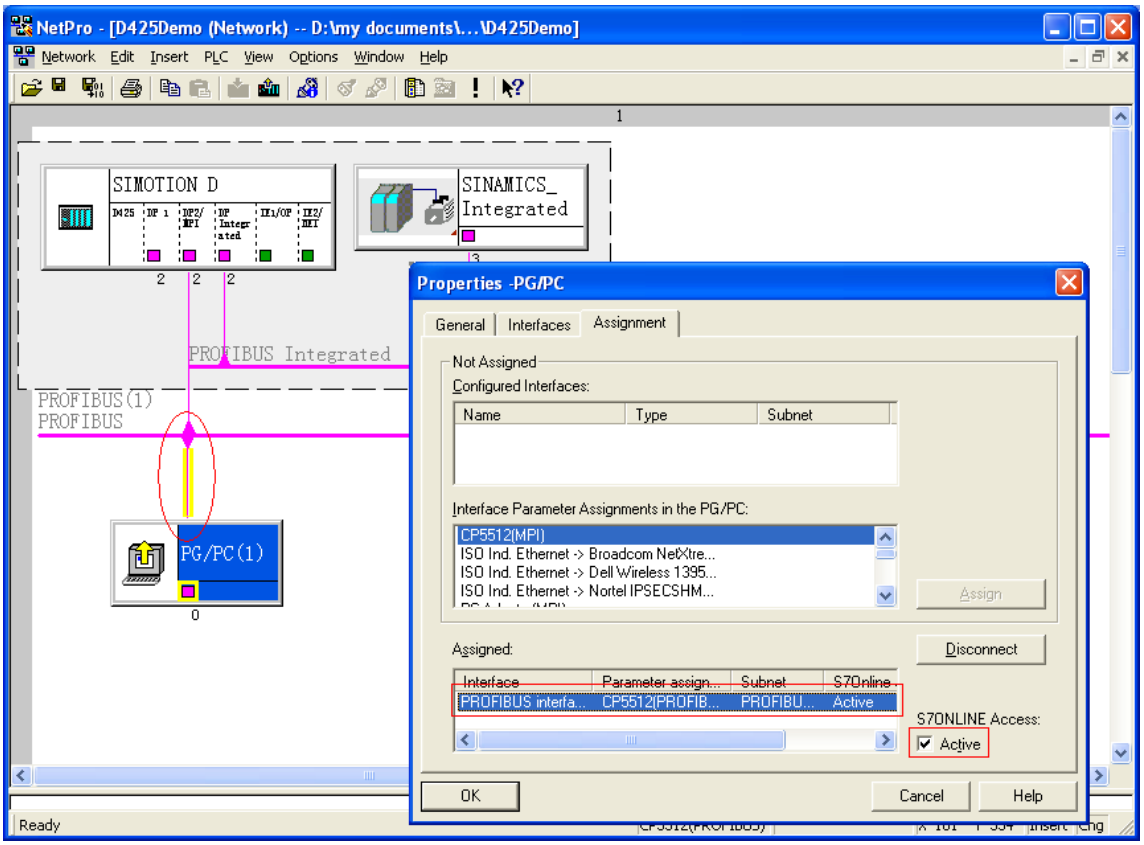


图 2.7 激活至 SINAMICS_Integrated 的路由

2.3.5 下载硬件组态

打开硬件组态窗口，编译保存并下载 SIMOTION D 硬件组态，如图 2.8 所示。

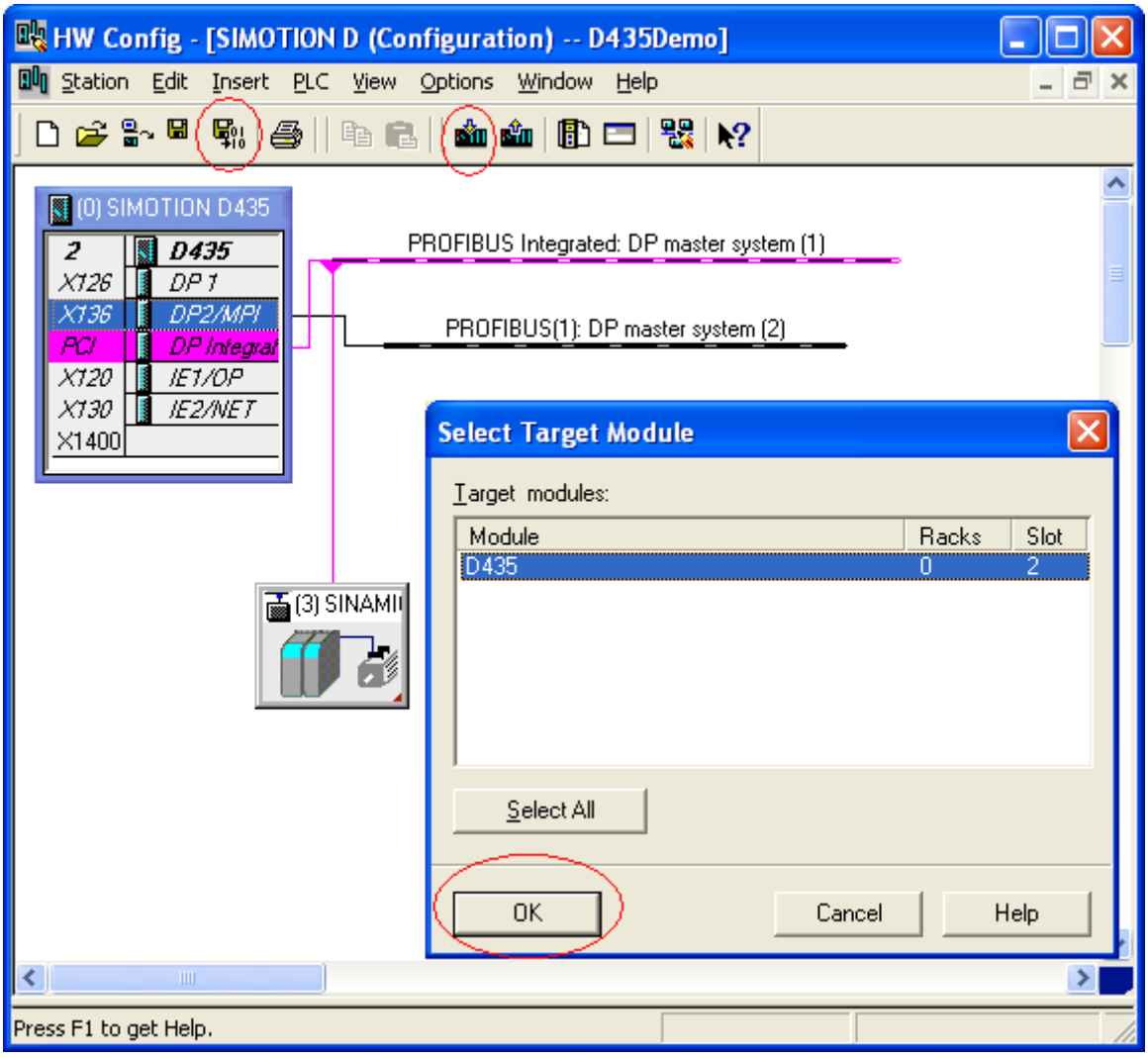


图 2.8 编译、保存并下载硬件组态

下载完成后将弹出提示对话框，选择 No， 不重新启动。如图 2.9 所示。SCOUT V4.1.4.1 以后的版本将不再出现该对话框。

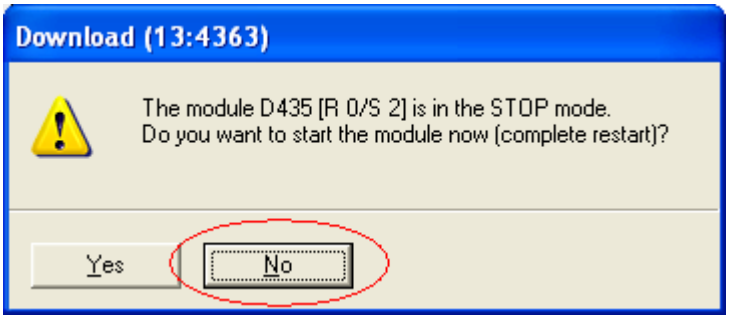


图 2.9 下载提示对话框

2.4 项目配置与调试

2.4.1 恢复 SINAMICS_Integrated 至工厂设置

等待 Simotion 的 Ready 指示灯变绿之后，点击在线按钮，Scout 将处于在线状态。选择左边导航栏中的 SINAMICS_Integrated，点击工具栏按钮，将 SINAMICS_Integrated 恢复工厂设置。如图 2.10 所示。

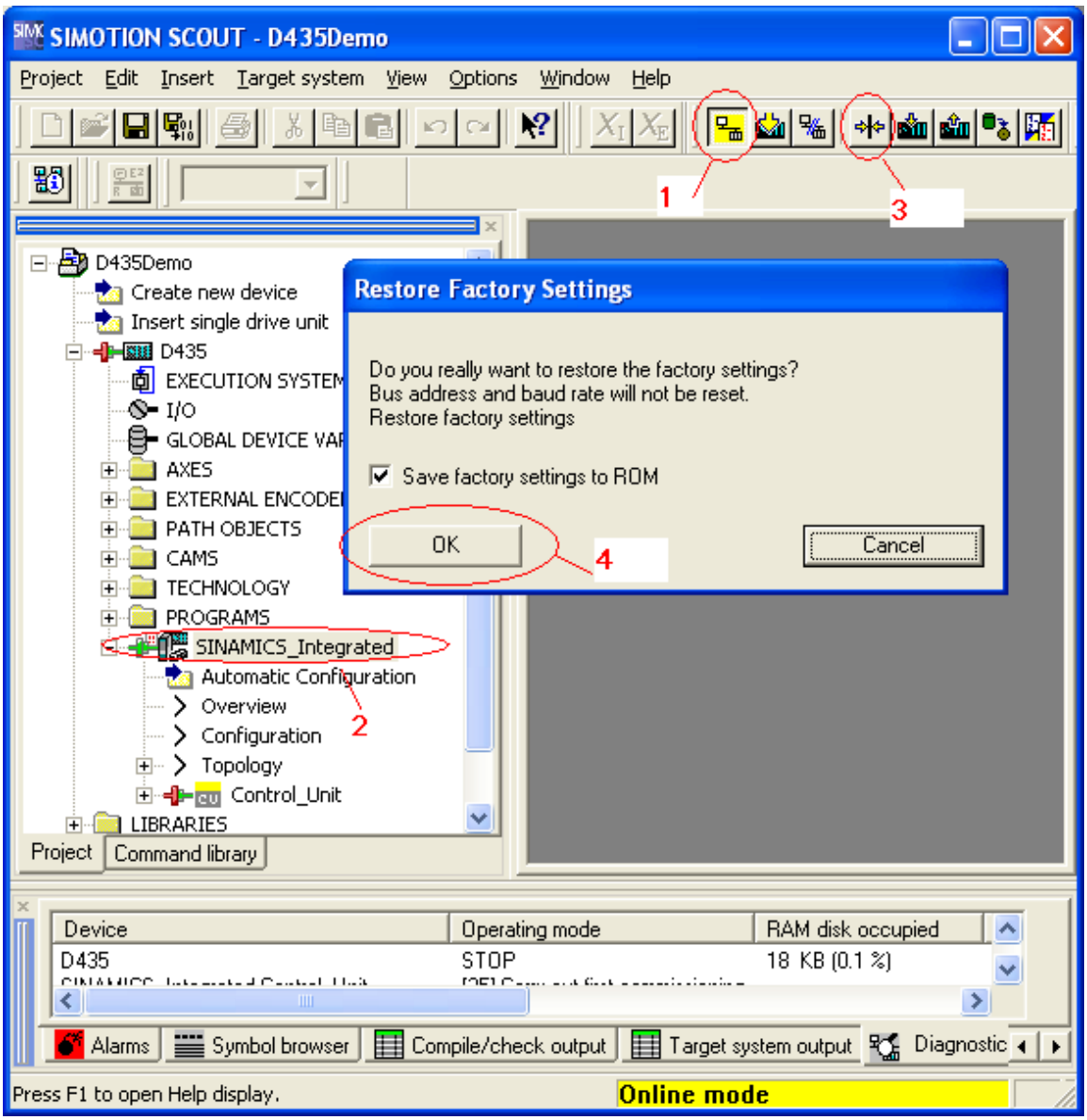


图 2.10 SINAMICS_Integrated 回复工厂设置

2.4.2 在线自动配置驱动

双击 Automatic Configuration，弹出自动配置对话框，点击 Load to PG（Upload）按钮。

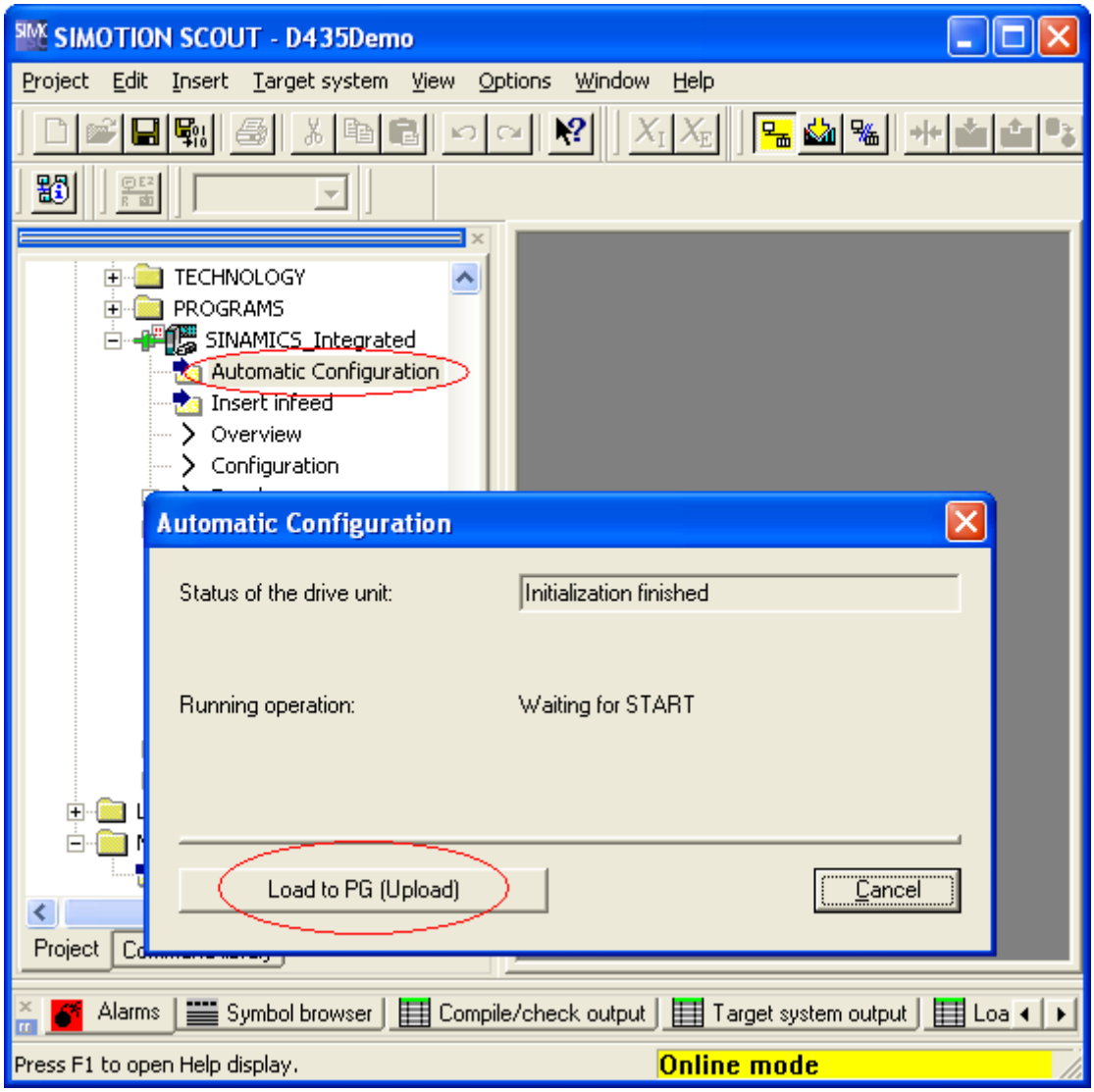


图 2.11 自动配置 Drive

自动配置过程中，弹出 Drive 类型组态对话框，这里选择 Servo 方式。如图 2.12 所示。

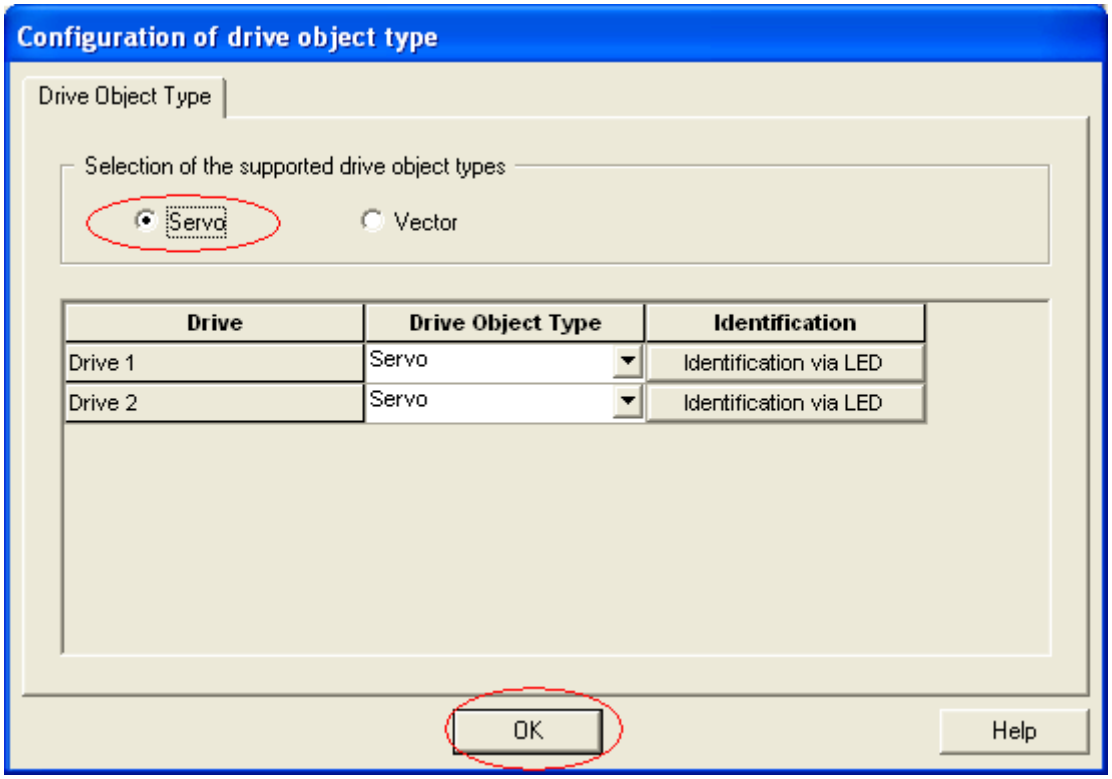


图 2.12 Drive 类型组态

自动配置完成后，点击 Close 按钮（如图 2.13），完成 Drive 的配置。

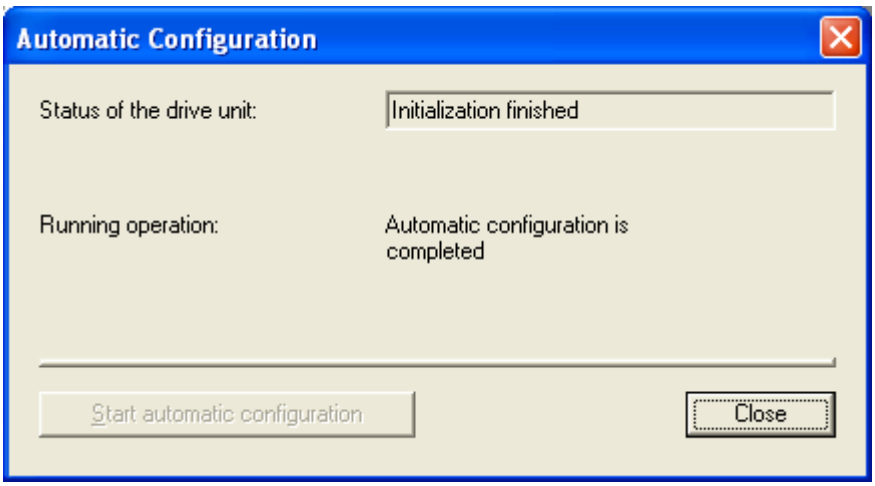


图 2.13 完成 Drive 配置

2.4.3 手动配置报文

为了与 Simotion 通讯，在自动组态完成后需要手动为 SINAMICS_Integrated 配置报文。伺服方式下常用的报文可以选择 105 报文。如图 2.14 所示。注意，配置报文前先将 Scout 离线。配置完报文后，点击 Transfer

to HW Config 将配置的报文信息传至硬件组态中，然后重新编译下载硬件组态（若硬件组态界面已经被关闭，双击设备名称“D435”可将其打开）。

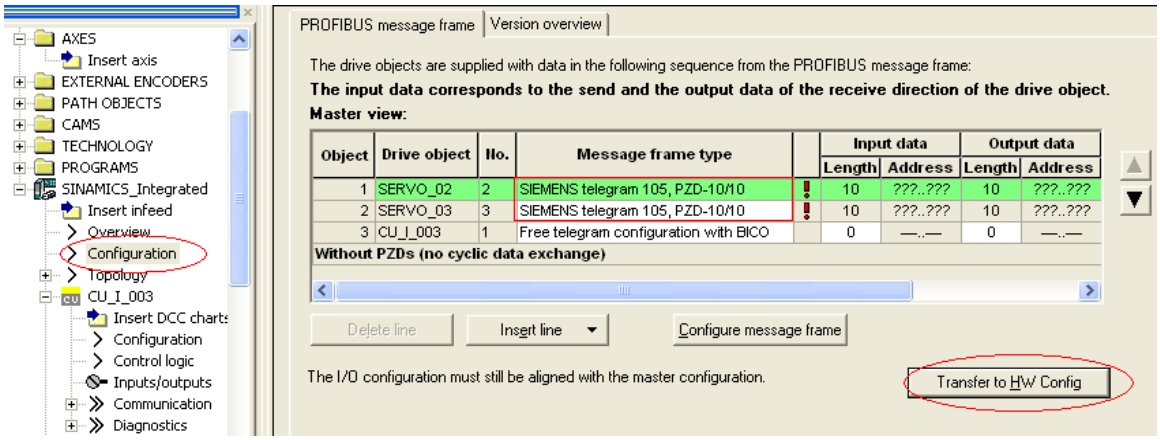


图 2.14 报文配置

2.4.4 轴配置

按照图 2.15~2.18 配置轴（没有插图的步骤按照缺省设置，按 Continue）。

双击 Insert Axis，在弹出的对话框中自定义轴的名称，并根据需求选择轴的控制类型（速度方式、位置方式、同步方式和差补方式），如图 2.15 所示。此时 Scout 仍然处于离线状态。

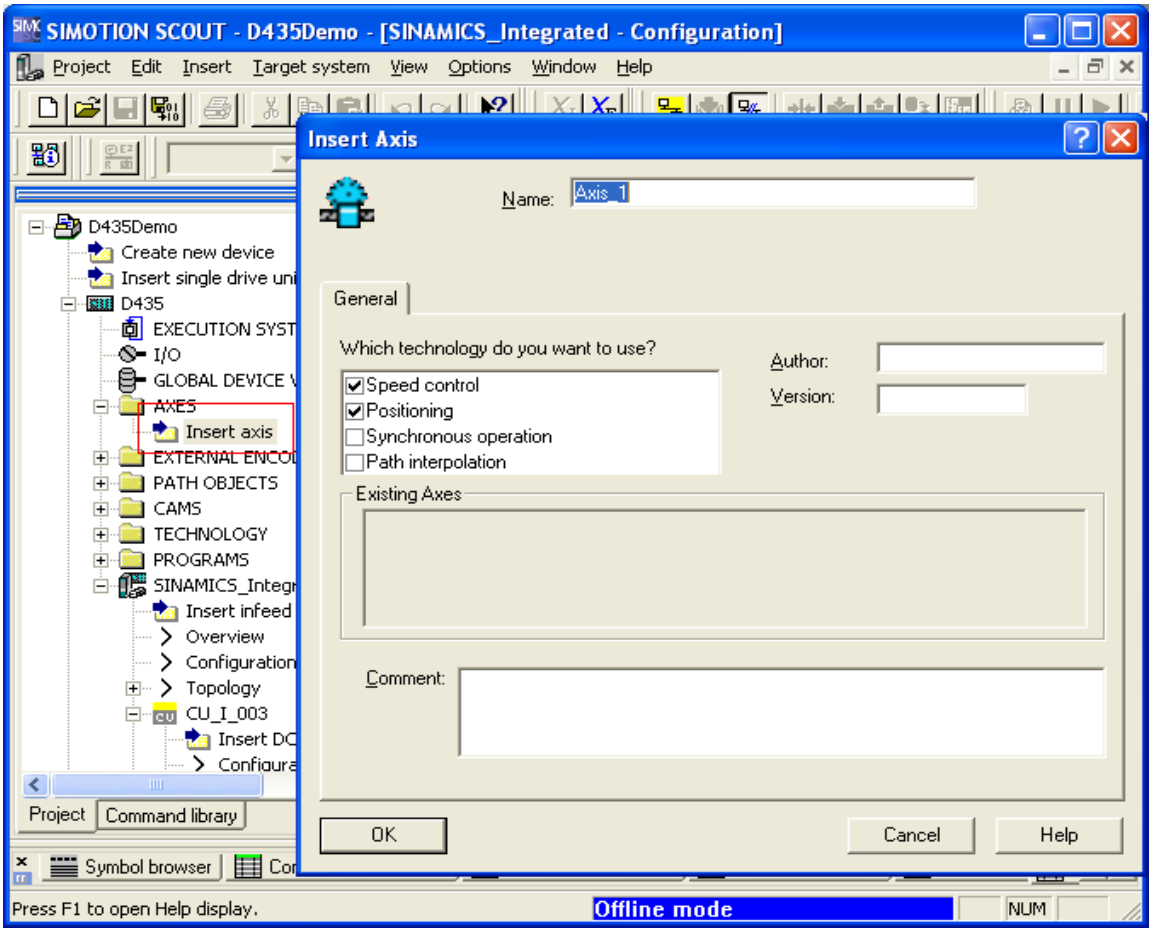


图 2.15 插入轴

在轴类型选择对话框中可以选择旋转轴或线性轴，电气轴、液压轴或虚拟轴。如图 2.16。

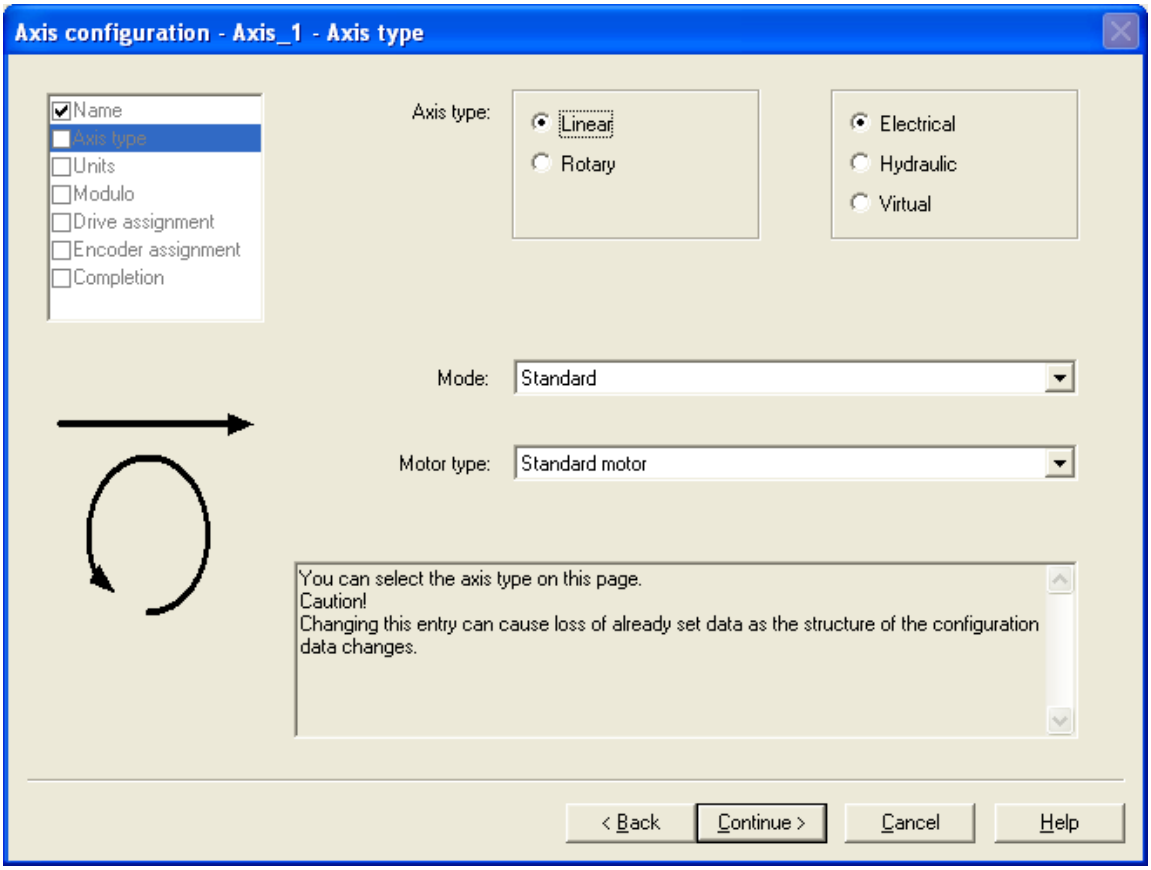


图 2.16 轴类型选择

选择轴的驱动单元及数据传送的报文格式，点击“Data Transfer from the drive”按钮可将驱动器相关参数传送过来。如图 2.17 所示。

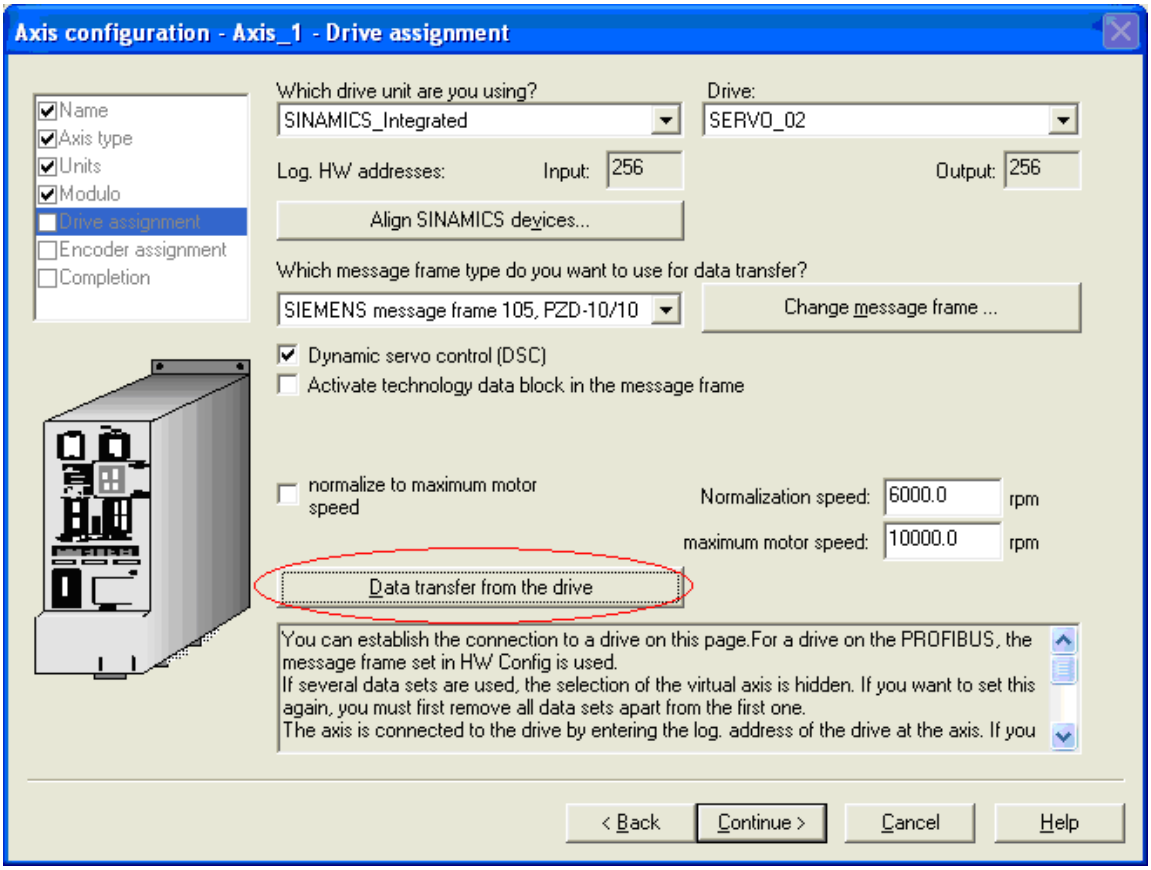


图 2.17 选择轴的驱动单元

对编码器进行配置，点击“Data Transfer from the drive”按钮可将编码器相关参数传送过来。如图 2.18 所示。

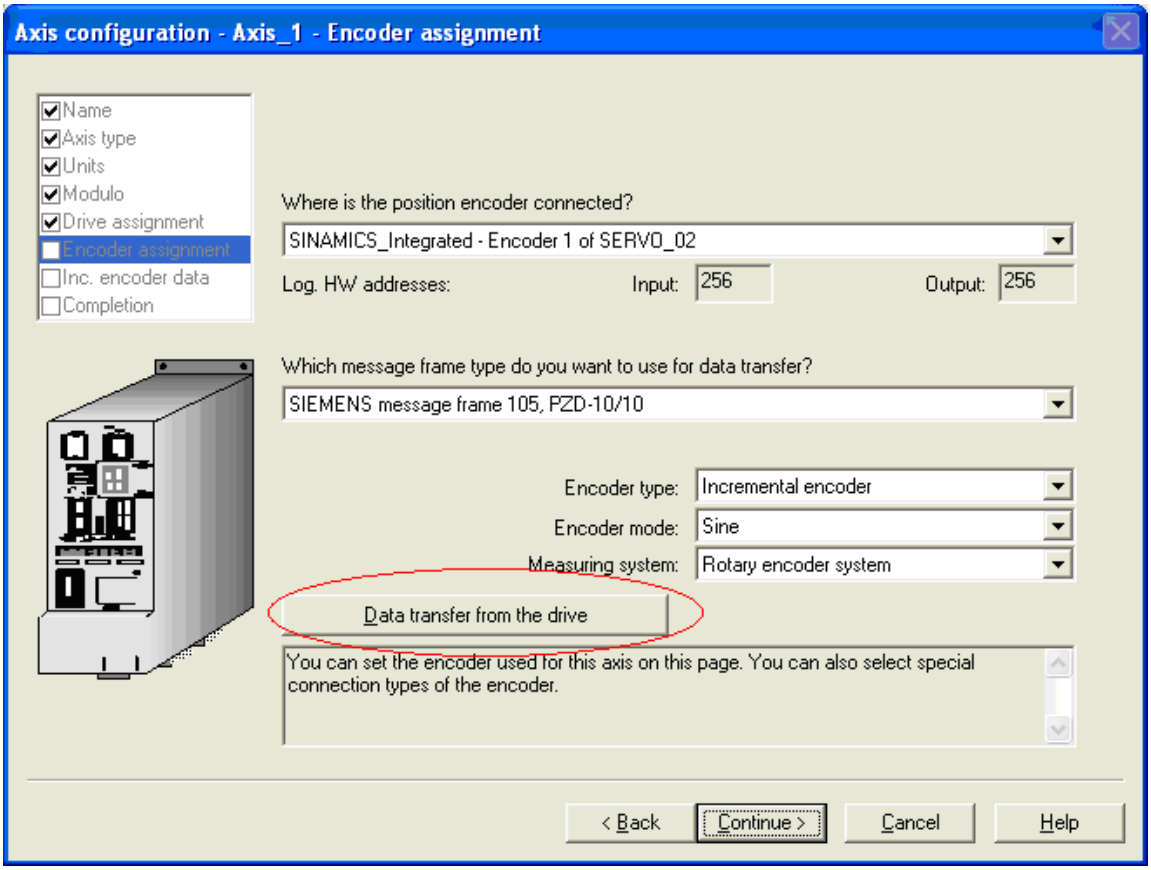


图 2.18 配置编码器参数

直到最后一步点击“finish”即完成轴的配置，可在 Scout 项目列表中看到刚刚配置好的轴 Axis_1， 如图 2.19 所示。

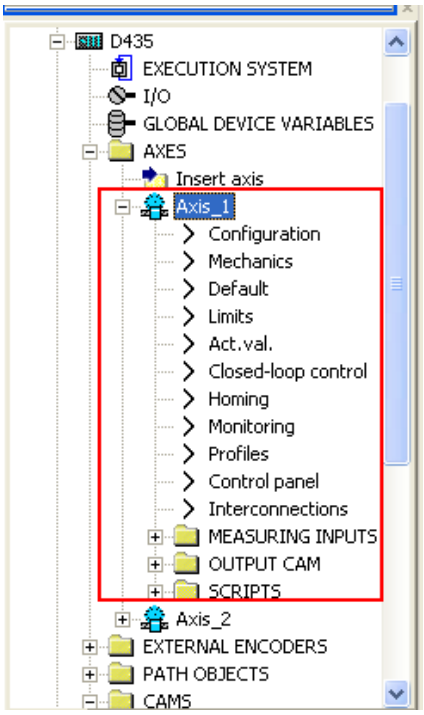


图 2.19 已配置的轴

按照同样的步骤配置同步轴 Axis_2， 在轴控制类型中选择 Synchronous Operation，如图 2.20 所示。

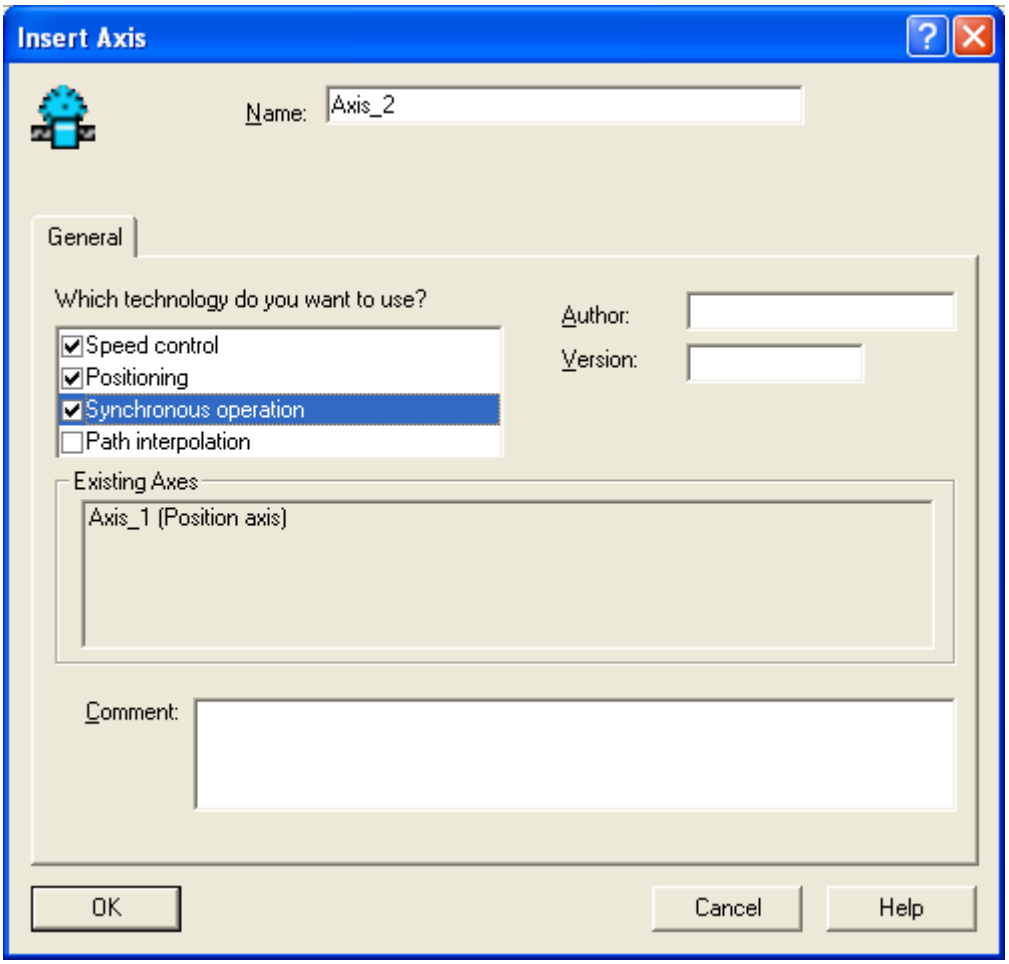


图 2.20 配置同步轴 Axis_2

2.4.5 下载整个项目

编译保存整个项目，按图 2.21 所示顺序将项目下载到设备中。当设备名称前面的图标出现红色（）时，表示离线配置和在线不一致，需要进行上载或下载操作。

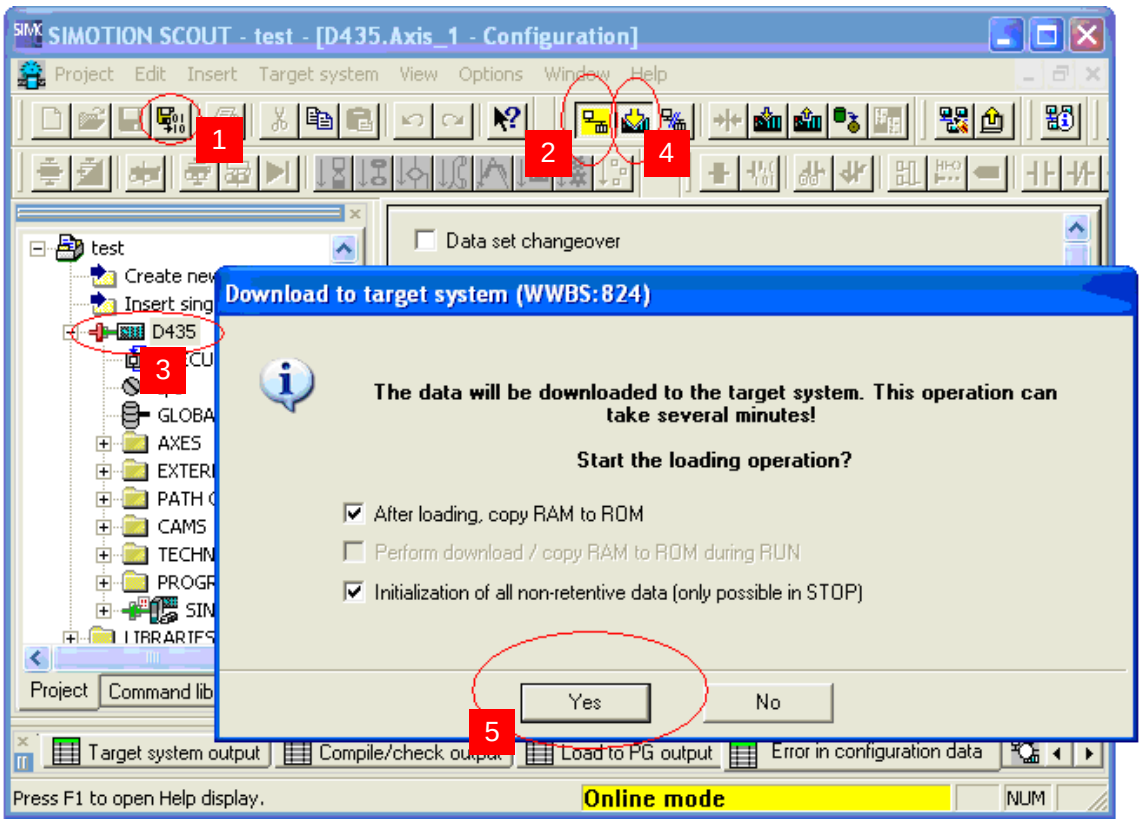


图 2.21 下载项目

2.4.6 使用控制面板调试轴

双击 Control Pannel 在屏幕下方出现调试控制面板，如图 2.22。

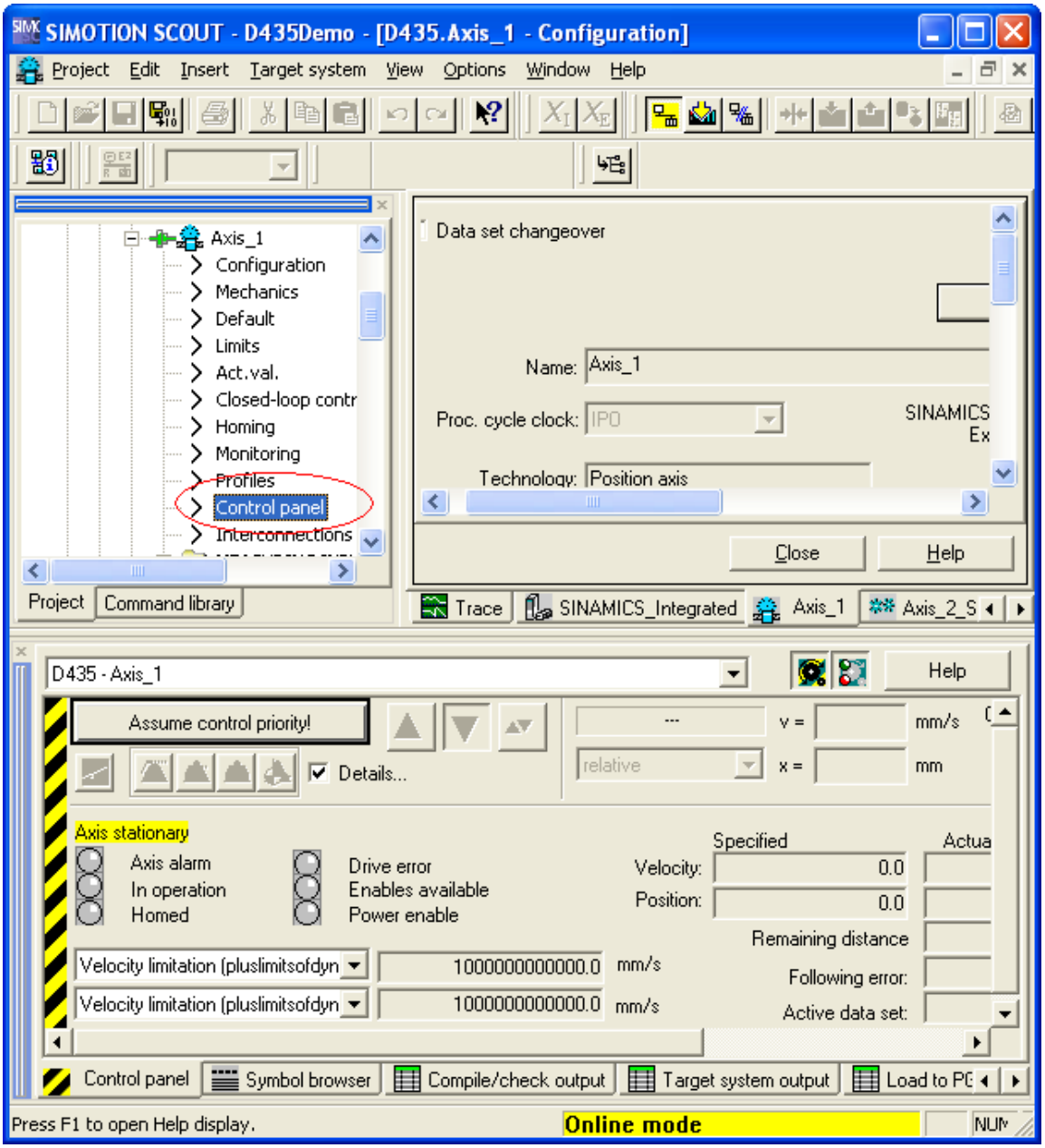


图 2.22 控制面板界面

按图 2.23 所示操作顺序确认控制优先级。

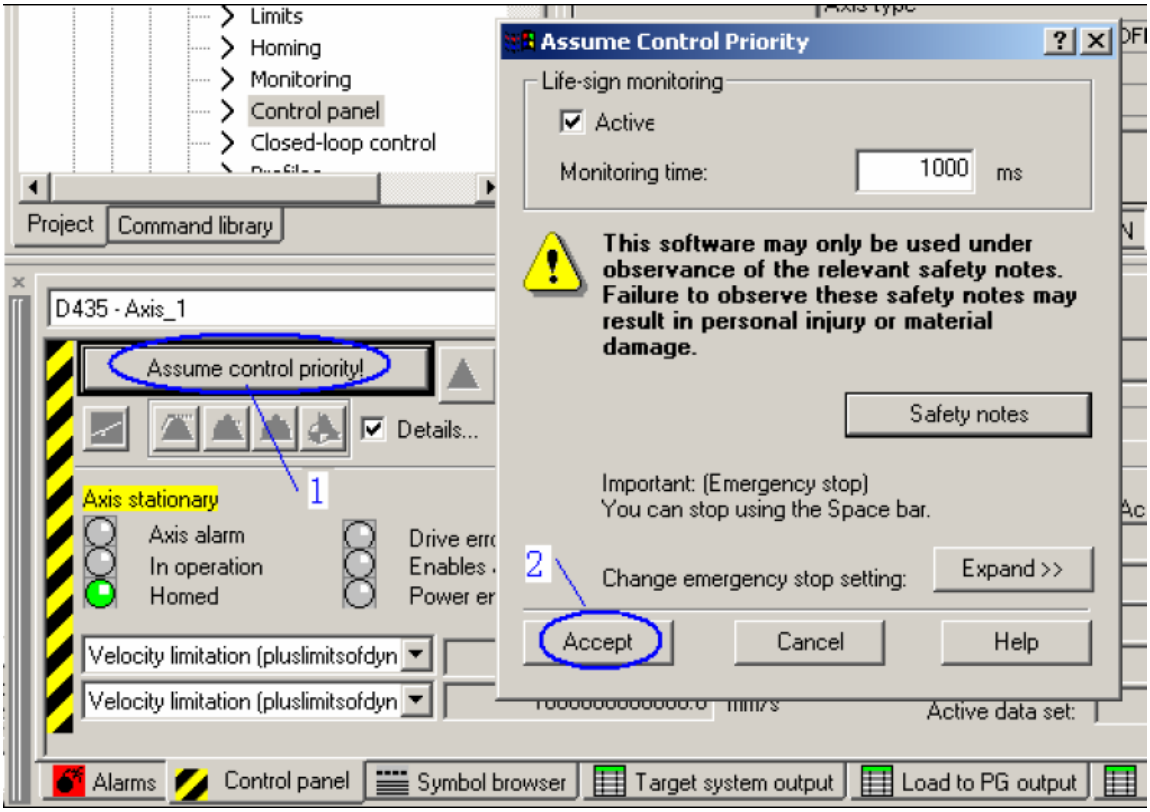


图 2.23 获得控制优先级

按照图 2.24 所示顺序操作，设置参数，电机可以运行。

1）使能 Axis_1。

2）选择一种运行方式。

3）启动轴（Axis_1）运行。

4）停止轴（Axis_1）运行。

5）退出控制面板。

如果通过控制面板进行轴的运行功能测试，轴运行正常证明前面轴的配置正确，否则检查轴配置。作为实际应用，还需要根据实际对轴的“Mechanics”、“Default”、“Limits”、“Homing”进行设置。

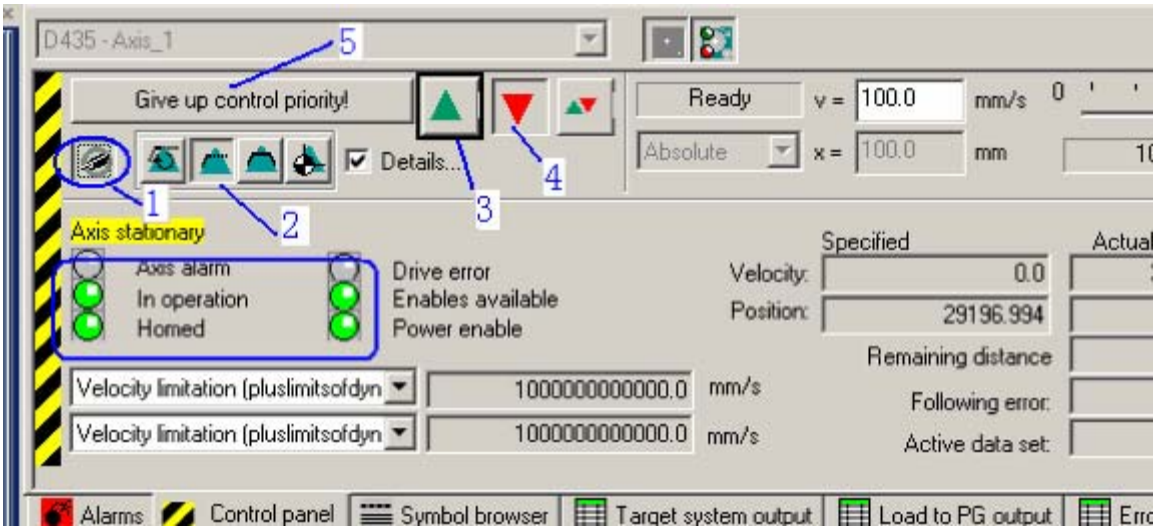


图 2.24 使用控制面板调试轴

以同样方法检测 Axis_2 的配置是否正确。

2.5 编程与测试

下面以定位命令为例介绍如何编写程序来控制轴的运动。

2.5.1 程序结构

SIMOTION 有多种编程方式，其中 MCC 为图形化的编程语言，简单易懂，适用于刚接触 SIMOTION 编程的用户。下面拟建立两个 MCC 程序，各程序功能如下：

MCC_1：使能轴，并对两轴进行回零，检测到启动信号后对轴进行定位。

MCC_fault：故障处理程序。

2.5.2 MCC 编程

（1）在项目导航栏中双击 PROGRAMS 下的“Insert MCC unit”，如图 2.25 所示。在弹出的界面中可以自定义该 unit 的名称，之后点击 OK 确认。

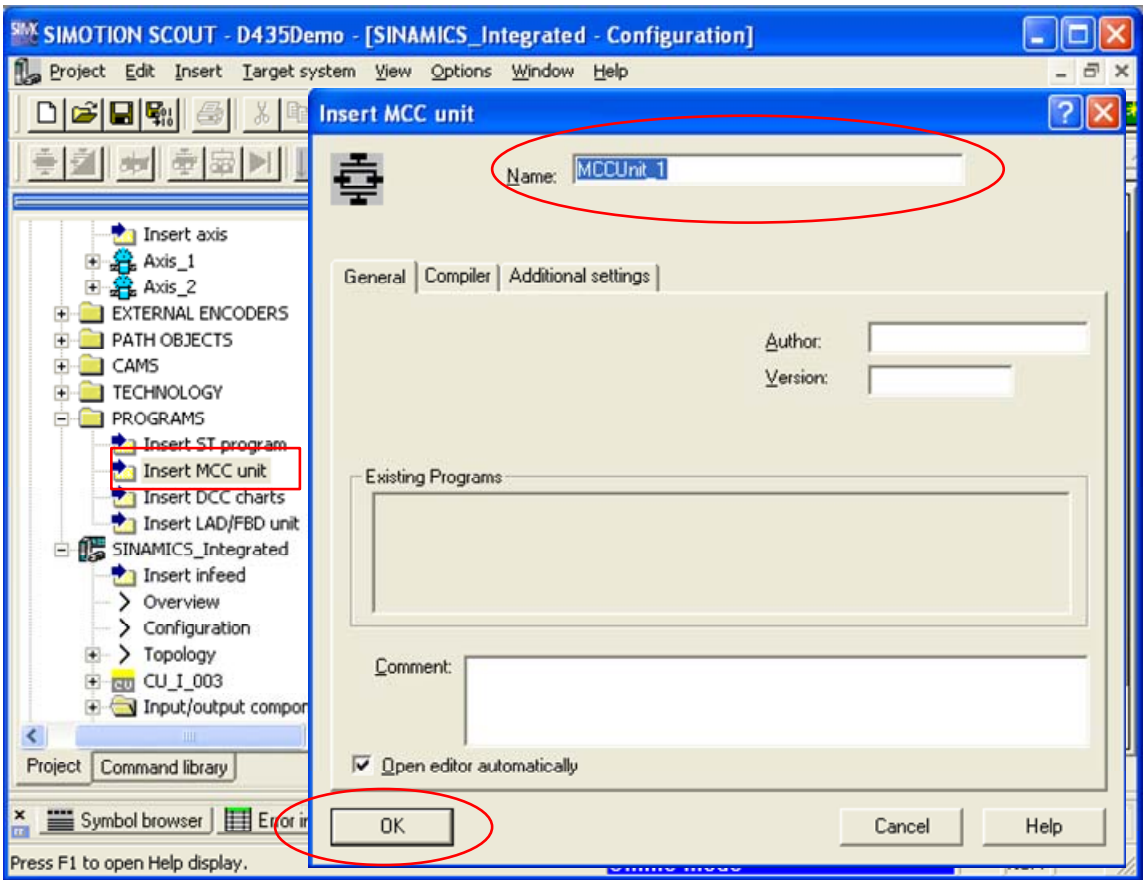


图 2.25 插入 MCCUnit_1

（2）双击 MCCUnit_1 下面的 Insert MCC Chart 插入 MCC_1 程序，Creation type 为 Program，如图 2.26 所示。

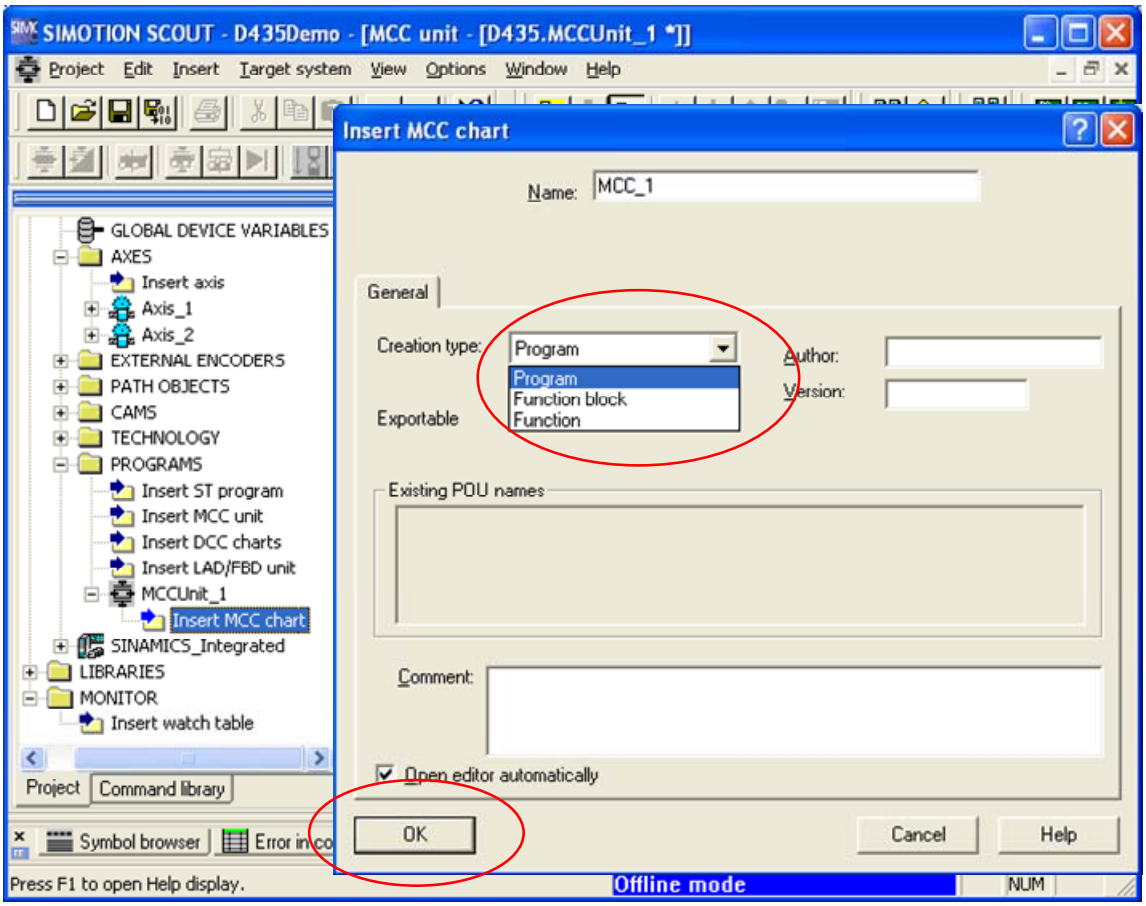


图 2.26 插入 MCC_1

（3）由于 SIMOTION 启动就绪所需要的时间比集成的 SINAMICS 短，因此在执行轴操作命令之前需要等待几秒钟以确保 SIMOTION 和集成的 SINAMICS 都已经启动好好。将鼠标放在工具栏中的  上，其下所包含的命令将自动弹出，选择如图 2.27 所示的等待时间命令插入到 MCC_1 中（注意在选择命令之前必须先选中编程区域的某处）。双击新插入的命令图标，在弹出的命令参数设置对面框中设置等待时间 6s。

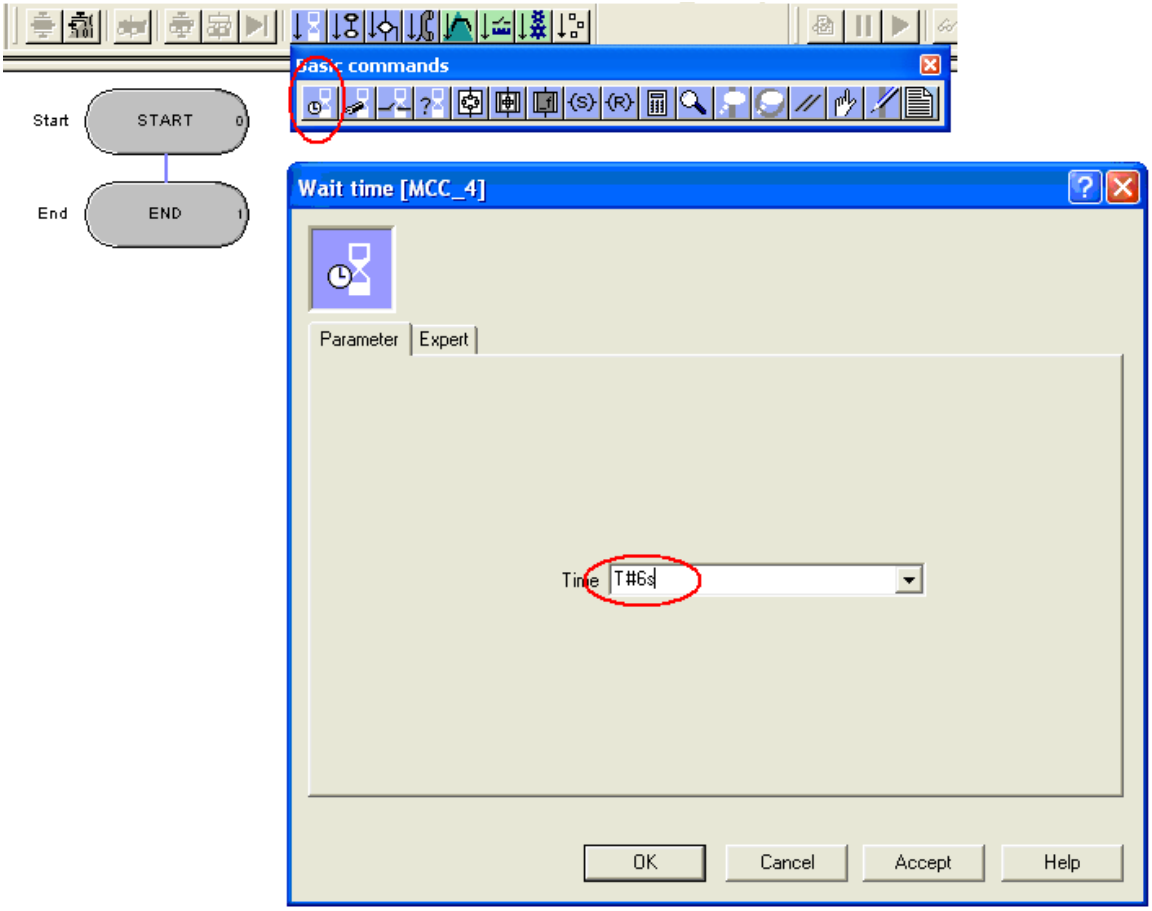


图 2.27 插入等待时间命令

（4）在等待时间命令之后插入  命令组下如图 2.28 所示的使能轴命令，在命令参数设置对面框中，选择 Axis_1。

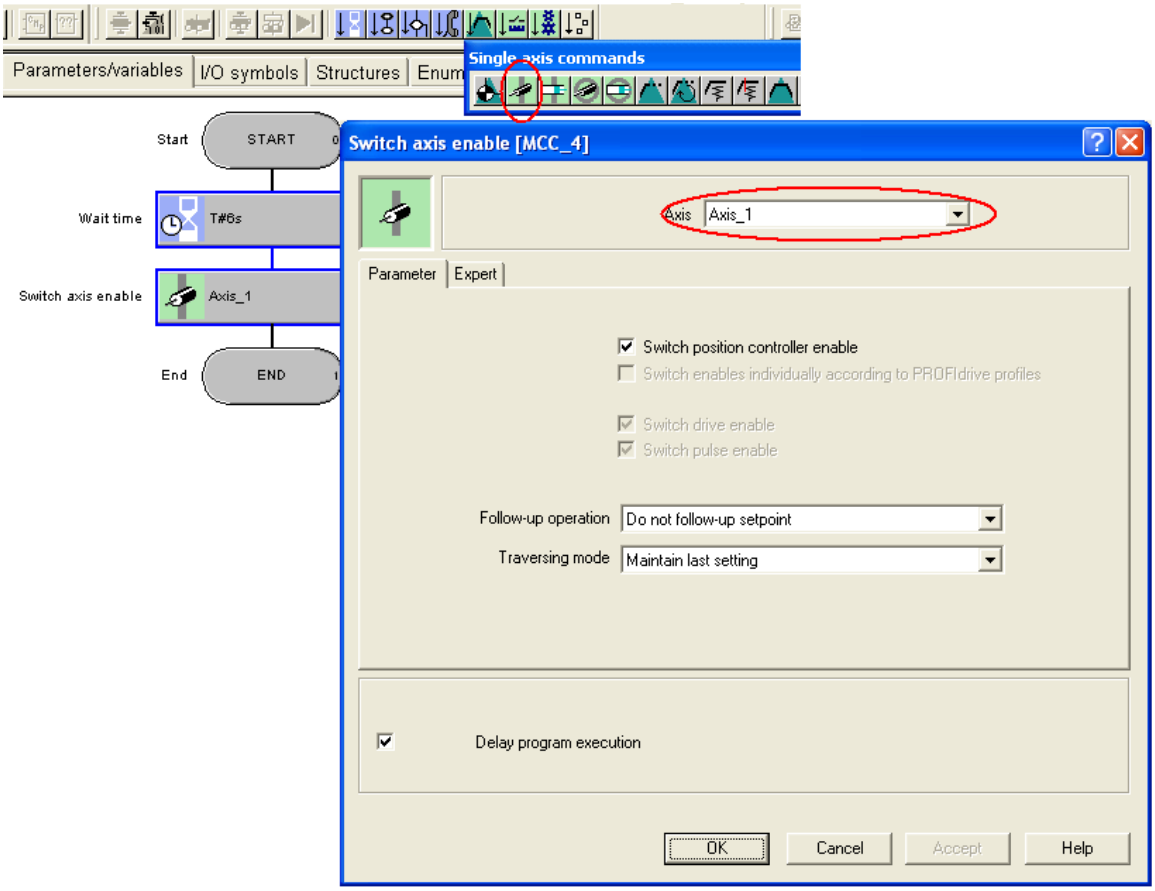


图 2.28 插入轴使能命令

（5）按照相同的方式，选择  下的回零命令，根据图 2.29 所示设置回零命令的参数。该命令执行后，轴的当前位置将被置成 0。

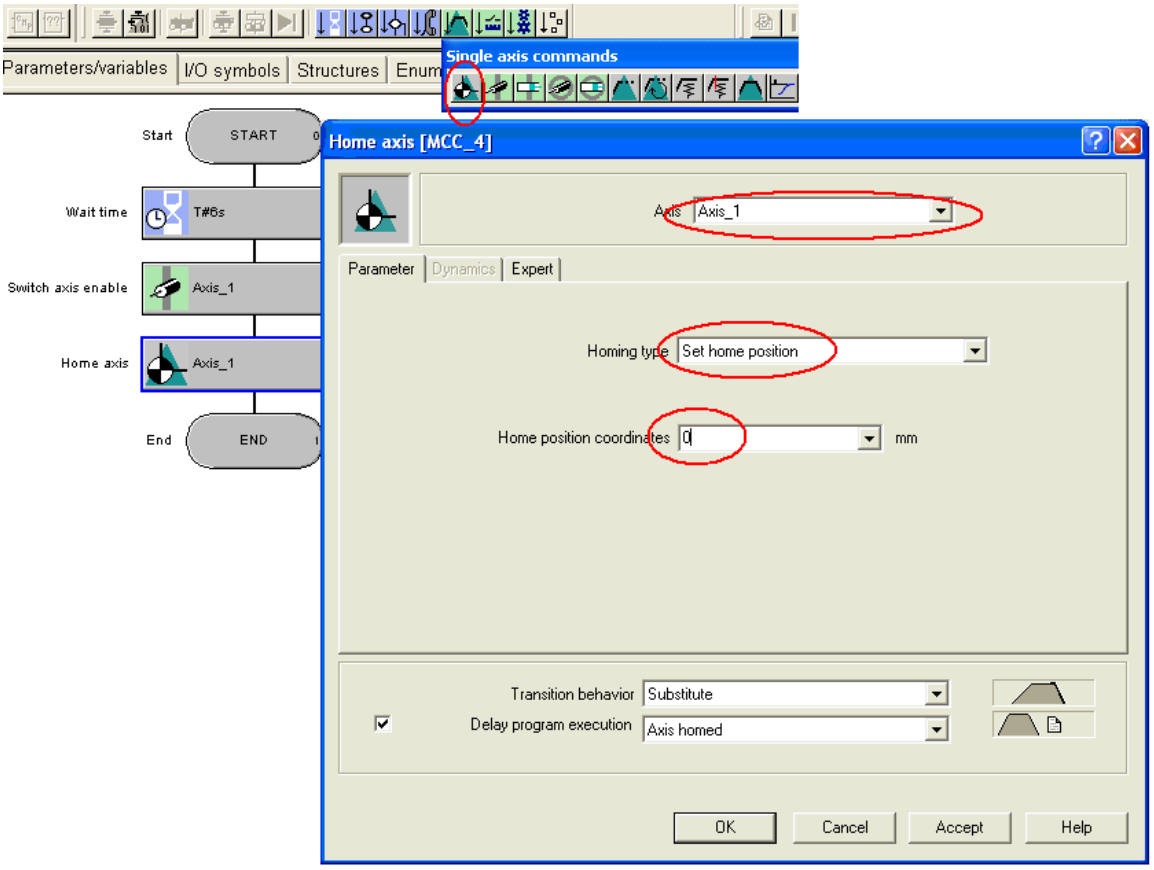


图 2.29 插入轴回零命令

（6）双机 MCCUnit_1，在 INTERFACE 区域中新建 startAxis1 变量用于启动轴定位命令。如图 2.30 所示。

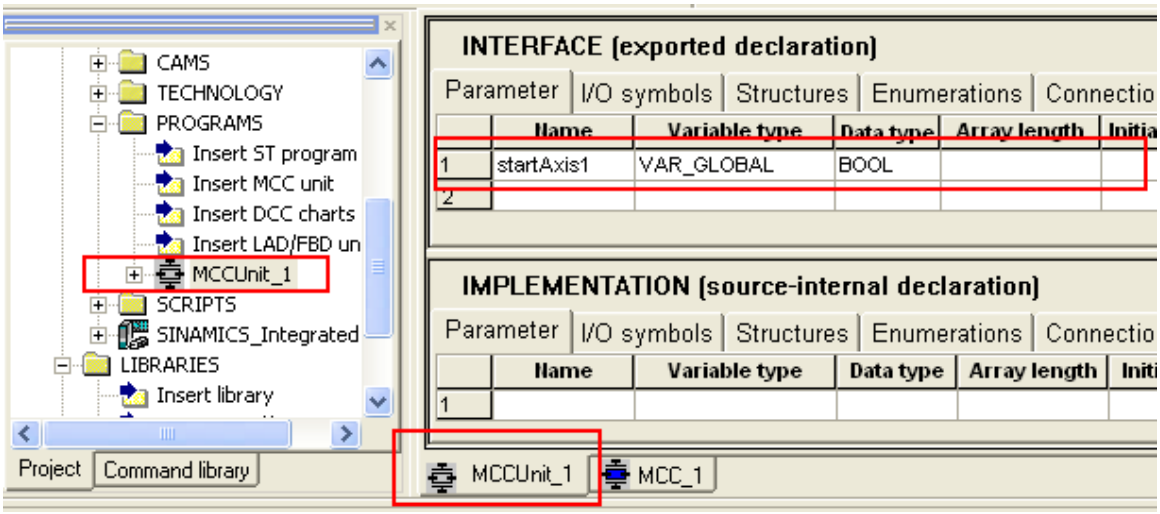


图 2.30 新建变量

（7）插入等待条件命令，如图 2.31 所示，当 startAxis1 变量为 true 时开始执行该命令后面的程序，否则程序停留在当前命令处。

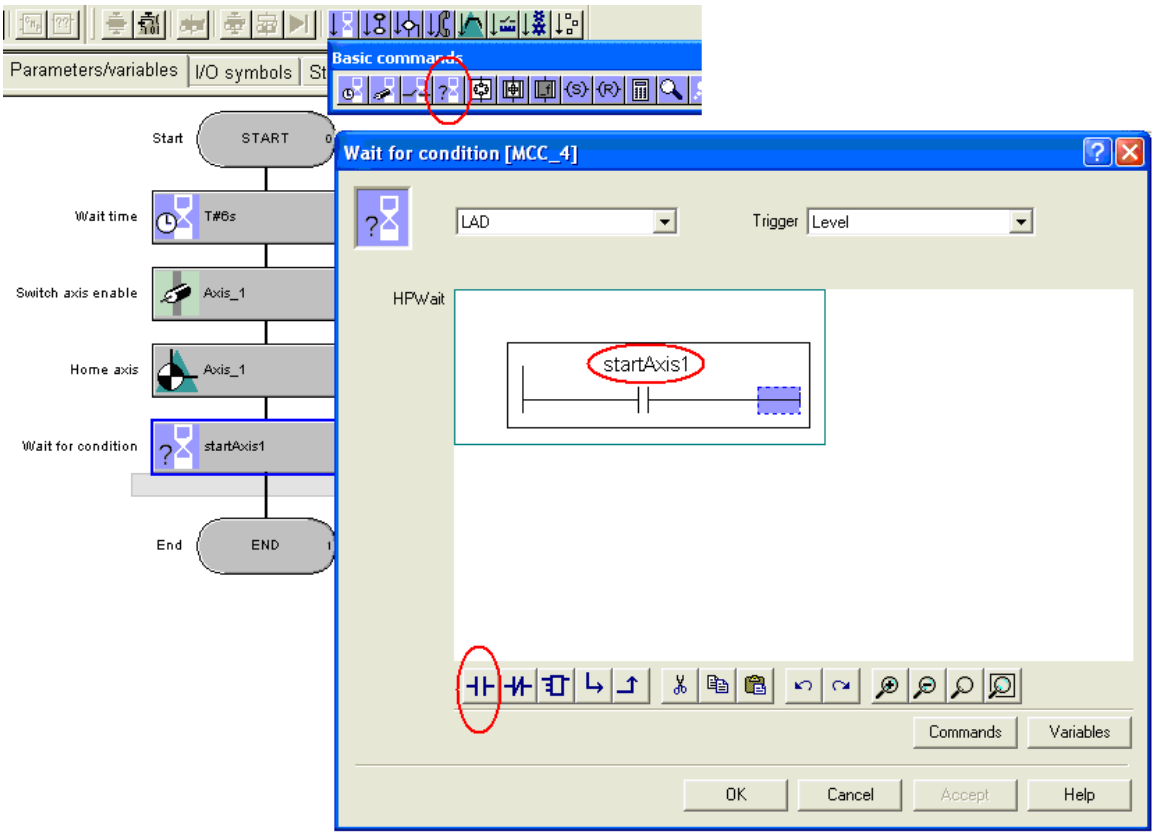


图 2.31 插入等待条件命令

（8）插入  下的轴定位命令，如图 2.32 所示。图中所示设置表示轴将以 200mm/s 的速度从当前位置（0mm）运动到 1000 处。

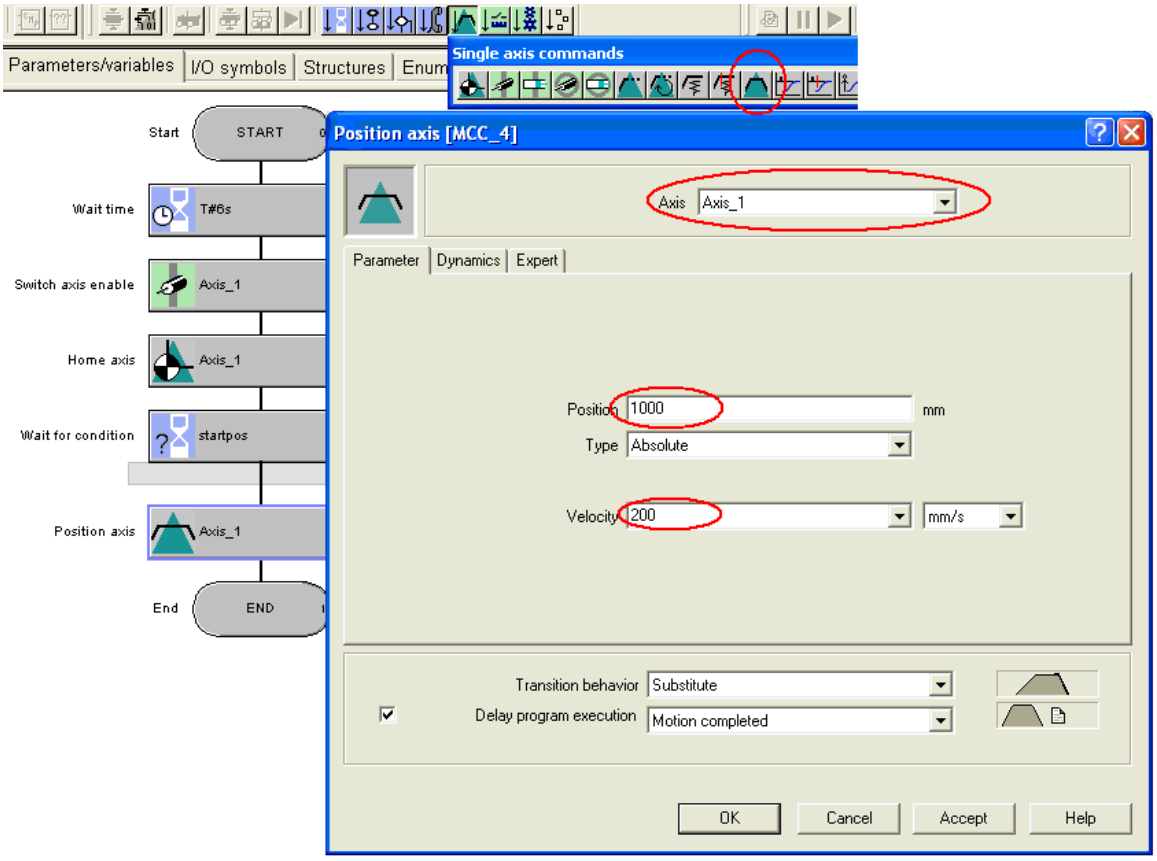


图 2.32 插入轴定位命令

（9）点击工具栏中的  按钮编译保存程序。

（10）在步骤（1）中建立的 MCCUnit_1 下面双击 Insert MCC Chart 新建 MCC_fault 作为故障处理程序。故障处理程序可以为空。

2.5.3 将程序分配到执行系统

任何程序只有被分配到执行系统的任务中才能被真正执行。双击 EXECUTION SYSTEM 打开 task 界面，按照图 2.33 所示步骤将 mcc_1 分配到 MotionTask_1 中。

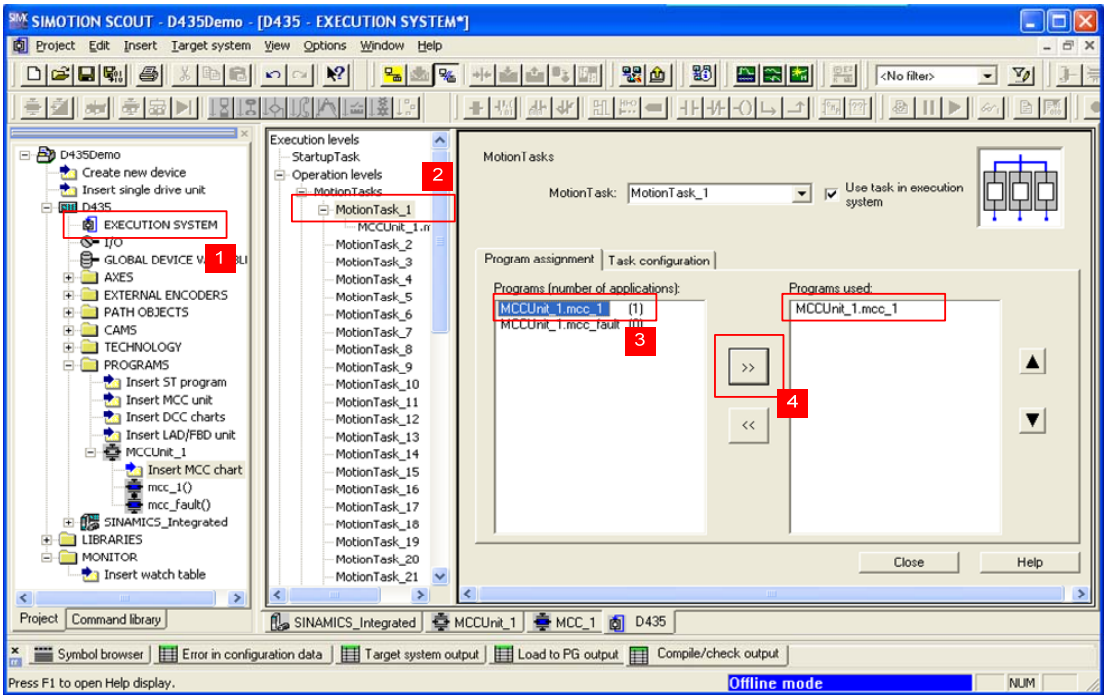


图 2.33 将程序分配到 Motion Task

在 MotionTask_1 的 Task configuration 标签页中，激活 Activation After Startup Task（如图 2.34），则分配到 MotionTask_1 中的程序将在设备上电启动进入运行状态后立即自动执行。

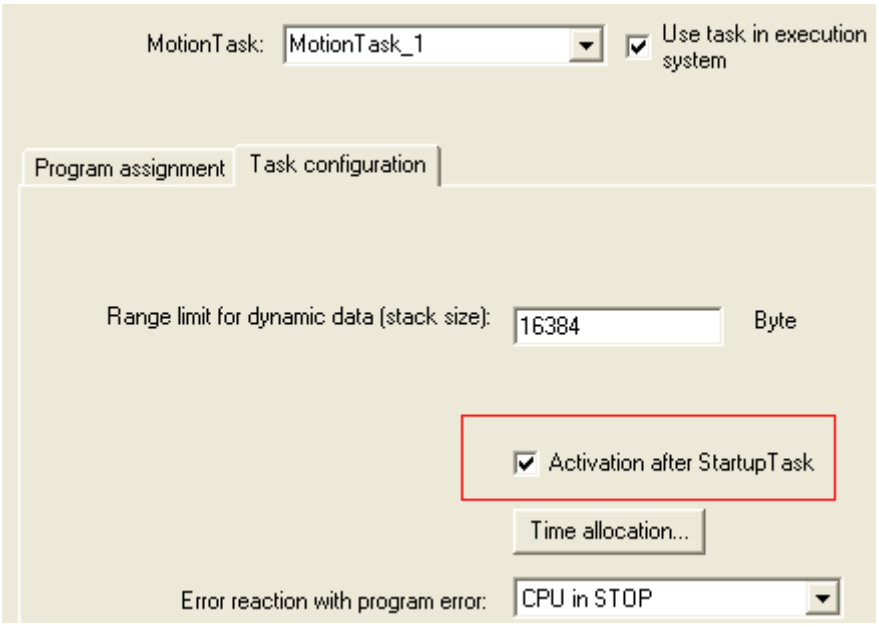


图 2.34 激活 Motion Task 的上电自动运行

按照类似的方法将 mcc_fault 分配到工艺故障 Task 和外设故障 Task 中，如图 2.35 所示。MCCUnit_1.mcc_fault 后面的数字表示该程序被分配的次数。

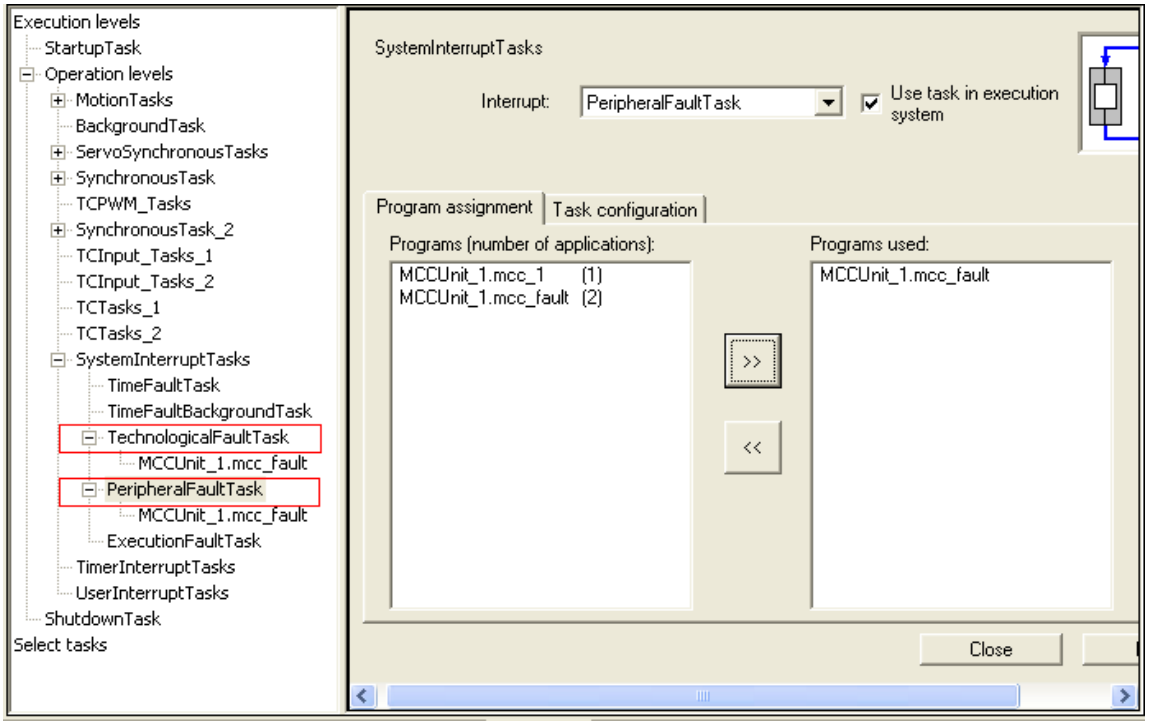


图 2.35 分配故障 Task

2.5.4 下载程序

点击工具栏按钮编译保存项目，编译无误后点击将 Scout 在线，然后下载整个项目到设备中。

SIMOTION 上 RDY 等变绿之后，如果 CPU 处于 STOP 状态（SIMOTION 上 STOP 灯为橙色），可以通过右键点击项目设备名称 D435 选择 TargetDevice->Operating Mode…打开操作模式控制面板，将 CPU 状态改为 RUN，如图 2.36 所示。此时 SIMOTION 上 RUN 指示灯将变为绿色。

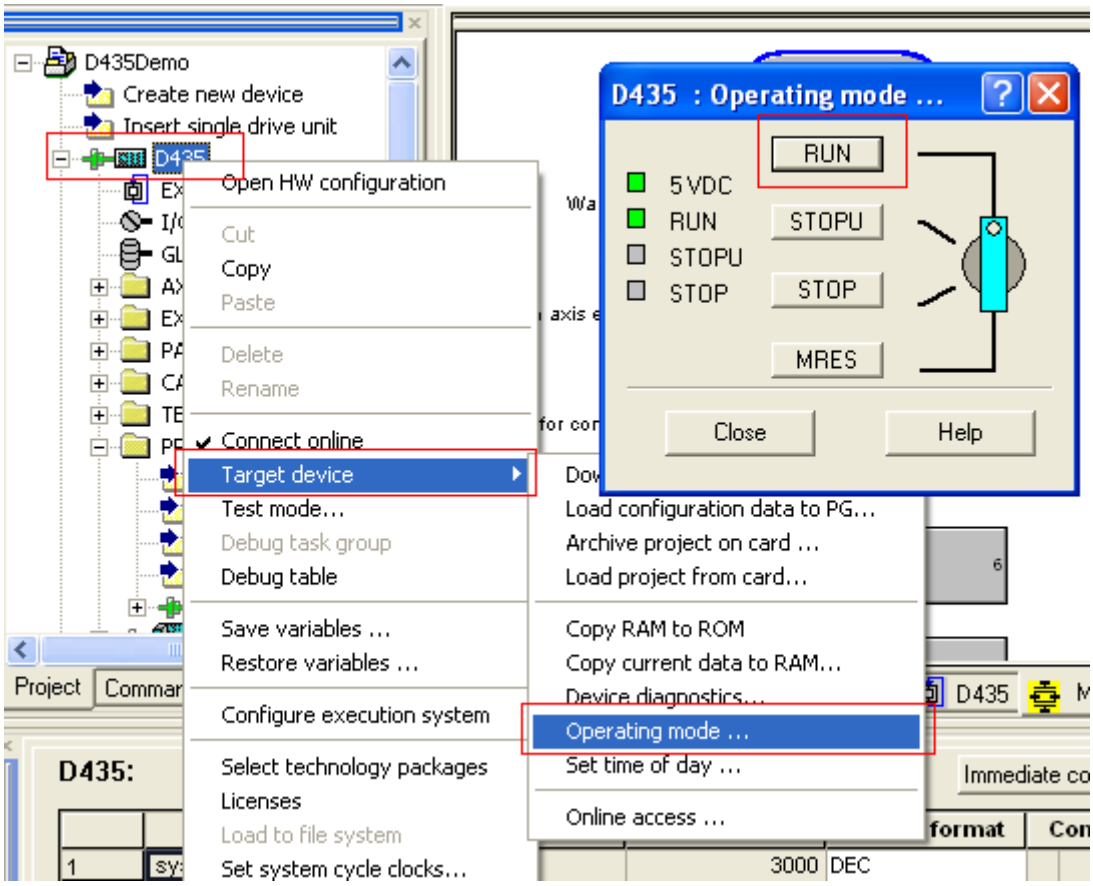


图 2.36 CPU 操作模式控制

2.5.5 程序测试

当 CPU 进入到 RUN 模式之后 MotionTask_1 中的程序立即执行。在等待 6s 之后，Axis1 依次执行使能、回零命令，执行完之后，程序停留在等待条件命令处。此时可以通过 Axis_1 的系统变量查看轴的当前位置值，如图 2.37 所示，positioningstate 下的 actualposition 和 commandposition 分别表示轴的当前位置和给定位置。

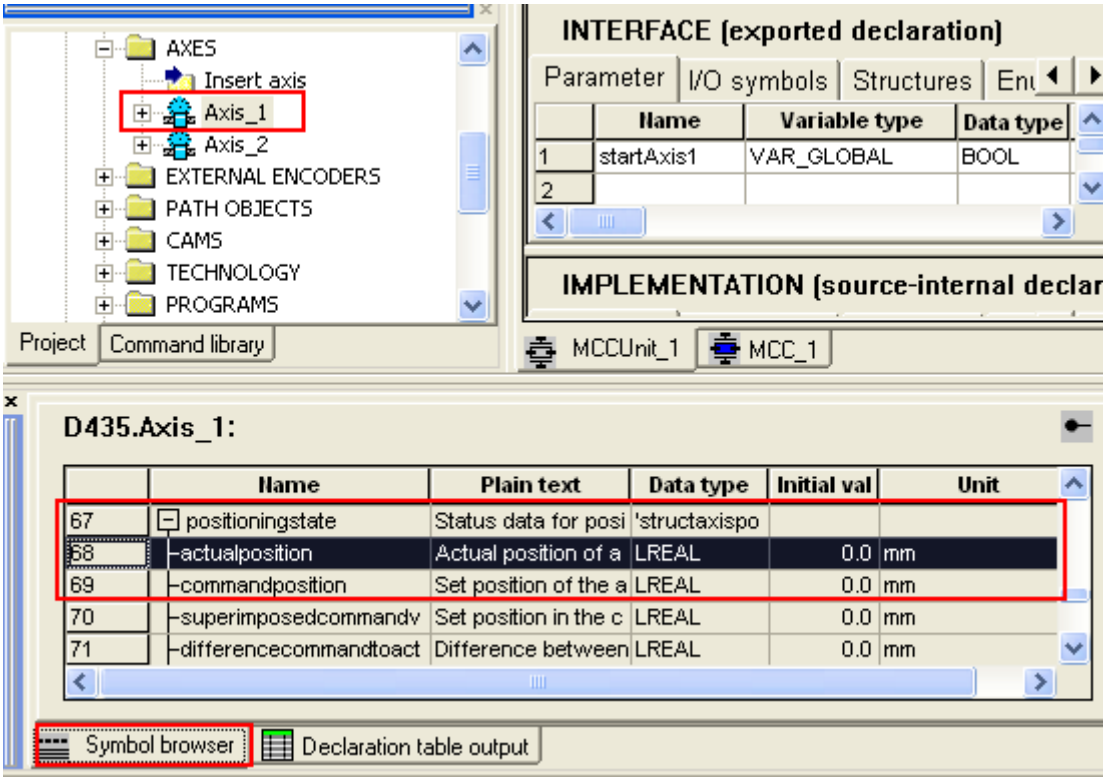


图 2.37 查看轴当前位置

现在只要将 startAxis1 置为 true，就能执行其后的定位命令。在项目导航栏中选中 MCCUnit_1，屏幕下方的 Symbol browser 中将列出其中的变量。在 Control Value 栏中设置值为 TRUE，并确保旁边方框中已打上钩，再点击 Imitate Control 按钮，startAxis1 的值即变为 true。程序中等待的条件成立，定位命令执行，轴开始以 200mm/s 的速度运行直到位置达到 1000mm。如图 2.38 所示。

轴停下来后，通过查看 positioningstate.actualposition 变量可知轴是否已经运行到指定位置。

到此程序执行结束，由于 MotionTask 为顺序执行的任务，不会自动重复执行，若要该程序再次执行，只能重新下载程序。

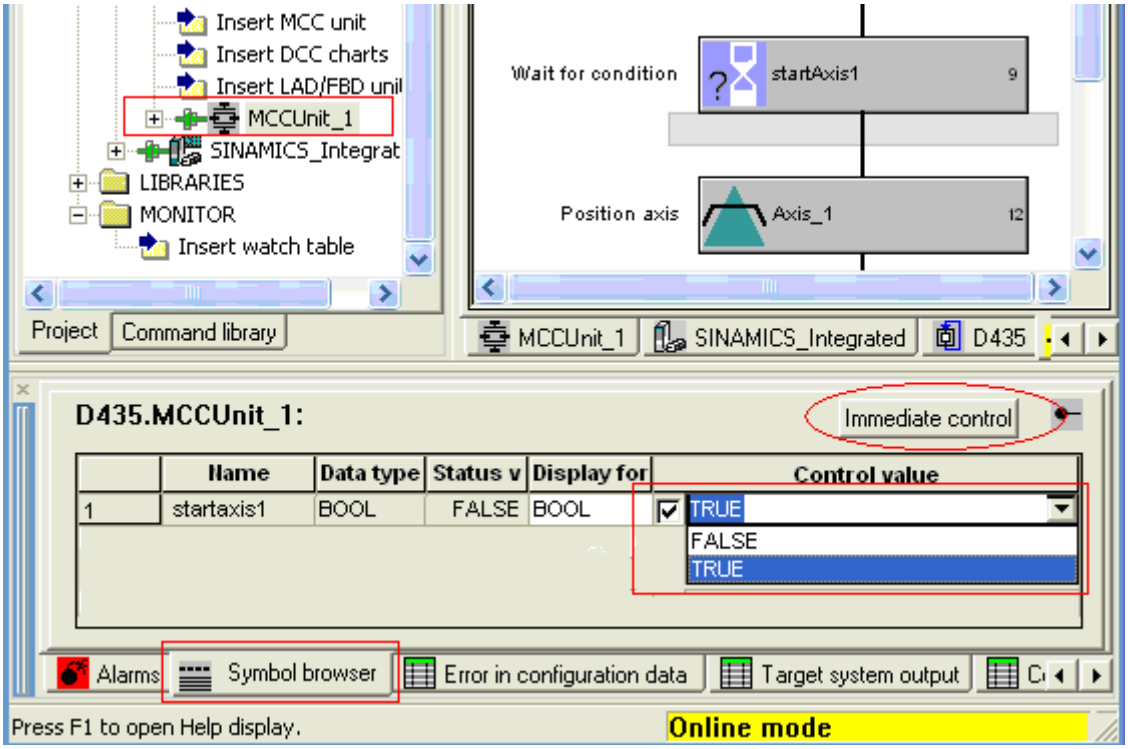

Insert MCC unit
Insert DCC charts
Insert LAD/FBD unit
MCCUnit_1
SINAMICS_Integrat
LIBRARIES
MONITOR
Insert watch table
Project
Command library
Wait for condition
startAxis1
9
Position axis
Axis_1
12
MCCUnit_1
SINAMICS_Integrated
D435
D435.MCCUnit_1:
Immediate control
Name
Data type
Status v
Display for
Control value
1
startaxis1
BOOL
FALSE
BOOL
TRUE
FALSE
TRUE
Alarms
Symbol browser
Error in configuration data
Target system output
Press F1 to open Help display.
Online mode

图 2.38 变量在线赋值

三、执行系统

SIMOTION 中的执行系统（EXECUTION SYSTEM）管理着系统任务以及用户任务的有序执行，每一个任务只有在满足特定条件的时候才会启动，而用户编写的程序只有分配到任务中才能真正被执行。执行系统分为不同的等级，每个等级可以包含一个或多个任务（TASK），每个任务中又可以分配一个或多个程序。因此，用户可以通过将程序分配到不同的任务，来指定程序运行时的优先级或执行顺序。同一个程序可以分配到不同的任务中，且相互之间不受影响。分配到任务的程序可以是 MCC、LAD 或 ST 程序。

3.1 执行等级

SIMOTION 中的执行系统根据其执行特点分为不同的执行等级。

- 同步任务以及时间控制的执行等级是按照指定的系统时钟周期或用户指定的时间周期执行。
- 事件驱动的执行等级是在事件发生时执行，包含用户中断任务和系统中断任务。
- 自由运行的执行等级，包含 MOTION TASK 和 BACKGROUND TASK，其中 MOTION TASK 为顺序执行的任务，每个 MOTION TASK 只执行一次；BACKGROUND TASK 为非周期的循环执行，执行的时间根据 TASK 中的程序长度而定。
- 系统启动和停止任务在特定的时候执行，其中 Startup task 在 CPU 从 STOP 状态切换到 RUN 状态时执行；Shutdown task 在 CPU 从 RUN 状态切换到 STOP 状态时执行。

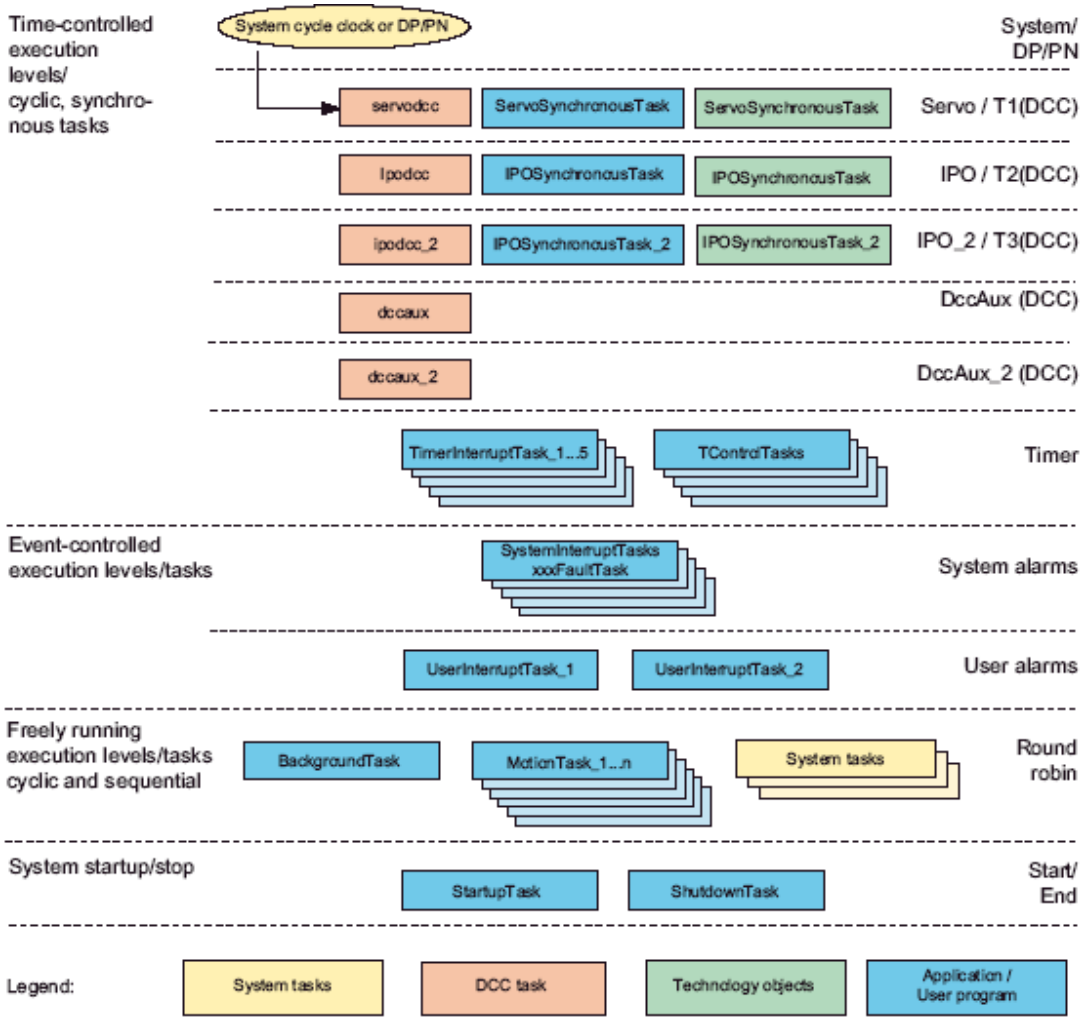


图 3.1 执行等级

除了用户级的任务之外，在各执行等级中还包含系统任务，由系统自动执行，用户无法分配程序到系统任务也不能改变其执行顺序，例如工艺功能、通讯（PROFIBUS DP、PROFINET IO）、TRACE 功能等，这些功能将被自动分配到系统任务自动执行，而不受用户程序影响。

DCC 任务在 SCOUT 的执行系统界面中是看不到的，但是可以在 DCC 编辑器中指定其执行等级。

将程序分配给任务时，双击 EXECUTION SYSTEM 打开任务分配界面，选中 MotionTask_1，在 Program assignment 列表框中选中程序 mcc_1，然后点击向右的箭头便将 mcc_1 分配给了 MotionTask_1。图 3.2 显示了分配的整个步骤。向其他任务（如 background task、startup task 等）中分配程序的过程与此相同。

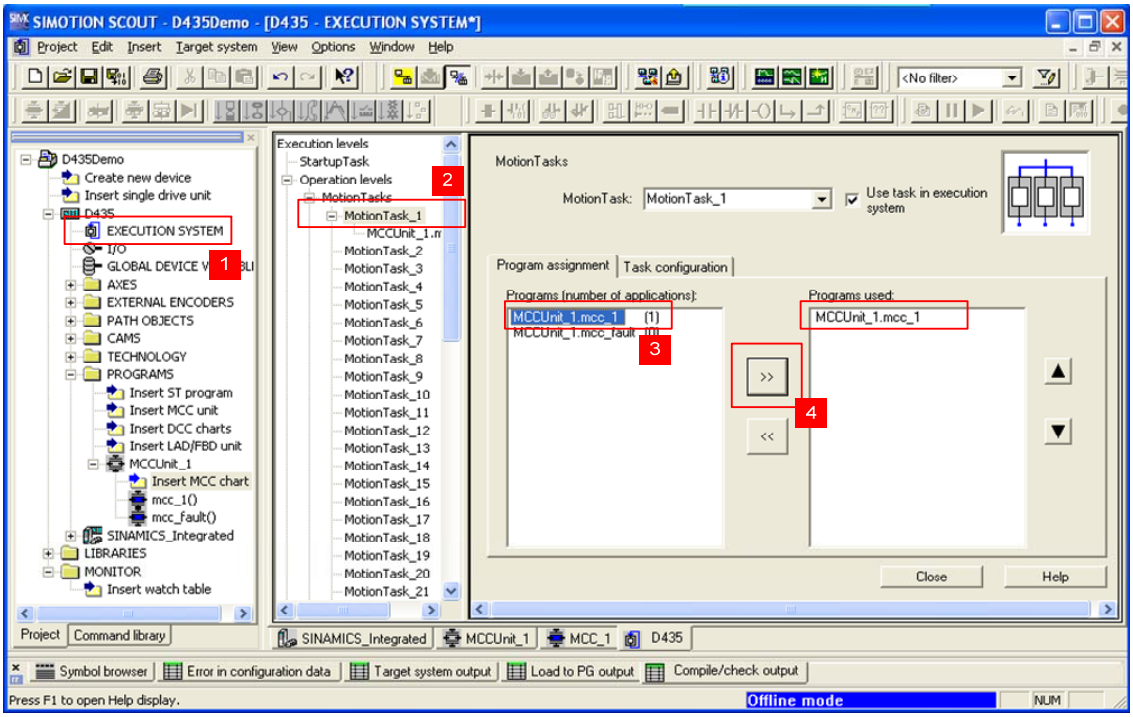


图 3.2 将程序分配到 Motion Task

在每个任务属性的 Task Configuration 标签页中可以设置任务的运行时间、启动特性、故障响应等，不同的任务具有不同的属性。例如在 MotionTask_1 的 Task configuration 标签页中，激活 Activation After Startup Task（如图 3.3），则分配到 MotionTask_1 中的程序将在设备上电启动进入运行状态后立即自动执行。其他参数设置请参考帮助手册。

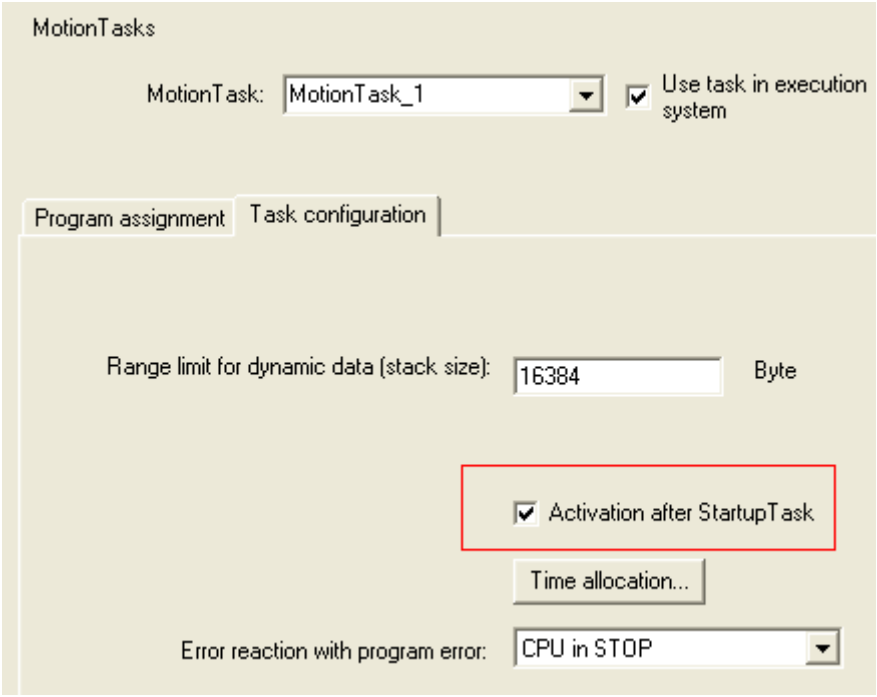


图 3.3 激活 Motion Task 的上电自动运行

如果程序中使用到定位、同步等工艺功能或者 SIMOTION 通过 PROFIBUE DP 等总线扩展了 IO 接口等，则需要为故障 Task 分配故障处理程序。故障处理程序可以是空程序也可以根据用户需要在程序中添加故障处理指令。图 3.4 显示了将 mcc_fault 分配到工艺故障 Task 和外设故障 Task 中。

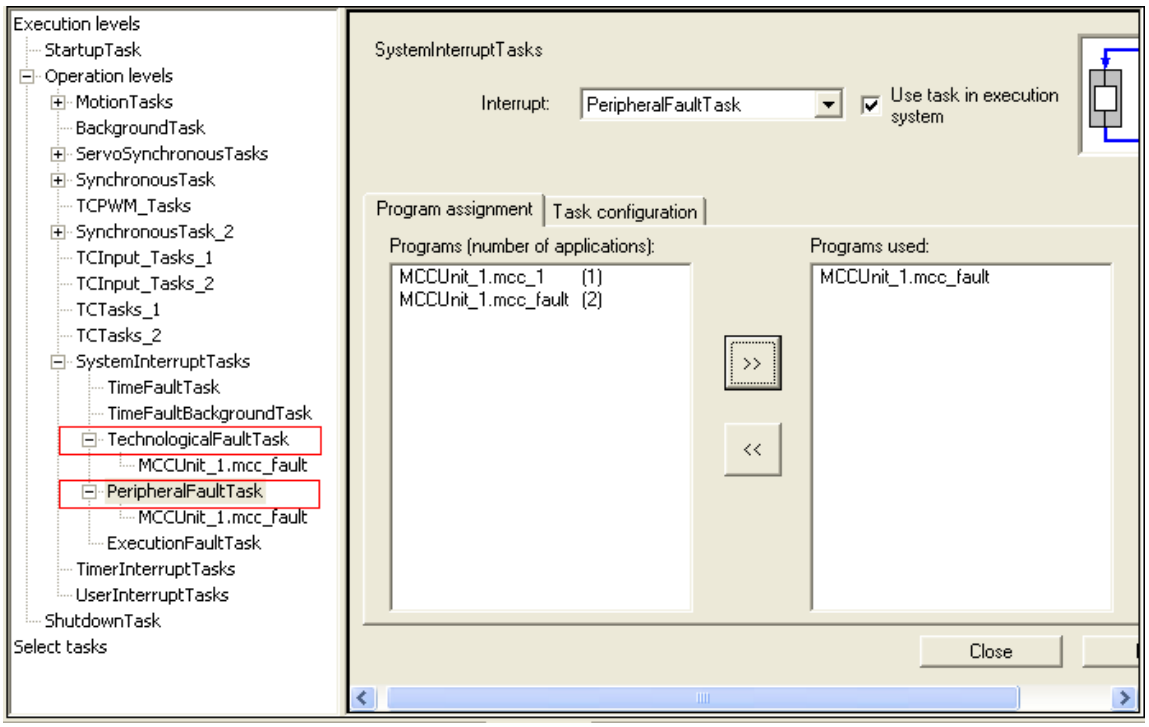


图 3.4 分配故障 Task

3.2 任务优先级

当多个任务在某一个时刻同时开始执行时，任务之间就会有冲突。执行系统规定不同的任务具有不同的优先级，优先级高的任务先执行，正在执行的低优先级的任务可以被高优先级中断，直到高优先级的任务执行完才继续执行低优先级的任务，这样就避免了任务之间的冲突。用户不能改变任务的优先级。

SIMOTION 中各任务的优先级如图 3.5 所示。图中带阴影的部分表示可以有多个任务。

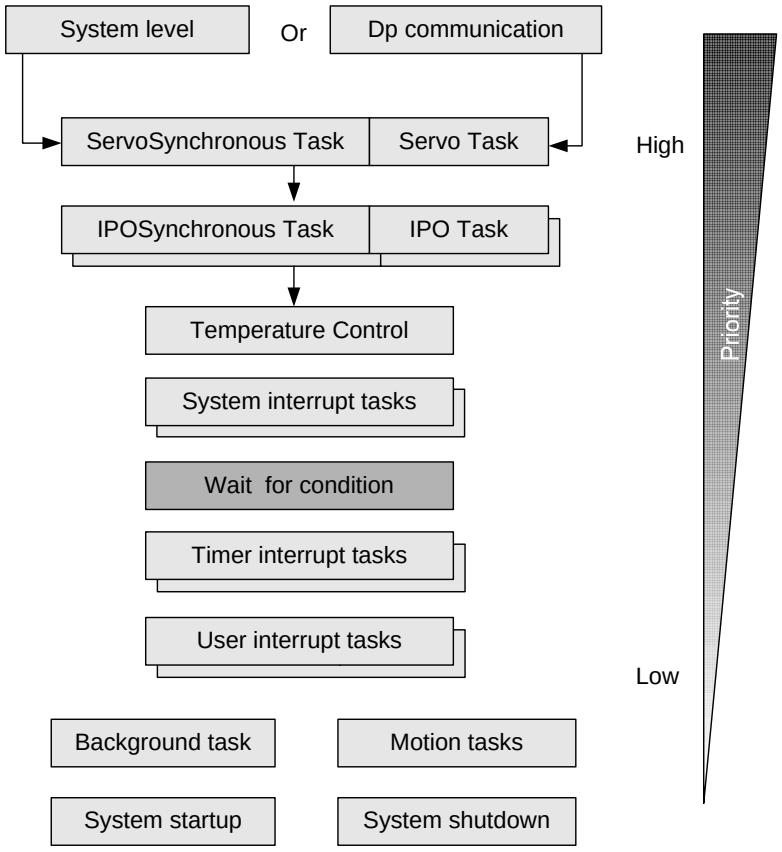


图 3.5 任务优先级

- 处于最高层的 System level 和 DP 通讯用于 PROFIBUS 或系统时钟同步以及读写 IO 信息。
- ServoSynchronous task 是用户可以分配 PROGRAM 的优先级最高的任务，以指定的系统时钟周期循环执行，它优先于 Servo task 执行。Servo task 是系统任务，用户不能给它分配 PROGRAM。
- 与 Servo 级相似，IPOSynchronous task 优先于 IPO task 执行，用户不能给 IPO task 分配 PROGRAM。与 Servo 级不同的是，IPO 级 有两组，除了图中

所示的一组之外，还有另一组为 IPOSynchronous task2 和 IPO task2，可以用于控制动态特性需求较低的轴控制。

- DP 通讯周期、Servo 执行周期以及 IPO 执行周期存在一定的比例，且以 DP 通讯周期为最小时间单位，称为 DP Cycle。可以通过右键点击设备名称 D435 选择“Set system cycle clocks…”来设置它们之间的比例。如图 3.6 所示。

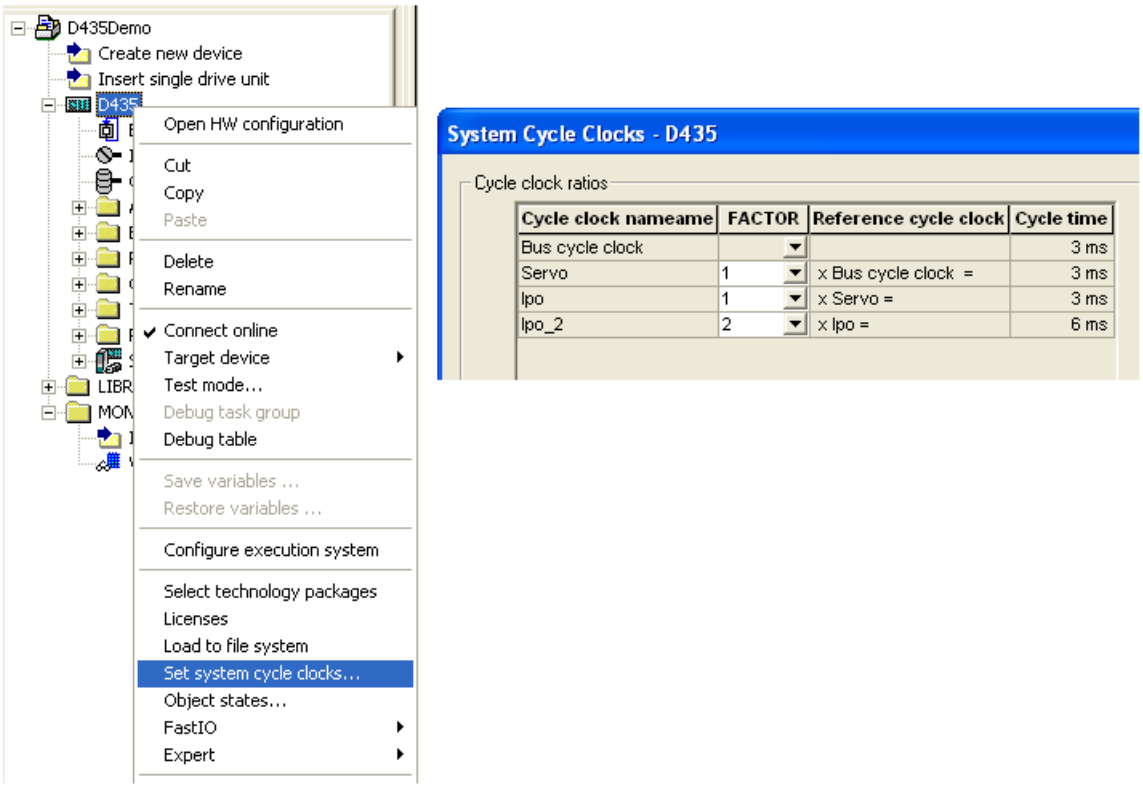


图 3.6 设置系统时钟周期

- 在 IPO 级，需要检查 Wait for condition 指令的条件以及用户中断的条件是否满足。如果 Wait for condition 指令条件满足，那么处于 WAITFORCONDITION 和 ENDWAITFORCONDITION 之间的程序（在 MCC 中即处于 wait for condition 指令下的阴影部分）将以更高的优先级执行（系统中断和时间中断之间）。
- Background task 和 motion task 处于最低优先级（前者相对较高），motion task 总共 32 个，主要用于顺序控制。
- Startup 和 shutdown task 只有在 CPU 状态切换的时候执行。

图 3.7 的例子中表示了执行系统中各任务的执行顺序。DP、Servo 和 IPO 周期的比为 1:1:2。DP 通讯处于最高优先级每个 DP cycle 都要执行一次，Servo 的

执行周期与 DP 通讯相同，优先级次之，因此每个 DP Cycle 中 DP 通讯执行完以后都要执行一次 Servo task。IPO 中的 task 每两个 DP Cycle 执行一次，图中的例子显示 IPO task 需要的时间超过一个 DP cycle，因此在某一个 DP Cycle 中没有执行完的 IPO task 将延迟到下一个 DP Cycle 中的 servo task 后面执行。

中断的优先级低于 IPO task，各个中断按照其优先级依次执行。用户中断条件的检查在 IPOSynchronous task 之前，但是中断的执行必须严格按照优先级。在中断的条件在 IPO 执行完中断 task 后的 DP Cycle 中剩下的时间用于 Background task 和 motion task。

与 motion task 相比，Background task 优先级较高，而且 Background task 有时间限定，如果在限定的时间内 Background task 的程序没有执行完，则报 timeout 故障导致 CPU 停机。因此，诸如 Wait for time 之类的等待型指令不能用于 Background task 中，而且 Background task 中用到的各种指令的 nextcommand 参数 也不能设置为 WHEN_MOTION_DONE。

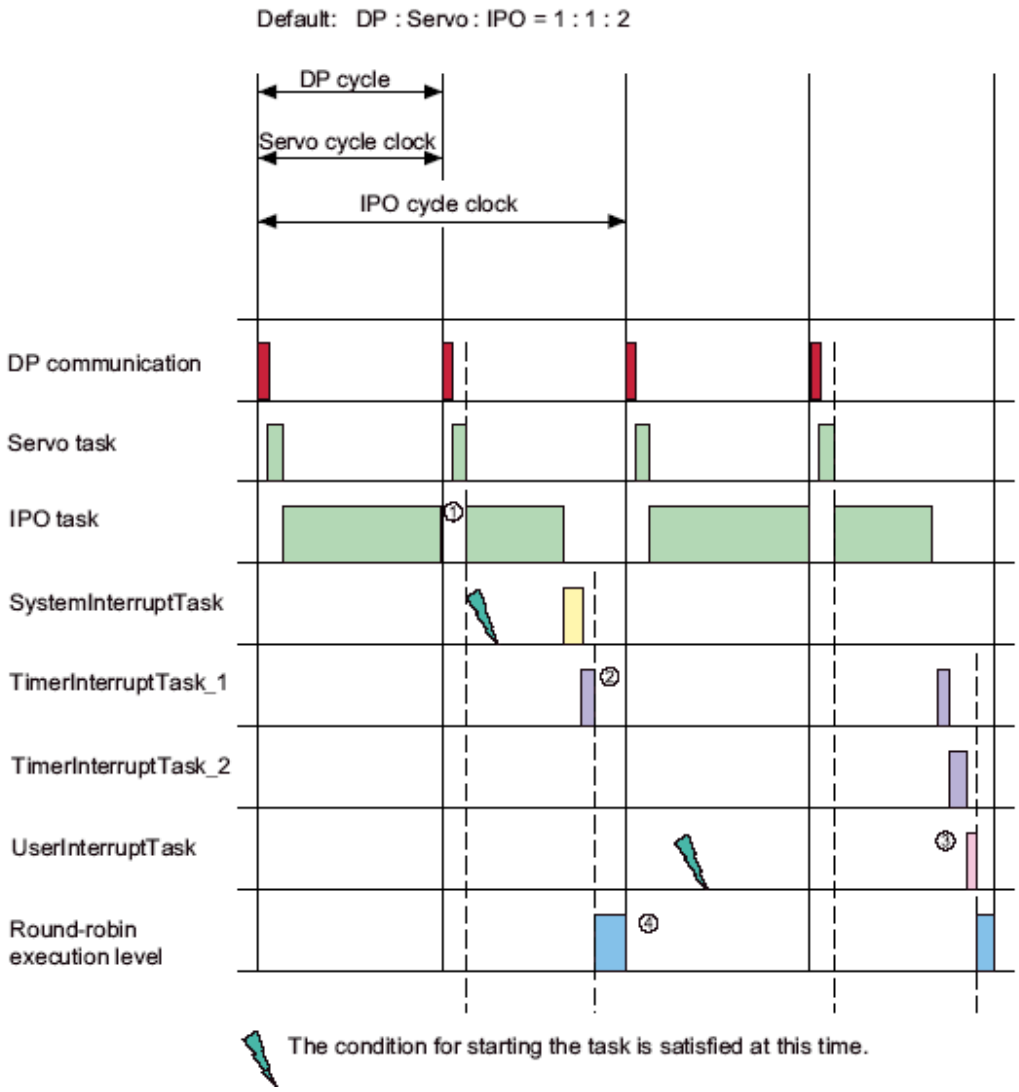


图 3.7 任务执行顺序

四、编程语言

4.1 概述

SIMOTION 具有多种编程语言，不同的编程语言具有不同的特色，从而使用户能够得心应手地处理各种任务。

SIMOTION 的编程语言主要包括以下三种：

1. MCC (Motion Control Chart)

MCC 是一种图形化的编程语言，几乎所有的功能（诸如定位、同步等）都有图形化的命令与之对应。它适用于编写顺序执行的程序，与 MontionTask 结

合使用使程序的执行更加简单明了。对于刚刚接触 SIMOTION 编程的人来说，这是一种非常好的入门语言。

2. LAD/FBD (Ladder Diagram)

梯形图是 PLC 中广泛使用的一种编程语言，对于熟悉 PLC 的用户，在使用 LAD 对 SIMOTION 进行编程时能够驾轻就熟。LAD 适用于边沿触发、二进制数据处理等逻辑控制程序。

3. ST (Structured Text)

ST 是一种基于文本的类似于 PACAL 的高级编程语言，它具有高级语言的各种元素，例如操作符、表达式等，因此适用于处理数据和包含数学算法的程序。

本章主要介绍 MCC 和 LAD 编程语言， ST 编程语言不在本书讨论范围之内。

4.2 MCC

MCC 程序的编写过程如同编写流程图，能够很大程序降低生产机械运动控制任务的复杂性。同时，它也支持逻辑控制命令，具有快速响应事件的等待命令，便于建立模块化 Program 以及 FC（Function）和 FB（Function Block）。图 4.1 为 MCC 编程的工作界面。

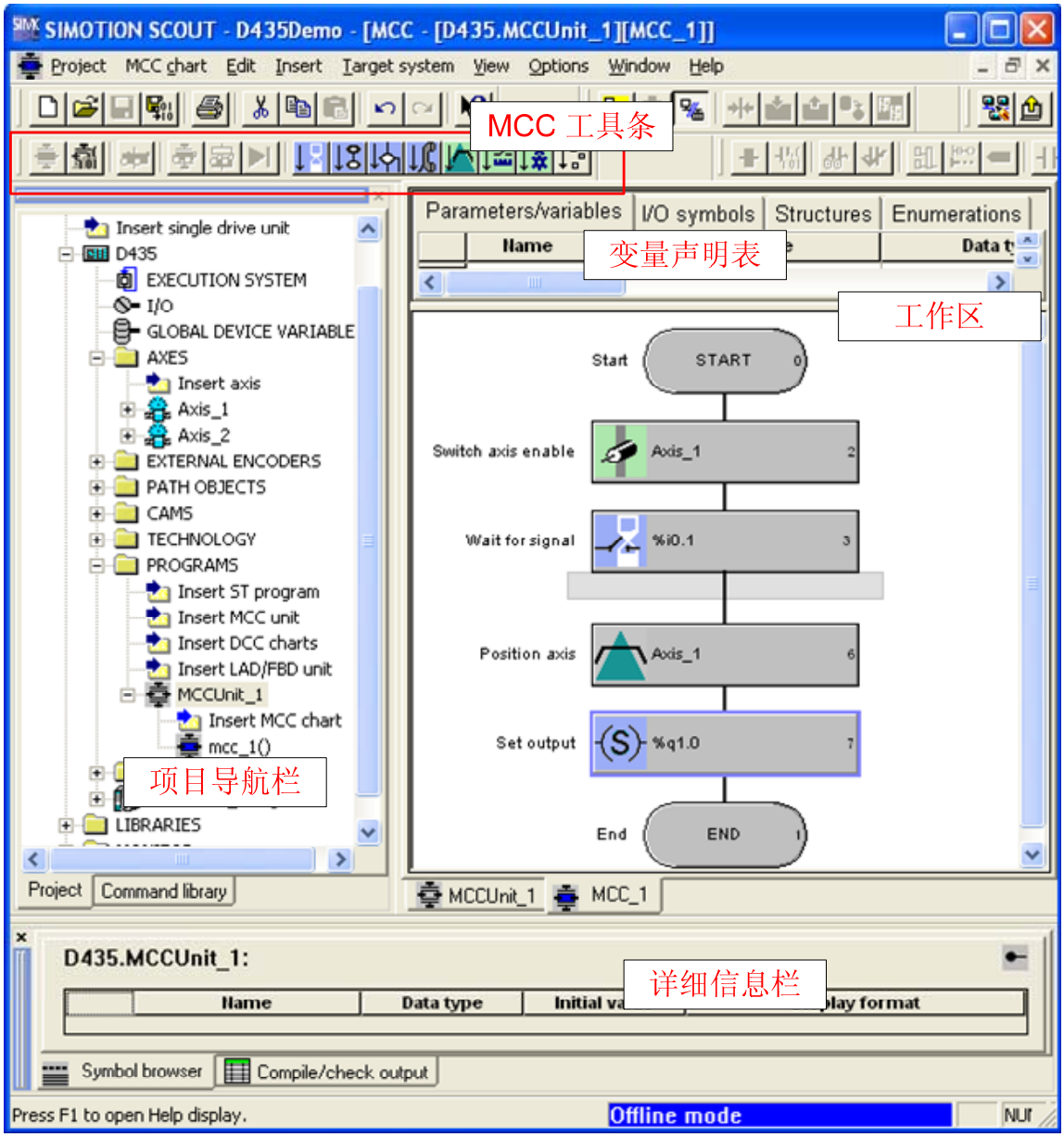


图 4.1 MCC 变成界面

4.2.1 MCC Unit 和 MCC Chart

MCC Unit 可以看作 MCC 程序的一个分组，在 SIMOTION 程序中可以插入多个 MCC Unit，而每个 MCC Unit 又可以插入多个 MCC Chart，每个 MCC Chart 都是一个独立的程序组织单元，它可以是 Program、FC、FB。

1. 插入 MCC Unit

在项目导航栏中双击 PROGRAMS 下的“Insert MCC unit”，如图 4.2 所示。

在弹出的界面中可以自定义该 unit 的名称，之后点击 OK 确认。

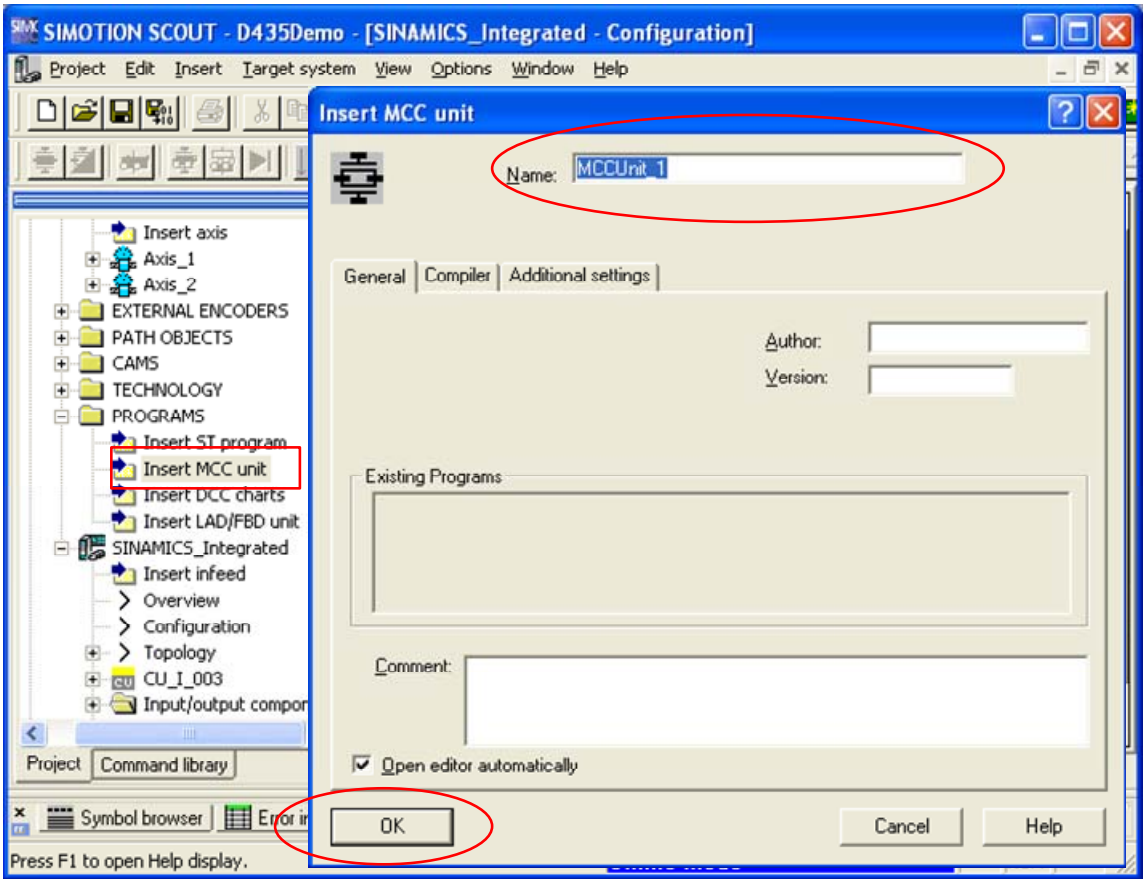


图 4.2 插入 MCC Unit

在 Compiler 标签页，可以设置该 unit 的编译选项，如图 4.3 所示。关于选项中每条的具体含义可以点击“Help”按钮参考帮助文档。

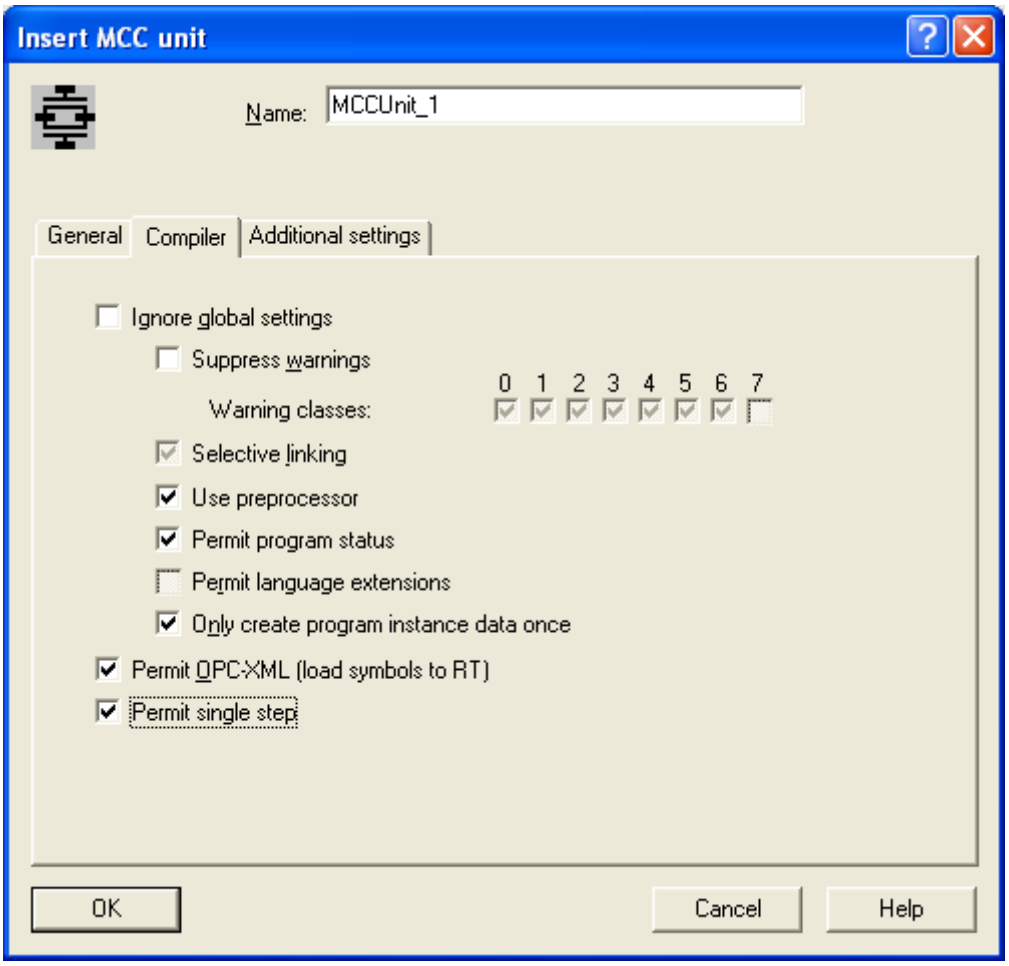


图 4.3 MCC unit 的编译选项

2. 插入 MCC Chart

双击刚刚建立的 MCCUnit_1 下面的“Insert MCC Chart”插入一个 MCC Chart 并对其进行命名，如图 4.4 所示。在创建类型中可以选择 Program、FC 或者 FB。SIMOTION 运行时以 PROGRAM 作为基本的执行单元，PROGRAM 中可以调用 FC 和 FB。

每个 MCC Chart 在建立的时候都自动包含 START 和 END 命令，用户可以在它们之间插入命令和注释，程序执行的时候从 START 开始，逐条顺序执行直到 END 结束。

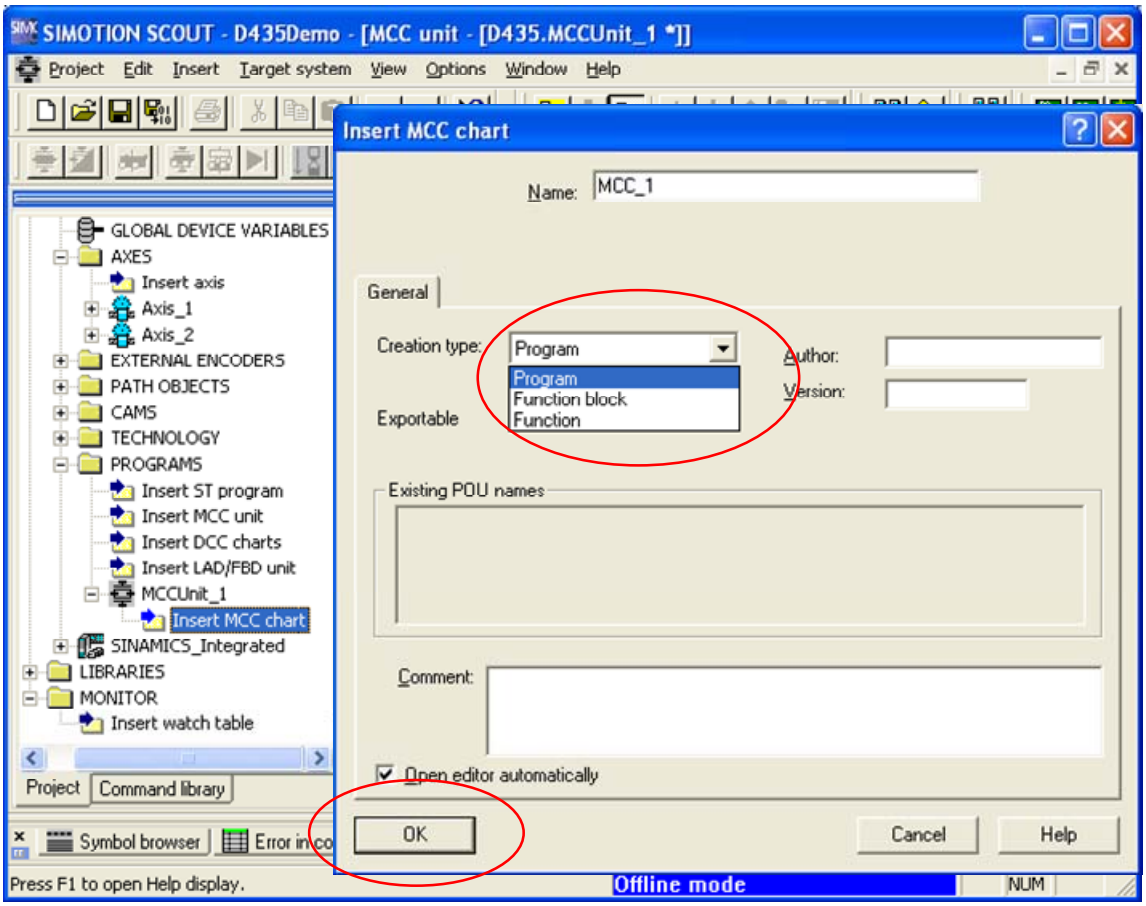
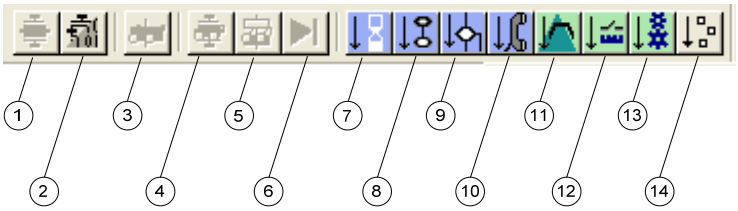


图 4.4 插入 MCC Chart

4.2.2 MCC 命令

在 MCC 工具条中包含重要的编程命令按钮（如插入 MCC Chart、保存和编译、监控等）和各种按照功能分组排列的命令组，如图 4.5 所示。①～⑥为编程命令按钮，允许用户插入 MCC Chart 以及编程后的保存、编译与调试；⑦～⑭为命令组，每个命令组都以一个按钮表示，当鼠标按下或移动到某个命令组按钮时，该命令组所包含的命令将自动在 MCC 工具条下列出来。



① 插入MCC Chart
② 保存编译程序
③ 程序状态
④ 监控
⑤ 单步调试
⑥ 下一步
⑦ 基本命令
⑧ 任务命令
⑨ 程序结构命令
⑩ 通讯命令
⑪ 单轴命令
⑫ 外部编码器、Measuring inputs以及Output Cam命令
⑬ 同步操作命令
⑭ 重要的常用命令

图 4.5 MCC 工具条

要插入命令使，用鼠标选中 MCC Chart 中 START 和 END 之间某处，再从 MCC 工具条中选择所需的命令后，该命令即插入到 MCC chart 中的光标所在位置。

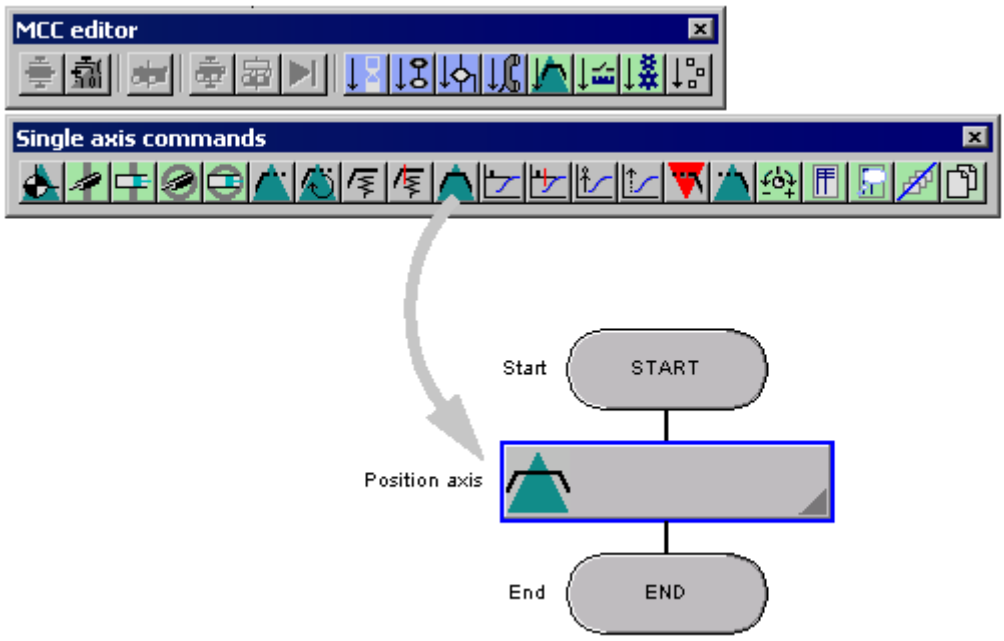


图 4.6 插入命令

双击插入到 MCC chart 中的某个命令，可以打开命令参数设置对话框，如图 4.7 所示，例如，图中打开的轴定位命令参数设置对话框中可以设置轴的位置、速度、定位方式等。

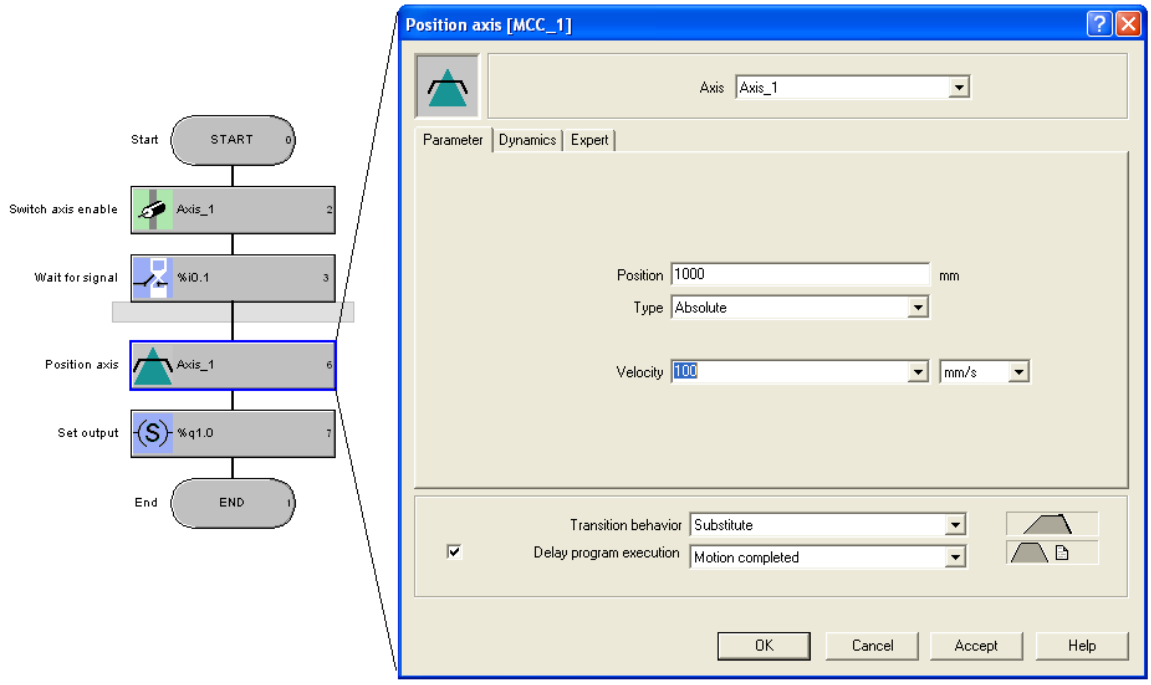


图 4.7 命令参数设置对话框

对于大部分轴操作命令来说，都包含 Transition behavior 和 Delay program execution 这两个参数，其中 Transition behavior 参数表示的是该命令与前一个命令的衔接关系（可以是替换、追加或者混合等），Delay program execution 参数表示的是该命令与下一个命令之间的关系（例如选择 Motion Completed 表示等待该命令执行动作完成才能开始下一个命令）。

MCC Chart 中的每个命令右边都一个数字，对于同一个 MCC Chart 来说，各个命令的数字都不相同，以示区别。在编译出错的时候通过这些数字能够很快找到出错的命令。

4.2.2.1 基本命令

基本命令组中包含等待、赋值、程序调用、工艺对象复位等命令，且利用其中一些命令可以使 MCC 实现逻辑控制的要求。

图 4.8 基本命令

如图 4.8 所示为基本命令组下所包含的各个命令，从左到右依次为：

- 等待时间。当程序执行到该命令时，包含该命令的任务被挂起，任务挂起的时间在该命令中给定。
- 等待轴。当程序执行到该命令时，包含该命令的任务被挂起，直到该命令中指定的轴的条件满足。命令中的条件可以是轴达到指定状态或（和）轴的某参数与具体值相比较满足指定的条件。
- 等待信号。当程序执行到该命令时，包含该命令的任务被挂起，直到指定的信号到达，信号必须是数字量输入 / 输出（BOOL 类型）。
- 等待条件。当程序执行到该命令时，包含该命令的任务被挂起，直到指定的条件满足。命令中的条件可以是单元变量、全局设备变量、常量、I/O 变量以及表达式。
- 模块。任意命令的组合，以使程序结构清晰。
- 子程序调用。可以调用用户定义或者 Library 中的 FC，FB 以及 Program。
- 系统函数调用。用于调用 Command Library 中的 FC，FB。
- 置位输出。可以置位的输出变量类型有 BOOL、BYTE、WORD、DWORD，置位对象的每个二进制位都被置位 1。
- 复位输出。可以复位的输出变量类型有 BOOL、BYTE、WORD、DWORD，复位对象的每个二进制位都被复位为 0。
- 变量赋值。为用户或系统变量赋值。
- 嵌入 ST 程序。允许用户在 MCC 程序中嵌入 ST 程序。
- 激活工艺对象（轴，同步对象，外部编码器等）仿真。
- 取消工艺对象仿真。
- 复位对象。将工艺对象复位到初始状态。
- 改变操作模式。使 SIMOTION 运行于 STOP 或 STOP U 模式。
- 激活跟踪。
- 注解块。为程序加入注解。

4.2.2.2 任务命令

任务命令组主要用于控制任务的执行时序、获取任务状态以及生成任务 ID。

图 4.9 任务命令

如图 4.9 所示为任务命令组下所包含的各个命令，从左到右依次为：

- 启动任务。启动命令中指定的 MotionTask。
- 中断任务。将指定的 MotionTask 中断在当前状态，但是已经执行的轴运动命令并不会被中断。被中断的 MotionTask 不会自动继续执行。
- 继续任务。继续执行指定的被中断的 MotionTask。
- 停止任务。停止指定的 MotionTask。
- 任务状态。查询指定的 MotionTask 的执行状态，并查询到的结果赋给制定的变量。
- 获取 TaskID。该命令根据指定的 MotionTask 名称生成项目唯一的 TaskID。

4.2.2.3 程序结构命令

程序结构命令包含控制语句执行顺序的条件、循环、跳转等命令。

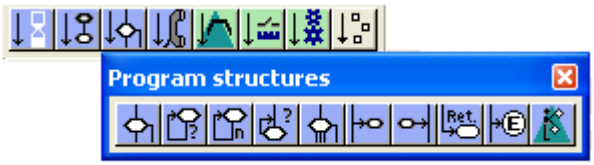


图 4.10 程序结构命令

如图 4.10 所示为程序结构命令组下所包含的各个命令，从左到右依次为：

- IF。条件命令，根据条件为 YES 活 NO，执行不同的分支。
- WHILE。循环命令，循环条件的判断在每次循环的开始，条件成立时执行循环体内的程序，否则跳出循环。
- FOR。执行制定次数的循环。
- UNTIL。循环命令，循环条件的判断在每次循环的结束，条件成立时执行循环体内的程序，否则跳出循环。
- CASE。条件命令，根据条件的值执行不同的分支，一般为多个分支。
- GO TO。跳转命令，使程序跳转到指定的标签处。
- SELECTION。跳转入口，用于指定标签。

- RETURN。用于结束 FC，FB，Program 的执行，当 RETURN 命令执行时，其后的语句都将被跳过。
- Exit。用于跳出 WHILE，FOR，UNTIL 循环。
- 同步开始。用于几个命令的并行执行。

4.2.2.4 通讯命令

通讯命令组主要用于系统消息的确认、通讯的建立以及数据交换。

图 4.11 通讯命令

如图 4.11 所示为通讯命令组下所包含的各个命令，从左到右依次为：

- 确认工艺对象报警。用于确认一个或多个工艺对象的所有报警信息。
- 确认指定的工艺对象报警。用于确认单个工艺对象的一个或所有报警信息。
- 到达的消息。该命令指示某个消息的到达，同时可以指定该消息的类型是否为可确认型。命令中的消息必须事先已被组态。
- 出去的消息。该命令指示某个消息即将离开，并且可确认的消息可以在 OP 中确认。命令中的消息必须事先已被组态。
- 建立 TCP/IP 连接。
- 断开 TCP/IP 连接。
- 发送数据。利用该命令可以通过不同的协议发送数据。
- 接收数据。利用该命令可以通过不同的协议接收数据。

4.2.2.5 单轴命令

单轴命令组包含各种单轴的运动命令以及运动无关的轴设置命令等。

图 4.12 单轴命令

如图 4.12 所示为单轴命令组下所包含的各个命令，从左到右依次为：

- 回零。设置位置轴或同步轴的参考坐标原点。
- 轴使能。使能电气轴，这是轴执行运动控制命令的条件之一。
- 轴去使能。去使能电气轴。
- 液压轴使能。使能液压轴，这是液压轴执行运动控制命令的条件之一。
- 液压轴去使能。去使能液压轴。
- 位置控制模式下的轴移动命令。在位置控制模式下，使位置轴或同步轴加速或减速到给定速度后保持恒速运动。
- 速度控制模式下的轴移动命令。在速度控制模式下，使轴加速或减速到给定速度后保持恒速运动。
- 开启扭矩限幅。
- 关闭扭矩限幅。
- 轴定位命令。使轴运动到指定位置，可以是相对或绝对方式。
- 运动到固定点。激活对运动到固定点的监控，达到固定点后保持钳位扭矩。
- 移初固定点。取消对运动到固定点的监控，达到固定点后不保持钳位扭矩。
- 基于时间的位置曲线。该命令使轴按照 CAM 曲线运动，其中曲线的 X 轴为时间，Y 轴为位置值。
- 基于时间的速度曲线。该命令使轴按照 CAM 曲线运动，其中曲线的 X 轴为时间，Y 轴为速度值。
- 轴停止命令。停止一根轴的运动。停止方式主要分为正常停止和快停，其中正常停止使用于所有单轴运动，但不适用于停止同步运动，快停不仅适用于所有单轴运动，而且可用于停止同步运动。
- 继续运动。对于在轴停止命令中选择 Normal stop without abort 停止方式的轴，可以用该命令使轴继续停止前的运动。
- 转换测量系统。重新定义位置值，可以选择重新定义的位置值是实际值还是给定值。
- 在线校正。可以对选择的工艺对象进行速度和加速度校正，且校正对当前命令及后序命令都有效。
- 轴参数设置。改变指定轴的速度、加速度、加加速度等缺省值。
- 设定虚轴参数。将实轴或者外部编码器的位置、速度、加速度值传递给虚轴，只有在实轴或外部编码器运动时值才能传递给虚轴。
- 删除命令队列。删除命令缓冲区中还没有被执行的命令。
- 转换参数组。为轴选择不同的参数组。

4.2.2.6 外部编码器、测量输入点以及 Output Cam 命令

图 4.13 外部编码器、Measuring Inputs 以及 Output Cam 命令

如图 4.13 所示为外部编码器、Measuring Inputs 以及 Output Cam 命令组下所包含的各个命令，从左到右依次为：

- 打开外部编码器。
- 关闭外部编码器。
- 同步外部编码器。对编码器进行回零。
- 打开编码器监控。要求必须至少有两个编码器系统接到同一根轴。两编码器系统之差被监控，超过最大允许误差时将产生报警。
- 关闭编码器监控。
- 激活测量输入点功能。激活轴或外部编码器的测量输入点功能，当测量输入点有信号时，轴或外部编码器的当前位置被记录。
- 关闭测量输入点功能。
- 同步测量系统。同步两个测量系统或者返回制定测量系统之间的差值。
- 打开 Output Cam。设置 Output Cam 的参数并开启它。
- 关闭 Output Cam。
- 打开/关闭 Output Cam。该命令会关闭当前已经打开的 Output Cam，另外时间偏移只对高速 Output Cam 有效。
- 打开 Output Cam Track。设置 Output Cam Track 的参数并开启它。
- 关闭 Output Cam Track。

4.2.2.7 同步操作命令

同步操作命令组包含两轴间的齿轮同步、凸轮同步操作命令，以及同步参数设置命令等。

图 4.14 同步操作命令

如图 4.14 所示为同步操作命令组下所包含的各个命令，从左到右依次为：

- 开启齿轮同步。
- 关闭齿轮同步。
- 设置齿轮同步偏差。在齿轮同步操作过程中，为主轴或从轴设置偏差。
- 开启速度同步。
- 关闭速度同步。
- 开启凸轮同步。
- 关闭凸轮同步。
- 设置凸轮缩放比例。在凸轮同步操作过程中，设置主轴或从轴的缩放比例。
- 设置凸轮同步偏差。在凸轮同步操作过程中，为主轴或从轴设置偏差。
- 切换主轴。切换同步关系中的主轴，切换可以发生的同步操作过程中。
- 参数化 CAM 曲线。设置 CAM 曲线的偏差和/或缩放比例。

4.2.3 数据类型和变量定义

4.2.3.1 数据类型

数据类型决定了一个变量或常量如何在程序中被使用。SIMOTION 中的数据类型主要有基本数据类型、用户定义的数据类型、工艺对象数据类型以及系统数据类型四种。

1. 基本数据类型

基本数据类型占有固定长度的内存区，不能再被分为更小的数据单元，主要包括二进制数据、整数、浮点数、时间、日期、字符串等。具体参考下表：

表 4.1 基本数据类型

类型	保留字	长度	范围
二 位	BOOL	1	0, 1 或 FALSE, TRUE
进 节	BYTE	8	16#0～16#FF

制类型	字	WORD	16	16#0～16#FFFF
	双字	DWORD	32	16#0～16#FFFF_FFFF
数字量类型	短整型	SINT	8	-128 ～127
	无符号短整型	USINT	8	0～255
	整型	INT	16	-32768～32767
	无符号整型	UINT	16	0～65535
	长整型	DINT	32	-2147483648～2147483647
	无符号长整型	UDINT	32	0 ～4294967295
	浮点型	REAL	32	-3.402823466E+38 ～ -1.175494351E-38, 0.0, +1.175494351E-38～+3.402823466E+38
	双精度浮点型	LREAL	64	-1.7976931348623158E+308～ -2.2250738585072014E-308, 0.0, +2.2250738585072014E-308 ～ +1.7976931348623158E+308
时间类型	时间段	TIME	32	T#0d0h0m0s0ms ～ T#49d17h2m47s295ms
	日期	DATE	32	D#1992-01-01 ～ D#2200-12-31
	时间	TIME_OF_DAY(TOD)	32	TOD#0:0:0.0 ～ TOD#23:59:59.999
	日期和时间	DATE_AND_TIME(DT)	64	DT#1992-01-01-0:0:0.0 to DT#2200-12-31-23:59:59.999
	字符串	STRING	8	ASCII 码的所有字符

除了以上基本数据类型外，还有几种通用数据类型，每种通用数据类型可以代表多个基本数据类型。通用数据类型一般用于 FC 和 FB 的输入/输出接口。如表所示。

标 4.2 通用数据类型

通用数据类型	包含的数据类型

ANY_BIT	BOOL，BYTE，WORD，DWORD
ANY_INT	SINT，INT，DINT，USINT，UINT，UDINT
ANY_REAL	REAL， LREAL
ANY_NUM	ANY_INT，ANY_REAL
ANY_DATE	DATE，TIME_OF_DAY (TOD)，DATE_AND_TIME (DT)
ANY_ELEMENTARY	ANY_BIT，ANY_NUM，ANY_DATE，TIME，STRING
ANY	ANY_ELEMENTARY，用户定义的数据类型，系统数据类型和工艺对象数据类型

2. 用户定义的数据类型（UDT）

用户定义的数据类型包括结构体和枚举类型。

图 4.15 中定义了名为 myStructure 的结构体，它包含四个元素（e1，e2，e3，e4），在 Data type 一栏中可以为每个元素指定数据类型，元素的数据类型可以是基本数据类型也可以是用户定义的数据类型。

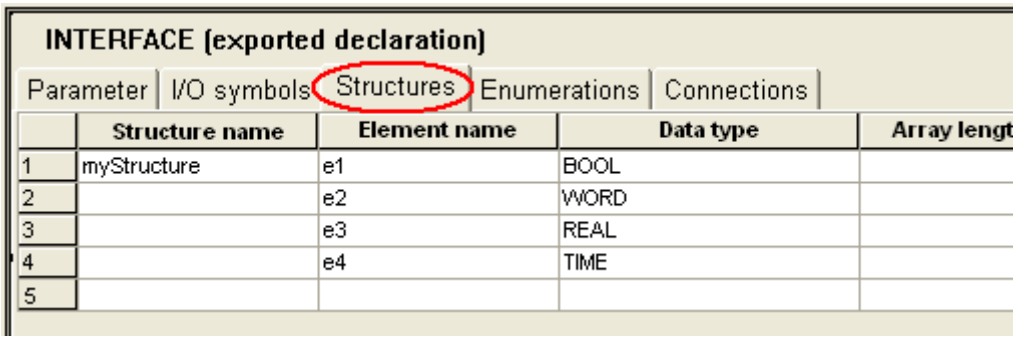


图 4.15 结构体

图 4.16 中定义了名为 Color 的枚举类型，它包含三个元素（red，blue，green），枚举类型变量的值可以是三个元素中的任何一个，Initialization value 栏中的 blue 表示枚举类型变量的缺省值为 blue。

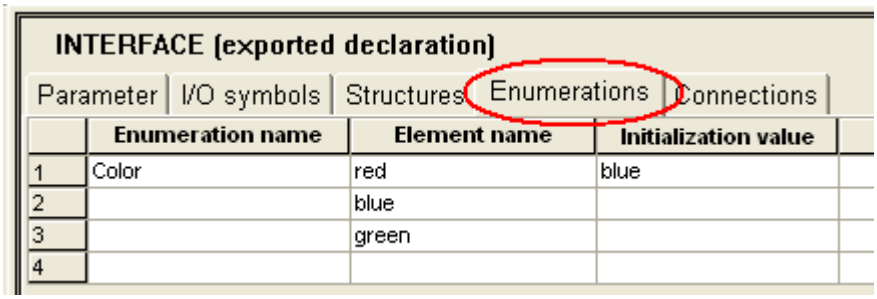


图 4.16 枚举类型

3. 工艺对象数据类型

以工艺对象（例如 DriveAxis、CAM、Output CAM 等）作为数据类型的变量称为工艺对象数据类型。具体请参考 SIMOTION 基本功能手册。

4. 系统数据类型

这类数据类型由系统定义，一般为结构体和枚举类型，用户可以直接使用，具体请参考 SIMOTION 基本功能手册。

4.2.3.2 变量定义

根据变量的作用范围和性质，SIMOTION 中的变量可以分为系统变量全局用户变量和局部用户变量三中。下面分别介绍各种类型的变量及其定义。

1. 系统变量

系统变量又分为 SIMOTION 设备变量和工艺对象变量，这些变量在建立项目或者新建工艺对象的时候由系统自动产生，用户不能对其进行更改，主要用来描述 SIMOTION 设备或者工艺对象的系统特性。在项目导航栏中选中 SIMOTION 设备或者某个工艺对象，其对应的系统变量便显示在界面下方详细信息栏的 Symbol Browser 中。例如图 4.17 中 Symbol Browser 中列出的是 SIMOTION 设备 D435 的系统变量。

系统变量的作用域为整个项目，可以被所有 Program、FC、FB 访问，也可以被 HMI 访问。

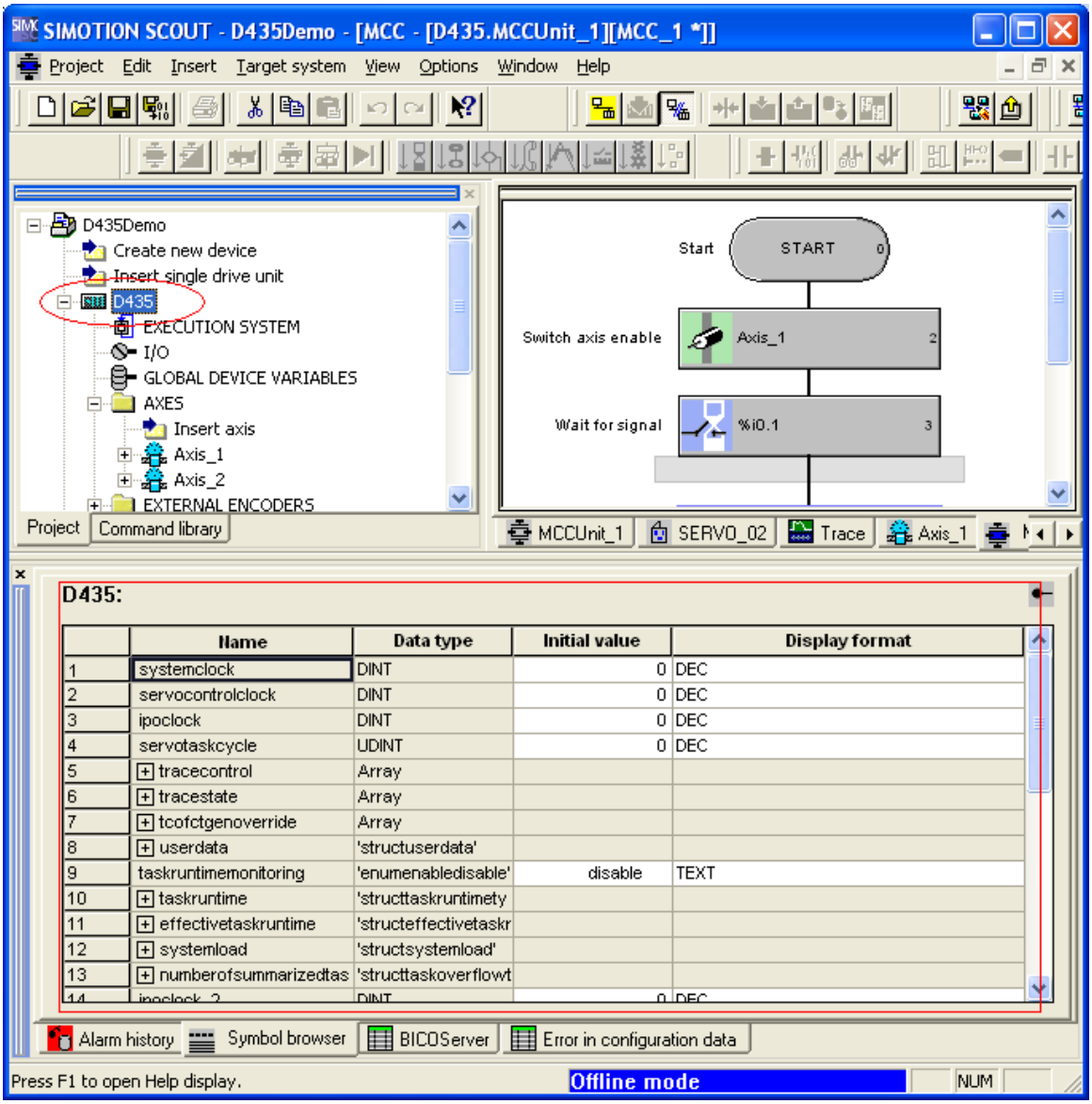


图 4.17 系统变量

2. 全局用户变量

全局用户变量包括 I/O 变量、全局设备变量以及单元变量。

I/O 变量和全局设备变量对应于项目导航栏中 I/O 和 GLOBAL DEVICE VARIABLES。例如图 4.18，在选中 GLOBAL DEVICE VARIABLES 后，用户便可以在详细信息栏的 Symbol Browser 中定义全局设备变量。I/O 变量和全局设备变量的作用域为整个项目，可以被所有 Program、FC、FB 访问，也可以被 HMI 访问。

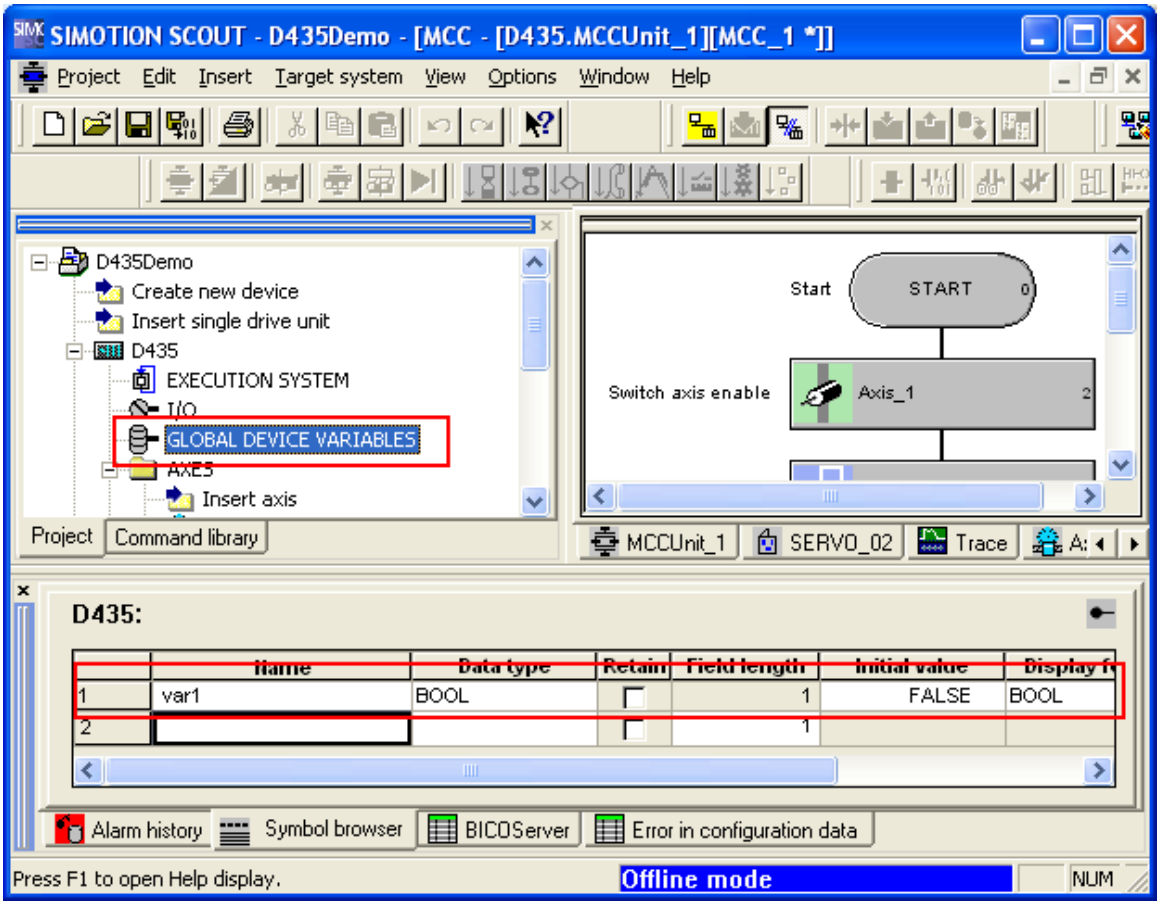


图 4.18 全局设备变量

单元变量是指用户在程序单元（Unit）的 INTERFACE 部分和 IMPLEMENTATION 部分定义的变量。图 4.19 显示的是在 MCCUnit_1 中定义单元变量。双击 MCC Unit 名称“MCCUnit_1”后，工作区中上半部分为 INTERFACE 部分，在这里定义的变量不仅可以被 HMI 以及本单元中所有 Program、FC、FB 访问，而且在本单元（MCCUnit_1）被其他单元引用后，该区域中定义的单元变量还可以被其他单元使用；工作区的下半部分为 IMPLEMENTATION 部分，该区域中定义的变量只能被本单元中的 Program、FC、FB 访问，不能被其他单元访问，也不能用 HMI 对其进行读写。

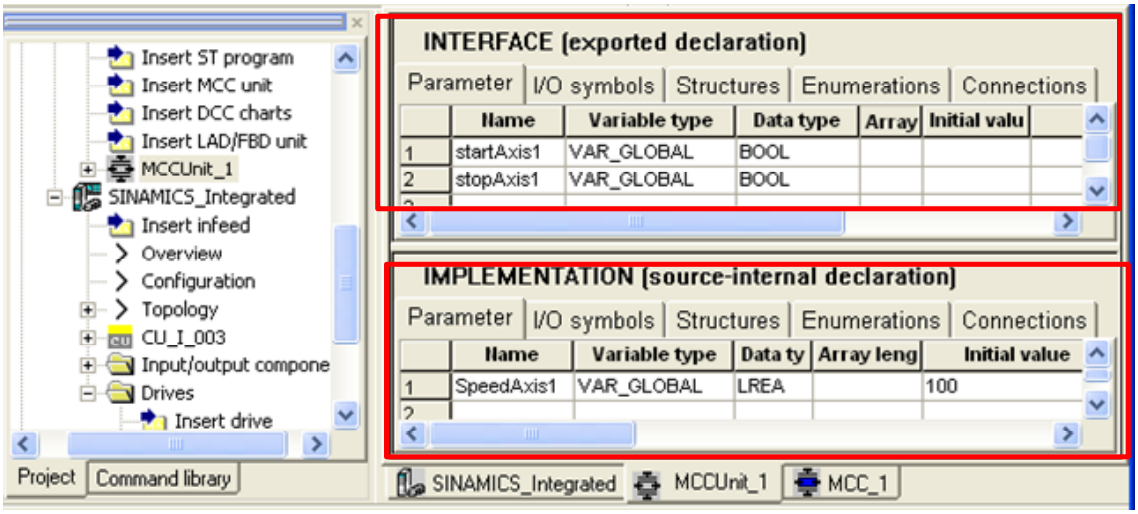


图 4.19 单元变量

图 4.19 中 Variable type 栏可以指定变量为全局类型（VAR_GLOBAL）、全局保持类型（VAR_GLOBAL_RETAIN）或全局常量（VAR_GLOBAL_CONSTANT）。

一个单元要引用另一个单元中的变量时，只要在单元界面的 Connections 标签页中添加对另一个单元的引用即可。图 4.20 中，MCCUnit_2 添加了对 MCCUnit_1 的引用，因此 MCCUnit_2 中的所有 Program、FC、FB 都可以访问 MCCUnit_1 中 INTERFACE 部分定义的单元变量。

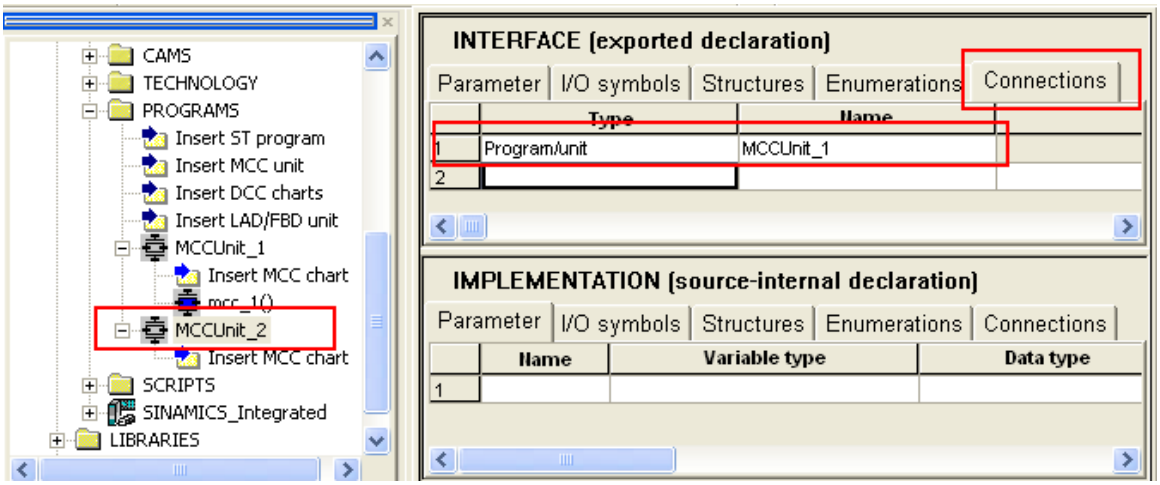


图 4.20 单元的引用

3. 局部用户变量

局部用户变量是指用户在 Program、FC、FB 中定义的变量。局部用户变量只能被定义它的 Program、FC 或 FB 访问，例如在某 Program 中定义的变量只

能被该 Program 访问，而不能其他的 Program、FC 和 FB 访问。以 MCC 编程为例，MCC Chart 中定义的变量即为局部用户变量。如图 4.21 所示，Variable type 中可指定变量为局部变量（VAR）、局部临时变量（VAR_TEMP）或者局部常量（VAR_CONSTANT）。局部临时变量所在程序被重新调用时都自动恢复为初始值。

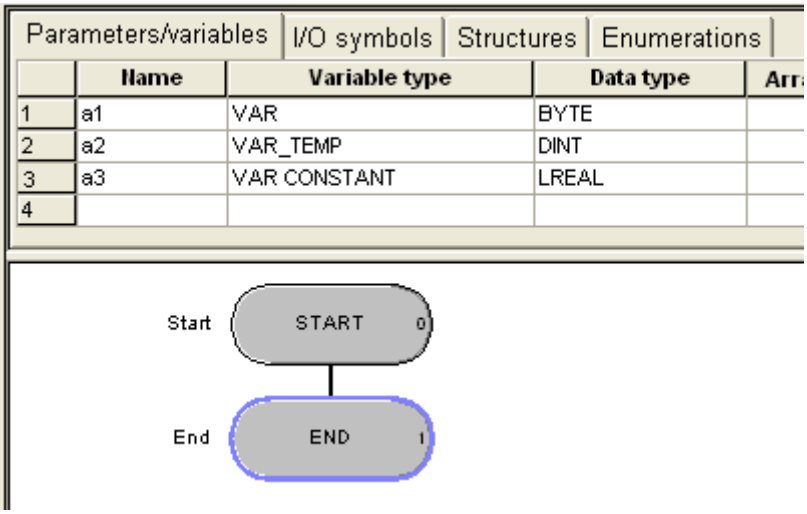


图 4.21 局部用户变量

4.2.4 子程序调用

通常，可重复使用的程序段可以子程序的形式被其他程序调用，从而省去重复编程的麻烦。子程序被调用的时候，程序的执行从主程序进入到子程序，当子程序执行完后再返回到主程序中继续未执行的语句，流程如图 4.22 所示。

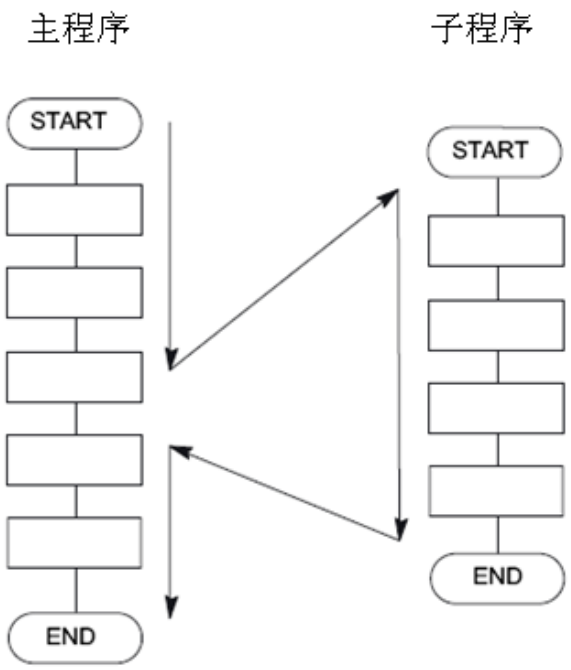


图 4.22 程序执行流程

作为主程序或子程序的可以是 FC、FB 或者 Program。

1. FC

FC 作为子程序时不带有静态数据，FC 中的局部变量在 FC 被重新调用的时候恢复到初始值。FC 可以通过其输入参数或者输入/输出参数从外部获得数据，也可以通过返回值向主程序返回数据（FC 以其名称作为返回值变量）。下面以实例说明 FC 的建立及其调用。

实例：建立一个计算圆周长的 FC，名称为 Circumference。圆周长计算公式为 Circumference=PI*2*radius，Radius 和 PI 变量在 FC 的声明部分定义。

步骤：

（1）插入一个 MCC Chart，将其命名为 Circumference，在 Creation Type 栏中选择 Function，Return type 为 REAL 类型。如图 4.23 所示。

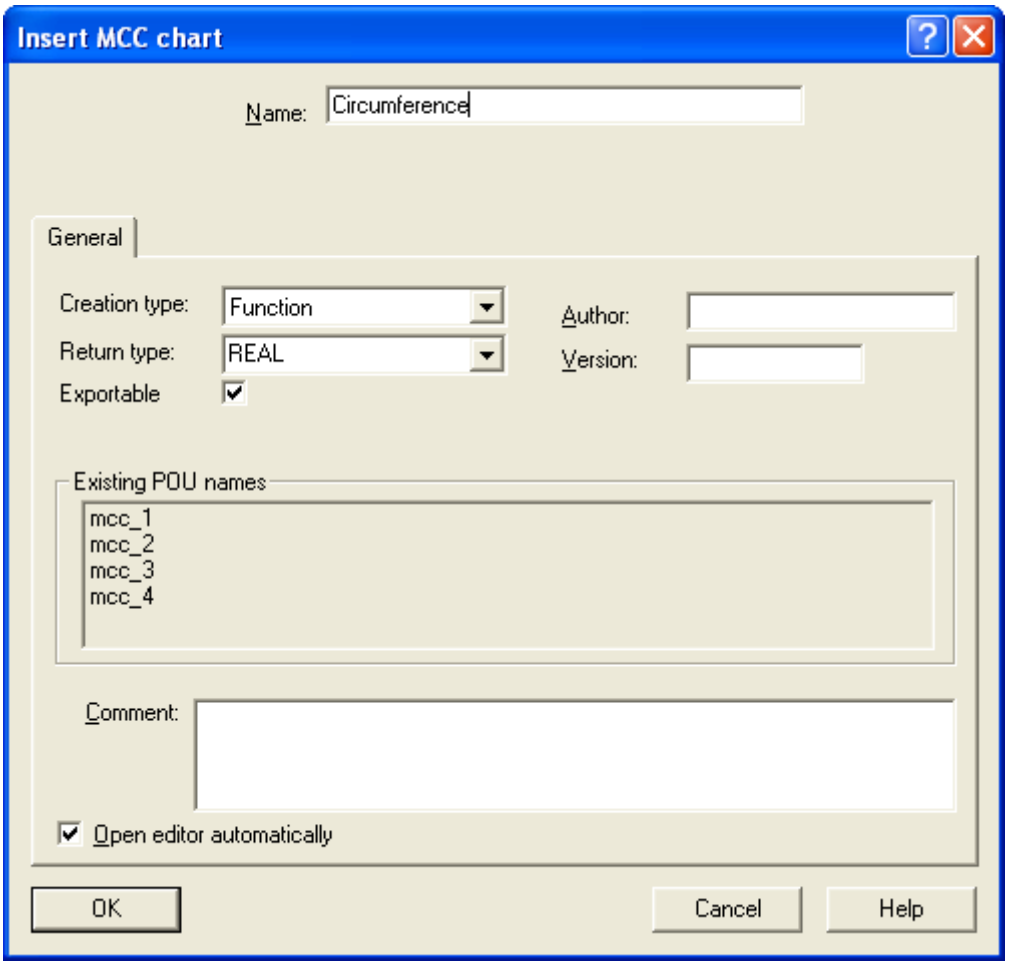


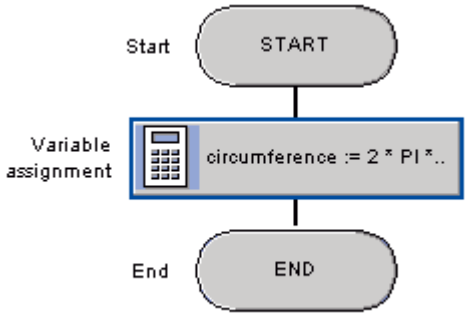
图 4.23 新建 FC

（2）在 FC 的变量声明中定义 Radius 和 PI 变量，如图 4.24 所示。

	Name	Variable type	Data type	Array length	Initial value	Comment
1	radius	VAR_INPUT	REAL			
2	PI	VAR CONSTANT	REAL		3.14159	
3						

图 4.24 定义 FC 的变量

（3）调用变量赋值命令，计算圆周长。（注意：FC 的名称即为返回值变量）如图 4.25 所示。



	Variable	:=	Expression
1	circumference	:=	2 * PI * radius
2		:=	

图 4.25 计算圆周长

（4）保存编译 FC。

（5）在同一个 MCC Unit 中插入一个新的 MCC Chart，Creation type 为 Program。

（6）在 Program 的变量声明中定义如图 4.26 所示的变量用于为 FC 参数赋值并获取其返回值。

	Name	Variable type	Data type	Array length	Initial value	Comment
1	mycircum	VAR	REAL			
2	myradius	VAR	REAL			
3						

图 4.26 在 Program 中定义变量

（7）插入子程序调用命令，并按照图 4.27 所示设置参数。

（8）保存编译 Program。

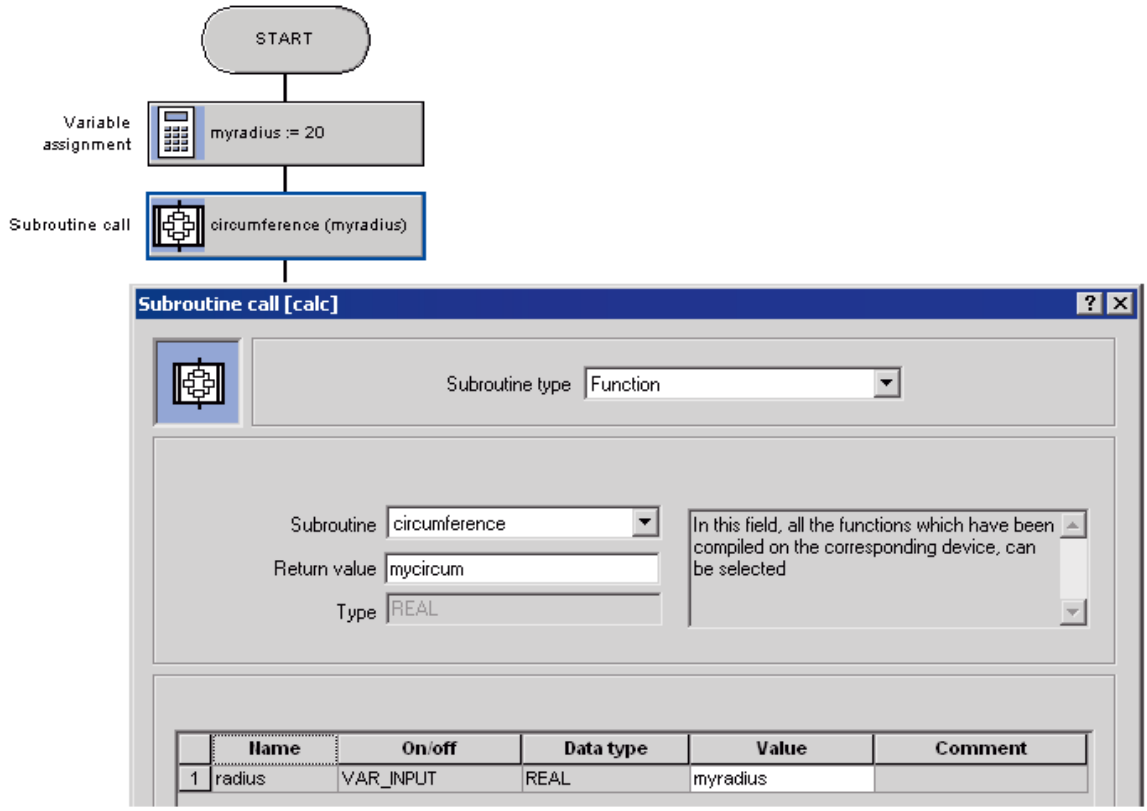
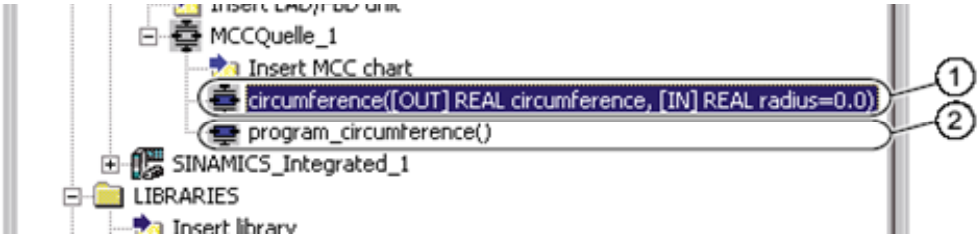


图 4.27 调用 FC

注意：在项目导航栏中，同一个 MCC Unit 中的子程序必须在调用它的主程序上面。右键点击 MCC Chart 在弹出的菜单中选择 Up 或 Down 可以调整其在 MCC Unit 中位置。到这里便完成了 FC 的定义及其调用，从项目导航栏中可以看到刚刚建立的两个 MCC Chart，如图 4.28 所示。



① 作为FC的MCC Chart
② 作为Program的MCC Chart

图 4.28 MCC Chart 的顺序

2. FB

FB 作为子程序时带有静态数据类，FB 中的局部变量（用 VAR 定义）在 FB 执行完后能够保留其值，即在下一次调用该 FB 时局部变量保留上一次调用结束时的值，只有用 VAR_TEMP 定义的局部临时变量才会在每次 FB 被调用的

时候重新初始化。在使用 FB 时，需要先定义一个 FB 的实例，FB 的静态类型数据都保存在实例中。同一个 FB 可以定义多个实例，而每个实例之间是独立。FB 可以通过其输入参数或者输入/输出参数从外部获得数据，也可以通过输出参数或者输入/输出参数向主程序返回数据。下面以实例说明 FB 的建立及其调用。

实例：建立一个计算跟随误差的 FB，名称为 FollError。跟随误差计算公式为 Difference = Set position – Actual position，Set position、Actual position 和 Difference 在 FB 的声明部分定义。

步骤：

（1）插入一个 MCC Chart，将其命名为 FollErr，在 Creation Type 栏中选择 Function block。如图 4.29 所示。

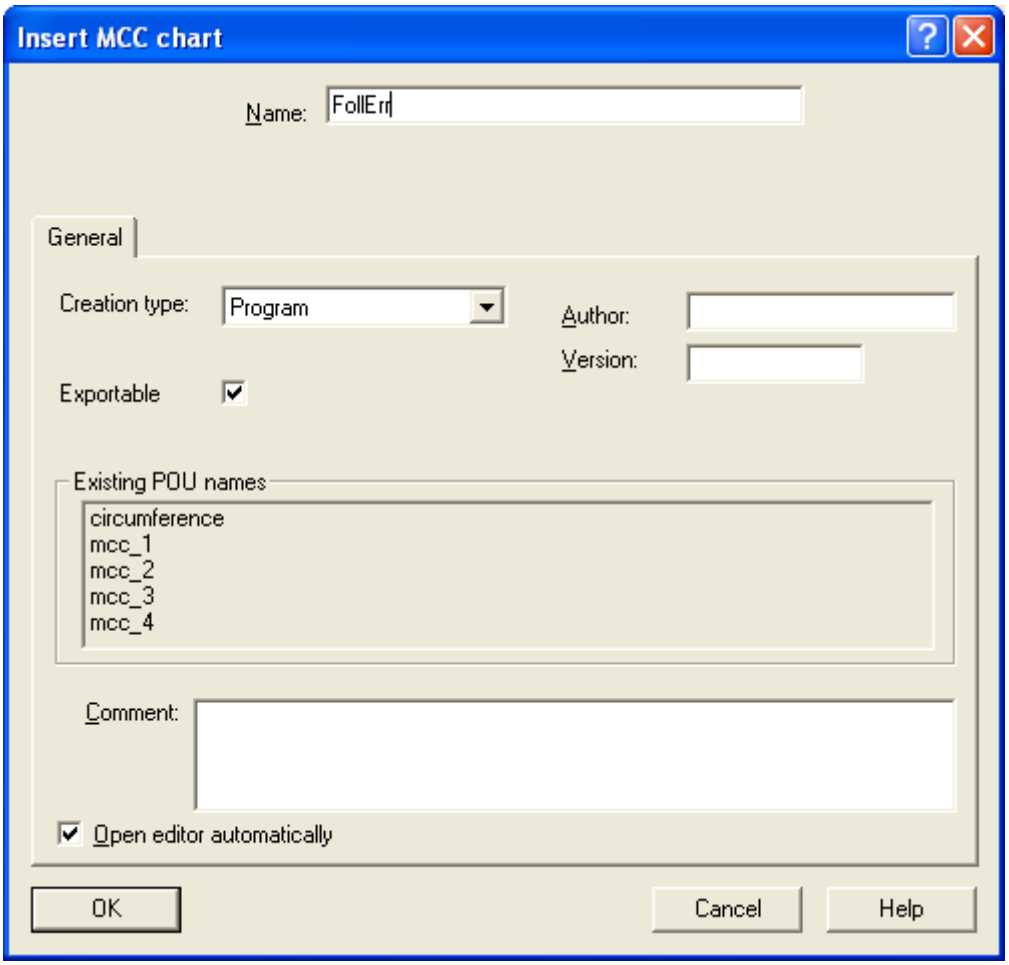


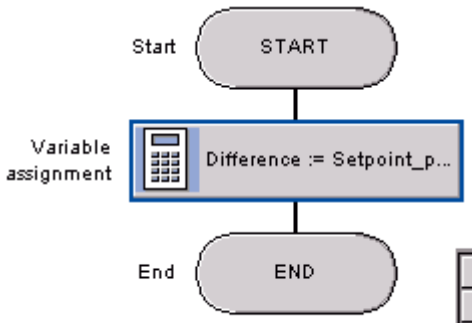
图 4.29 新建 FB

（2）在 FB 的变量声明中定义输出与输出变量，如图 4.30 所示。

	Name	Variable type	Data type	Array length	Initial value	Comment
1	Setpoint_position	VAR_INPUT	LREAL			
2	Actual_position	VAR_INPUT	LREAL			
3	Difference	VAR_OUTPUT	LREAL			
4						

图 4.30 定义 FB 的变量

（3）调用变量赋值命令，计算跟随误差，如图 4.31 所示。



Formula language

	Variable	:=	Expression
1	Difference	:=	Setpoint_position - Actual_position
2		:=	

图 4.31 计算跟随误差

（4）保存编译 FB。

（5）在同一个 MCC Unit 中插入一个新的 MCC Chart，Creation type 为 Program。

（6）在 Program 的变量声明中定义 FB 的实例及相关变量以获取 FB 的返回值，如图 4.32 所示，其中 Result 用于调用 FB 时获取跟随误差返回值，Result_2 用于在 FB 调用后获取跟随误差值。

	Name	Variable type	Data type	Array length	Initial value	Comment
1	myFollErr	VAR	FollErr			
2	Result	VAR	LREAL			
3	Result_2	VAR	LREAL			
4						

图 4.32 在 Program 中定义变量

（7）插入子程序调用命令，并按照图 4.33 所示设置参数，赋给 FB 的两个输入参数的是轴对象变量，可以直接从 Symbol browser 中拖入到子程序调用命令窗口的对应位置。

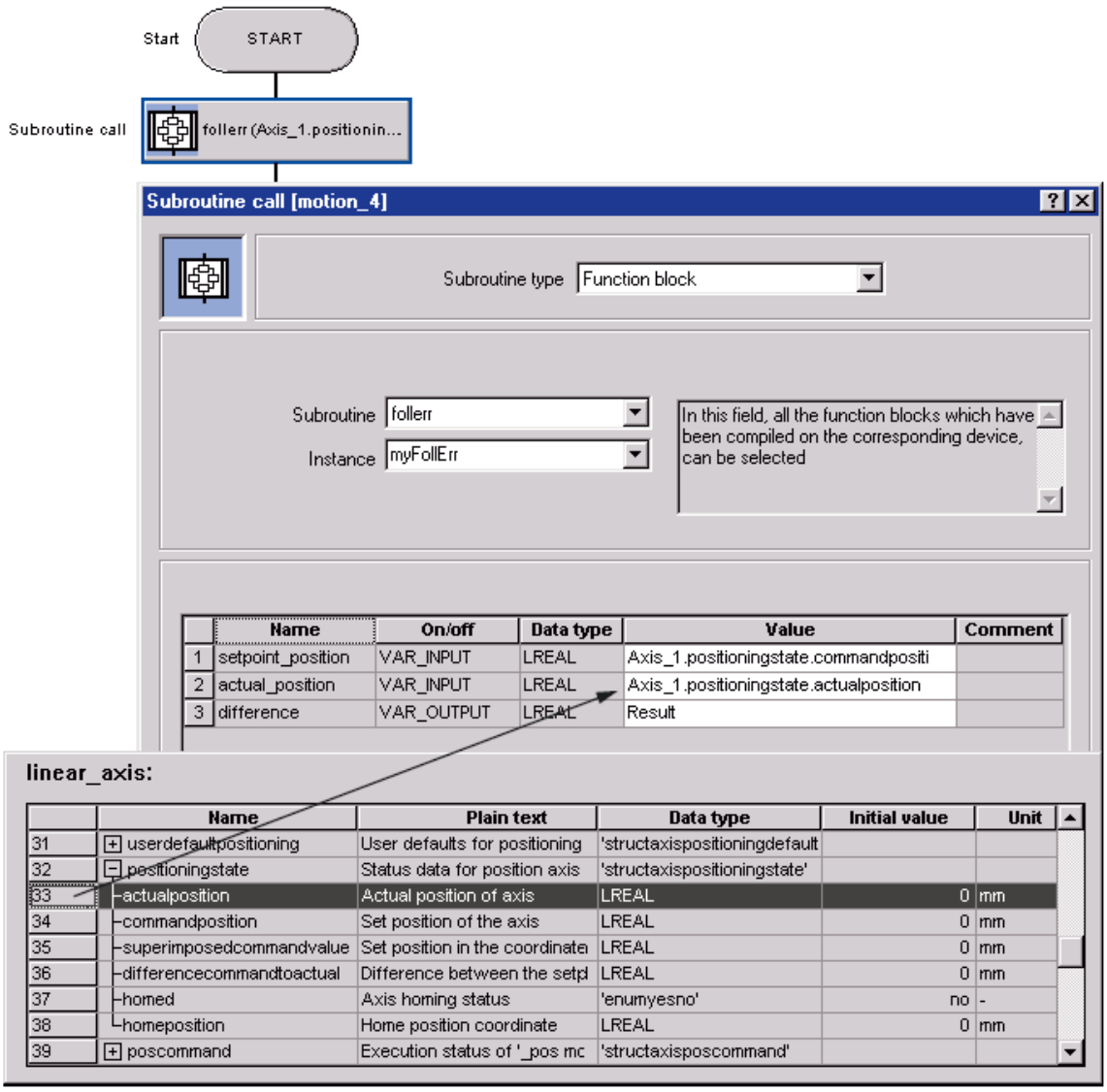


图 4.33 调用 FB

（8）访问 FB 的输出参数。FB 被调用后其静态变量的值能够保留，因此在主程序的任何地方仍然可以访问 FB 的输出参数。如图 4.34 所示，调用变量赋值命令，将 FB 的 Difference 参数赋给 Result_2。

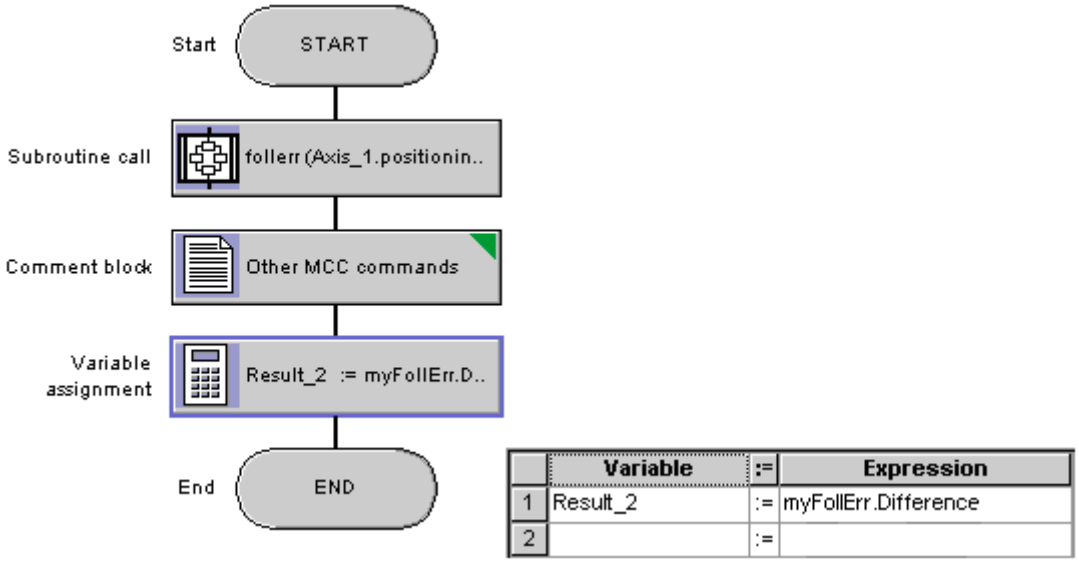


图 4.34 访问 FB 的输出参数

（9）保存编译 Program。到这里便完成了 FB 的定义及其调用。

3. Program

Program 作为子程序时，调用它的主程序可以是另一个 Program 和 FB，另外子程序和主程序所在程序单元（Unit）的编译选项需要做相应的设置，如图 4.35 所示，主程序所在程序单元必须钩选 Permit language extensions 选项，而子程序所在程序单元必须钩选 Only create program instance data once 选项。

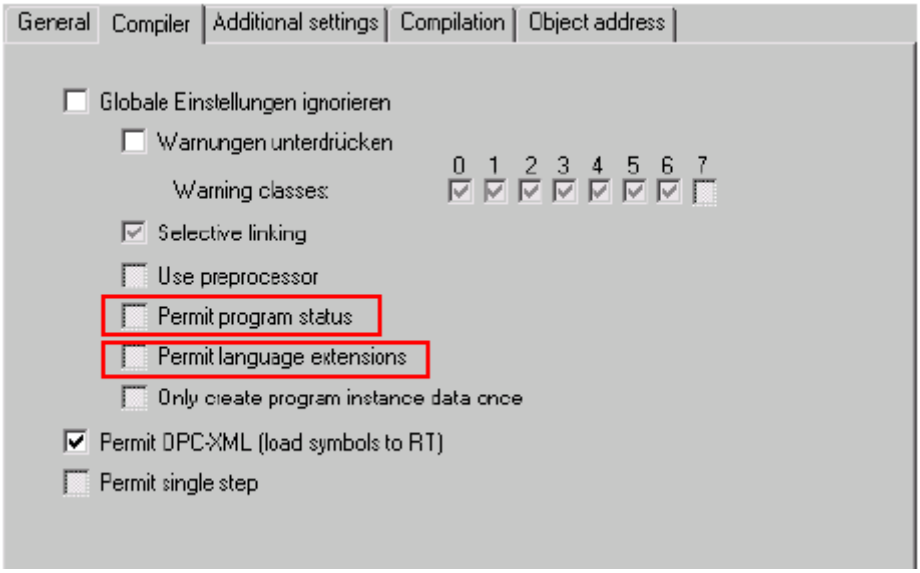


图 4.35 编译选项设置

4.3 LAD

梯形图与电器控制系统的电路图很相似，具有直观易懂的优点，很容易被工厂电气人员掌握，特别适用于开关量逻辑控制。

梯形图中的某些编程元件沿用了继电器这一名称，如输入继电器、输出继电器、内部辅助继电器等，但是它们不是真实的物理继电器，而是一些存储单元（软继电器），每一软继电器对应于存储器中映像寄存器的一个存储单元。梯形图按从左至右、从上到下的顺序执行的，上一个网络的执行结果可以立即被下一个网络使用。梯形图和功能块图（FBD）之间可以互相切换。图 4.36 为 SIMOTION 的梯形图编程界面。

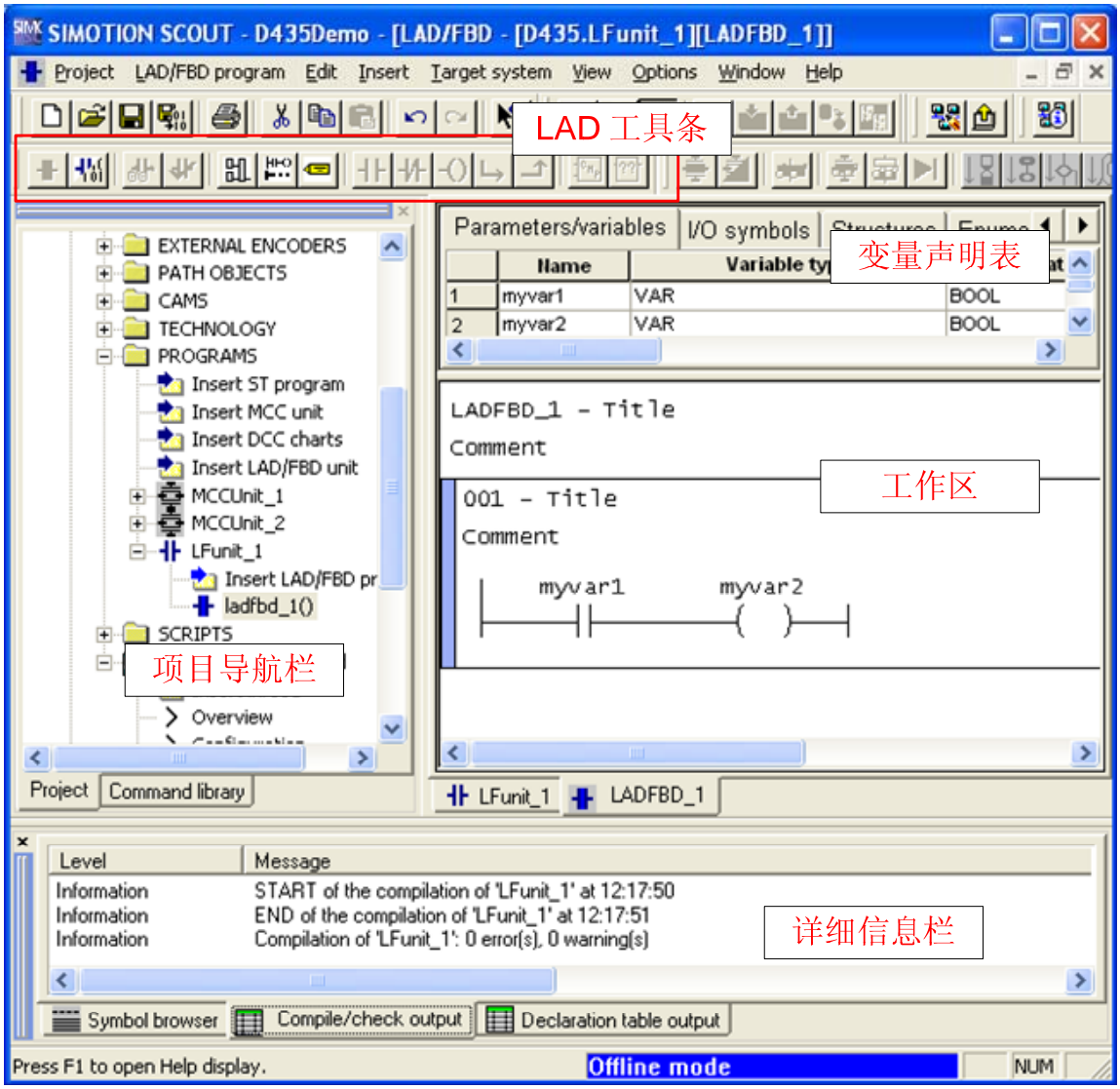
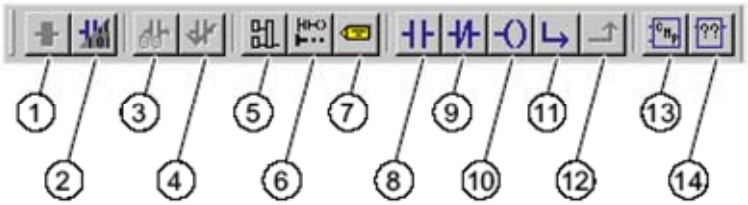


图 4.36 梯形图编程界面

LAD 工具条包含了一些重要的常用的命令。如图 4.37 所示。



① 插入LAD/FBD Unit
② 保存编译程序
③ 程序状态
④ 符号检查和类型更新
⑤ 切换到FBD
⑥ 插入网络
⑦ 打开/关闭跳转标签
⑧ 插入常开出点
⑨ 插入常闭触电
⑩ 插入线圈
⑪ 打开分支
⑫ 关闭分支
⑬ 插入比较器
⑭ 插入一个空块

图 4.37 LAD 工具条

除工具条中的命令外，更多的命令可以在 Command Library 中找到。如图 4.38 所示。

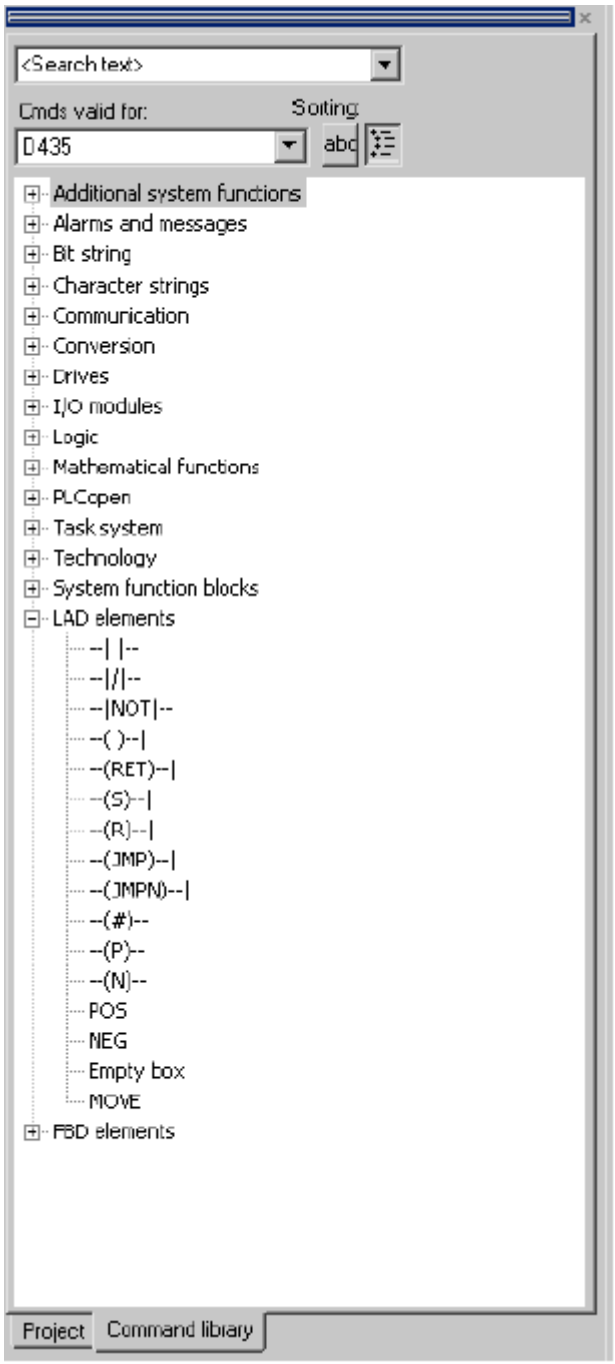


图 4.38 Command Library

同 MCC 类似，LAD 程序也是以 LAD Unit 作为程序组，同一个 LAD Unit 下可以插入多个 LAD Program、FC 和 FB。具体编程规则这里不作更多介绍，详细信息请参考 LAD/FBD 编程手册。

五、常用运动控制功能

5.1 单轴运动控制命令

5.1.1 轴使能命令(Enable Axis)

轴使能命令用于使能电气驱动轴（对于液压驱动轴，使用 Switch QF-axis enable 命令）。

轴必须满足以下条件才能执行运动命令：

- 驱动使能
- 脉冲使能（功率单元使能）
- 对于位置轴和同步轴需要附加条件：位置控制器使能
- 取消 Follow-up operation

5.1.1.1 命令参数说明

参数	说明
Switch position controller enable	只对位置轴和同步轴有效。如果要激活位置控制器则勾选该复选框。 勾选该复选框： • Switch enables individually according to PROFIdrive profiles 复选框被清除。 • Switch drive enable 和 Switch pulse enable 复选框被勾选。 如果该复选框没有被勾选，位置控制器的当前状态不变。 对于虚轴而言，位置控制器使能始终被激活，即使该复选框没有被勾选。 实轴位置控制器使能的当前状态可以通过系统变量 servoMonitorings.controlState 查看。

Switch enables individually according to PROFIdrive profiles	如果想根据 PROFIdrive 行规进行单个使能，则勾选该复选框。 勾选该复选框： • Switch drive enable 和 Switch pulse enable 复选框被隐藏。 • 控制字 1（STW1）的 7 个位的复选框显示。 (1) 勾选对应位的复选框以激活对应使能。 (2) 清除对应位的复选框以保持先前的使能设置。 必须激活所有使能以保证驱动的正常运行。 清除该复选框，Switch drive enable 和 Switch pulse enable 复选框出现。
Switch drive enable	仅当 Switch enables individually according to PROFIdrive profiles 复选框未被勾选时才有效。 如果要激活驱动使能则勾选该复选框。 如果该复选框未被勾选，驱动使能的当前状态保持不变。 实轴驱动使能的当前状态可以通过系统变量 actorMonitorings.driveState 来查询。
Switch pulse enable	仅当 Switch enables individually according to PROFIdrive profiles 复选框未被勾选时才有效。 如果要激活脉冲使能（功率单元使能）则勾选该复选框。 如果该复选框未被勾选，脉冲使能的当前状态保持不变。 实轴驱动使能的当前状态可以通过系统变量 actorMonitorings.power 来查询。
Follow-up mode	✓ Do not follow up setpoint (缺省值) 轴可以执行运动命令。 对于实轴，只有当所有的使能都被激活时 Do not follow up setpoint 才有效。 ✓ Follow up setpoint 轴不能执行运动命令。 可以通过系统变量 control 来判断实轴可不可以执行运动命令。
Traversing mode	✓ Maintain last setting (缺省值) 轴按照最近设置的运行模式（位置和速度控制）被使能。 ✓ Enable for speed- and position-controlled operation 轴使能以进行速度和位置控制操作。 该参数对于驱动轴不可选。 ✓ Enable for speed-controlled operation 轴被使能以进行速度控制操作。 对于位置轴，如果只选择速度控制操作，位置控制器仍然被激活。
Delay program execution	如果希望系统在该命令执行完毕后才执行 MCC chart 下面的命令，则勾选该复选框。 如果没有勾选该复选框，那么下一个命令将会立即执行。

5.1.1.2 范例

例 1：

1. 插入一个新的 MCC Unit，在 Compiler 标签页对编译器进行设置，如勾选 Permit program status 复选框则可以监控程序的运行状态。如图 5.1 所示。

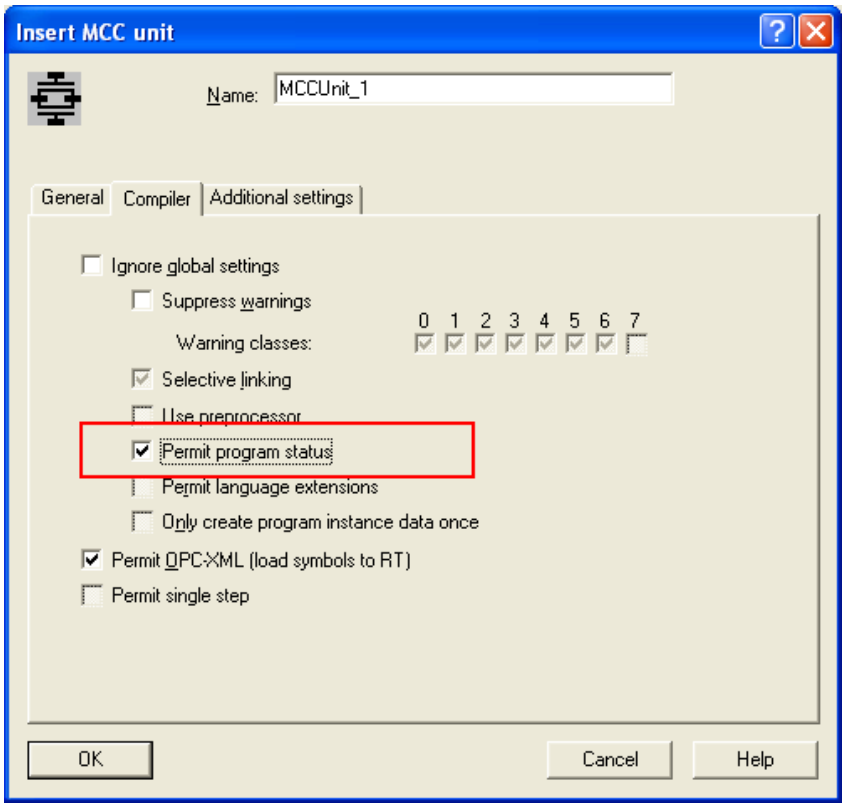


图 5.1 修改编译选项

2. 在 MCCUnit_1 下插入一个 MCC chart，命名为 MCC_EnableAxis。

3. 插入轴使能命令，并根据图 5.2 设置其参数。

1) Axis: 要执行使能操作的轴。选择 Axis_1。

2) 勾选 Switch position controller enable 复选框。

3) Follow-up operation： Do not follow-up setpoint。

4) Traversing mode： Maintain last setting。

5) 勾选 Delay program execution 复选框。系统会在使能轴命令执行完成后再执行下面的命令。

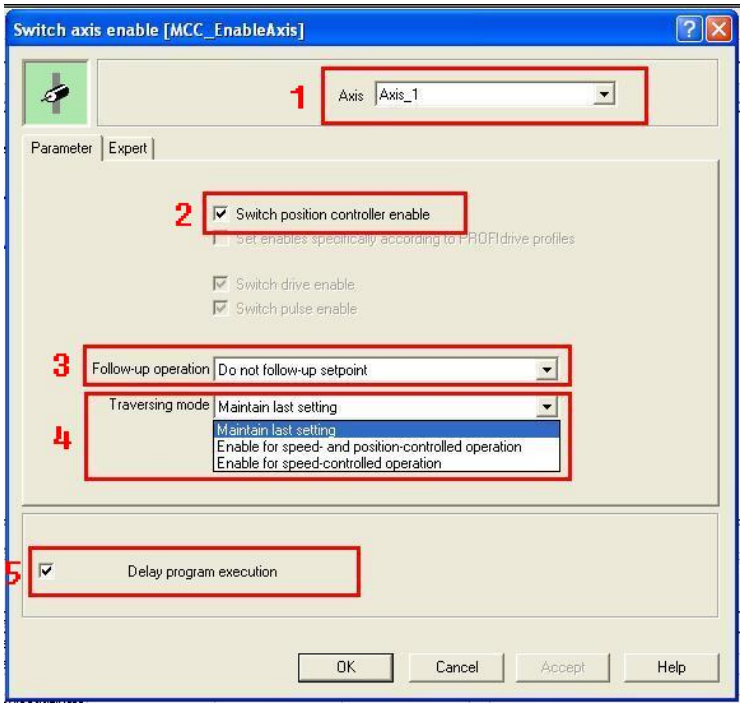


图 5.2 使能轴命令参数设置

4. 为了能够更加方便的监控程序的执行过程，在使能命令之前加入一个等待条件命令，并在 GLOBAL DEVICE VARIABLE 下面新建 Bool 类型变量 g_boStart，将它的高电平作为等待条件。如图 5.3 所示。

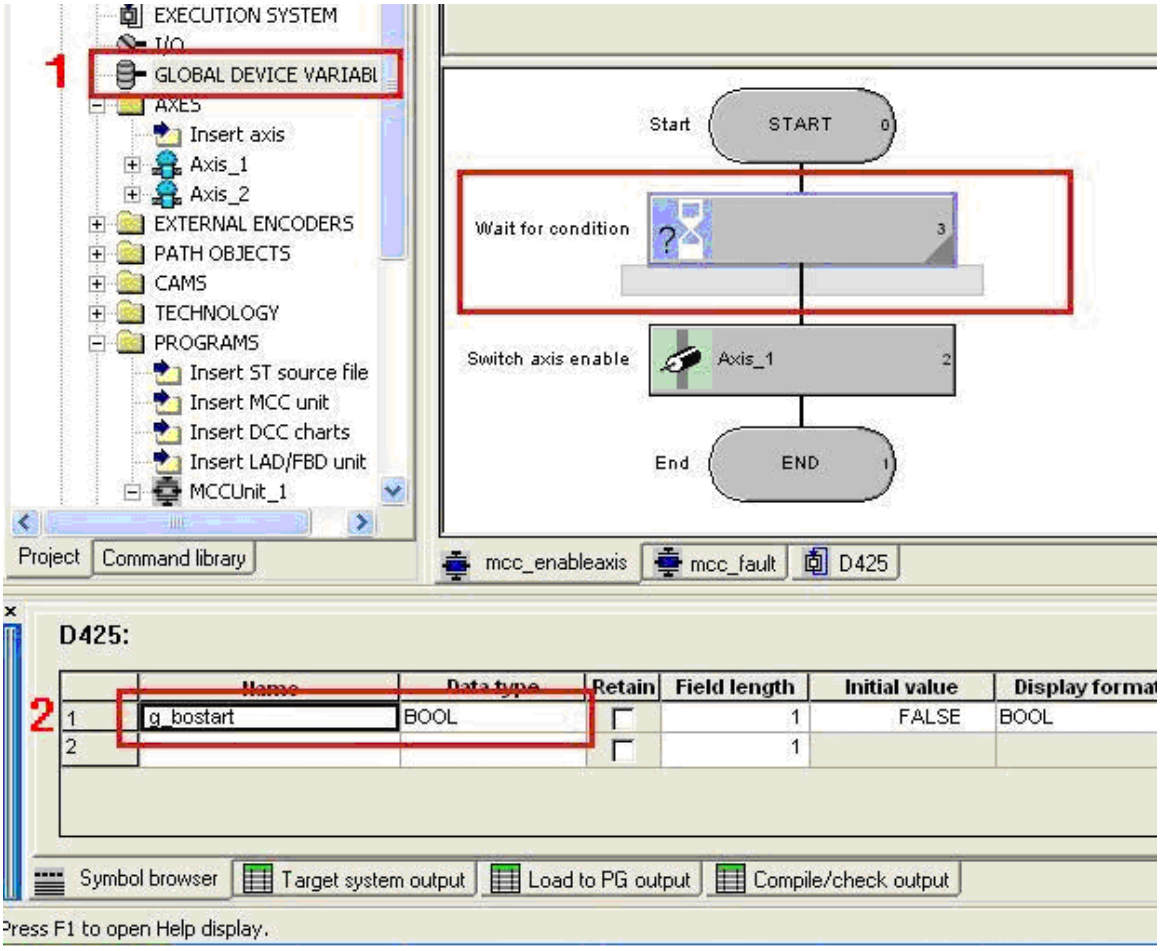


图 5.3 添加变量

5. 点击工具栏上的 Compiler 按钮对程序进行编译保存。

6. 将程序分配给执行系统

将编译无错的 MCC_EnableAxis 分配到 MotionTask_1 中，同时，在 MotionTask_1 的 Task configuration 标签页中，激活 Activation After Startup Task 使分配到 MotionTask_1 中的程序将在设备上电启动进入运行状态后立即自动执行。

7. 在线并下载程序

8. 运行程序

将 Simotion 的操作模式从 STOP 变成 RUN，点击 MCC 工具栏上的 按钮激活程序监控功能。等待条件命令呈黄色显示，表示程序在等待条件满足， 如图 5.4 所示。

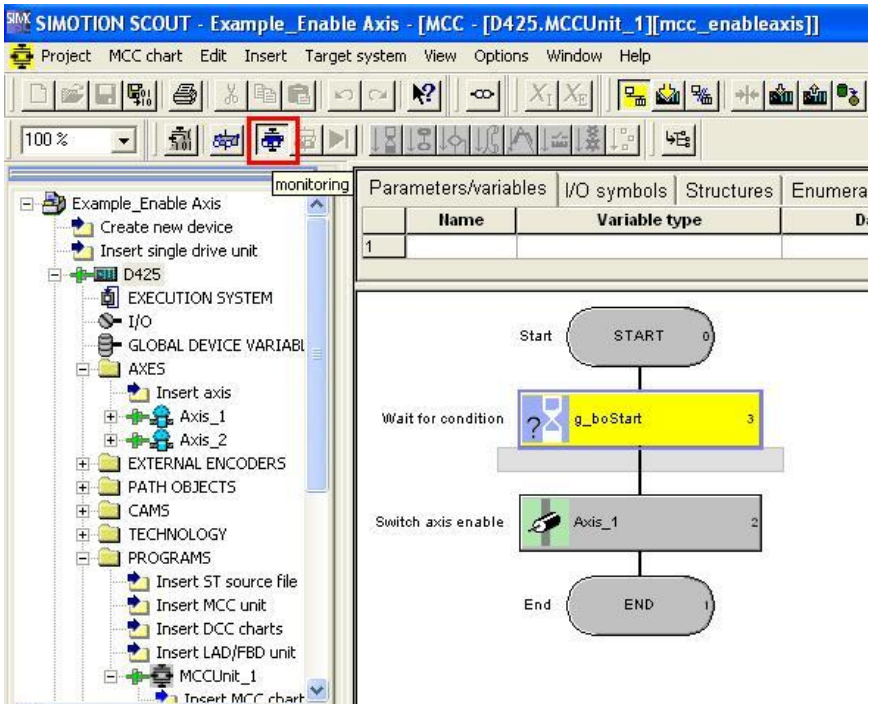


图 5.4 激活程序监控功能

在线修改变量 g_boStart 的值为 TRUE，等待的条件满足，程序继续向下运行，开始执行轴使能命令，该命令执行完成后，整个程序运行结束。

9. 状态查询

轴有没有被使能可以通过系统变量 control 来查看。

方法如下：高亮想要查看的轴，在 Symbol browser 中找到 control 这个系统变量，如果该变量的值为 active，则表明该轴已经被使能，否则轴没有被使能。如图 5.5 所示。

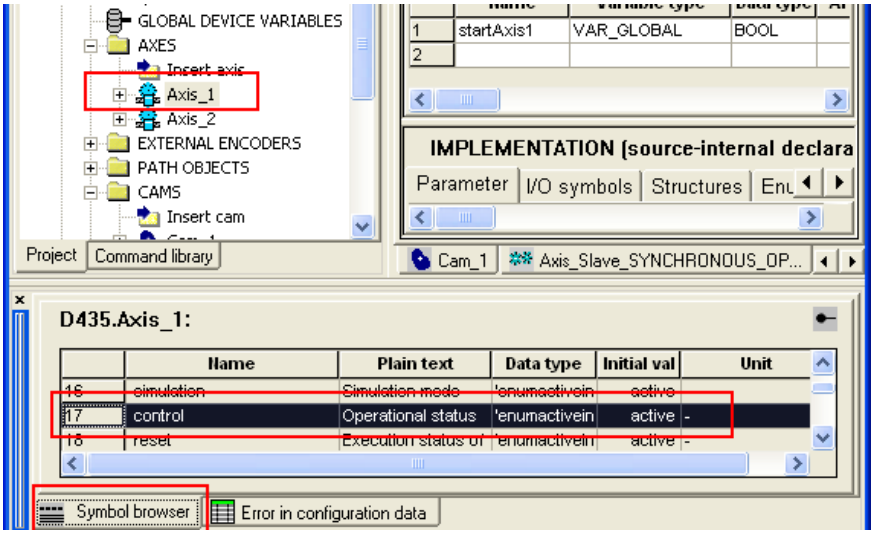


图 5.5 查看轴使能状态

5.1.2 轴回零命令 (Home Axis)

该命令用于回零轴。

对于位置轴或者同步轴，显示或输入的位置值是参考轴的坐标系的。轴的坐标系必须同轴的实际物理位置相关联。

不同类型的编码器，其回零方式有所区别。

对于增量式编码器，每次上电都需要进行回零操作。增量式编码器有三种回零方式：

- 主动回零（Active Homing）
- 被动回零（Passive Homing）
- 设置参考点（Setting Actual Position Value）

1. 主动回零（Active Homing）

主动回零存在三种模式：

- 使用接近开关和编码器零脉冲回零（Homing output cam and encoder zero mark）：轴运动直到遇到撞块后开始寻找第一个编码器零脉冲，找到第一个编码器零脉冲后走指定偏移量再停下来，并将指定的零点位置坐标作为轴当前所在位置。
- 仅用编码器零脉冲回零（Encoder zero mark only）：轴运动直到遇到第一个编码器零脉冲后走指定偏移量再停下来，并将指定的零点位置坐标作为轴当前所在位置。
- 仅用外部零脉冲回零（External zero mark only）：轴运动直到遇到外部零脉冲后走指定偏移量再停下来，并将指定的零点位置坐标作为轴当前所在位置。

2. 被动回零（Passive Homing）

被动回零是指在轴运动过程中（该运动由其他指令产生），遇到回零信号开始回零。被动回零指令本身不能使轴产生运动。它有四种回零模式：

- 带撞块和编码器零脉冲：轴运动过程中，一旦检测到外部撞块则开始寻找接下来的第一个编码器零脉冲，当编码器零脉冲检测到时便将指定的零点位置坐标作为轴当前位置。
- 仅外部零脉冲：轴运动过程中，一旦检测到外部零脉冲信号，则将指定的零点位置坐标作为当前轴的位置。
- 仅编码器零脉冲：轴运动过程中，一旦检测到编码器零脉冲，则将指定的零点位置坐标作为当前轴的位置。
- 缺省设置（取决于编码器）：对于增量式 Sin/Cos 编码器、TTL 编码器或者旋转变压器，采用编码器零脉冲回零；对于 Endat 编码器采用外部零脉冲回零。

3. 设置参考点（Setting Actual Position Value）

设置参考点存在三种模式：

- 直接设置参考点

这种回零方式不能使轴产生运动，它直接将轴的当前位置设置为指定的零点坐标值。

- 相对设置参考点

这种回零方式不能使轴产生运动，它将轴的当前位置加上指定的偏移量作为零点位置坐标并赋给当前位置。

对于绝对值编码器，则只需要进行一次编码器校正（Absolute Encoder Adjustment），以后每次上电都不需要重新回零。也可以通过设置参考点来对绝对值编码器进行回零。

5.1.2.1 命令参数说明

参数	说明
Homing type	回零方式。 ✓ Active homing (缺省值): 主动回零 ✓ Passive homing 被动回零 ✓ Set home position home position coordinate 的值被设置成轴的当前位置（实际值）。该回零方式不会触发轴运动。

	✓ Relative home position setting home position coordinate 的值被加到轴的当前位置（实际值）。该回零方式不会触发轴运动。 ✓ Absolute encoder adjustment: 只有当轴的编码器为绝对值编码器时，Absolute encoder adjustment 才有效。轴的零点和编码器零点由 absolute encoder offset 值决定。该回零方式不会触发轴运动。 ✓ Absolute encoder adjustment with specification of the position value: 只有当轴的编码器为绝对值编码器时，Absolute encoder adjustment with specification of the position value 才有效。

5.1.2.2 范例

例 2 仅用编码器零脉冲回零（Encoder zero mark only）

1. 回零参数设置

双击项目导航栏中轴 Axis_1(Axis_1 带增量式编码器)下的 Homing，在右侧页面中进行参数设置，如图 5.6 所示：

1) Homing mode： Encoder zero mark only。

2) Homing procedure: 搜寻方向。选择 Start in positive direction。

3) Entry velocity: 检测到编码器零脉冲后继续运行 Home position offset 的速度。设为 20。

4) Reduced velocity: 寻找编码器零脉冲的速度。设为 10。

5) Home position offset: 零点位置偏差值。设为 0

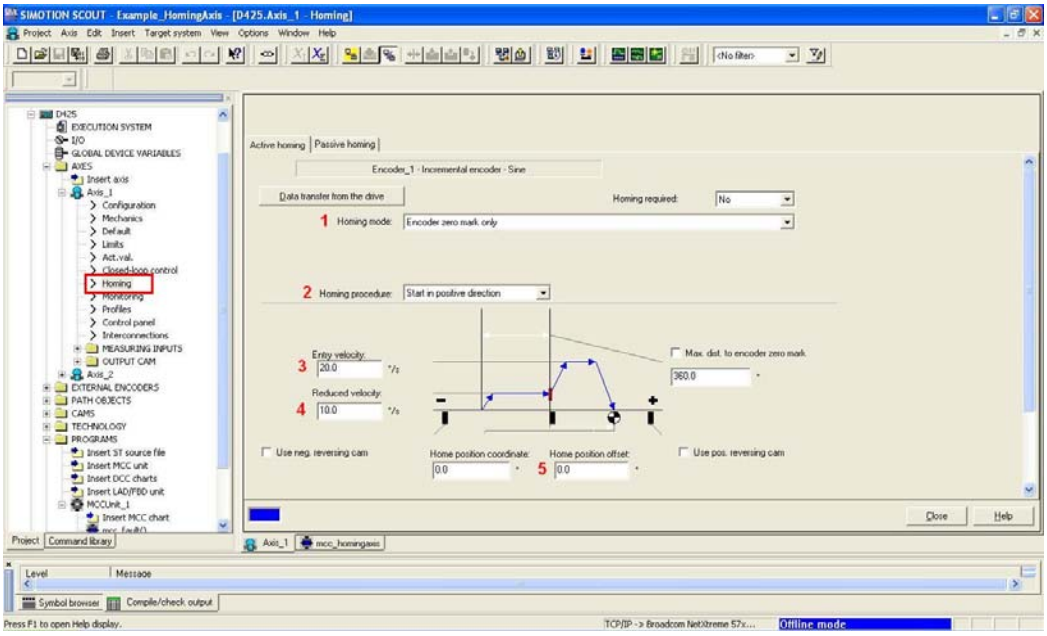


图 5.6 轴回零参数设置

2. 插入 MCC Unit 和 MCC chart，并将其分别命名为 MCCUnit_1 和 MCC_HomingAxis。

3. 在 GLOBAL DEVICE VARIABLES 中定义两个 bool 型的变量 g_boStart 和 g_boStartHomeAxis1 用于控制程序的运行。

4. 在 MCC_HomingAxis 中依次插入等待条件命令 (g_boStart 的上升沿作为触发条件)、轴使能命令（使能轴 Axis_1）、等待条件命令(g_boStartHomeAxis1 的上升沿作为出发条件)和轴回零命令（回零轴 Axis_1）。

5. 回零命令参数设置

双击插入的轴回零命令，在弹出的对话框中进行参数设置，之后点击 OK 确认。

1) Axis: 执行回零操作的轴。选择 Axis_1。

2) Homing Type: Active homing(主动回零)。

3) Home position coordinates: 零点坐标。在该编辑框中输入值。也可以输入变量。设为 0。

4) Homing approach velocity: 回零速度。在该编辑框中输入值。也可以输入变量。该参数只有当回零方式为使用接近开关和编码器零脉冲回零（Homing output cam and encoder zero mark）或者仅用外部零脉冲回零（External zero mark only）时有效。因此在本例中该参数不起作用。

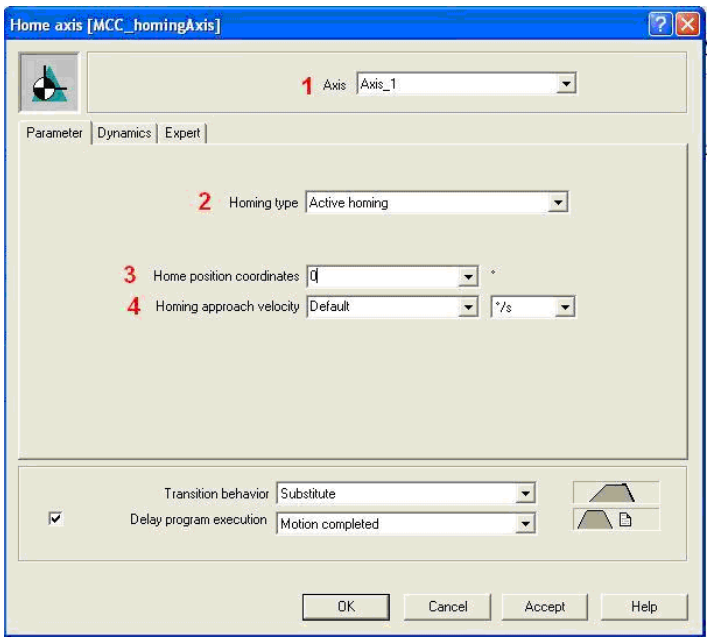


图 5.7 轴回零命令参数设置

在 Dynamics 页面进行加速度，加加速度设置。可以在编辑框中输入值，也可以输入变量。本例全部采用默认设置。

6. 编译并保存项目。

7. 将程序分配给执行系统。

将 mcc_homingaxis 分配给 MotionTask_1，并将 MotionTask_1 的 Activation after StartupTask 激活。

8. 在线并下载程序。

9. 运行程序。

将 Simotion 的操作模式从 STOP 变成 RUN。

在线修改变量 g_boStart 的值为 TRUE，开始执行轴使能命令，该命令执行完成后，置位 g_boStartHomingAxis1，轴 Axis_1 开始回零。

回零过程：轴按照回零速度(Reduced velocity)沿着搜寻方向(Homing procedure)运行直到检测到编码器的零脉冲，然后以 Entry velocity 运行 Home position offset，最后将轴的位置设置成回零命令中 Home position coordinates 项中的值。

10. 状态查询

轴有没有回零点可以通过系统变量<Axis>.positioningState.homed 来查看。

方法如下：高亮想要查看的轴，在 Symbol browser 中找到 positioningState.homed 这个系统变量，如果该变量的值为 TRUE，则表明该轴已经回过零点，否则轴没有进行回零。

例 3 仅用外部零脉冲回零（External zero mark only）

1. SINAMICS 设置

该回零方式需要将外部零脉冲信号与 Sinamics 上的 DI 相连。例如 p495.0 = 1 表示 DI9，如图 5.8 所示：

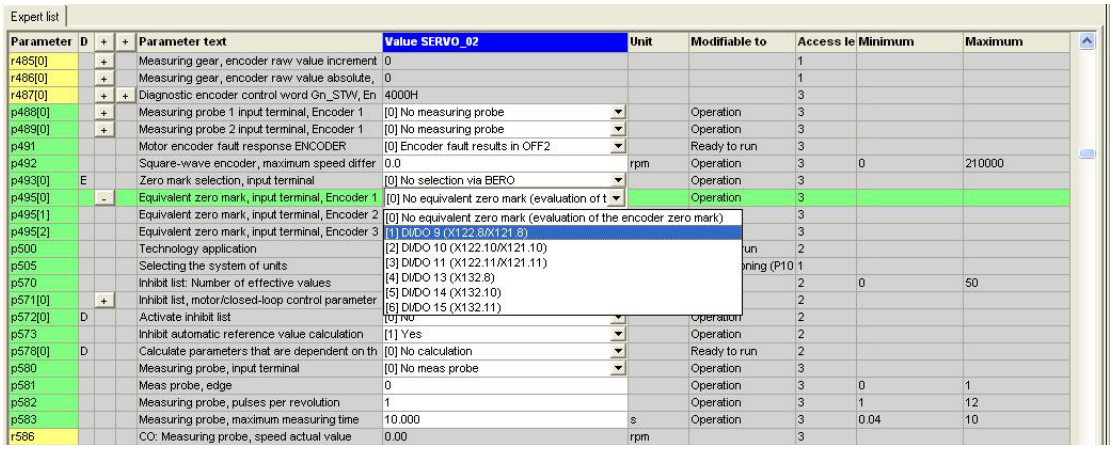


图 5.8 外部零脉冲信号与 Sinamics 上的 DI 相连

这样，DI/DO9 即为外部零脉冲。此外，在 SINAMICS 中需要将外部零脉冲信号与上面的端子硬件相联。

电机的旋转方向由零脉冲信号决定：正转对应上升沿，反转对应下降沿。

如果需要进行相反的操作，需要在 CU 中将信号反转，如图 5.9：

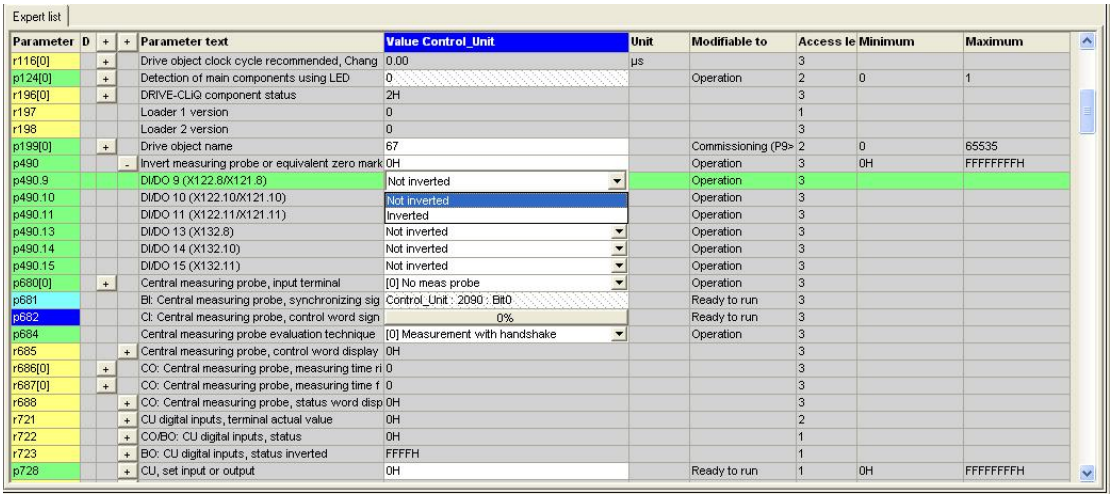


图 5.9 DI 信号反转

本例中，外部零脉冲的地址为 CU 报文的第九位。

2. CU 报文格式设置

为 CU 选择一种报文格式，如 SIEMENS telegram 390。

3. 回零参数设置

1) Homing mode: 回零模式，选择 External zero mark only

2) Signal transition: 外部零脉冲信号的沿。选择 Hign->Low(negative)

3) Ext. zero mark signal from drive available: Yes

4) Log address of ext. zero mark: 外部零脉冲的地址。CU 上端子 DI/DO9 对应的报文地址。

5) Bit number: 本例为 9(DI/DO9) 。

6) On the external zero mark side: 寻找外部零脉冲信号的方向。选择 Right

7) Homing procedure: 执行回零操作的方向。选择 Start in positive direction

8) Approach velocity: 寻找撞块时的速度。本例为 20。

9) Entry velocity: 走 Home position offset 时的速度。本例为 10。

10) Reduced velocity: 寻找外部零脉冲边缘的速度。

11) Home position coordinate: 零点位置。本例为 0。

12) Home position offset: 偏差。本例为 0。

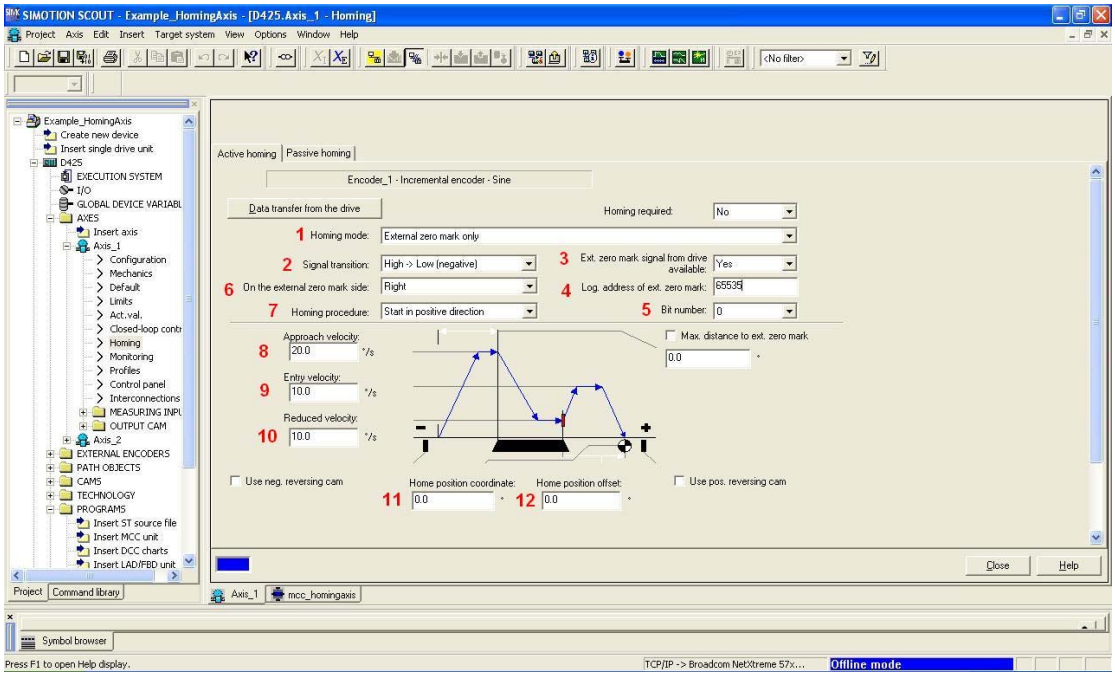


图 5.10 轴回零参数设置

4. 回零命令参数设置

双击插入的 Home axis 命令，在弹出的对话框中进行参数设置，点击 OK 确认。

1) Axis: 执行回零操作的轴。选择 Axis_1。

2) Homing Type: Active homing(主动回零)。

3) Home position coordinates: 零点坐标。设为 0

4) Homing approach velocity: 回零速度。设为 10。

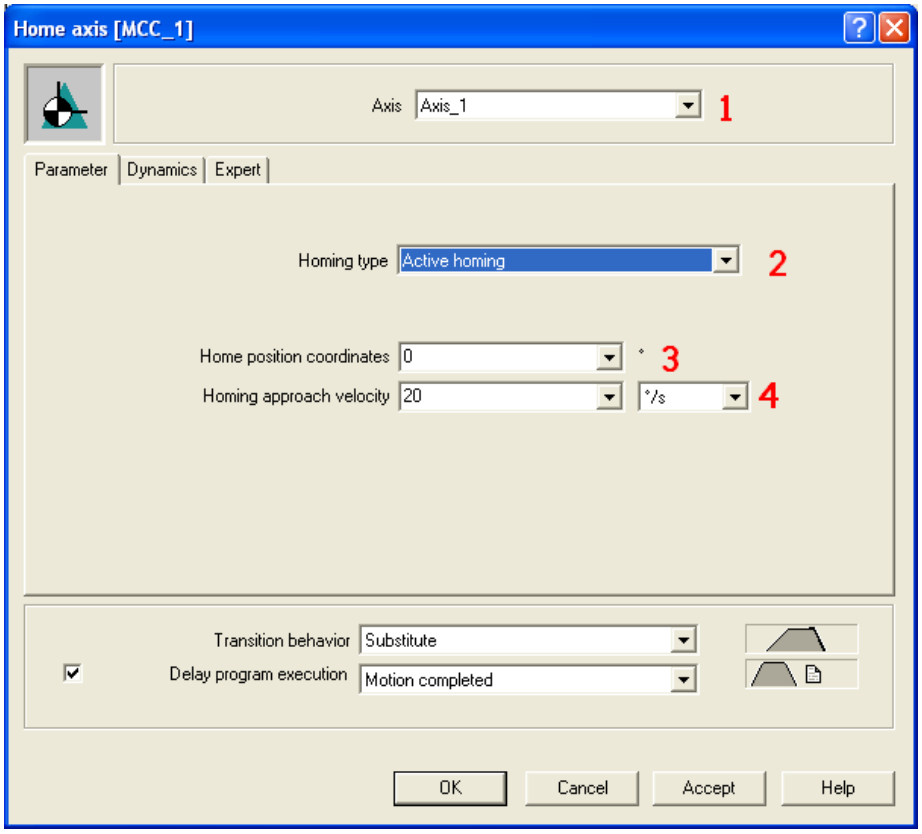


图 5.11 轴回零命令参数设置

其它操作均与例二相同。

例 4 使用外部撞块和编码器零脉冲回零（Homing output cam and encoder zero mark）

1. 回零参数设置

1) Homing mode: Homing output cam and encoder zero mark

2) Encoder zero mark: In front of homing output cam

3) Log. Address of homing output: 外部撞块的地址(必须大于 64)。在本例中，外部撞块的信号由 ET200M 的 DI 来模拟，其地址为 100。

4) Bit number: 0

5) Homing procedure: Start in positive direction

6) Approach velocity: 10

7) Entry velocity: 15

8) Reduced velocity: 20

9) Home position coordinate: 0

10) Home position offset: 10

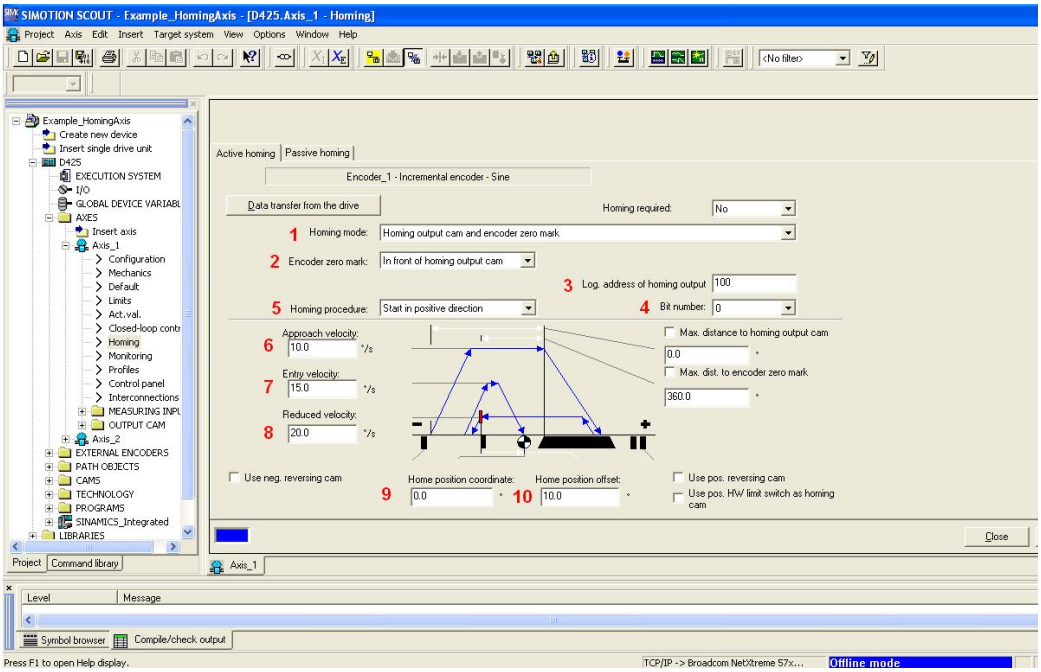


图 5.12 轴回零参数设置

2. 回零命令参数设置

1) Axis: Axis_1

2) Homing type: Active homing

3) Home position coordinate: Default

4) Homing approach velocity: Default

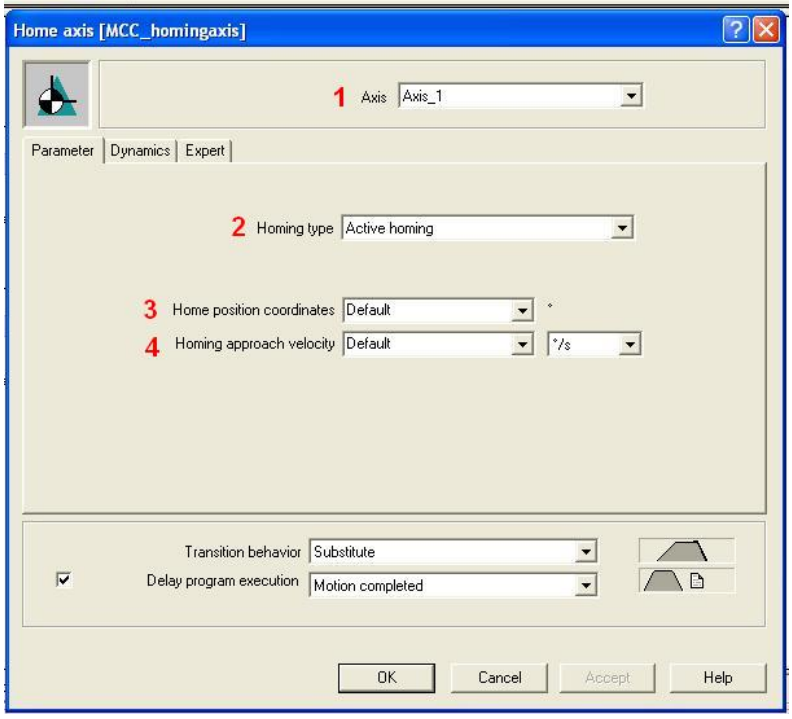


图 5.13 轴回零命令参数设置

3. 添加 IO 变量

在项目导航栏中高亮 D425 下的 I/O，在下面的 symbol browser 中新建一个 Bool 型的变量，命名为 i_boHomingCam，地址为 PI100.0，该变量即为撞块信号。

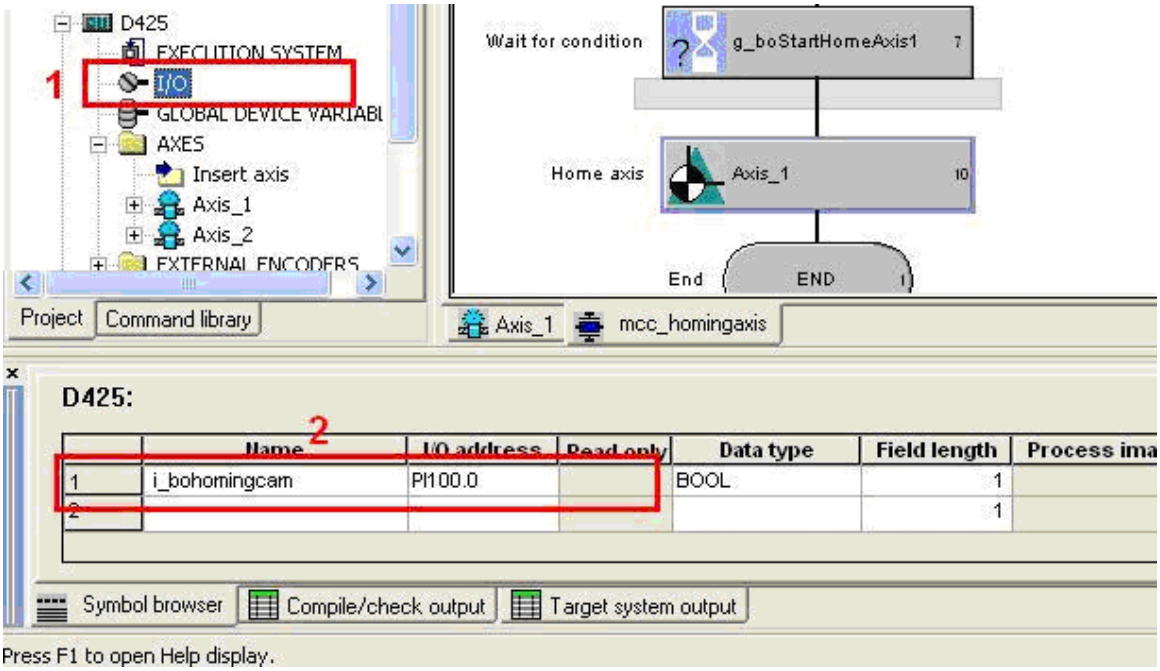


图 5.14 添加 Simotion IO 变量

4. Trace 设置

Simotion 的 trace 功能可以实时跟踪轴的各种状态，如位置，速度等等。当 Simotion 在线后点击工具栏上的按钮，trace 界面出现在窗口中。如图 5.15

所示。

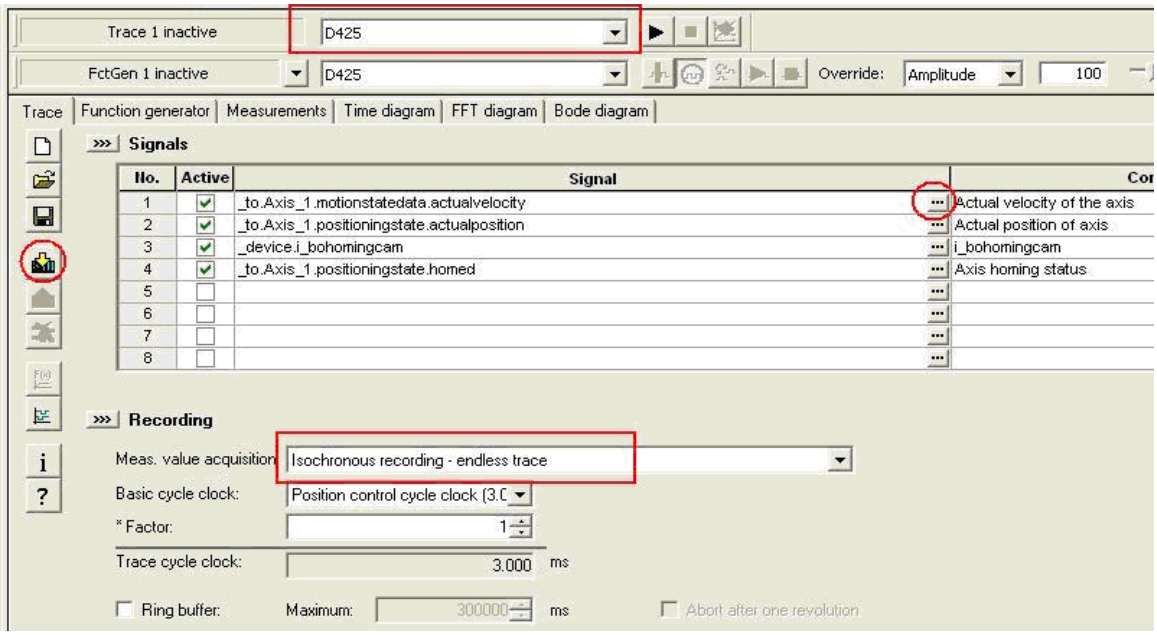


图 5.15 Trace 窗口

点击 Signals 窗口中的…按钮弹出 Signal Selection Trace 对话框，从中可以选择待 trace 的变量。如下图所示。例如，点击左边栏中 TO->Axis_1，右边窗口中将列出 Axis_1 的所有系统变量，从中选择 motionstatedata.actualvelocity（轴的实际速度）然后点击想要放置该变量的栏的序号即可，如图 5.16 所示。按照相同的方法，将如下变量添加到 trace 列表中：

1) i_boHomingCam
2) Axis_1.positioningState.homed
3) Axis_1.motionstatedata.actualvelocity
4) Axis_1.positioningstate.actualposition

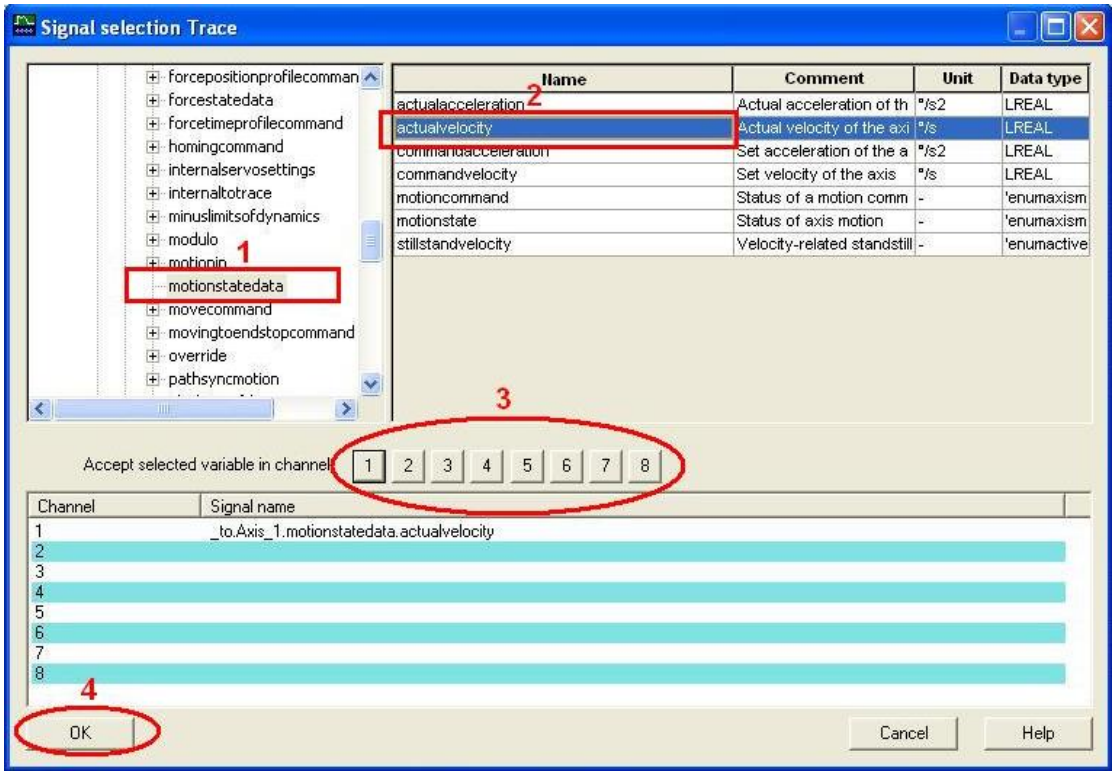


图 5.16 选择 Trace 信号

将 trace 下载到目标设备中，然后点击 trace 栏中的▶开始 trace，在 trace 窗口的 Time diagram 标签页中可以看到 trace 的实时状态。要停止 Trace 只要点击 trace 栏中的■按钮即可以。

5. 运行程序。

确认 Simotion 的 CPU 处于 RUN 状态，置位 g_boStart，轴 Axis_1 使能，使能完成后，点击▶开始 trace，然后置位 g_boStartHomeAxis1，轴 Axis_1 开始回零。

回零过程:

轴沿着 Homing procedure 中设置的搜索方向，加速至 Approach velocity，碰到撞块信号后（ET200M 的 DI0 闭合）减速停止。然后再反向加速至 Reduced velocity，离开撞块后（ET200M 的 DI0 断开）遇到的第一个编码器零脉冲后轴停止，然后轴反向加速至 Entry velocity 运行 Home position offset 距离后停止在参考点，将参考点位置设为 Home position coordinates。如图 5.17 所示。

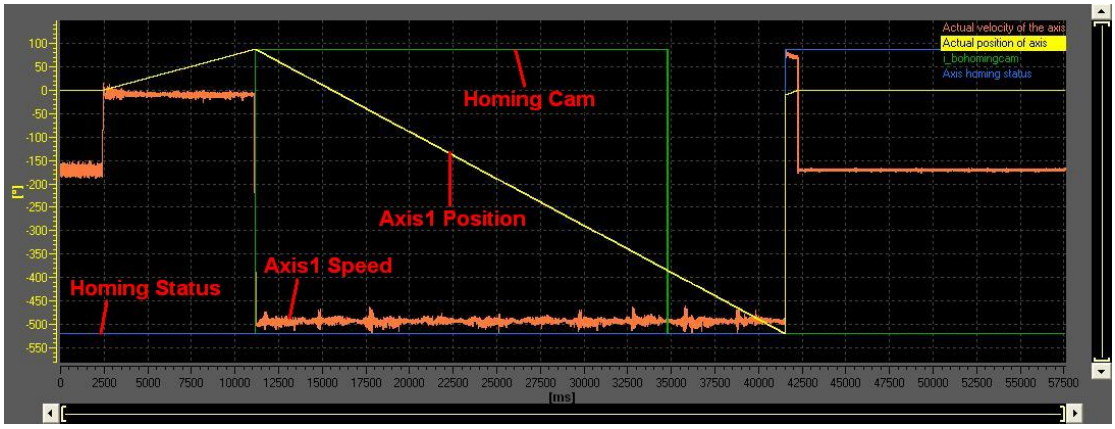


图 5.17 回零过程 Trace 图

其余操作均与例二相同。

例 5 设置参考点（Set home position）

1. 插入 MCC 程序和 MCC chart，并将其分别命名为 MCCUnit_1 和 MCC_HomingAxis。

2. 定义变量

在 GLOBAL DEVICE VARIABLES 中定义两个 bool 型的变量 g_boStart 和 g_boStartHomeAxis1 用于控制程序的运行。

3. 在 MCC_HomingAxis 中依次插入等待条件命令(g_boStart 的上升沿作为出发条件)，轴使能（使能轴 Axis_1）、等待条件命令(g_boStartHomeAxis1 的上升沿作为出发条件)和轴回零命令（回零 Axis_1）。

4. 回零命令参数设置

双击插入的 Home axis 命令，在弹出的对话框中进行参数设置，点击 OK 确认。

1) Axis: Axis_1。

2) Homing Type: Set home position。

3) Home position coordinates: 0。

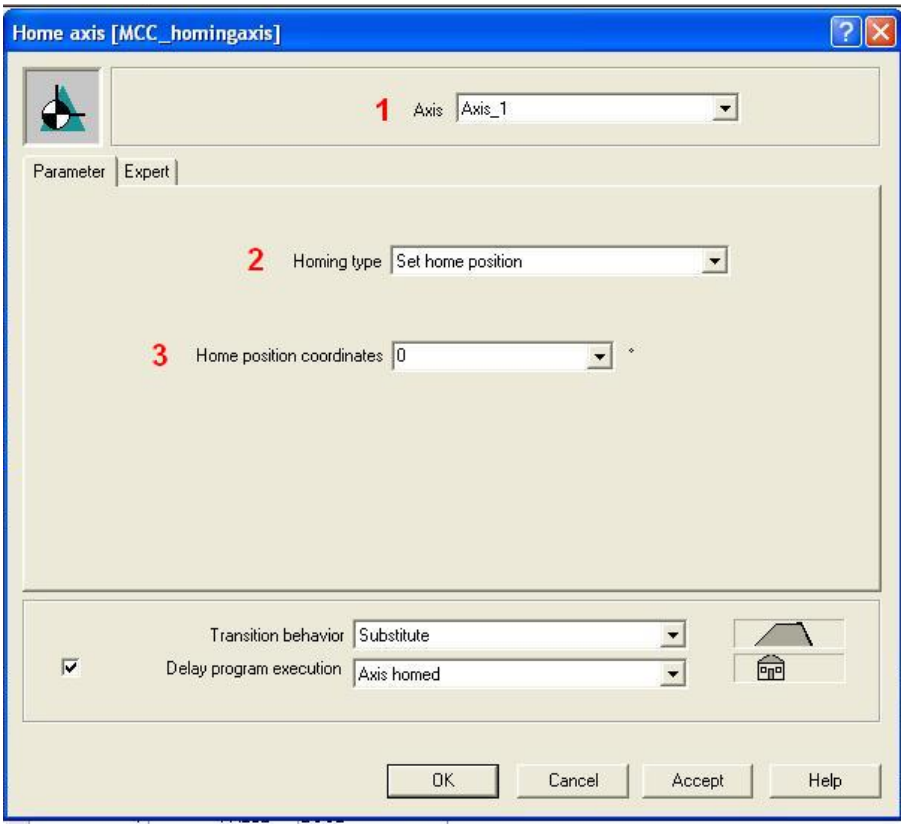


图 5.18 轴回零命令参数设置

5. 编译并保存。

6. 将程序分配给执行系统。

7. 在线并下载程序。

8. 运行程序。

将 Simotion 的操作模式从 STOP 变成 RUN。

将变量 g_boStart 的值从 FALSE 变为 TRUE，等待的条件满足，程序向下运行，开始执行轴使能命令，该命令执行完成后，置位 g_boStartHomingAxis1，轴 Axis_1 的当前位置被设置成 Home position coordinates 中的值，回零结束。整个过程中轴不发生任何运动。

5.1.3 轴移动命令(Move Axis)

和

该命令使轴加速或减速到程序设定的速度，然后保持不变。如果限制了恒速运动时间（Dynamics 页面的 Constant traversing time 参数），那么轴在设定时间到达后减速到 0。

轴的移动命令有两种：位置控制模式的移动和速度控制模式的移动。前者使轴运行于位置控制模式下，后者使轴运行于速度控制模式下，它分别对应于单轴命令组下的  （Speed Specification）命令和  （Start axis position-controlled）。由于它们功能相似，现以位置控制模式的移动命令为例，介绍它的使用。

5.1.3.1 命令参数说明

参数	说明
Axis	执行移动命令的轴
Speed	恒速运动阶段的速度值。在编辑下拉框中输入值。也可以输入变量。
Direction	轴的移动方向。 ✓ From speed sign：根据速度值的符号。 ✓ Positive：正向。 ✓ Negative：反向。 ✓ Last direction programmed：上一次的设定。 Preassigned value(default)：缺省值为系统变量 userDefaultDynamics.direction
Velocity profile	各个运动阶段之间的过渡方式。 缺省值为系统变量 userDefaultDynamics.profile。
Constant traversing time	恒速运行时间。 勾选该复选框，在编辑框中输入值。

5.1.3.2.2 范例

例 6: Axis_1 以 60 度/秒的速度沿正向移动。

1. 插入 MCC Unit 和 MCC chart，分别将其命名为 MCCUnit_1 和 mcc_moveaxis。

2. 定义变量。在 GLOBAL DEVICE VARIABLES 中定义两个 Bool 型的变量 g_boStart 和 g_boStartMove。

3. 在 mcc_moveaxis 中依次插入等待条件(g_boStart 的上升沿作为触发条件)命令、轴使能命令（使能 Axis_1）命令和等待条件(g_boStartMove 的上升沿作为触发条件)命令。

4. 插入轴移动命令。

在 MCC 工具条的单轴命令组中单击位置控制模式下的轴移动命令，将其插入 mcc_moveaxis 中第二个等待条件命令之后。

5. 移动命令参数设置

双击插入的轴移动命令，在弹出的对话框中进行参数设置，点击 OK 确认。

1) Axis: Axis_1。

2) Speed: 60 度/秒。

3) Direction: Positive

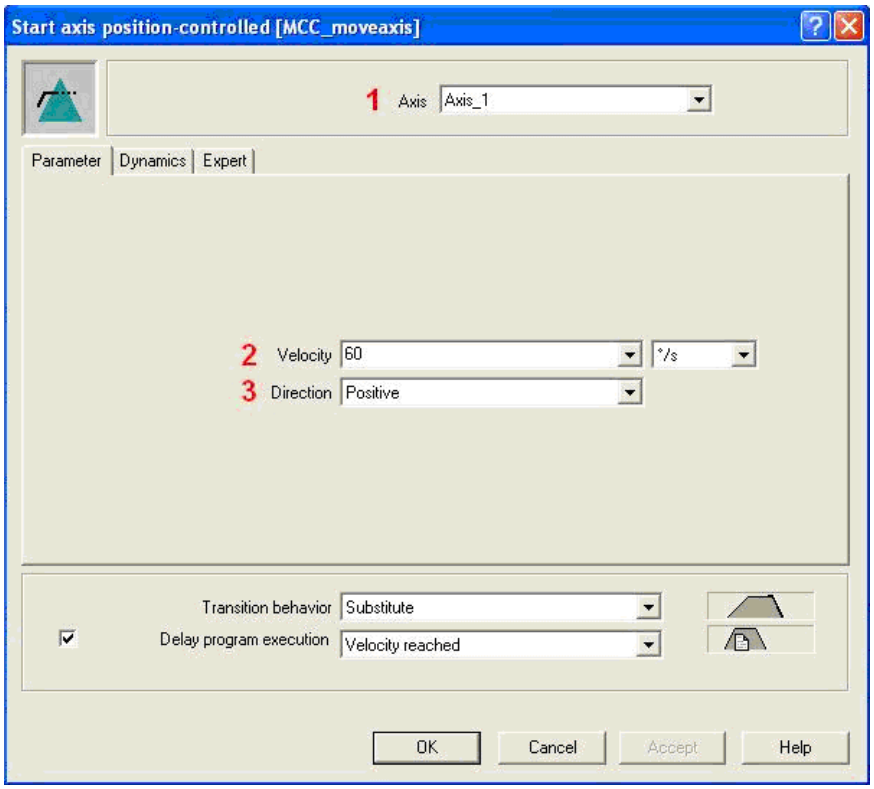


图 5.19 轴移动命令参数设置

在 Dynamics 页面进行加速度，加加速度设置。可以在编辑框中输入值，也可以输入变量。本例全部为默认设置。

在 Dynamics 页面中有个 Time 复选框，如果激活则可以定义一个时间间隔。该时间间隔表示轴开始恒速运行到开始减速的时间跨度。如果不激活 Time 复选框，轴将一直运行直到接受一个新命令。

6. 编译并保存。

7. 将程序分配给执行系统。

8. 在线并下载程序。

9. Trace 功能设置

参考例 4，将 Axis_1. motionstatedata.actualvelocity 系统变量添加到 Trace 信号列表中。

10. 运行程序。

确认 Simotion 的 CPU 处于 RUN 状态，置位 g_boStart，轴 Axis_1 使能，使能完成后，点击  开始 trace，然后置位 g_boStartMove，轴 Axis_1 首先从零速加速到 60 度/秒，然后稳定在 60 度/秒的速度沿正向移动。如图 5.20 所示。

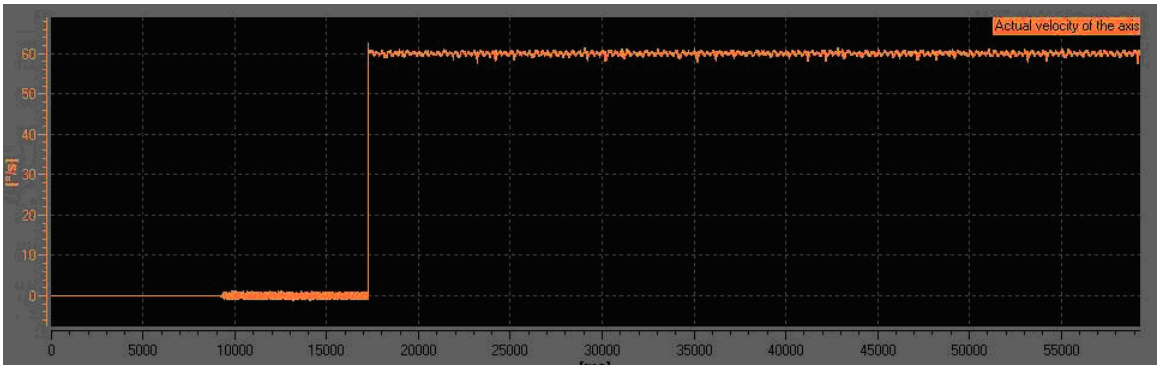


图 5.20 轴移动 Trace 图

5.1.4 轴定位命令(Position Axis)

轴定位命令将轴（旋转轴，模态轴或线性轴）移动到一个位置。位置可以是绝对位置或者相对位置。模态轴还可以通过“最短路径”进行定位。

定义的位置必须位于软限位开关内。

5.1.4.1 命令参数说明

参数	说明
Position	位置值。取决于 type 参数。 • "Absolute"类型：运动的终点。 • "Relative"类型：以从轴的当前位置算起的运动距离。 位置值要以浮点数类型输入。
Type	绝对定位或相对定位。 ✓ Absolute(缺省值): 绝对定位。 ✓ Relative: 相对定位。
Direction	运动方向。 下面两种情况必须要定义运动方向。 • 相对定位（对于所有轴）。 • 绝对定位且轴为模态旋转轴。 如果选择了正方向或反方向，那么方向的优先级比位置高。速度的符号由定义的方向决定。 ✓ Positive: 运动方向为轴的正方向。对于相对运动，位置的符号被忽略。 ✓ Negative: 运动方向为轴的负方向。对于相对运动，位置的符号被忽略。 ✓ From position (仅对相对定位有效): 运动方向由位置的符号决定。 ✓ Shortest path (仅对绝对定位且模态旋转轴有效): 运动方向为到达目标位置的最短路径的方向。 ✓ Last direction set in the program
Velocity	恒速运行阶段的速度。 在编辑框中输入值，也可以输入变量。 ✓ Current ✓ Last programmed velocity Preassigned value (缺省值)：缺省值为系统变量 userDefaultDynamics.velocity。

5.1.4.2 范例

例 7: 绝对定位

Axis_1 以 20 度/秒的速度移动到 100 度的位置。

1. 插入 MCC Unit 和 MCC chart，分别将其命名为 MCCUnit_1 和 MCC_PositioningAxis。

2. 定义变量

在 GLOBAL DEVICE VARIABLES 中定义两个 Bool 型的变量 g_boStart 和 g_boStartPos。

3. 在 MCC_PositioningAxis 中依次插入等待条件命令 (g_boStart 的上升沿作为触发条件) , 轴使能命令（使能 Axis_1），等待条件命令(g_boStartPos 的上升沿作为触发条件)。

4. 插入定位轴命令。

在 MCC 工具条的单轴命令组中单击轴定位命令，将其插入 MCC_PositioningAxis 中第二个等待条件命令之后。

5. 定位轴命令参数设置

双击刚刚插入的 Position axis 命令，在弹出的对话框中进行参数设置，点击 OK 确认。如图 5.21 所示。Dynamics 页面的设置均采用默认值。

1) Axis: Axis_1

2) Position: 100

3) Type: Absolute

4) Direction: Positive

5) Velocity: 20 度/秒

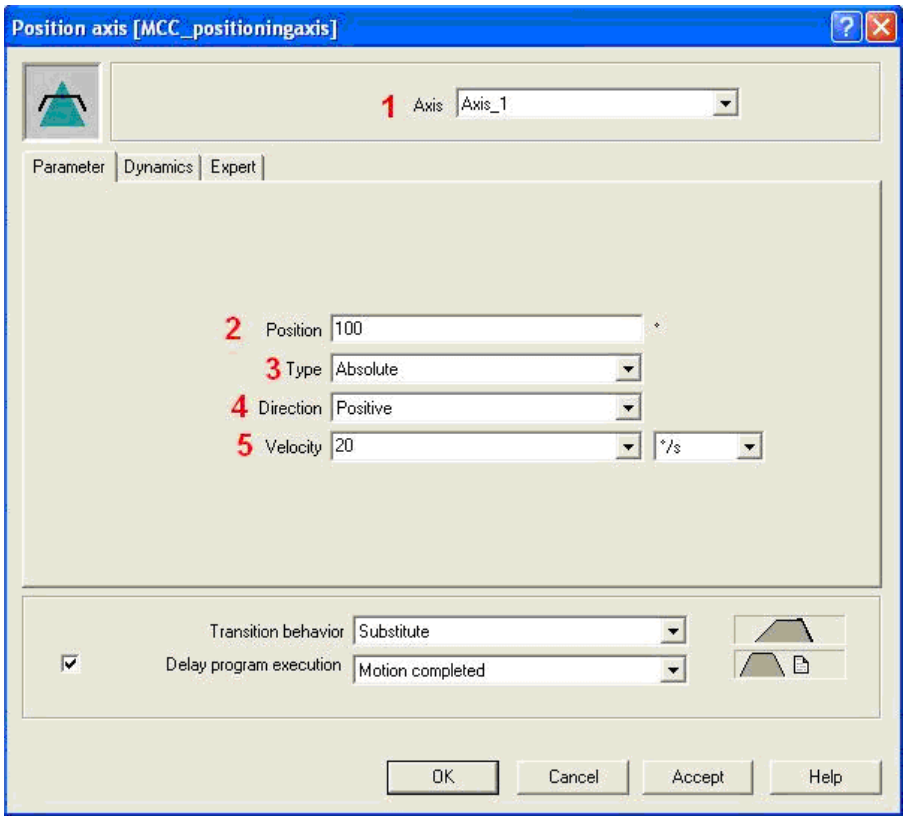


图 5.21 轴定位命令参数设置

6. 编译并保存。

7. 将程序分配给执行系统。

将 MCC_PositioningAxis 分配给 MotionTask_1，并将 MotionTask_1 的 Activation after StartupTask 激活。

8. 在线并下载程序。

9. Trace 设置

将系统变量 Axis_1. positioningstate->actualposition 和 Axis_1.motionstatedata.actualvelocity 添加到 Trace 信号列表中。

10. 运行程序。

将 Simotion 的 CPU 操作模式从 STOP 变成 RUN。点击 开始 trace。

置位 g_boStart，Axis_1 使能。置位 g_boStartPos，轴 Axis_1 开始运动，以 20 度/秒的速度运动到 100 度的位置然后静止。

点击 停止 trace，在 Time diagram 标签页中可以查看 trace 的结果，如图 5.22 所示。

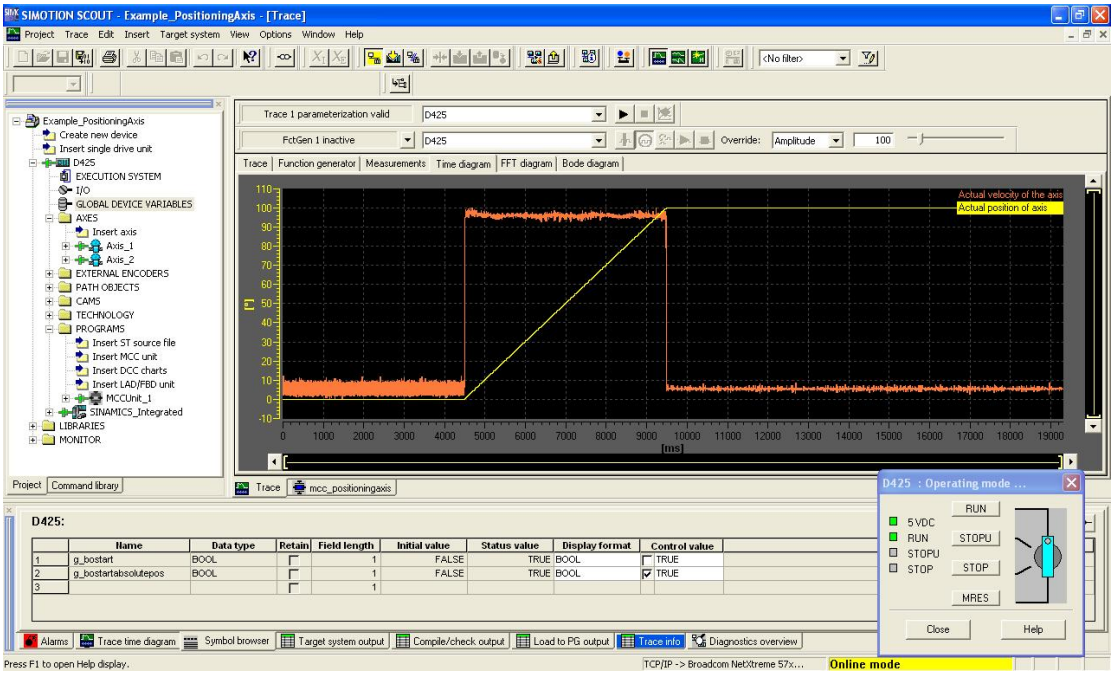


图 5.22 轴绝对定位 Trace 图

例 8: 相对定位

轴 Axis_1 以 40 度/秒的速度从当前位置正向移动 200 度。

轴定位命令参数设置:

1) Axis: Axis_1

2) Position: 200

3) Type: Relative

4) Direction: Positive

5) Velocity: 40 度/秒

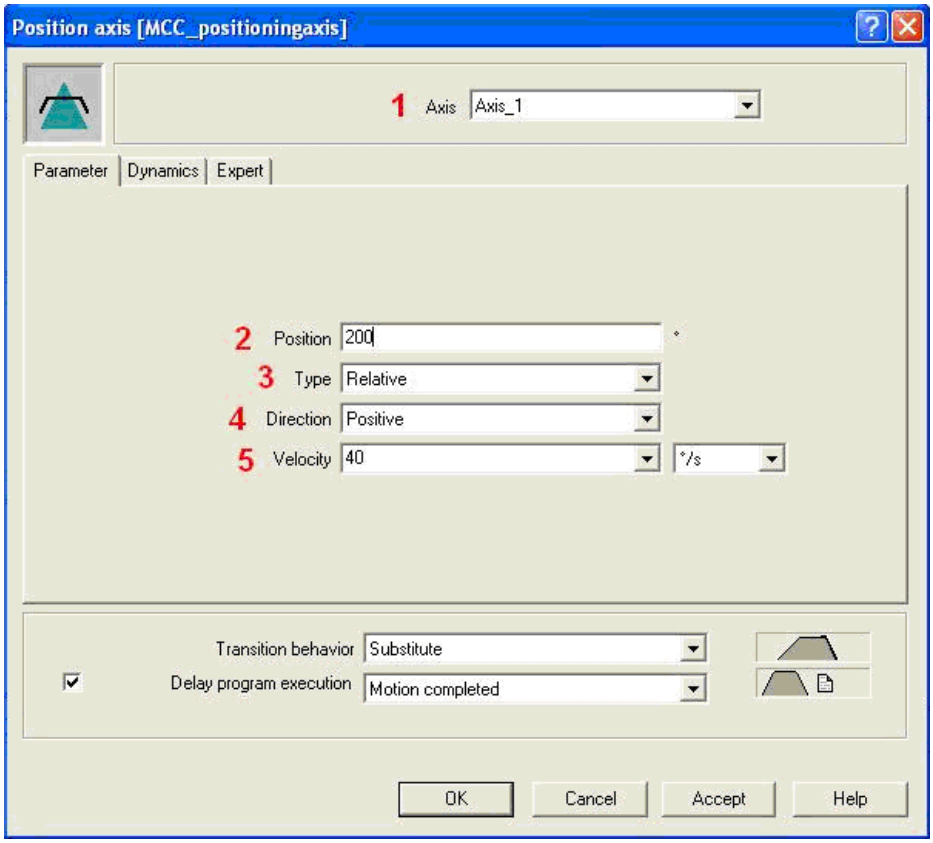
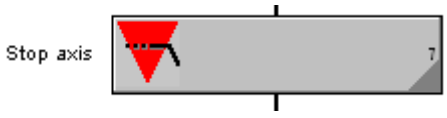


图 5.23 轴定位命令参数设置

其余操作均与例 7 相同。

5.1.5 轴停止命令(Stop Axis)



轴停止命令用于停止轴的运动。该命令适用于所有单轴运动。运动可以通过普通停止(Normal stop)或快停(Quick stop)来停止。

普通停止(Normal stop): 该命令停止所有单轴运动（定位运动和速度运动），但不能停止同步运动。

快停(Quick stop): 该命令即可以停止所有单轴运动（定位运动和速度运动），又可以停止同步运动。此外，轴的使能被移除。

5.1.5.1 命令参数说明

参数	说明
Stop mode	停止模式。 只有使用 normal stop without abort 模式停止的运动可以通过 Continue motion 命令恢复。其它运动均不可以被恢复。 ✓ Normal stop without abort（缺省值） Selection 项中的运动按照 Dynamics 页面中设定的动态参数停止。该运动可以被 Continue motion 命令恢复。但是在停止命令和恢复命令之间不能有其它命令。该命令对同步运动不会产生影响。 ✓ Normal stop with abort Selection 项中的运动按照 Dynamics 页面中设定的动态参数被停止。该运动不能被恢复。该命令对同步运动不会产生影响。 ✓ Quick stop within defined period 运动可以在设定的时间内停止。在 Dynamics 页面中的 Time for deceleration 参数中设定时间。运动可以被恢复。 ✓ Quick stop at maximum deceleration 运动按照轴的最高动态性参数停止。运动不能被恢复。 ✓ Quick stop with preassigned braking ramp 运动按照控制器中的制动器斜坡停止。制动器斜坡在组态时进行设置。运动不能被恢复。 ✓ Quick stop with dynamics parameters 运动按照 Dynamics 页面中设定的动态性参数停止。运动不能被恢复。
Selection	需要停止的运动。 该参数只有当停止模式为 Normal stop without abort 或 Normal stop with abort 时才有效。 ✓ All motions: 所有运动。 ✓ Basic motion: 基本运动。 ✓ Superimposed motion: 叠加运动。
Traversing mode	运行模式。 ✓ Position-controlled 只用于位置轴和同步轴。轴从当前的运行模式(例如速度控制，力控制或者扭矩控制)切换到位置控制，然后停止。 ✓ Closed-loop speed controlled 轴从当前的运行模式(例如速度控制，力控制或者扭矩控制)切换到速度控制，然后停止。速度斜坡立刻起作用。先前存在的跟随误差不必移除。 如果位置控制的运动在速度控制模式下被停止，轴的使能会被移除。

	✓ Last set traversing mode (缺省值) 轴从当前的运行模式(例如位置控制，速度控制，力控制或者扭矩控制)切换到上次设定的运行模式(位置或速度控制)，然后停止。
Time for deceleration	✓ Quick stop within defined period 停止模式下的制动时间。 缺省值为系统变量 userDefaultDynamics.stopTime。

5.1.5.2 范例

例 9:

1. 插入 MCC Unit 和 MCC chart，将其分别命名为 MCCUnit_1 和 MCC_StopAxis。

2 在 GLOBAL DEVICE VARIABLES 中定义三个 Bool 型的全局变量 g_boStart, g_boStartMove 和 g_boStop。

3. 在 MCC_StopAxis 中依次插入等待条件命令（g_boStart 的上升沿作为触发条件），轴使能命令（使能轴 Axis_1），等待条件命令（g_boStartMove 的上升沿作为触发条件），位置控制模式下的轴移动命令（轴为 Axis_1，速度为 20 度/秒，方向为正向）和等待条件命令（g_boStop 的上升沿作为触发条件）。

4. 插入停止轴命令。

在 MCC 工具条的单轴命令组中点击轴停止命令，将其插入到 MCC_StopAxis 中。程序如图 5.24 所示。

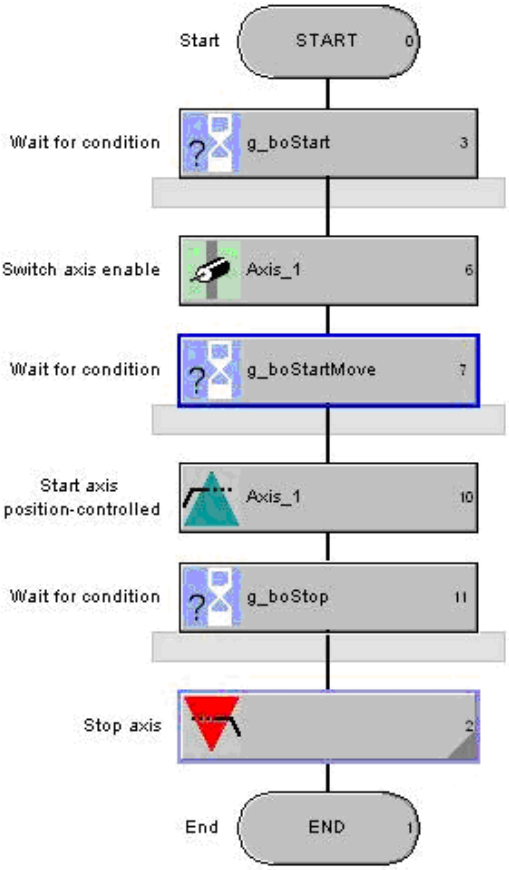


图 5.24 轴停止程序

5. 停止轴命令参数设置。

双击插入的 Stop Axis 命令，在弹出的对话框中进行参数设置，点击 OK 确认。

1) Axis: Axis_1。

2) Stop mode: Normal Stop without Abort.

3) Selection: All motions。

4) Traversing mode: Last set traversing mode。

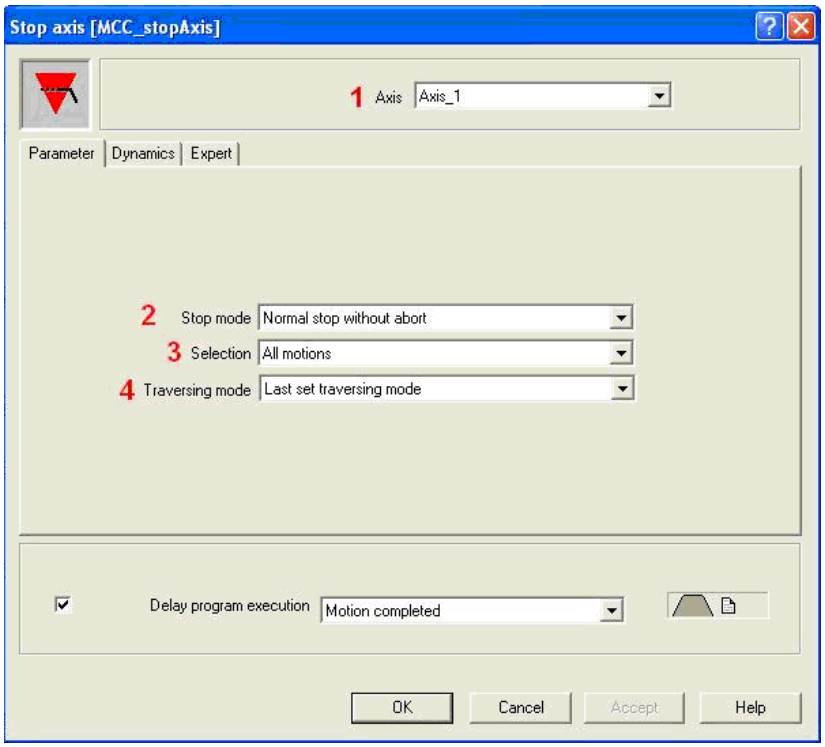


图 5.25 轴停止命令参数设置

6. 编译并保存。

7. 将程序分配给执行系统。

8. 在线并下载程序。

9. Trace 设置

将系统变量 Axis_1.motionstatedata.actualvelocity 添加到 Trace 信号列表中。

10. 运行程序。

启动 Trace，置位 g_boStart，使能轴 Axis_1。置位 g_boStartMove，轴 Axis_1 以 Start axis position-controlled 命令中设定的速度和方向移动。置位

g_boStop，轴 Axis_1 减速到 0，然后静止。如图 5.26 所示

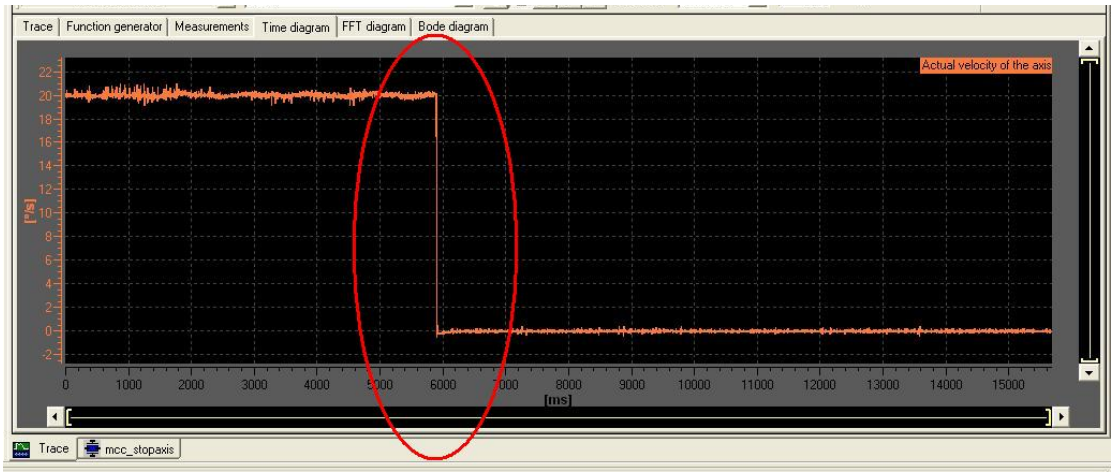


图 5.26 轴停止 Trace 图

11. 状态查询

轴有没有处于静止状态可以通过系统变量<Axis>.motionstatedata.motionstate 来查看。

方法如下：高亮想要查看的轴，本例为 Axis_1，在 Symbol browser 中找到 motionstatedata.motionstate 系统变量，如果该变量的值为 STANDSTILL，则表明该轴处于静止状态。

5.2 同步运动控制命令

5.2.1 同步操作简介

5.2.1.1 功能介绍

同步操作是指，按照一定的标准（齿轮比，比例，偏移，凸轮），将主对象的值经过处理后作为给定值分配给从对象。

同步操作功能可以分为以下三类：

1. 齿轮同步（Gearing）

与机械式齿轮同步相对应，主对象和从对象之间为线性传递关系，可指定齿轮比。如图 5.27 所示。

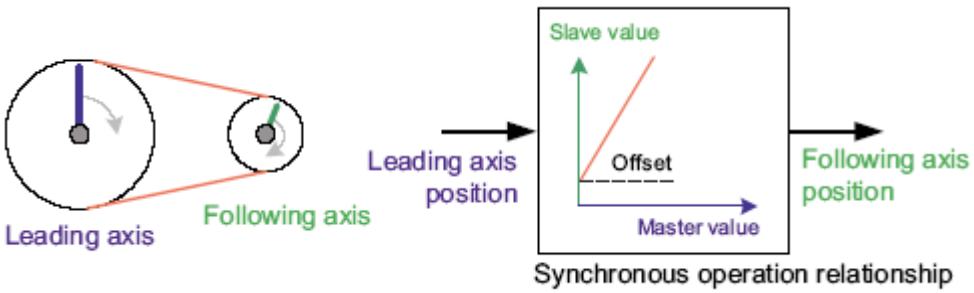


图 5.27 齿轮同步

2. 速度同步

可以实现固定的速度比

3. 凸轮同步（Camming）

与机械式凸轮相对应，主、从对象之间为非线性传递关系，主对象的值按照已定义好的凸轮曲线处理后传递给从对象。如图 5.28。

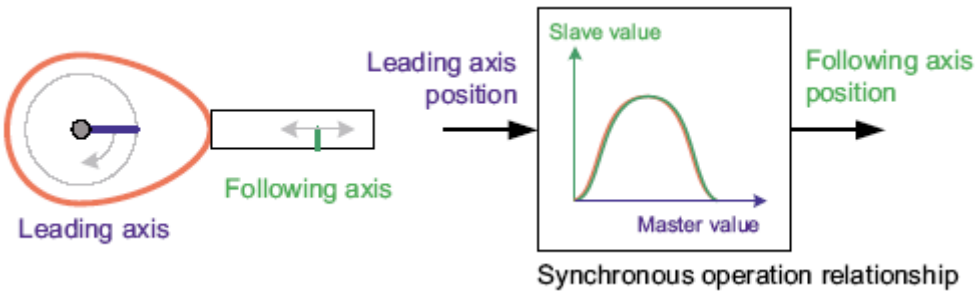


图 5.28 凸轮同步

同步关系的建立至少需要一个主对象和一个同步轴，同步轴由一个从轴、一个或两个同步对象（在凸轮同步的情况下还包含一个或多个凸轮曲线）组成。同步对象在建立同步轴的时候自动产生，即项目导航栏中的 Axis 目录下的 。主对象、同步对象及从轴之间的关系如图 5.29 所示。

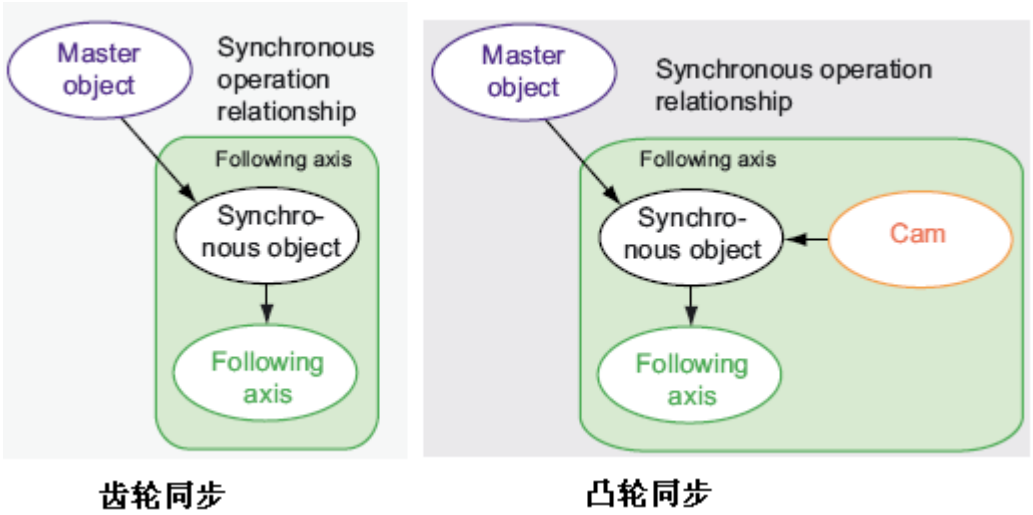


图 5.29 同步关系

5.2.1.2 齿轮同步

齿轮同步的主、从对象关系可以用以下公式概括：

从值=主值 x 比例系数+偏移

公式中，主、从值可以为主、从轴的速度或位置；比例系数即齿轮比，可以以分数或者小数的形式设置；偏移为同步时从值相对于主值的位置偏移，当不明显指定偏差时，系统将自动在同步时产生便宜，且不可控。

齿轮同步可以分为绝对同步（Gearing Type=ABSOLUTE）和相对同步（Gearing Type=RELATIVE）。

绝对同步方式下，同步的主值和从值相对于各自的绝对坐标系，偏移值为同步上时主从间的绝对偏移，而不管同步前主从之间的差值有多少。如图 5.30 所示。

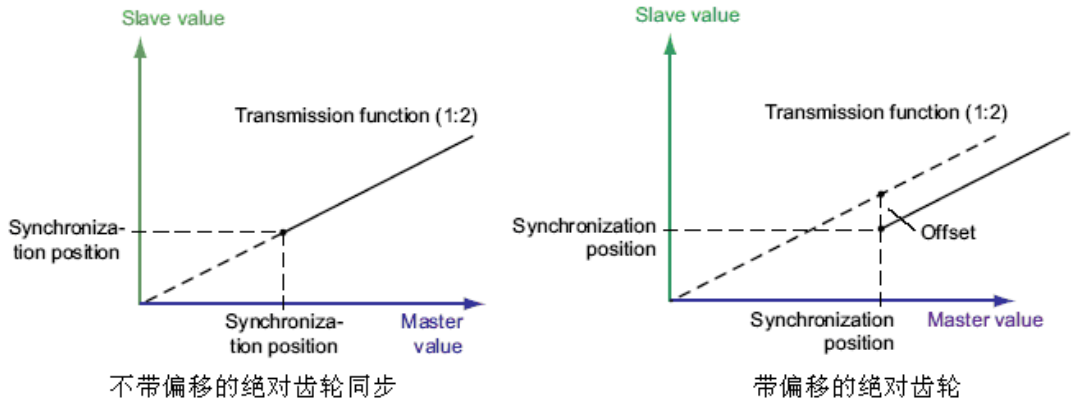


图 5.30 绝对齿轮同步

相对同步方式下，主值和从值以开始同步点为参考坐标原点，偏移值为同步上时主从间的相对偏移，此时的主、从值之间的绝对偏移除了包含此偏移值之外还包含同步前主、从值之间的差异。如图 5.31 所示。

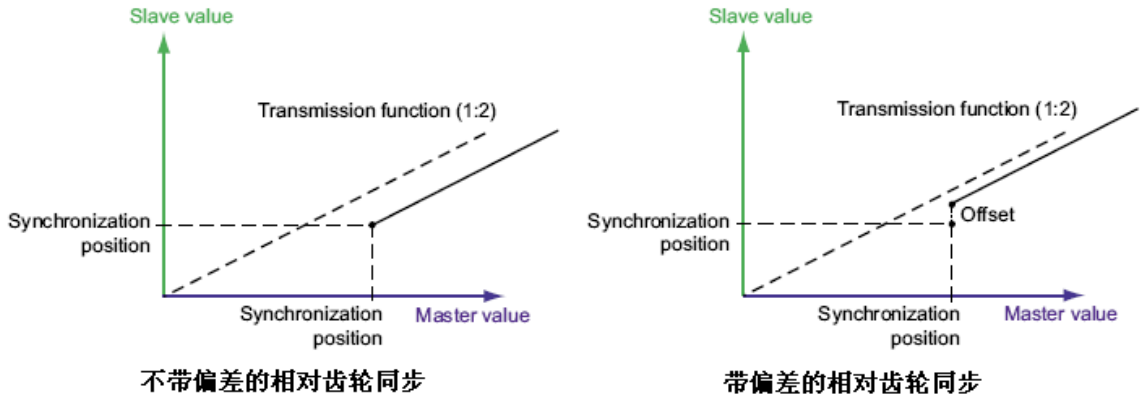


图 5.31 相对齿轮同步

5.2.1.3 凸轮同步

凸轮同步用于主、从轴之间的非线性同步，可以用以下公式概括：

从值=KS（主值+主值偏移）+从值偏移

其中 KS 为凸轮曲线。在定义域（主值）和值域（从值）凸轮都可以被定义为绝对方式或相对方式。绝对方式指主或从值相对于各自的绝对坐标原点，相对方式指主或从值以同步开始时的位置作为参考坐标原点。绝对/相对凸轮同步有可能的组合如图 5.32 所示。

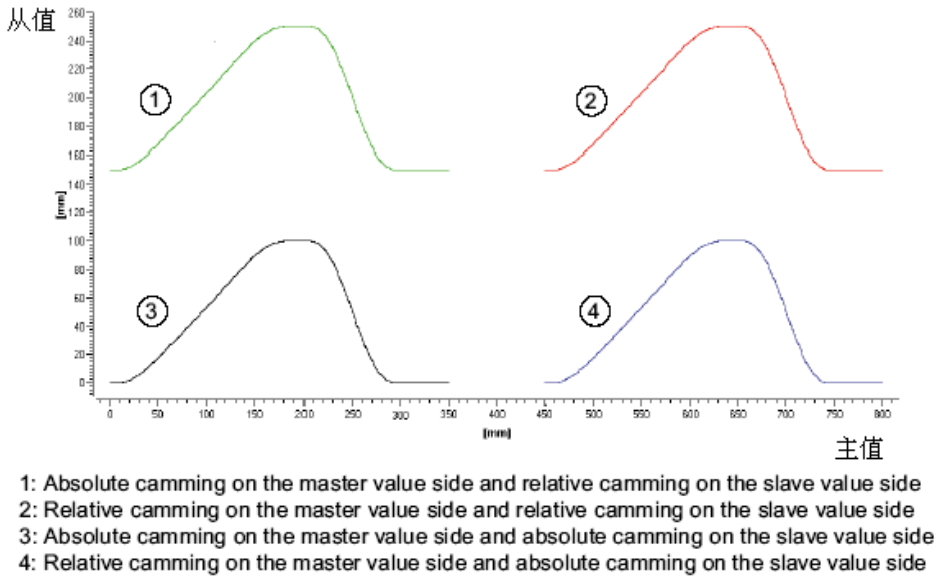


图 5.32 绝对/相对凸轮同步

通过设置 cammingMode 参数可以指定凸轮同步为非循环或循环。

非循环(NO-CYCLIC) 凸轮同步指凸轮在定义的主值范围内只执行一次。当到达凸轮终点或起点时，凸轮自行结束。循环凸轮同步(CYCLIC) 指凸轮的定义区间循环地映射到主值上，凸轮同步按照曲线循环被执行。如图 5.33。

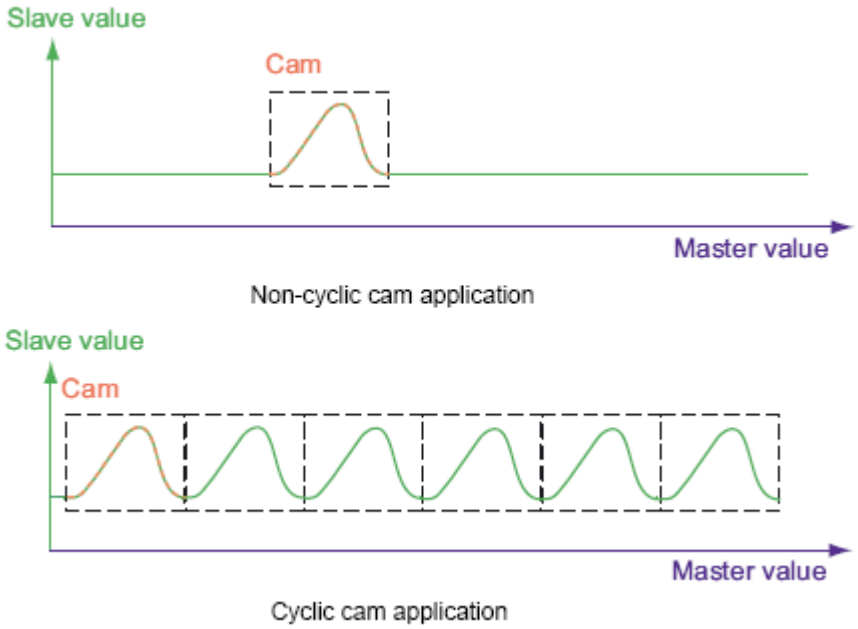


图 5.33 非循环/循环凸轮同步

5.2.2 齿轮同步命令(Gearing On)

5.2.2.1 命令参数说明

参数	说明
Following axis	指定要进行同步的轴。从轴可以为: • 所有同步轴： 轴定义在项目导航栏的 AXES 文件夹下。 • <Reference> ： 如果要进行同步操作的轴没有定义在设备中而是指定为参考(变量)，则选择<Reference>。 MCC Unit 或 MCC chart 中所有已声明的 followingObjectType 类型的变量都可以在 Synchronous operation 下拉编辑框中找到。
Synchronous operation	根据所选的 Following axis，所有与所选的从轴相关联的同步对象在该下拉列

	表中显示。
Reset master value	如果想要设置同步关系的主对象值，则选择该复选框（缺省值）。 如果清除该复选框则保持上次主对象值。
Master axis/encoder/external master value	如果勾选了 Reset master value 复选框则需要设置该参数。 可以选择： • 设备或 DP 主站中所有可用的定位轴，同步轴和外部编码器。 • 所有 MCC 源文件或 MCC chart 中声明的 posAxis, followingAxis 或 externalEncoderType 类型的变量。
Gear direction	齿轮同步方向。 ✓ From sign of gear ratio (缺省值) 同步方向由齿轮比的符号决定。 ✓ Opposite direction 从轴运动方向与主轴相反。 ✓ Same direction 从轴运动方向与主轴相同。 ✓ Opposite current gearing direction 与当前同步方向相反。 ✓ Current direction 当前同步方向。 ✓ Last programmed direction ✓ Preassigned value 缺省值。缺省值为系统变量: userDefault.gearingSettings.direction
Type of Gear ratio	齿轮比指从轴移动距离和主轴移动距离之间的比值。齿轮比可以为分数（分子/分母）或浮点数。 ✓ Fraction (numerator / denominator) 齿轮比为分数形式。 ✓ Floating-point number 齿轮比为浮点数。 ✓ Last programmed ✓ Preassigned value 缺省值。缺省值为系统变量: userDefault.gearingSettings.defineMode
Gear ratio type	齿轮比的输入值类型。 ✓ Value entry 直接在 Gear ratio numerator/denominator 或者 Gear ratio field 中输入值。 ✓ Last programmed value ✓ Preassigned value 缺省值。 缺省值为系统变量: • userDefault.gearingSettings.numerator 和 userDefault.gearingSettings.denominator 或者 • userDefault.gearingSettings.ratio

Gear ratio numerator	以整数形式输入齿轮比的分子
Gear ratio denominator	以整数形式输入齿轮比的分母。
Transformation ratio	以浮点数形式输入齿轮比。
Reference point	齿轮同步的参考点。 ✓ Gearing takes place relative to axis zero (缺省值) 绝对同步：从值和主对象值之间的线性关系每次都参考轴的零点。 同步开始时主对象值和从值之间的偏差在同步过程中被补偿。但是，可以在 Synchronization 页面的 Start of synchronization 中指定偏差。同步后，该偏差作为从值和主对象值之间恒定相位偏差存在。否则，相位偏差为 0。 ✓ Gearing relative to start position 相对同步：从值和主对象值之间的线性关系每次都参考开始同步时的轴位置。 在同步开始时主设定值和从值之间的偏差在同步过程中不会被补偿。同步后，该偏差作为从值和主设定值之间恒定相位偏差存在。 此外，还可以在 Synchronization 页面的 Start of synchronization 中指定偏差。同步后，该值作为从值和主设定值之间恒定相位偏差存在。 ✓ Last programmed reference point ✓ Preassigned value 缺省值。缺省值为系统变量:userDefault.syncProfile.syncProfileReference
Synchronization reference	同步参考。 ✓ Leading axis 长度相关的同步：同步在指定的主值区间（同步长度）内完成。 • 优点：同步操作可以在定义的主值区间（同步长度）内完成。 • 缺点：同步操作的动态响应由主值（速度）决定，不会考虑从轴的动态响应限制。 ✓ Time 时间相关的同步：同步根据指定的动态响应完成。动态响应在 Dynamics 页面中进行设置。 • 优点：同步操作总是根据设定的动态响应参数进行。 • 缺点：不能预知完成同步所需要的主值区间。 ✓ Last programmed setting ✓ Default value : 缺省值。缺省值为系统变量:userdefault.syncProfile.syncProfileReference。
Start of synchronization	齿轮同步的开始时间。 ✓ at leading axis position： 当主轴到达指定位置时，齿轮同步开始。 需要对以下参数进行设置： • Reference point of leading axis • Leading axis position 同步会根据以下值进行：

	• 主对象值= 上次设置的主轴位置 • 从值：取决于 Reference point 选项： (1). 对于绝对同步: 从值 = 齿轮比 * 主轴位置 (2). 对于相对同步: 从值 = 当前从值 ✓ at master axis position with offset 当主轴到达指定位置时，齿轮同步开始。需要为从轴指定偏差。 需要对以下参数进行设置： • Offset of the following axis • Reference point of leading axis • Leading axis position 同步会根据以下值进行： • 主对象值= 上次设置的主轴位置 • 从值：参考 Offset of the following axis 选项描述。 ✓ Synchronize immediately 立即开始同步。 同步会根据以下值进行： • 主对象值= 当前主对象值 • 从值：取决于 Reference point 选项中的设置： (1). 对于绝对同步（缺省值）: 从值 = 齿轮比 * 当前主值设定值 (2). 对于相对同步: 从值 = 当前从值 ✓ Synchronize immediately with offset 立即开始同步。需要额外设定从轴偏差。 需要对 Offset of the following axis 项进行设置： 同步会根据以下值进行： • 主对象值= 当前主对象值 • 从值：参考 Offset of the following axis 选项描述。 ✓ At the following axis position 当从轴位于指定位置时，同步开始。 需要对以下参数进行设置： • Reference point of leading axis • Following axis position 同步会根据以下值进行： • 主对象值：取决于 Reference point 参数设置： (1). 对于绝对同步（缺省值）：主对象值 = 主轴位置/齿轮比 (2). 对于相对同步: 主对象值= 当前主对象值 • 从值 = 上次设定的从轴位置 ✓ Last programmed setting ✓ Default value 缺省值。缺省值为系统变量: userdefault.gearingSettings.synchronizingMode
Offset of the following axis	如果在 Start of synchronization 选项中选择了如下选项，则需要设置该参数。

	• at master axis position with offset • Synchronize immediately with offset 在编辑框中输入偏差值。 根据 Reference point 选项中的设置，从值为： • 对于绝对同步，从值 = 设定偏差 • 对于相对同步，从值 = 当前从值 + 设定偏差
Reference point of leading axis	如果在 Start of synchronization 选项中选择了如下选项，则需要设置该参数。 • at leading axis position • at master axis position with offset • At the following axis position ✓ Synchronize before synchronization position 同步在设定的位置完成。 ✓ Symmetrical 当位于设定的位置，主轴只走了同步长度的一半。 ✓ Synchronize from synchronization position 同步在设定的位置开始。 ✓ Last programmed reference point of leading axis ✓ Preassigned value 缺省值。缺省值为系统变量: userdefault.syncProfile.syncPositionReference
Synchronization length	如果在 Synchronization reference 选项框中选择了 Leading axis，则需要设置该参数。 在该编辑框中输入同步长度。 ✓ Last programmed synchronization length ✓ Preassigned value 缺省值。缺省值为系统变量: userdefault.syncProfile.syncLength
Following axis position	如果在 Start of synchronization 选项框中选择了 At the following axis position，则需要设置该参数。 在该编辑框中输入从轴位置。 ✓ Last programmed following axis position ✓ Preassigned value 缺省值。缺省值为系统变量: userdefault.gearingSettings.syncPositionSlave
Leading axis position	如果在 Start of synchronization 选项框中选择了如下选项，则需要设置该参数。 • at leading axis position • at master axis position with offset 在该编辑框中输入主从轴位置。 ✓ Last programmed leading axis position ✓ Preassigned value 缺省值。缺省值为系统变量:userdefault.gearingSettings.syncPositionMaster
Synchronization direction	从轴同步时的运动方向。 ✓ Retain system behavior

	同步按照最短路径进行。在此种情况下，当移动轴时，会进行检查以决定是否维持当前的运动方向。 ✓ Maintain direction of following axis 同步按照从轴运动方向进行。 ✓ Positive 同步按照正向进行。 ✓ Negative 同步按照反向进行。 ✓ Shortest path 同步不考虑方向而是根据最短路径进行。 ✓ Preassigned value (缺省值) 缺省值。缺省值为系统变量: userdefault.gearingSettings.synchronizingDirection

5.2.2.2 范例

例 10: 相对同步

参数：Gear ration: 1:1

Reference point: Gearing relative to start position

Start of Synchronization: Synchronize immediately

Synchronization length: 30mm

主轴(Axis_Master)以 20mm/s 的速度运行，从轴(Axis_Slave)静止在 0mm 处，同步立即开始，30mm 后，从轴(Axis_Slave)和主轴(Axis_Master)之间相对同步。

1. 定义同步操作配置

在离线状态下，双击 Axis_Slave（同步轴）下的同步对象 Axis_Slave_SYNCHRONOUS_OPERATION，然后在右边界面勾选 Axis_Master，表示将其作为 Axis_Slave 的主轴，在 Coupling Type 中选择使用主轴的设定值（Setpoint）或实际值（Actual value with extrapolation）作为主值。如图 5.34 所示。

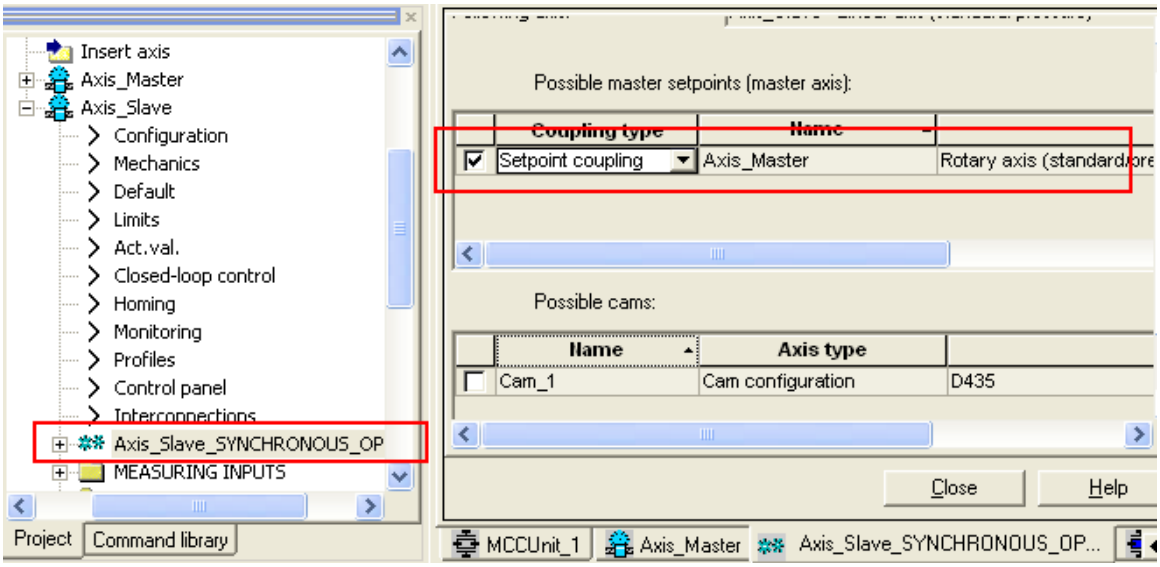


图 5.34 定义同步操作配置

2. 插入 MCC Unit 和 MCC chart，并将其分别命名为 MCCUnit_1 和 MCC_Gearing。

3. 在 GLOBAL DEVICE VARIABLES 中定义三个 Bool 型的变量 g_boStart, g_boStartMove 和 g_boStartGearing。

4. 在 mcc_gearing 中依次插入等待条件指令（g_boStart 的上升沿为触发条件）、轴使能指令（用于使能 Axis_Slave），轴使能命令（用于使能 Axis_Master），轴回零命令（用于回零 Axis_Slave，回零方式为 set home position）, 轴回零命令（用于回零 Axis_Master，回零方式为 set home position），等待条件指令（g_boStartMove 的上升沿为触发条件），位置控制模式下的轴移动命令（用于移动 Axis_Master，速度为 20mm/s），等待条件命令（g_boStartGearing 的上升沿为触发条件）。

5. 插入齿轮同步命令

在 MCC 工具条的同步操作命令组中单击齿轮同步命令，如下图所示，该命令即被插入到 MCC chart 中。

6. 齿轮同步命令参数设置。

双击插入的齿轮同步命令，在弹出的对话框中进行参数设置，点击 OK 确认。

Paremeters 页，如图 5.35：

1) Following axis: Axis_Slave。

2) Leading axis/encoder/external master value: Axis_Master.

3) Gear direction: From sign of gear ratio

4) Type of gear ration: Fraction(nominator/denominator)

5) Gear ratio type: Value entry

6) Gear ratio numerator: 1

7) Gear ratio denominator: 1

8) Reference point: Gearing takes place relative to axis zero(相对同步)

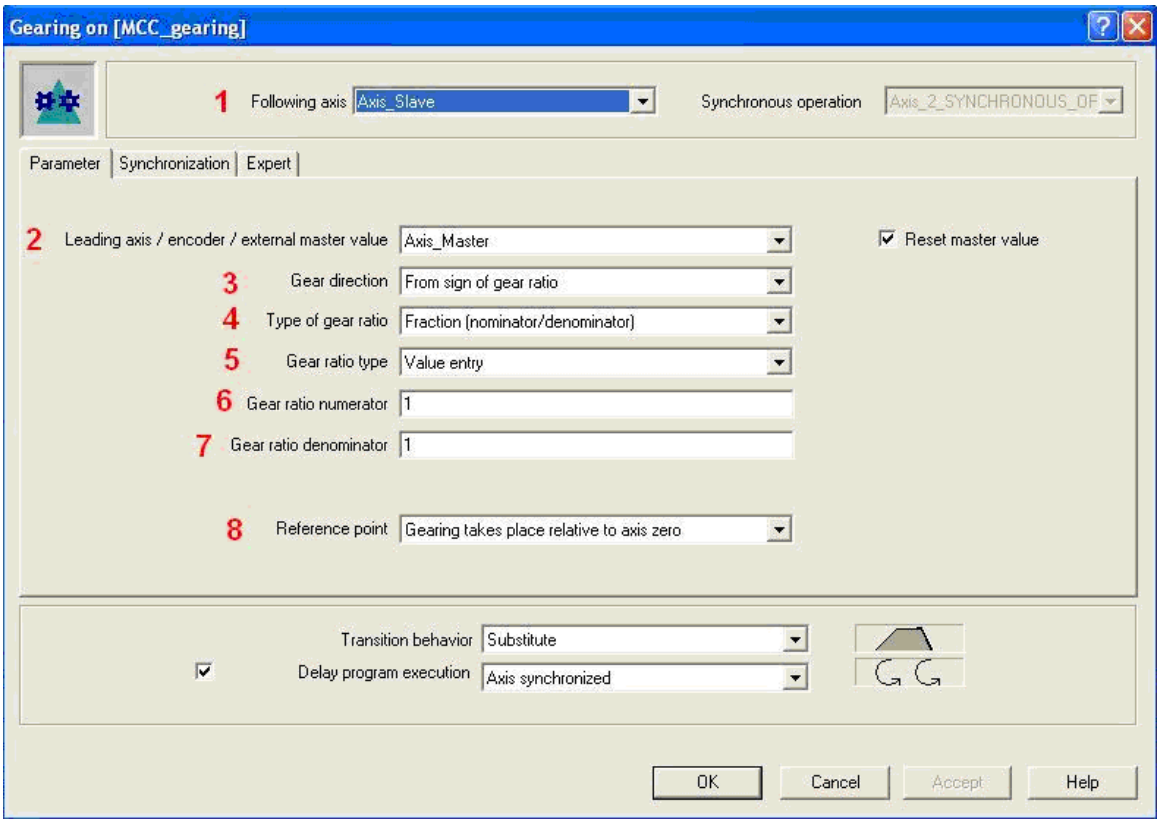


图 5.35 齿轮同步命令参数设置 1

Synchronization 页，如图 5.36 所示：

1) Synchronization reference: Leading axis.

2) Start of synchronization: Synchronize immediately（立即同步）

3) Synchronization length: 30mm

4) Synchronization with look-ahead: Default

5) Synchronization direction: Positive

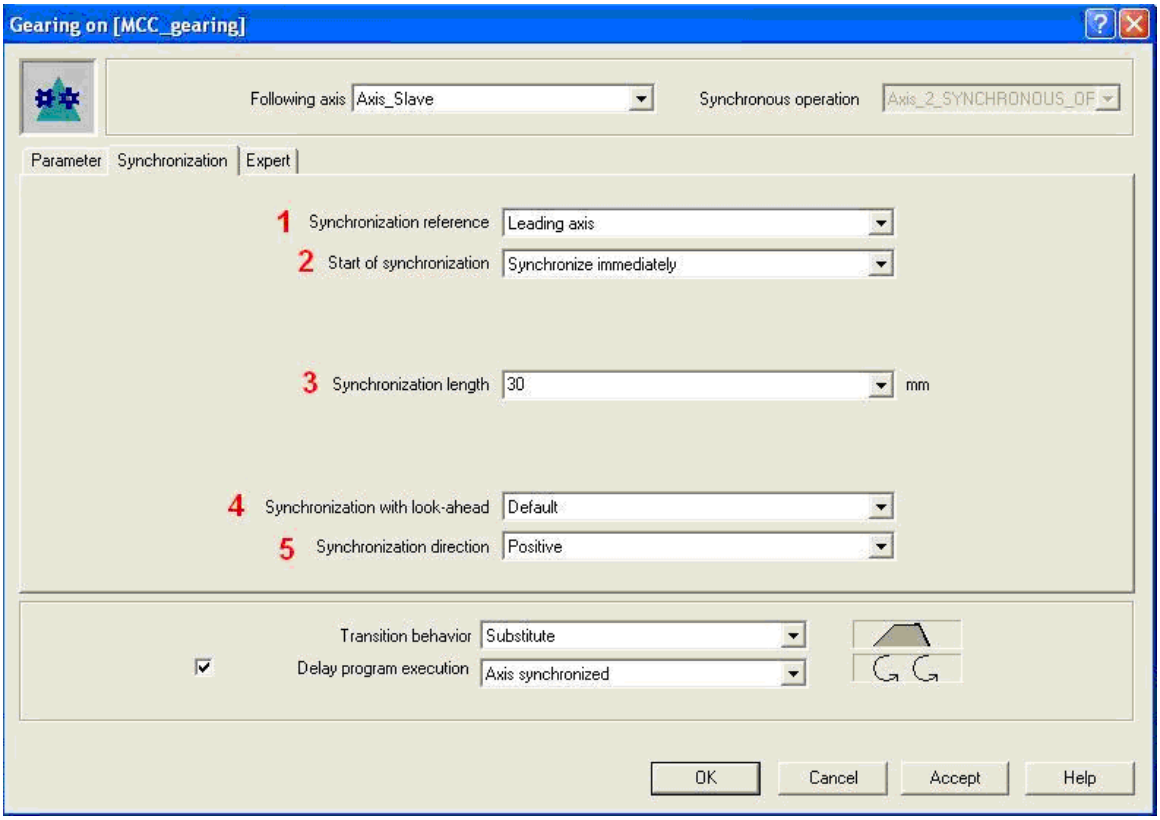


图 5.36 齿轮同步命令参数设置 2

7. 编译并保存。

8. 将程序分配给执行系统。

9. 在线并下载程序。

10. Trace 设置

将以下变量添加到 Trace 列表中：

1) _to.Axis_Master.positioningstate.actualposition

2) _to.Axis_Slave.positioningstate.actualposition

3) _to.Axis_Master.motionstatedata.actualvelocity

4) _to.Axis_Slave.motionstatedata.actualvelocity

5) _to.Axis_Slave_SYNCHRONOUS_OPERATION.state

6) _to.Axis_Slave_SYNCHRONOUS_OPERATION.syncstate

11. 运行程序。

确认 CPU 处于 RUN 状态，置位 g_boStart，轴 Axis_Slave 和 Axis_Master 使能，通过 Axis_Slave.control 和 Axis_Master.control 系统变量可以查看轴有没有被使能，使能后轴 Axis_Slave 和 Axis_Master 进行回零，同样通过查看

Axis_Slave 和 Axis_Master 的系统变量 PositioningState.Homed 可以知道回零是否成功。回零成功后，两轴都位于原点位置，点击 开始 trace， 置位 g_boStartMove，轴 Axis_Master 开始移动，然后置位 g_boStartGearing，同步立即开始，轴 Axis_Slave 开始移动，轴 Axis_Master 移动 30mm 后轴 Axis_Slave 与主轴 Axis_Master 同步。Trace 结果如图 5.37 所示：

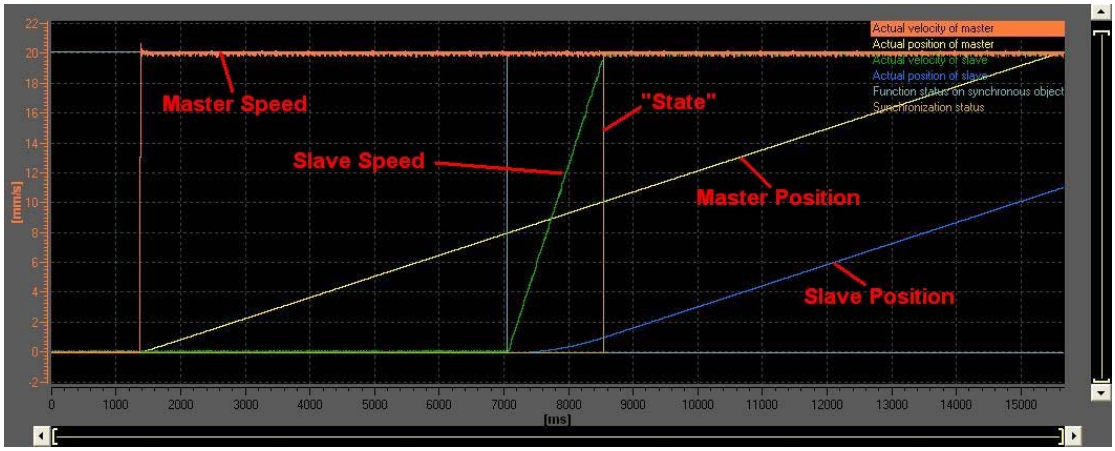


图 5.37 齿轮同步 Trace 图

12. 状态查询

同步对象 Axis_Slave_SYNCHRONOUS_OPERATION 的系统变量 syncstate 为 yes，表示主、从轴正在同步，另一个变量 state 为 Gearing，表示正在进行的同步类型为齿轮同步。

例 11: 相对同步

参数：Gear ration: 1:1

Reference point: Gearing relative to start position

Start of Synchronization: At leading axis position

Reference point of leading axis: Synchronize before synchronization position

Synchronization length: 20mm

Leading axis position: 200mm

Synchronization direction: Positive

主轴(Axis_Master)以 20mm/s 的速度运行，从轴(Axis_Slave)静止在 0mm 处，当主轴位置为 180mm 时同步开始，20mm 后即主轴位置为 200mm，从轴(Axis_Slave)和主轴(Axis_Master)之间相对同步。

同步命令参数设置如图 5.38，5.39 所示，其余操作同例 10。

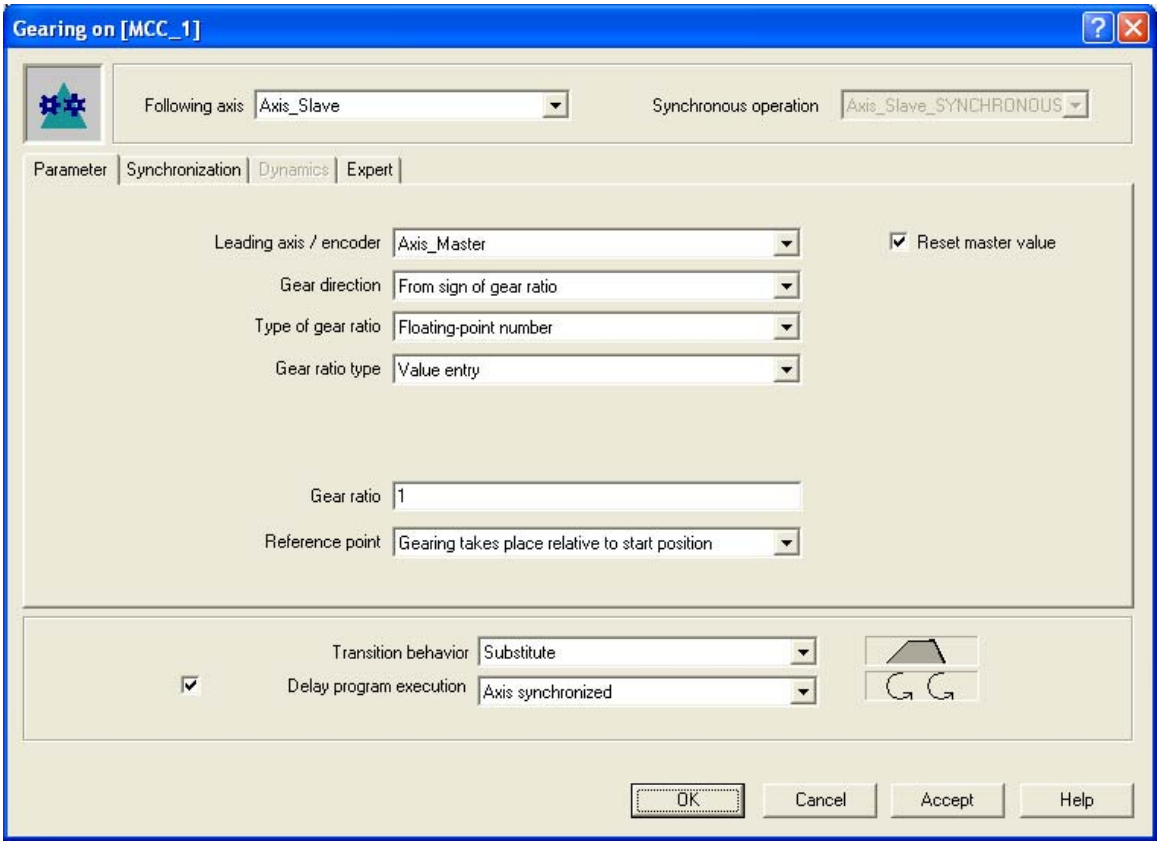


图 5.38 齿轮同步命令参数设置 1

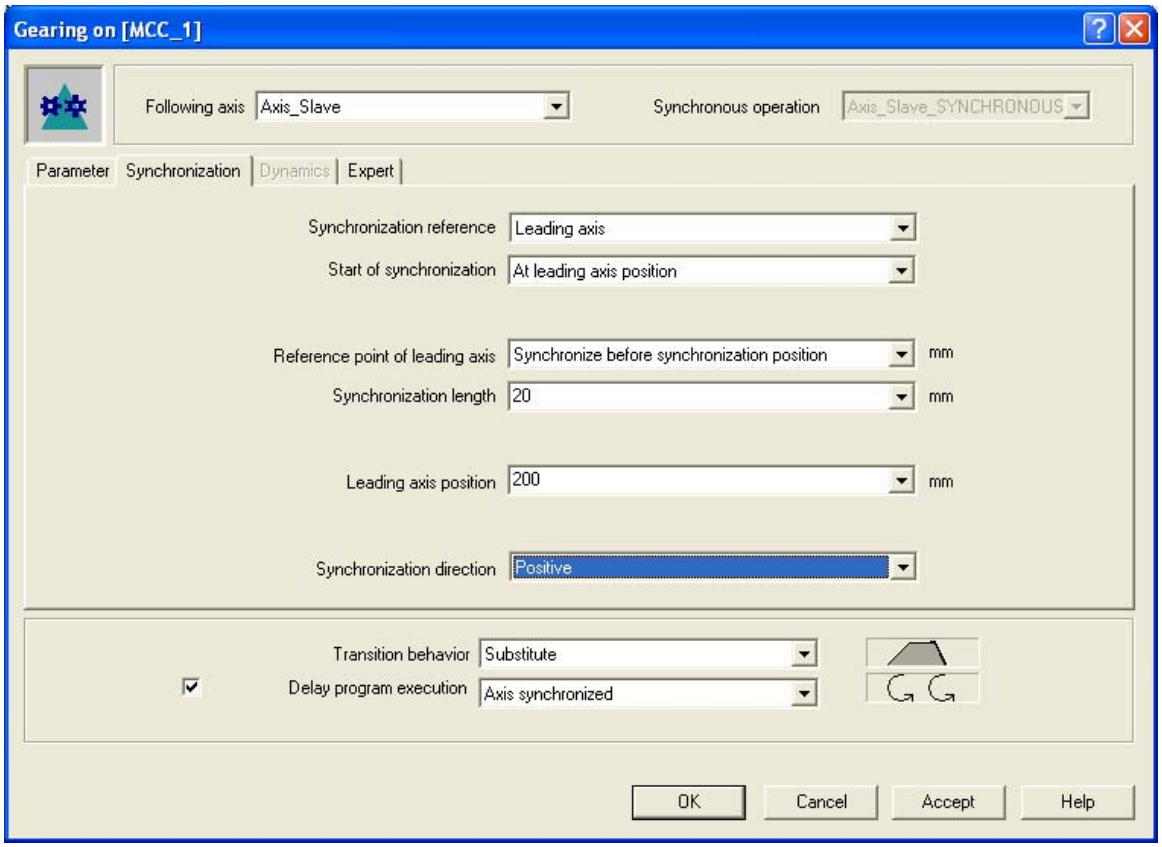


图 5.39 齿轮同步命令参数设置 2

Trace 结果：

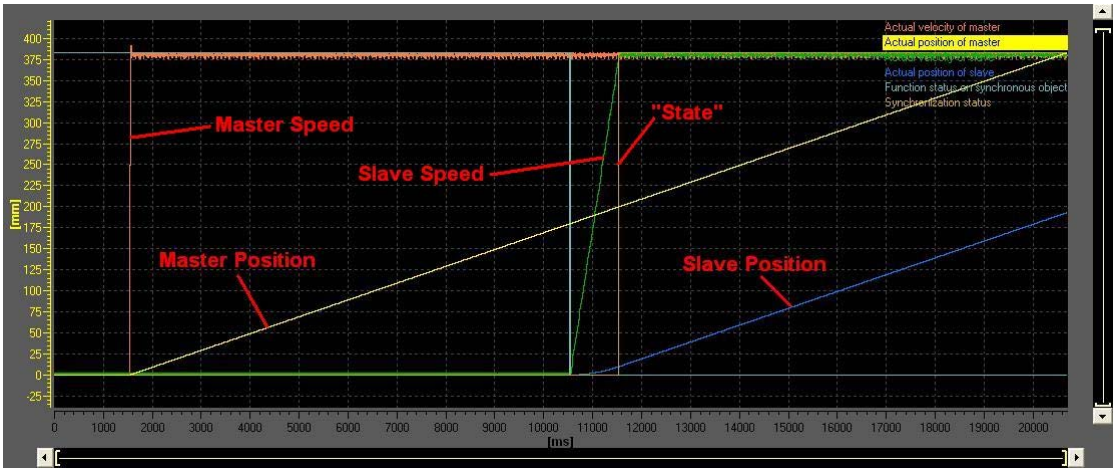


图 5.40 齿轮同步 Trace 图

例 12: 相对同步

参数：Gear ration: 1:1

Reference point: Gearing relative to start position

Start of Synchronization: At leading axis position with offset

Offset of the following axis: 30mm

Reference point of leading axis: Synchronize from synchronization position

Synchronization length: 20mm

Leading axis position: 200mm

Synchronization direction: Positive

主轴(Axis_Master)以 20 mm/s 的速度运行，从轴(Axis_Slave)静止在 0 mm 处，当主轴位置为 200mm 时，同步开始，当主轴位置为 220mm 时，从轴速度与主轴相同，从轴位置为 30mm（偏差+执行同步命令时从轴的位置），之后从轴(Axis_Slave)和主轴(Axis_Master)之间相对同步。

同步命令参数设置如图 5.41，5.42 所示，其余操作同例 10。

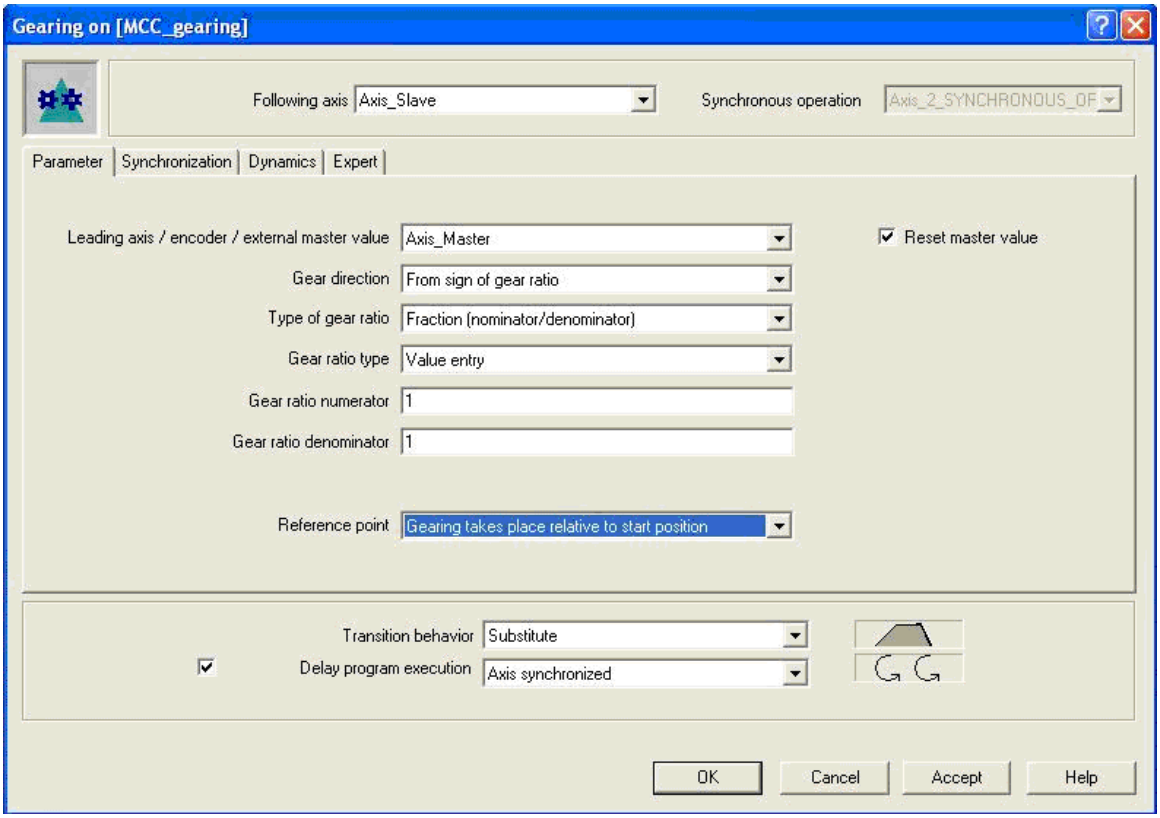


图 5.41 齿轮同步命令参数设置 1

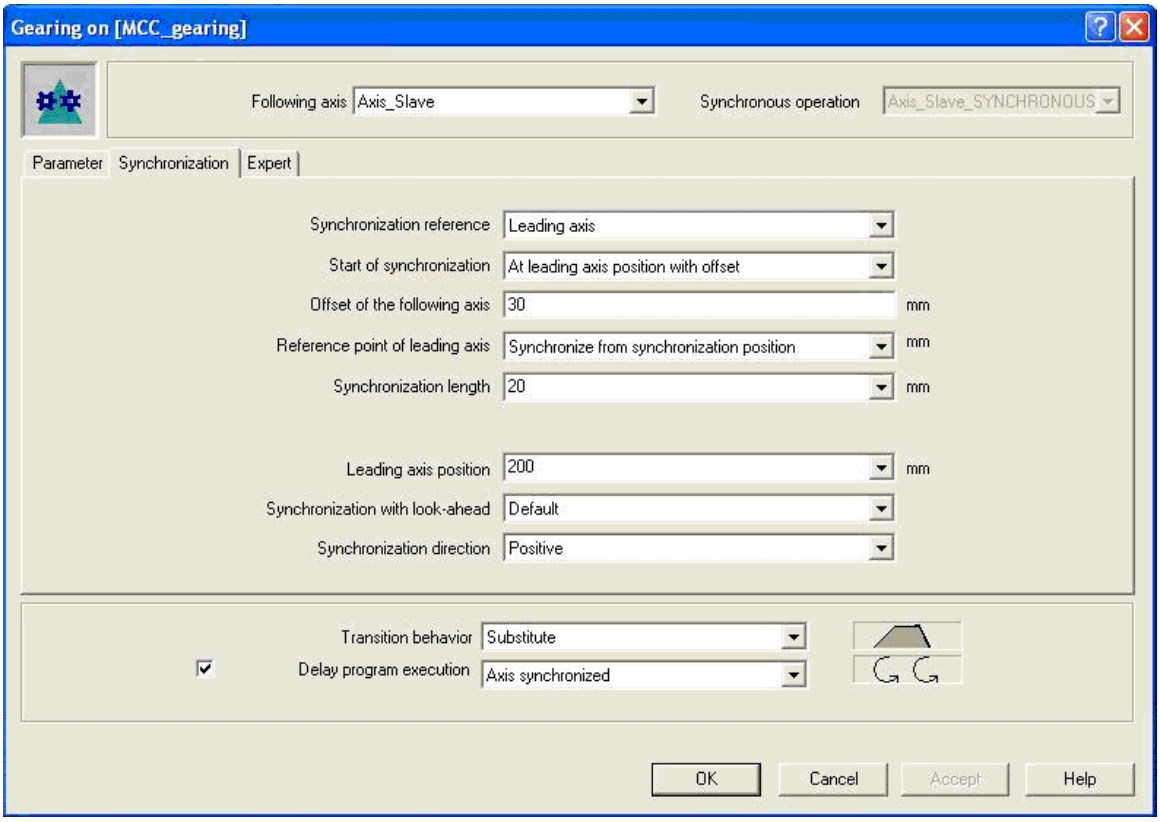


图 5.42 齿轮同步命令参数设置 2

Trace 结果：

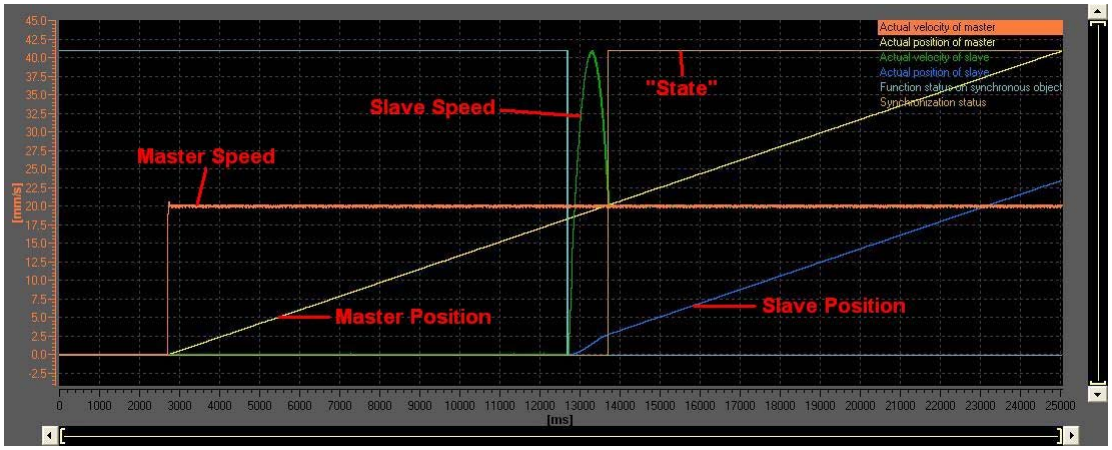


图 5.43 齿轮同步 Trace 图

例 13: 绝对同步

参数：Gear ration: 1:1

Reference point: Gearing relative to axis zero

Start of Synchronization: Synchronize immediately

Synchronization length: 30mm

主轴(Axis_Master)以 20mm/s 的速度运行，从轴(Axis_Slave)静止在 0mm 处，同步立即开始，30mm 后，从轴(Axis_Slave)和主轴(Axis_Master)之间绝对同步。

同步命令参数设置如下图，其余操作同例 10。

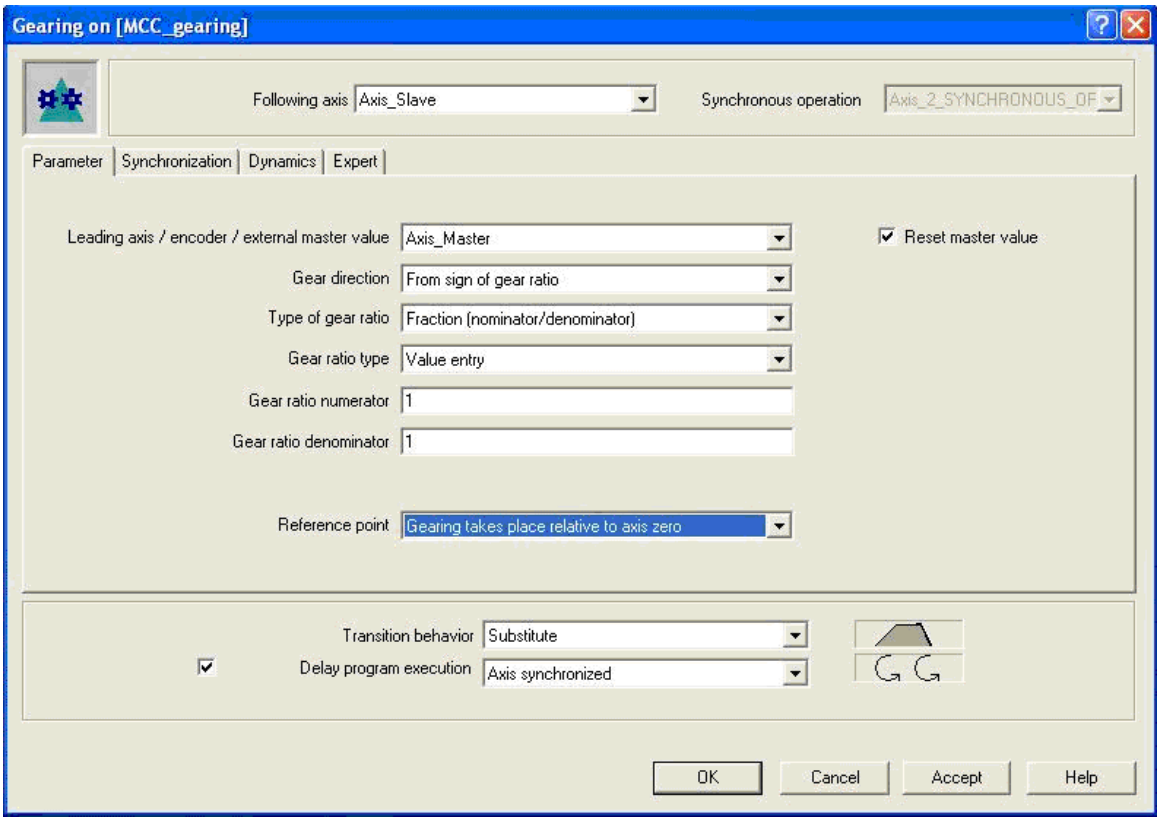


图 5.44 齿轮同步命令参数设置 1

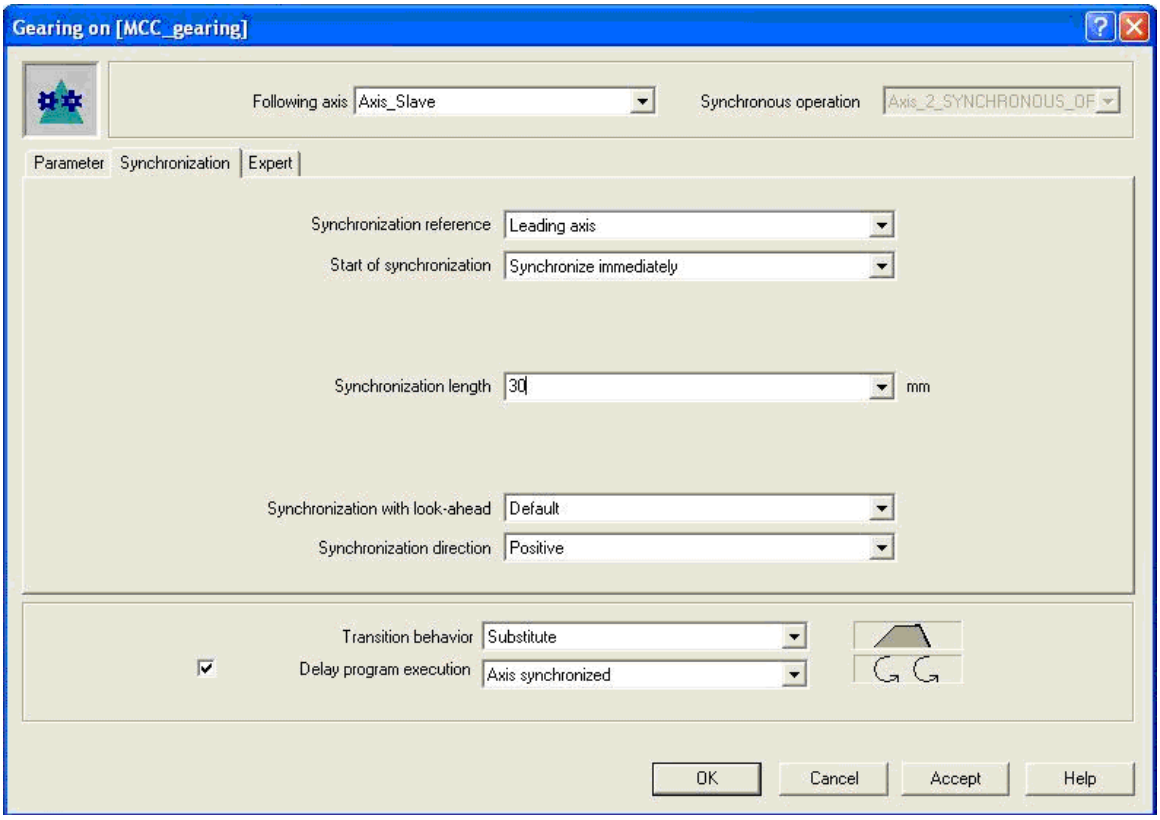


图 5.45 齿轮同步命令参数设置 2

Trace 结果：

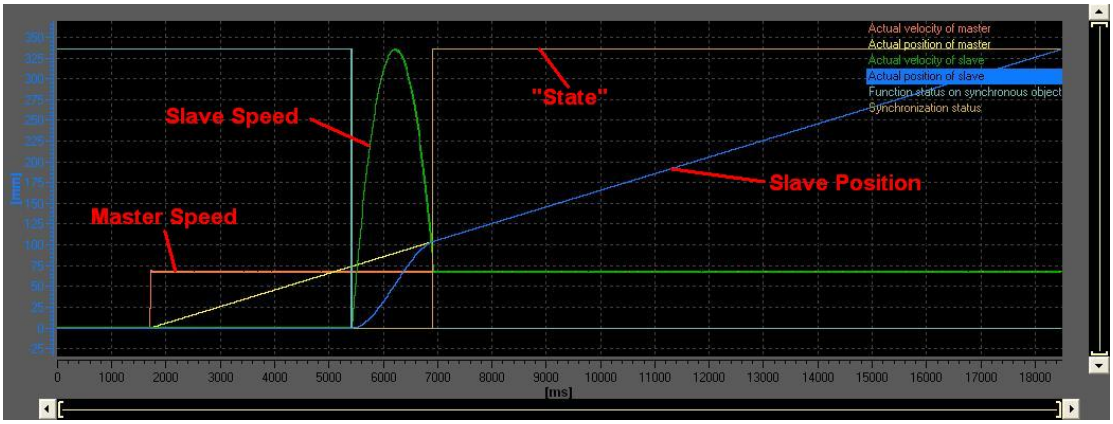


图 5.46 齿轮同步 Trace 图

例 14: 绝对同步

参数：Gear ration: 1:1

Reference point: Gearing relative to axis zero

Start of Synchronization: Time

Reference point of leading axis: Immediately

Velocity: 30 mm/s

Acceleration: 40 mm/s2

Deceleration: 40 mm/s2

主轴(Axis_Master)以 20 mm/s 的速度运行，从轴(Axis_Slave)静止在 0 mm 处，同步立即开始，从轴加速到同步速度（30 mm/s），同步过程结束后，从轴的速度降到和主轴速度一致，从轴(Axis_Slave)和主轴(Axis_Master)之间位置绝对同步。

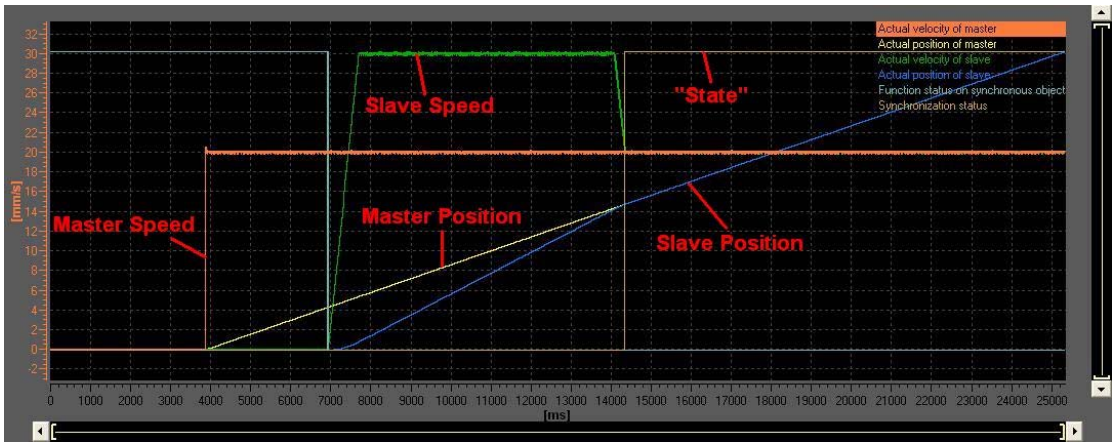


图 5.47 齿轮同步 Trace 图

例 15：绝对同步

参数：Gear ration: 1:1

Reference point: Gearing relative to axis zero

Start of Synchronization: At leading axis position with offset

Offset of the following axis: 30mm

Reference point of leading axis: Symmetrical

Synchronization length: 20mm

Leading axis position: 200mm

Synchronization direction: Positive

主轴(Axis_Master)以 20 mm/s 的速度运行，从轴(Axis_Slave)静止在 0 mm 处，当主轴位置为 190mm 时，同步开始，当主轴位置为 210mm 时，从轴速度与主轴相同，从轴位置为 240mm（偏差+主轴位置），之后从轴(Axis_Slave)和主轴(Axis_Master)之间绝对同步。

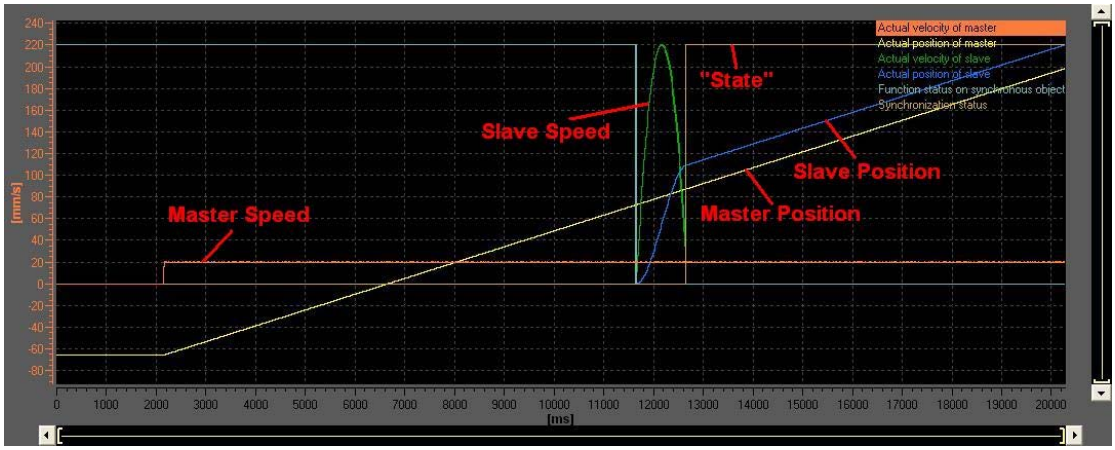


图 5.48 齿轮同步 Trace 图

5.2.3 解除齿轮同步(Gearing Off)

该命令用于解除齿轮同步。

5.2.3.1 命令参数说明

参数	说明
Following axis	指定要执行解除同步操作的轴。从轴可以为: • 所有同步轴。轴定义在项目导航栏的 AXES 文件夹下。 • <Reference>：如果要进行解除同步操作的轴没有定义在设备中而是指定为参考(变量)，则选择<Reference>。 MCC Unit 或 MCC chart 中所有已声明的 followingObjectType 类型的变量都可以在 Synchronous operation 下拉编辑框中找到。
Synchronous operation	根据所选的 Following axis，所有与所选的从轴相关联的同步对象在该下拉列表中显示
Synchronization reference	指定解除同步操作的参考。 ✓ Leading axis 长度相关的同步：同步在指定的主值区间（解除同步长度）内完成。 • 优点：同步操作可以在定义的主值区间（解除同步长度）内完成。 • 缺点：同步操作的动态响应由主值（速度）决定，不会考虑从轴的动态响应限制。 ✓ Time 时间相关的同步：同步根据指定的动态响应完成。动态响应在 Dynamics 页面

	中进行设置。 • 优点：同步操作总是根据设定的动态响应参数进行。 • 缺点：不能预知完成同步所需要的主值区间。 ✓ Last programmed setting ✓ Default value 缺省值。缺省值为系统变量: userdefault.syncProfile.syncProfileReference。
Desynchronization position	解除齿轮同步的开始时间。 ✓ At leading axis position 当主轴到达指定位置时，解除同步开始。 需要对以下参数进行设置： • Reference point of desynchronization position • Leading axis position ✓ Desynchronize immediately 立即开始解除同步。 ✓ At following axis position 当从轴位于指定位置时，解除同步开始。 需要对以下参数进行设置： • Reference point of leading axis • Following axis position ✓ Last programmed start of desynchronization ✓ Preassigned value(default value) 缺省值。缺省值为系统变量: userdefault.gearingSettings.syncOffMode
Reference point of desynchronization positio	如果在 Desynchronization position 选项中选择了如下选项，则需要设置该参数。 • At leading axis position • At following axis position ✓ Stop before desynchronization position 同步在设定的位置完成解除。 ✓ Symmetrical 当位于设定的位置，主轴只走了解除同步长度的一半。 ✓ Start from desynchronization position 在设定的位置开始解除同步。 ✓ Last programmed reference point of leading axis position ✓ Preassigned value 缺省值。缺省值为系统变量：userdefault.syncProfile.syncOffPositionReference
Desynchronization length	如果在 Synchronization reference 选项框中选择了 Leading axis，则需要设置该参数。 在该编辑框中输入同步长度。 ✓ Last programmed synchronization length ✓ Preassigned value 缺省值。 缺省值为系统变量: userdefault.syncProfile.syncOffLength
Following axis	如果在 Desynchronization position 选项框中选择了 At following axis position，则

position	需要设置该参数。 在该编辑框中输入从轴位置。 ✓ Default 缺省值。缺省值为系统变量: userdefault.gearingSettings.syncOffPositions.Slave
Leading axis position	如果在 Desynchronization position 选项框中选择了 at leading axis position，则需要设置该参数。 在该编辑框中输入主从轴位置。 ✓ Default 缺省值。缺省值为系统变量:userdefault.gearingSettings.syncOffPositions.Master
Synchronization direction	从轴同步时的运动方向。 ✓ Retain system behavior 同步按照最短路径进行。在此种情况下，当移动轴时，会进行检查以决定是否维持当前的运动方向。 ✓ Maintain direction of following axis 同步按照从轴运动方向进行。 ✓ Positive 同步按照正向进行。 ✓ Negative 同步按照反向进行。 ✓ Shortest path 同步不考虑方向而是根据最短路径进行。 ✓ Preassigned value (缺省值) 缺省值。缺省值为系统变量:userdefault.gearingSettings.synchronizingDirection

5.2.3.2 范例

例 16:

参数: Synchronization referenc: Leading axis

Desynchronization position: Desynchronize immediately

Desynchronization length: 10

Synchronization direction: Positive

立即解除同步，在 10mm 内从轴与主轴脱离同步。

1. 添加解除齿轮同步命令

在齿轮同步例 10 的 MCC chart 中继续添加 wait for condition 命令（g_boStopGearing 的上升沿作为触发条件）。

在 MCC 工具条的同步操作命令组中单击解除齿轮同步命令，将其插入到 MCC chart 中。

2. 解除齿轮同步命令参数设置

双击插入的 Gearing off 命令，在弹出的对话框中进行参数设置，点击 OK 确认。

Paremeters 页，如图 5.49 所示：

1) Following axis: Axis_Slave。

2) Synchronization referenc: Leading axis

3) Desynchronization position: Desynchronize immediately

4) Desynchronization length: 10

5) Synchronization direction: Positive

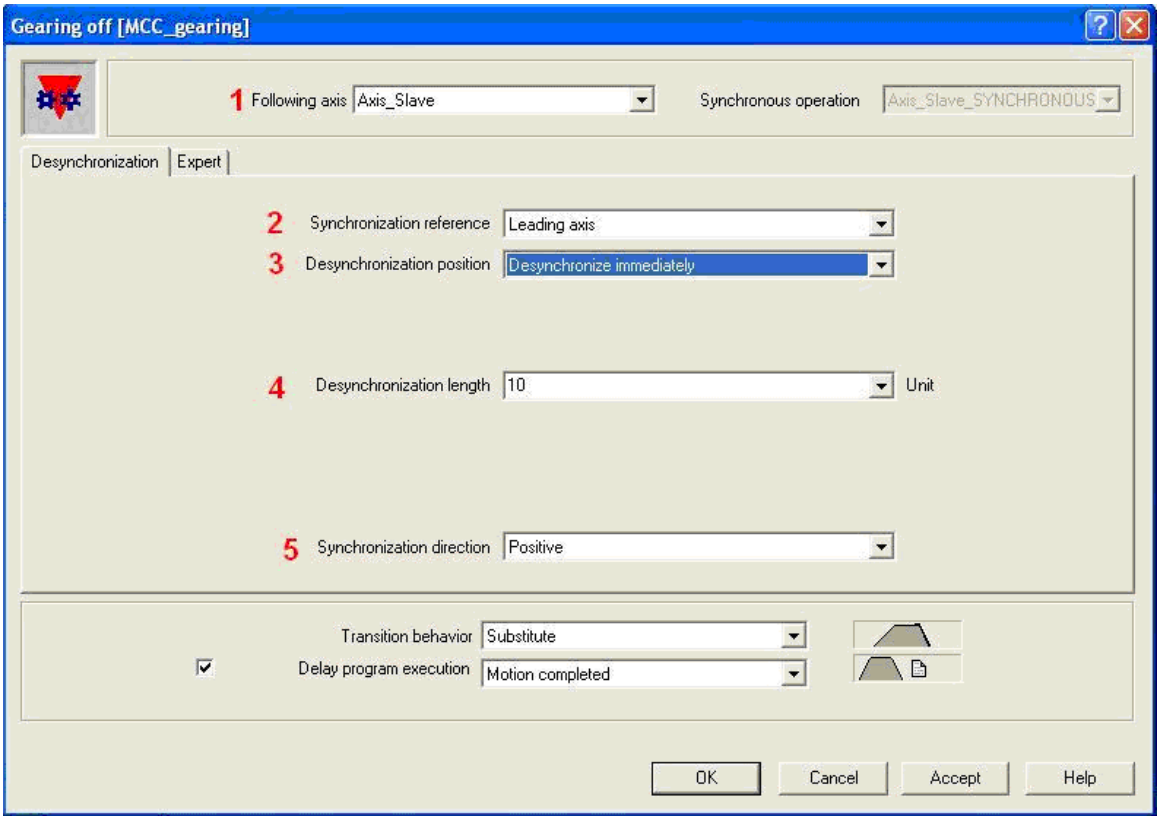


图 5.49 解除齿轮同步命令参数设置

其余操作均同例 10。

当从轴与主轴同步后，置位 g_boStopGearing，从轴立即开始解除与主轴的齿轮同步。结果如下图所示：

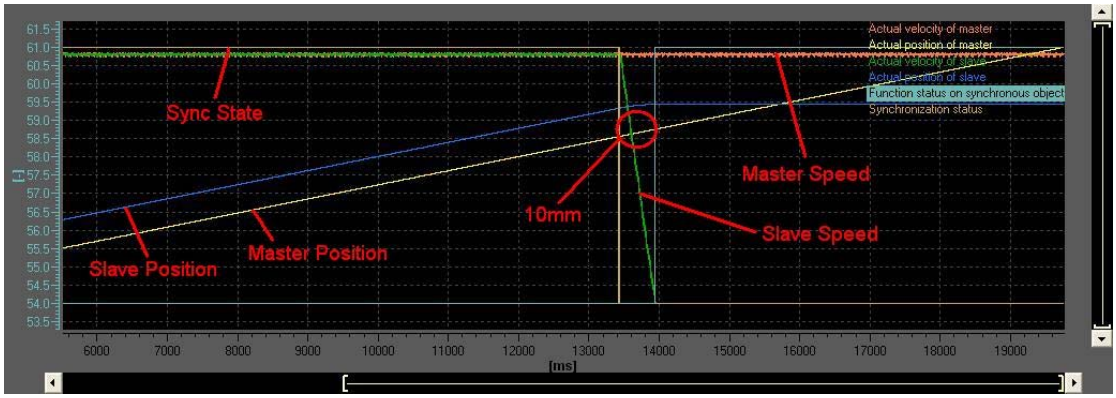
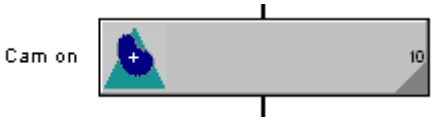


图 5.50 解除齿轮同步 Trace 图

5.2.4 凸轮同步（Cam On）



5.2.4.1 命令参数说明

参数	说明
Following axis	指定要进行凸轮同步的轴。从轴可以为: • 所有同步轴。轴定义在项目导航栏的 AXES 文件夹下。 • <Reference> ：如果要进行同步操作的轴没有定义在设备中而是指定为参考(变量)，则选择<Reference>。MCC Unit 或 MCC chart 中所有已声明的 followingObjectType 类型的变量都可以在 Synchronous operation 下拉编辑框中找到。
Synchronous operation	根据所选的 Following axis，所有可用的同步对象在该下拉列表中显示。
Reset master value	如果想要设置同步关系的主值，则激活该复选框（缺省值）。 如果清除该复选框则保持上次主值。 如果勾选了 Reset master value 复选框则需要设置该参数。
Master axis/encoder/external master value	可以选择： • 设备或 DP 主战中所有可用的定位和同步轴和外部编码器。 • 所有 MCC 源文件或 MCCchart 中声明 posAxis, followingAxis, or externalEncoderType 类型的工艺对象变量。
Cam	选择描述同步关系的凸轮。可以为： • 所有相关设备上已定义的凸轮。凸轮位于项目导航栏的 CAMS 文件夹中。 • 所有 MCC 源文件或 MCCchart 中声明的 camType 类型的工艺对象变量。 注意：在后一种情况中，必须将项目导航中的凸轮和从轴的同步对象建立

联系。

选择当主值增加时，凸轮的运动方向。

Cam direction	✓ Positive 凸轮的运动方向和主值的变化方向一致。 • 如果主值正向变化，凸轮则朝着定义区间值增加的方向（向右）运动。 • 如果主值负向变化，凸轮则朝着定义区间值减少的方向（向左）运动。 ✓ Negative 凸轮的运动方向和主值的变化方向相反。 • 如果主值正向变化，凸轮则朝着定义区间值减少的方向（向左）运动。 • 如果主值负向变化，凸轮则朝着定义区间值增加的方向（向右）运动。 ✓ Last programmed direction ✓ Preassigned value (缺省值) 缺省值为: userdefault.cammingSettings.direction
Evaluation of the leading axis	主值为绝对值或相对值。 ✓ Absolute 在凸轮定义区间内主值为绝对值。 ✓ Relative 在凸轮定义区间内主值为相对值（相对于凸轮的起点）。 ✓ Last programmed value ✓ Default (缺省值) 缺省值为:userDefault.cammingSettings.masterMode
Evaluation of the following axis	从值为绝对值或相对值 ✓ Absolute 如果凸轮为循环运行，那么每次新的凸轮循环，从值都从初始值开始。 ✓ Relative 如果凸轮为循环运行，那么每次新的凸轮循环，从值从上次凸轮循环的终点开始。 ✓ Last programmed value ✓ Default (缺省值) 缺省值为系统变量:userDefault.cammingSettings.slaveMode
Processing the cam	选择凸轮是否循环运行。 ✓ Cyclic processing 如果主值到达了凸轮定义区间的重点，则凸轮从起点继续运行。 ✓ Non-cyclic processing 主值限定在凸轮定义区间内。凸轮只在定义区间内执行一次。 如果主值达到凸轮定义区间的起点或终点，凸轮不再运行。当主值同方向再此穿过，从值不变。

✓ Last programmed cam mode

✓ Preassignment (缺省值)

缺省值为系统变量:userdefault.CammingSettings.cammingMode

同步参考

✓ Leading axis

长度相关的同步：同步在指定的主值区间（同步长度）内完成。

- 优点：同步在可定义的主值区间（同步长度）内完成。
- 缺点：同步操作的动态响应由主值（速度）决定。从轴的动态响应限制不会被考虑。

Synchronization reference

✓ Time

时间相关的同步：同步根据指定的动态响应性完成。动态响应在 Dynamics 页面中进行设置。

- 优点：同步操作总是根据指定的动态响应参数进行。
- 缺点：不能预知完成同步所需要的主值设定值区间。

✓ Last programmed setting

✓ Default value

缺省值。缺省值为系统变量: userdefault.syncProfile.syncProfileReference。

齿轮同步的开始时间。

✓ at leading axis position

当主轴到达指定位置时，凸轮同步开始。

需要在以下选项中进行设置：

- Reference point of leading axis
- Leading axis position

同步根据下值进行：

- 主值设定值 = 上次设置的主轴位置
- 从值：取决于 Evaluation of the following axis 选项：

(1). 对于绝对同步: 从值会根据设定的主轴位置并且考虑以下参数进行计

Start of synchronization

算：

Evaluation of the leading axis

Processing the cam

Any offset to cam starting point

(2). 对于相对同步: 从值 = 当前从值

✓ at master axis position with offset

当主轴到达指定位置时，凸轮同步开始。需要为从轴指定偏差。

需要在以下选项中进行设置：

- Offset of the following axis
- Reference point of leading axis
- Leading axis position

同步根据下值进行：

- 主值设定值 = 上次设置的主轴位置

- 从值：取决于 Evaluation of the following axis 选项。

(1). 对于绝对同步：

从值 = 经过计算得出的从值 + 设定偏差；

经过计算得出的从值与 at leading axis position 中的一样。

(2). 对于相对同步：

从值 = 当前从值 ＋ 设定偏差

✓ Synchronize immediately

立即开始同步。

同步根据下值进行：

- 主值 = 当前主值
- 从值：取决于 Evaluation of the following axis 选项：

(1). 对于绝对同步:

从值会根据设定的主轴位置并且考虑以下参数进行计算：

Evaluation of the leading axis

Processing the cam

Any offset to cam starting point

(2).对于相对同步: 从值 = 当前从值

✓ Synchronize immediately with offset

立即开始同步。需要额外设定从轴偏差。

需要在 Offset of the following axis 选项中进行设置：

同步根据下值进行：

- 主值 = 当前主值
- 从值：取决于 Evaluation of the following axis 选项。

(1). 对于绝对同步：

从值 = 经过计算得出的从值 + 设定偏差；

经过方法与 Synchronize immediately 中的一样。

(2). 对于相对同步：从值 = 当前从值 ＋ 设定偏差

✓ At end of cam cycle

该选项只有当 Evaluation of the leading axis 为 Relative 时才可选。

凸轮同步在下面两种情况下都会开始：

- 该同步对象已经在用另一个凸轮进行同步。
- 处于凸轮同步中的主值到达了凸轮终点或者凸轮循环周期的终点。

这样，会在指定点及时的从一个凸轮切换到另一个凸轮。

需要在 Reference point of leading axis 选项中进行设置。

同步根据下值进行：

- 主值 ＝ 凸轮循环周期终点的主值
- 从值：取决于 Evaluation of the following axis 选项。

(1). 对于绝对同步：

从值根据凸轮循环终点的主值并且参考以下值进行计算：

Processing the cam

Offset in relation to cam starting point

(2). 对于相对同步：

从值 = 当前从值

✓ Last programmed setting

✓ Default value

缺省值。缺省值为系统变量: userdefault.cammingSettings.synchronizingMode

如果在 Start of synchronization 选项中选择了如下，则需要定义改参数。

Offset of the following axis

- at master axis position with offset
- Synchronize immediately with offset

在该编辑框中输入值，该值将叠加到计算的或当前从轴位置。

如果在 Start of synchronization 选项框中选择了如下选项，则需要设置该参数。

- at leading axis position
- at master axis position with offset
- At end of cam cycle

✓ Synchronize before synchronization position

Reference point of leading axis

同步在设定的位置完成。

✓ Symmetrical

当位于设定的位置，主轴只走了同步长度的一半。

✓ Synchronize from synchronization position

同步在设定的位置开始

✓ Last programmed reference point of leading axis

✓ Preassigned value

缺省值。缺省值为系统变量：userdefault.syncProfile.syncPositionReference

如果在 Synchronization reference 选项框中选择了 Leading axis，则需要设置该参数。

Synchronization length

在该编辑框中输入同步长度。

✓ Last programmed synchronization length

✓ Preassigned value

缺省值。缺省值为系统变量:userdefault.syncProfile.syncLength

如果在 Evaluation of the leading axis field 选项框中选择了 Relative，则需设定该参数。

Offset in relation to cam starting point

该参数指定了凸轮在其定义区间内的起点。在编辑框中输入值。

✓ Last programmed starting point

✓ Preassignment (缺省值)

缺省值为系统变量:userdefault.cammingSettings.camStartPosition

在该编辑框中输入从轴位置。

Following axis position

✓ Last programmed following axis position

✓ Preassigned value

缺省值。缺省值为系统变量:cammingSettings.syncPositionSlave

Leading axis

如果在 Start of synchronization 选项框中选择了如下选项，则需要设置该参

position	数。 • at leading axis position • at master axis position with offset 在该编辑框中输入主从轴位置。 ✓ Last programmed leading axis position ✓ Preassigned value 缺省值。缺省值为系统变量:userdefault.cammingSettings.syncPositionMaster 指定从轴同步的运动方向。 ✓ Retain system behavior 同步按照最短路径进行。此种情况下，当移动轴时，会进行检查以决定是否维持当前的运动方向。 ✓ Maintain direction of following axis 同步按照从轴运动方向进行。
Synchronization direction	✓ Positive 同步按照正向进行。 ✓ Negative 同步按照反向进行。 ✓ Shortest path 同步不考虑方向而是根据最短路径进行。 ✓ Preassigned value (缺省值) 缺省值。缺省值为系统变量： userdefault.cammingSettings.synchronizingDirection

5.2.4.2 范例

例 17:

1. 轴配置

轴 Axis_Master 为模态线性定位轴，模态长度为 1000mm，轴 Axis_Slave 为模态线性同步轴，模态长度为 1000mm。

2. 插入凸轮曲线

双击项目导航栏中 CAMS 文件夹下的 Insert cam，在弹出的对话框进行设置，点击 OK 按钮确认。如图 5.51 所示：

1) Name：所创建的凸轮曲线的名称。本例为 Cam_1。

2) Type：凸轮曲线的类型。创建凸轮时，可以选择多项式(Polynomials)或插值点列表(Interpolation point table)来定义凸轮。本例为多项式（Polynomials）。

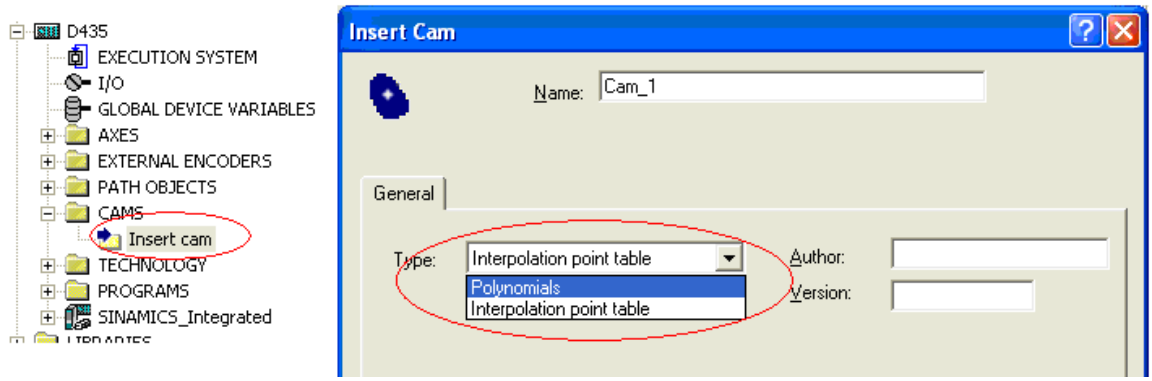


图 5.51 创建 Cam

3. 编辑凸轮曲线

在 Normalized Polynomial 中选择相应的曲线段，然后点击 VDI Wizard 按钮，如图 5.52 所示。

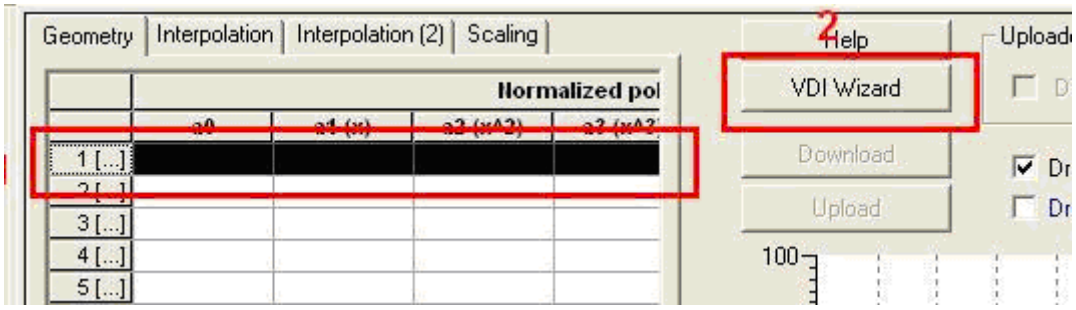


图 5.52 选择曲线段

在弹出的对话框中，选择运动段的类型，本例选择 Dwell to Dwell，单击 Next 按钮，如图 5.53 所示。

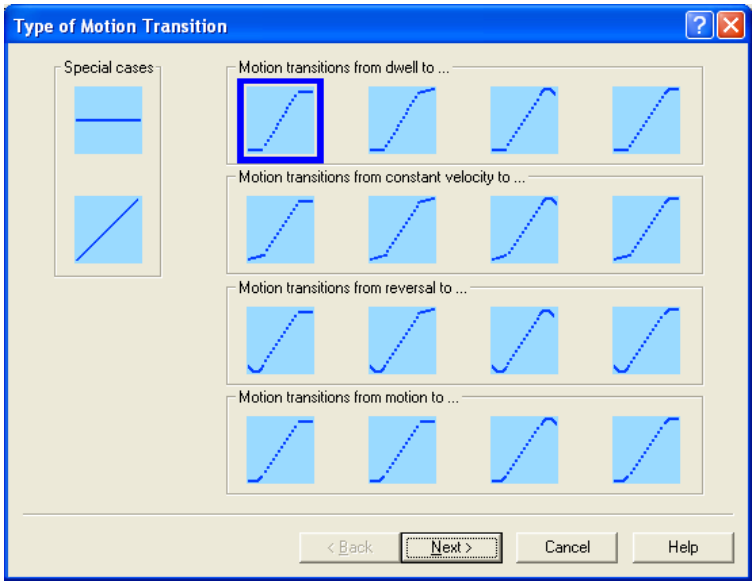


图 5.53 选择运动段的类型

选择 Symmetric(对称)或者 Asymmetric（非对称），本例为 Symmetric(对称)，点击 Next 按钮。

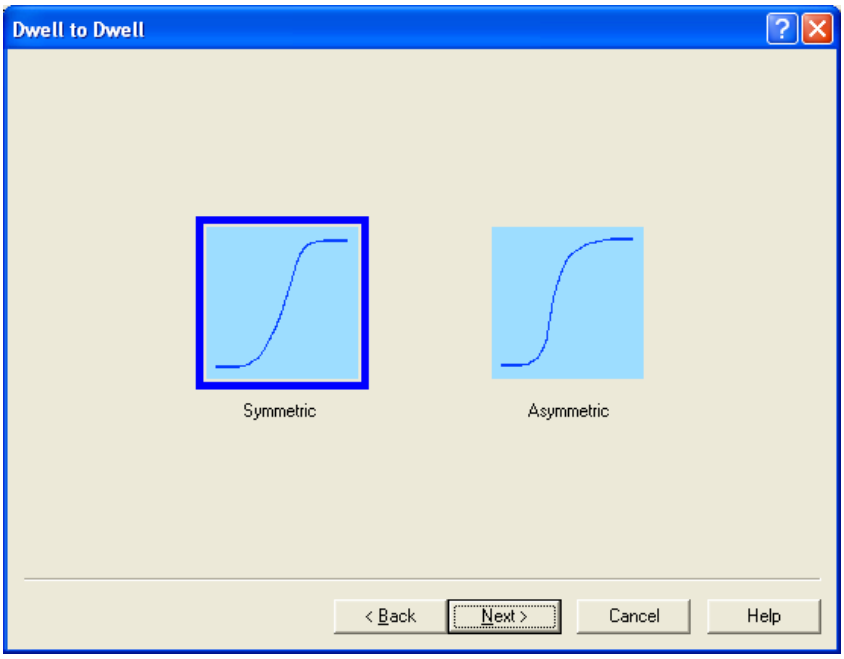


图 5.54 选择 Symmetric(对称)或者 Asymmetric（非对称）

选择多项式的类型，直线，5 次多项式等等，本例选择 Polynomial 5th order，点击 Next 按钮。

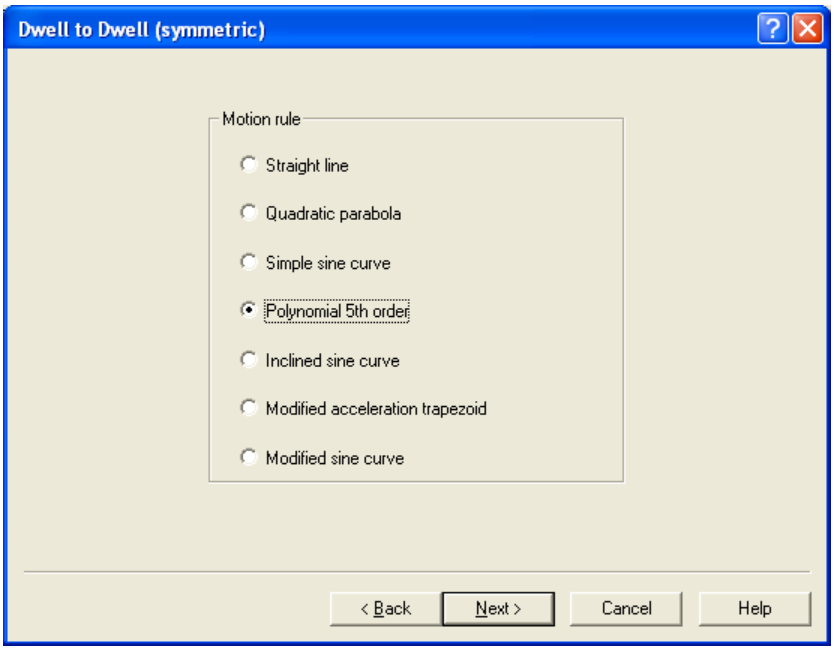


图 5.55 选择多项式的类型

定义曲线段的起点和终点坐标，如图 5.56 所示。点击 Next 按钮。

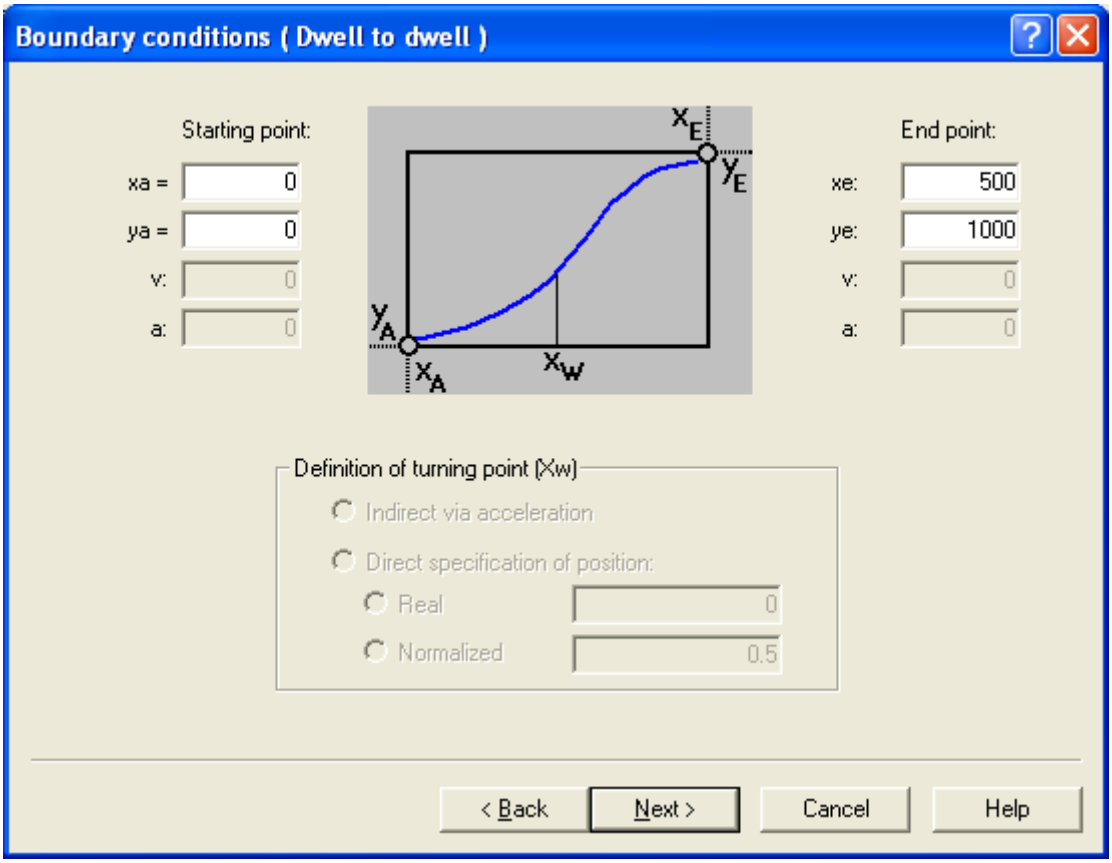


图 5.56 定义曲线段的起点和终点坐标

可以点击 Preview 按钮预览曲线段，确认无误后点击 Finish 按钮结束该曲线段的编辑。在右边窗口中可以看到刚刚创建的 CAM 曲线。

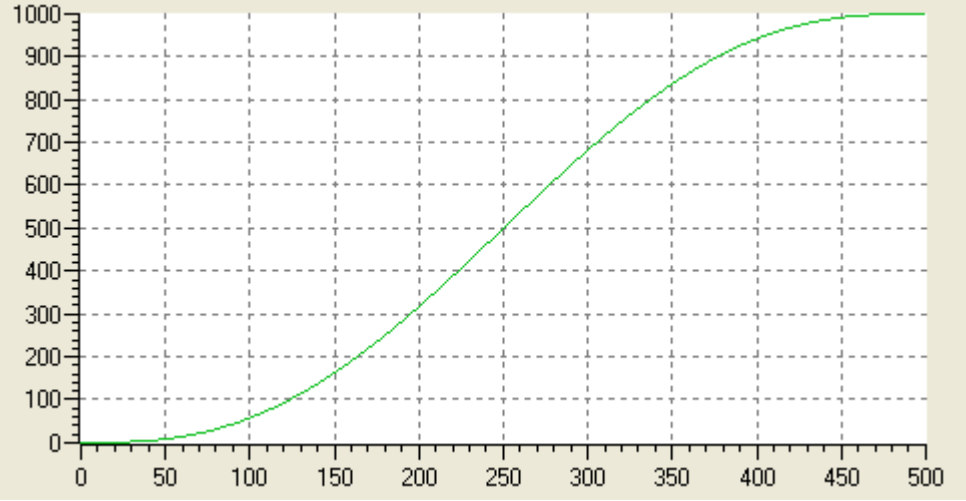


图 5.57 创建的曲线段

按照同样的方法创建第二条曲线段，向导的最后一步根据如图 5.58 所示设置起点和终点。

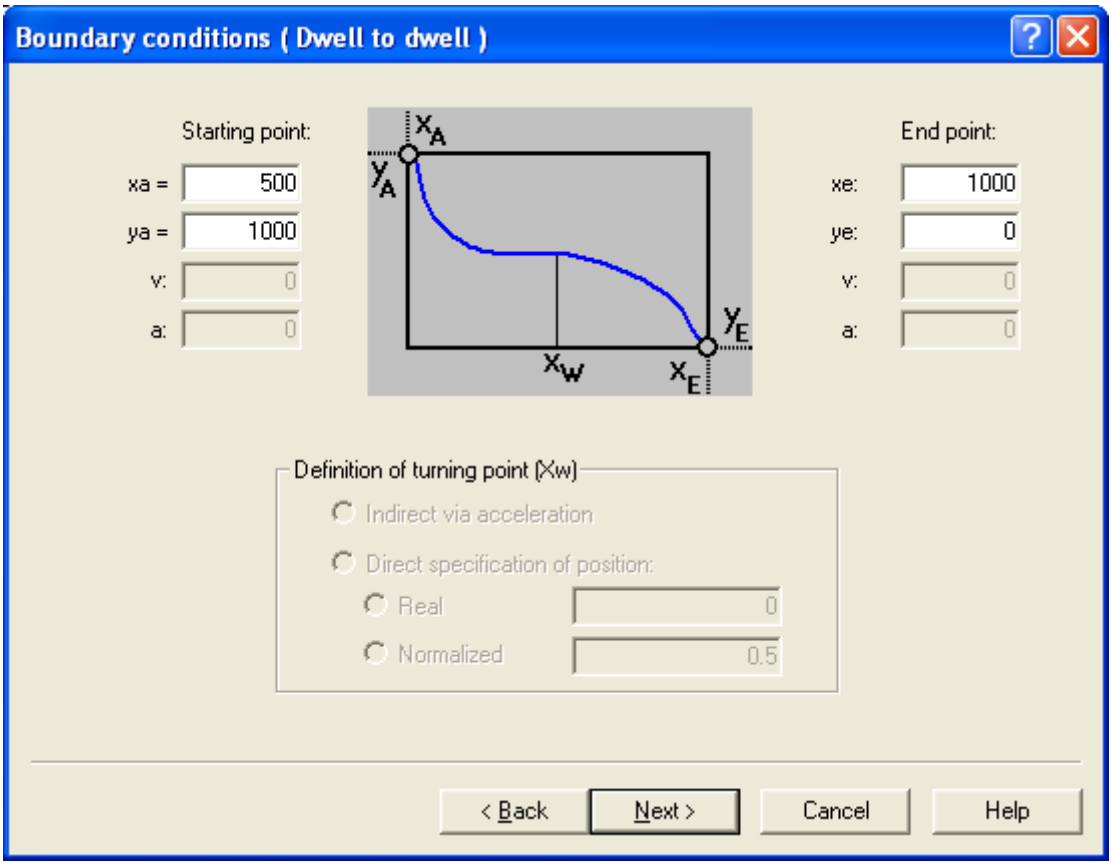


图 5.58 第二条曲线段设置

整个凸轮曲线如图 5.59 所示。

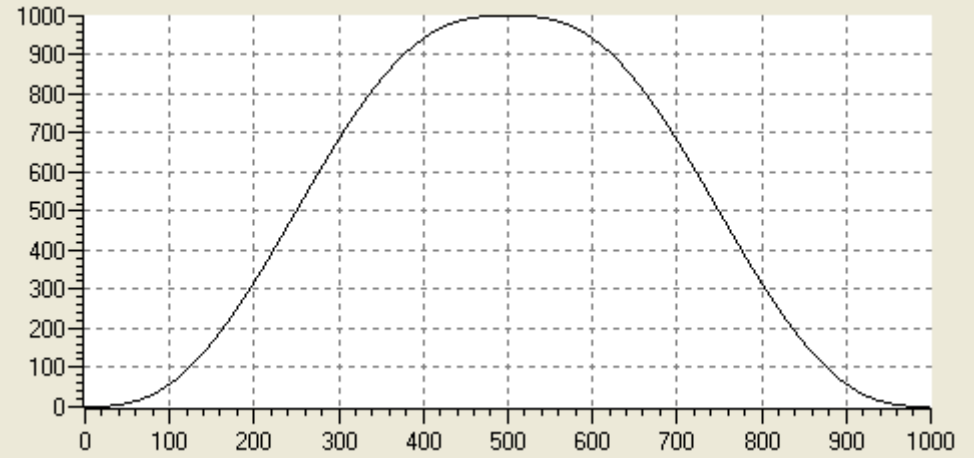


图 5.59 完整的 Cam 曲线

4. 定义同步操作配置

在离线状态下，双击 Axis_Slave（同步轴）下的同步对象 Axis_Slave_SYNCHRONOUS_OPERATION，然后在右边界面的

Interconnections with cams 中勾选已经创建的凸轮曲线，本例为 Cam_1，这样 Cam_1 就与 Axis_Slave 的同步对象相关联了，如图 5.60 所示。

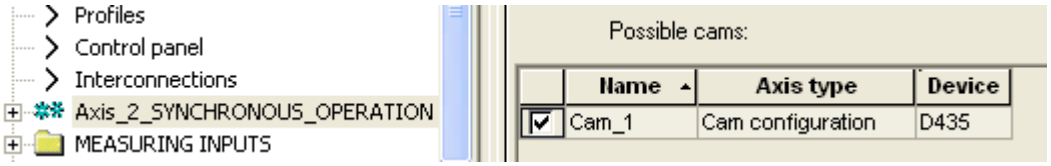


图 5.60 定义同步操作配置

5. 插入 MCC Unit 和 MCC chart，分别将其命名为 MCCUnit_1 和 MCC_Camming。

6. 定义变量

在 GLOBAL DEVICE VARIABLES 中定义三个 Bool 型的变量用于控制程序运行，本例中为 g_boStart, g_boStartMove 和 g_boStartCamming。

7. 在 mcc_camming 中依次插入 Wait for condition 指令（g_boStart 的上升沿为触发条件）、Switch axis enable 指令（用于使能 Axis_Slave），Switch axis enable 指令（用于使能 Axis_Master）， Homing axis（用于回零 Axis_Slave，回零方式为 set home position）, Homing axis（用于回零 Axis_Master，回零方式为 set home position）, Wait for condition 指令（g_boStartMove 的上升沿为触发条件）,Start axis position-controlled 指令（用于移动 Axis_Master，速度为 20mm/s），Wait for condition 指令（g_boStartCamming 的上升沿为触发条件）。

8. 插入凸轮同步命令

在 MCC 工具栏的同步操作命令组中单击凸轮同步命令，将其插入到 MCC chart 中。

9. 凸轮同步命令参数设置

双击插入的 Cam on 命令，在弹出的对话框中进行参数设置，点击 OK 确认。

Paremeters 页，如图 5.61 所示：

1) Following axis: Axis_Slave

2) Leading axis/encoder/external master value: Axis_Master

3) Cam: Cam_1

4) Cam direction: Positive

5) Evaluation of the leading axis: Relative

6) Evaluation of the following axis: Relative

7) Cam processing: Cyclic processing

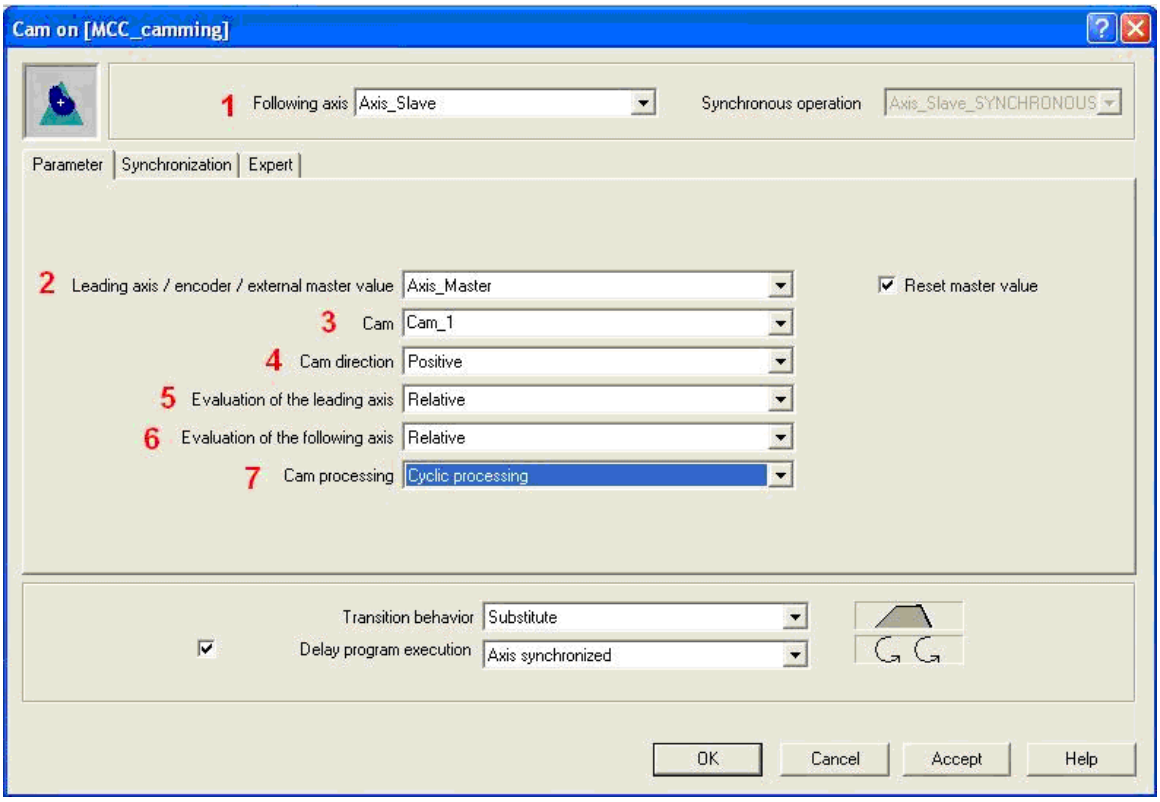


图 5.61 凸轮同步命令参数设置 1

Synchronization 页，如图 5.62 所示：

1) Synchronization reference: Leading axis.

2) Start of synchronization: Synchronize immediately

3) Synchronization length: 10

4) offset to cam starting point: 0

5) Synchronization direction: Positive

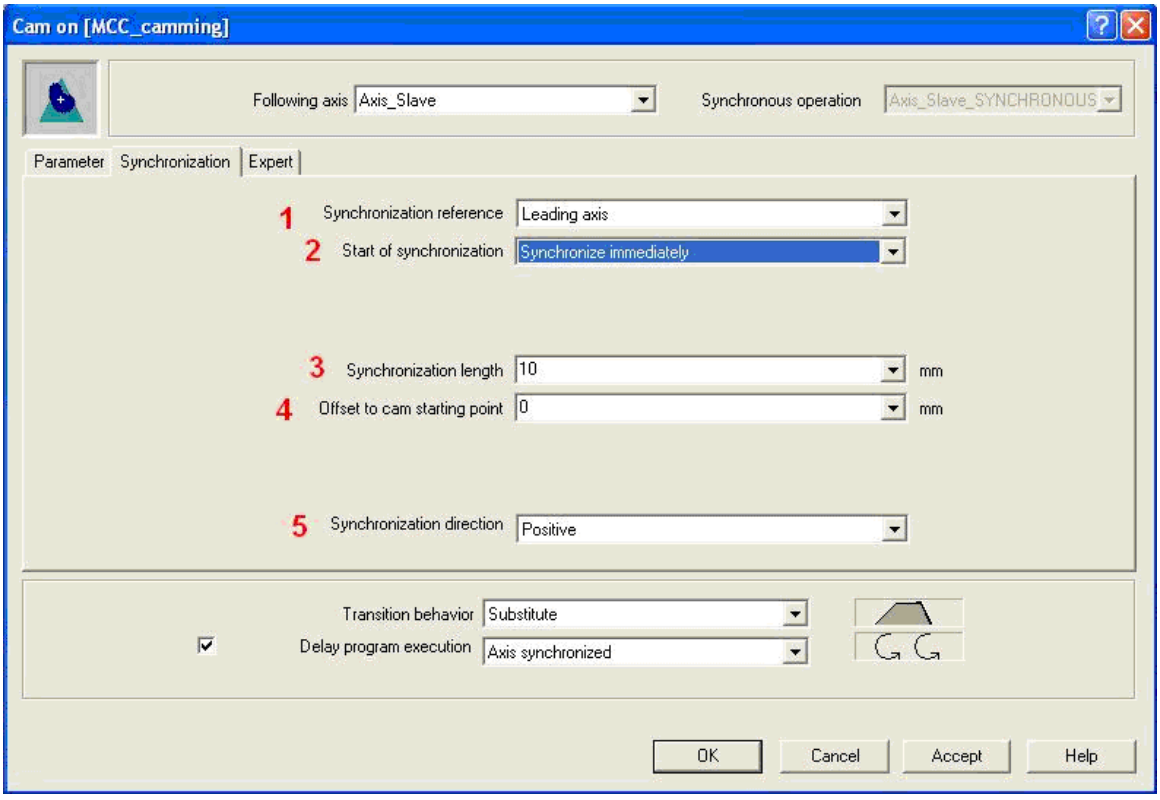


图 5.62 凸轮同步命令参数设置 2

10 编译并保存。

11 将程序分配给执行系统。

12 在线并下载程序。

13 Trace 设置

将以下变量添加到 Trace 列表中：

1) _to.Axis_Master.positioningstate.actualposition

2) _to.Axis_Slave.positioningstate.actualposition

3) _to.Axis_Master.motionstatedata.actualvelocity

4) _to.Axis_Slave.motionstatedata.actualvelocity

5) _to.Axis_Slave_SYNCHRONOUS_OPERATION.state

6) _to.Axis_Slave_SYNCHRONOUS_OPERATION.syncstate

14 运行程序。

确认 CPU 处于 RUN 状态，置位 g_boStart，轴 Axis_Slave 和 Axis_Master 使能，通过 Axis_Slave.control 和 Axis_Master.control 系统变量可以查看轴有没有被使能，使能后轴 Axis_Slave 和 Axis_Master 进行回零，同样通过查看

Axis_Slave 和 Axis_Master 的系统变量 PositioningState.Homed 可以知道回零是否成功。回零成功后，两轴都位于原点位置，点击开始 trace， 置位 g_boStartMove，轴 Axis_Master 以 20 mm/s 的速度运行，然后置位 g_boStartCamming，当主轴位置为 0mm 时，凸轮同步开始，当主轴位置为 10mm 时，从轴与主轴之间凸轮同步。

Trace 结果如图 5.63 所示：

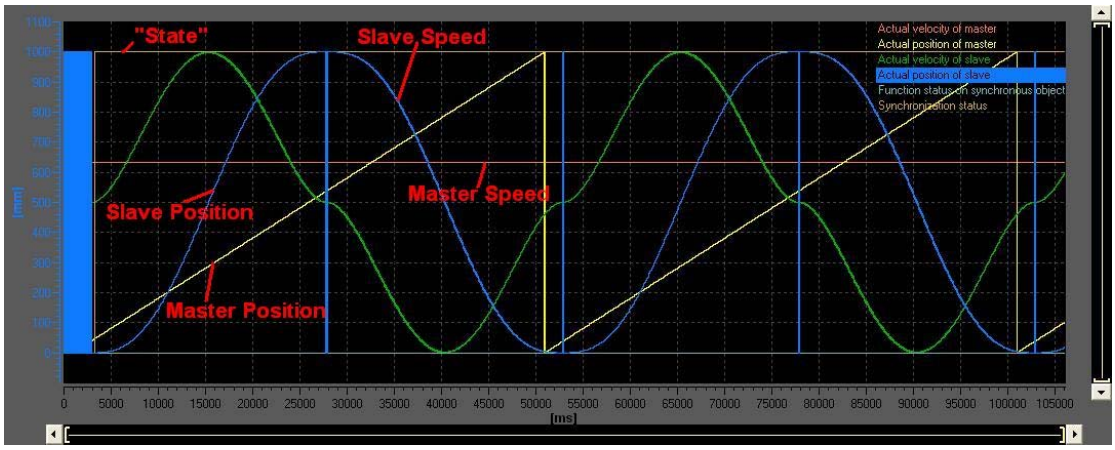
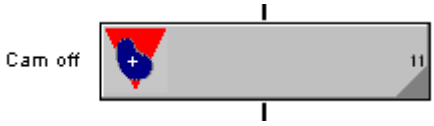


图 5.63 凸轮同步 Trace 图

从图中可以看出，从轴位置按照凸轮曲线周期地增加。

5.2.5 解除凸轮同步命令（Cam Off）



该命令用于解除凸轮同步。

5.2.5.1 命令参数说明

参数	说明
Following axis	指定要进行解除凸轮同步的轴，可以为: • 所有同步轴。轴定义在项目导航栏的 AXES 文件夹下。 • <Reference> ：如果要进行同步操作的轴没有定义在设备中而是指定为参考(变量)，则选择<Reference>。MCC Unit 或 MCC chart 中所有已声明的 followingObjectType 类型的变量都可以在 Synchronous operation 下拉编辑框中找到。

Synchronous operation

根据所选的 Following axis，所有可用的同步对象在该下拉列表中显示。

同步参考

✓ Leading axis

长度相关的同步：同步在指定的主值区间（解除同步长度）内完成。

- 优点：同步在可定义的主值区间（解除同步长度）内完成。
- 缺点：同步操作的动态响应由主值（速度）决定。从轴的动态响应限制不会被考虑。

Synchronization reference

✓ Time

时间相关的同步：同步根据指定的动态响应性完成。动态响应在 Dynamics 页面中进行设置。

- 优点：同步操作总是根据指定的动态响应参数进行。
- 缺点：不能预知完成同步所需要的主值设定值区间。

✓ Last programmed setting

✓ Default value

缺省值。缺省值为系统变量: userdefault.syncProfile.syncProfileReference。

Desynchronization position

解除凸轮同步的开始时间。

✓ at leading axis position

当主轴到达指定位置时，凸轮同步解除。

需要在以下选项中进行设置：

- Reference point of desynchronization axis
- Leading axis position

✓ Desynchronize immediately

立即解除凸轮同步。

✓ At following axis value

当从轴到达指定位置时，凸轮同步解除。

需要在以下选项中进行设置：

- Reference point of desynchronization axis
- Leading axis position

✓ At end of cam cycle

当从轴到达凸轮终点时解除同步。需要在 Reference point of desynchronization position 选项中进行设置。

✓ Last programmed start of desynchronization

✓ Default value

缺省值。缺省值为系统变量: userdefault.cammingSettings.syncOffMode

Reference point of desynchronization position

如果在 Desynchronization position 选项框中选择了如下选项，则需要设置该参数。

- At leading axis value
- At following axis value
- At end of cam cycle

✓ Stop before desynchronization position

	同步在设定的位置完成。 ✓ Symmetrical 当位于设定的位置，主轴只走了同步长度的一半。 ✓ Stop from desynchronization position 同步在设定的位置开始。 ✓ Last programmed reference point of leading axis position ✓ Preassigned value 缺省值。缺省值为系统变量： userdefault.syncProfile.syncOffPositionReference 如果在 Synchronization reference 选项框中选择了 Leading axis，则需要设置该参数。
Desynchronization length	在该编辑框中输入解同步长度。 ✓ Last programmed desynchronization length ✓ Preassigned value 缺省值。缺省值为系统变量: userdefault.syncProfile.syncOffLength 如果在 Desynchronization position 选项框中选择了 at following axis value，则需要设置该参数。
Following axis position	在该编辑框中输入从轴位置。 ✓ Default 缺省值。缺省值为系统变量: userdefault.syncOffPositions.Slave 如果在 Desynchronization position 选项框中选择了 at leading axis value，则需要设置该参数。
Leading axis position	在该编辑框中输入从轴位置。 ✓ Default 缺省值。缺省值为系统变量:userdefault.syncOffPositions.Master
Synchronization direction	指定从轴同步的运动方向。 ✓ Retain system behavior 同步按照最短路径进行。此种情况下，当移动轴时，会进行检查以决定是否维持当前的运动方向。 ✓ Maintain direction of following axis 同步按照从轴运动方向进行。 ✓ Positive 同步按照正向进行。 ✓ Negative 同步按照反向进行。 ✓ Shortest path 同步不考虑方向而是根据最短路径进行。 ✓ Preassigned value (缺省值) 缺省值。缺省值为系统变量： userdefault.cammingSettings.synchronizingDirection

5.2.5.2 范例

例 18:

参数: Synchronization referenc: Leading axis

Desynchronization position: Desynchronize immediately

Desynchronization length: 20

Synchronization direction: Positive

1. 添加解除凸轮同步命令

在上面凸轮同步例子的 MCC chart 中继续添加 wait for condition 命令（g_boStopCamming 的上升沿作为触发条件）。

在 MCC 工具条的同步操作命令组中单击解除凸轮同步命令，将其插入到 MCC chart 中。

2. 解除凸轮同步命令参数设置

双击插入的 Cam off 命令，在弹出的对话框中进行参数设置，点击 OK 确认。

Paremeters 页，见图 5.64 所示：

1) Following axis: Axis_Slave。

2) Synchronization referenc: Leading axis

3) Desynchronization position: Desynchronize immediately

4) Desynchronization length: 20

5) Synchronization direction: Positive

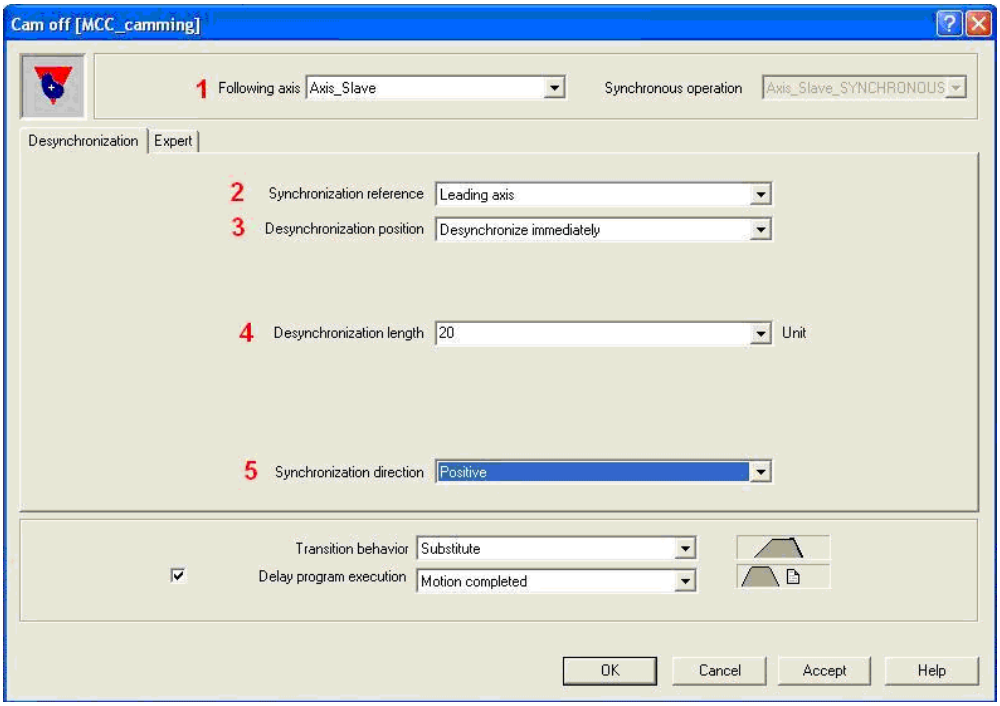


图 5.64 解除凸轮同步命令参数设置

其余操作均同例 17。

当从轴与主轴凸轮同步时，置位 g_boStopCamming，从轴开始解除与主轴的凸轮同步，结果如下图所示。

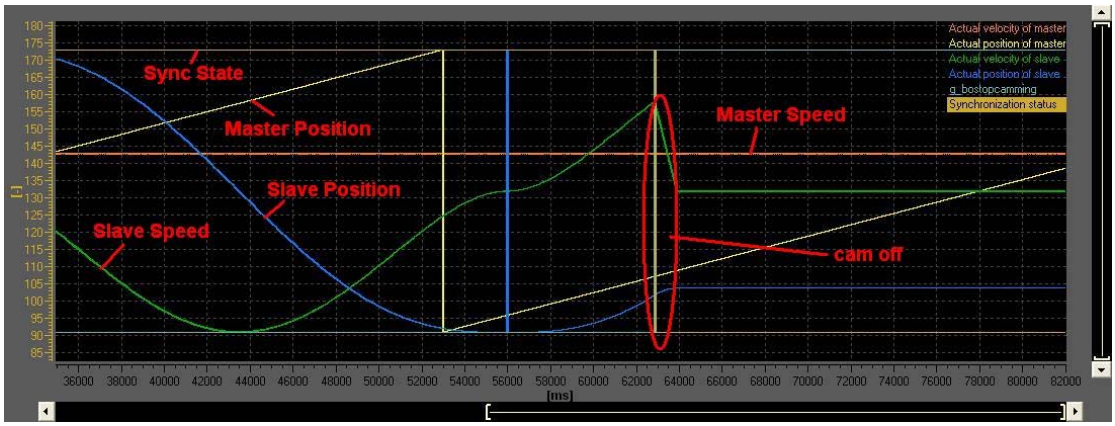


图 5.65 解除凸轮同步 Trace 图

5.3 应用实例

5.3.1 机械结构

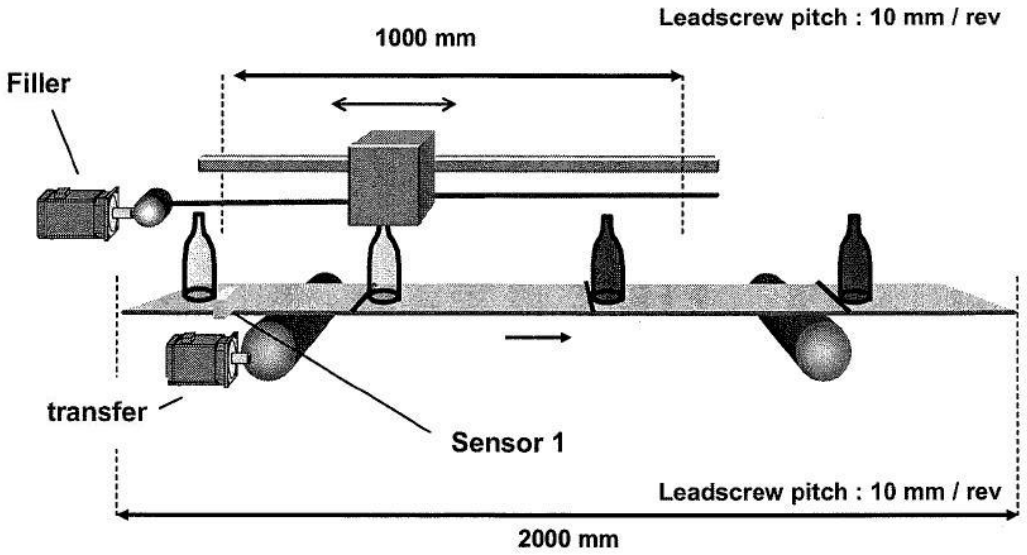


图 5.66 灌装生产线

5.3.2 工艺描述

Transfer（传送带）沿正向一直运行，当 Sensor1 检测到有瓶子，Filler 开始齿轮同步，当 Transfer 向前运动 100mm 后，Filler 与 Transfer 实现速度同步，同时 output cam 输出信号（开始灌装），当主轴位置为 800mm 时，output cam 关闭输出信号（停止灌装），同时 Filler 开始解除与 Transfer 的速度同步，当 Transfer 向前运动 100mm 后，Filler 完成同步解除，然后快速回到起点，等待下一次运动。38893094

5.3.3 系统拓扑结构

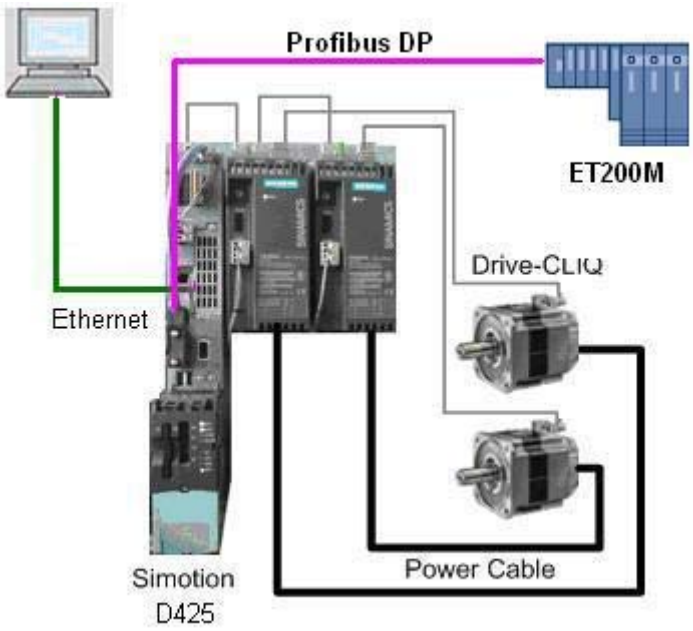


图 5.67 系统拓扑结构图

硬件：Simotion Standard D425 Demo

软件：Simotion Firmware version: 4.1.4

Sinamics Firmware version: 2.6.2

Scout software version: 4.1.4.1

5.3.4 系统组态

1. 参考第二章，创建一个名为 “SimotionD425_Example”的新项目，并插入设备 SIMOTION D425，设置通讯接口为 TCP/IP。

2. 参考图 5.68 在硬件组态中配置 ET200M 模块

（1）配置 DP1 通讯接口，这里设置 DP 地址为 2。接口的具体参数如波特率、最高站地址等可以通过点击“Properties..”按钮进行设置。缺省方式下，波特率为 1.5Mbps。

（2）根据实际硬件信息选择正确的 ET200M 模块，将其拖入 PROFIBUS DP 总线上。

（3）修改 ET200M 模块的 DP 地址。ET200M 模块上有 DP 地址拨码开关，硬件组态中的设置需要与拨码开关的设置匹配，这里设置为 8。

（4）根据实际硬件信息为 ET200M 添加数字两输入输出模块，将其拖入屏幕下方的列表中。

（5）修改 DI/DO 的起始地址。这里 DI/DO 的起始地址都设置为 80。

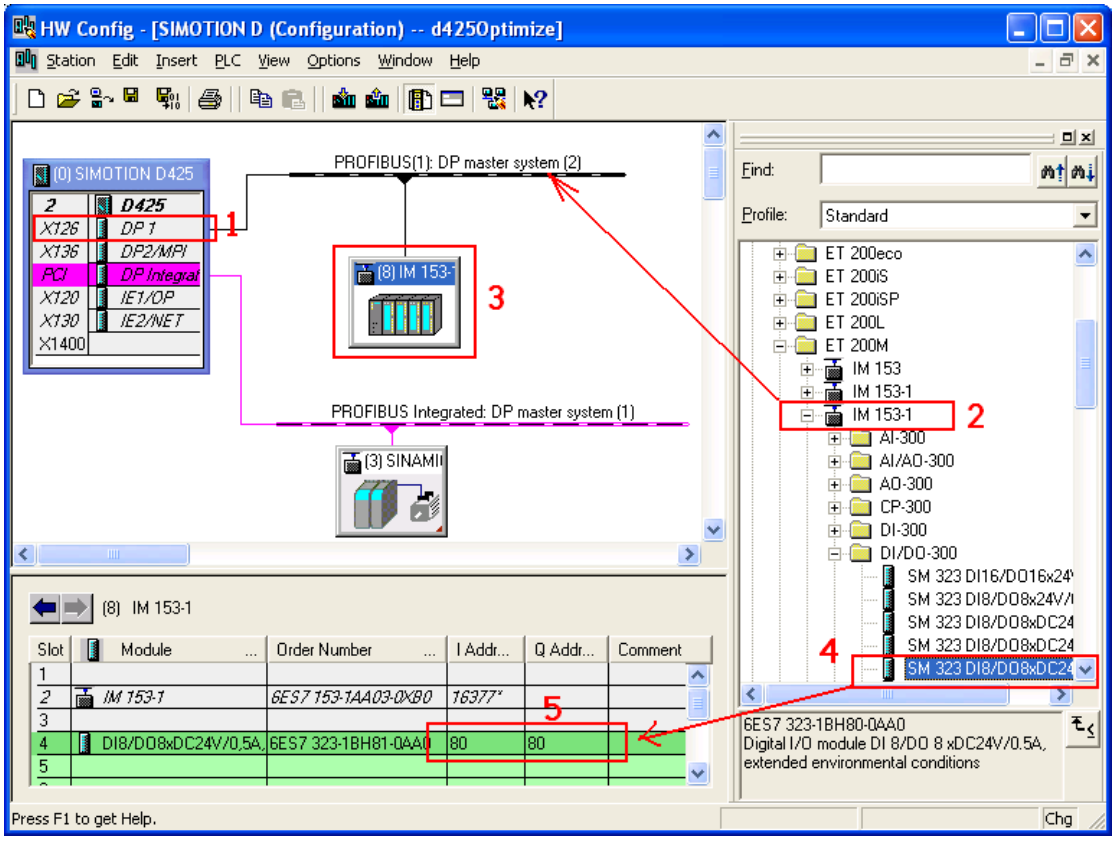


图 5.68 配置 ET200M

3. 其他组态与第二章中相同，下载硬件组态后在线自动配置驱动，并为驱动配置 105 报文。

4. 创建名为 Axis_Transfer 的线性定位轴，并钩选模态轴选项，模态值为 0~2000mm。如图 5.69 所示。

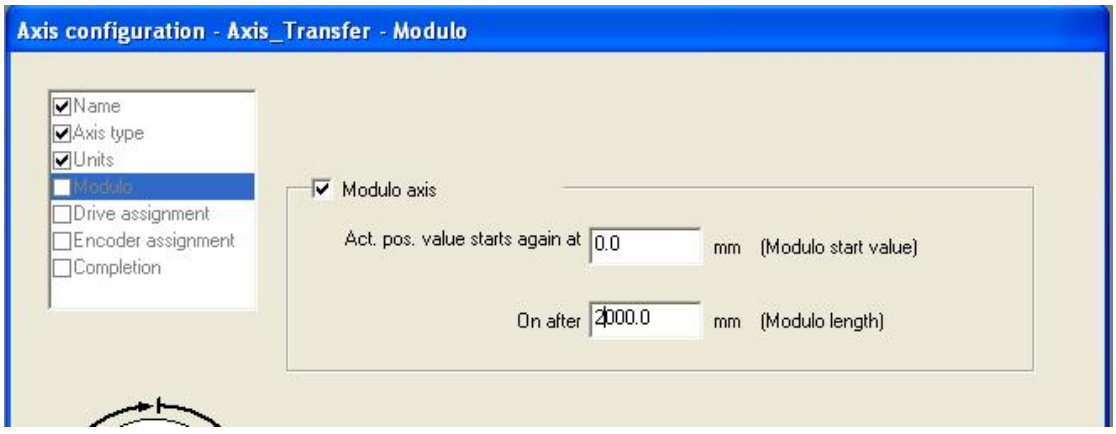


图 5.69 配置 Axis_Transfer 为模态轴

5. 创建名为 Axis_Filler 的线性同步轴。

6. 创建 Output Cam。

（1） 双击位于 Axis_Filler 的 OUTPUT CAM 文件夹下的 Insert output cam，在弹出的对话框中进行参数设置，点击 OK 按钮确认。

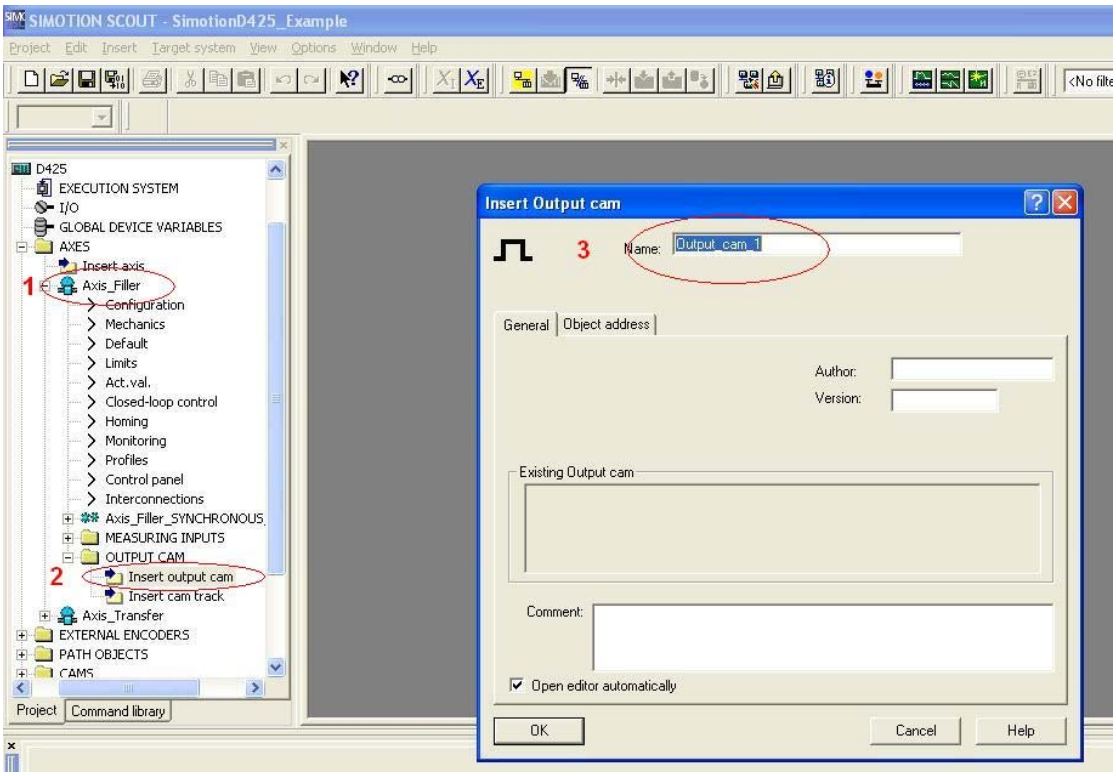


图 5.70 插入 output cam

（2）Output Cam 参数设置

双击刚刚创建的 output_cam_1 下的 configuration，在右边页面进行设置，参考图 5.71 设置。其中 HW address 对应 ET200M 的输出端子。

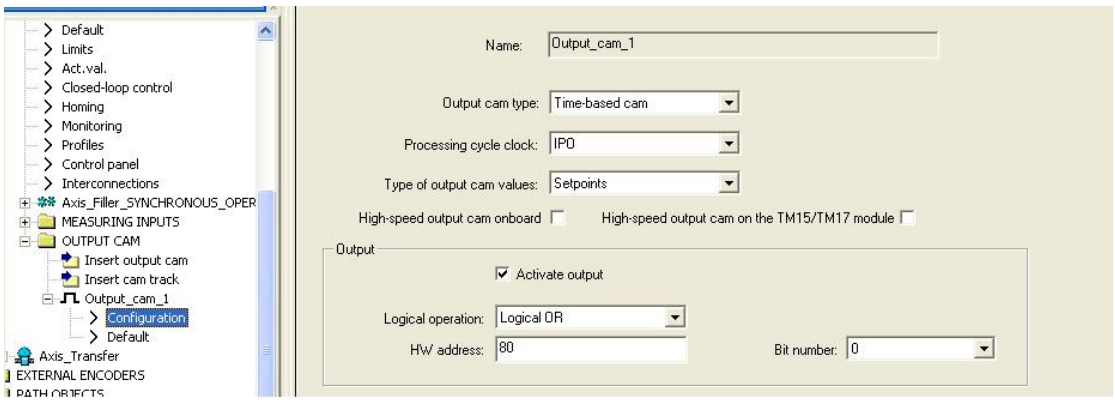
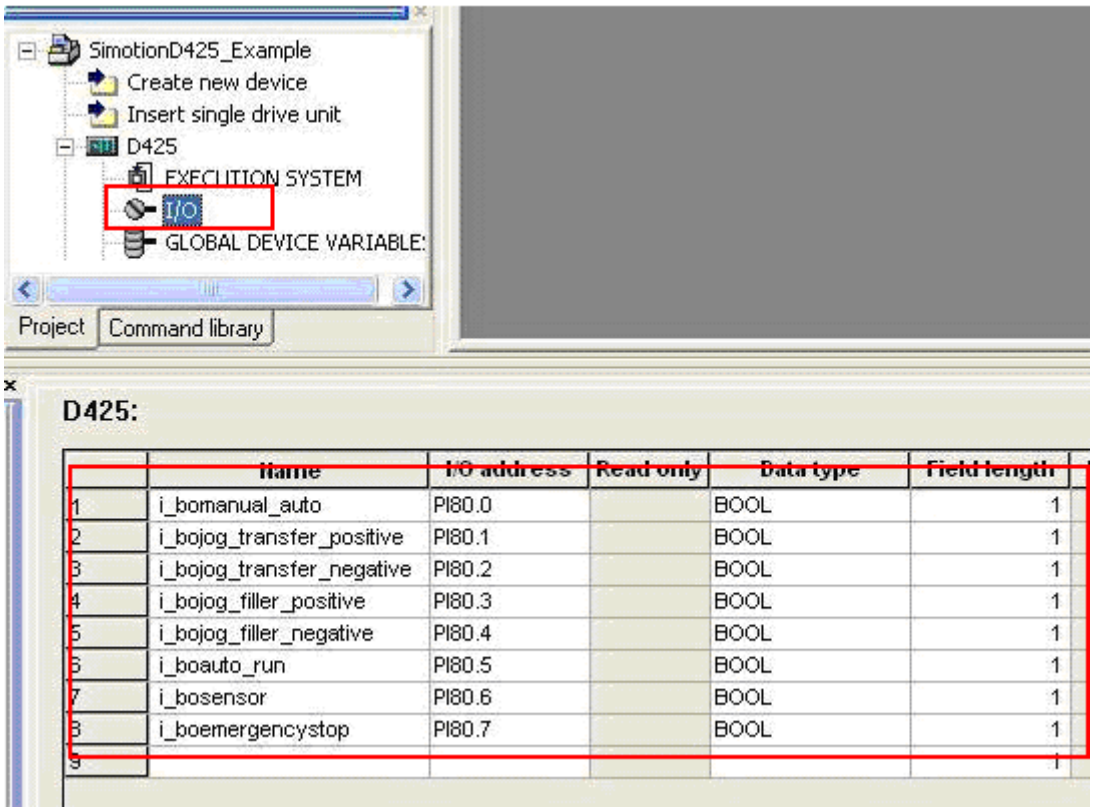


图 5.71 output cam 参数设置

5.3.5 程序实现

5.3.5.1 创建 IO 变量表

点击 D425 下面的 IO，参考图 5.72 创建 Simotion IO 变量表。



	Name	I/O address	Read only	Data type	Field length
1	i_bomanual_auto	PI80.0		BOOL	1
2	i_bojog_transfer_positive	PI80.1		BOOL	1
3	i_bojog_transfer_negative	PI80.2		BOOL	1
4	i_bojog_filler_positive	PI80.3		BOOL	1
5	i_bojog_filler_negative	PI80.4		BOOL	1
6	i_boauto_run	PI80.5		BOOL	1
7	i_bosensor	PI80.6		BOOL	1
8	i_boemergencystop	PI80.7		BOOL	1
9					1

图 5.72 Simotion IO 变量表

- i_bomanual_auto(PI80.0): 模式选择开关。TRUE: 自动模式。FALSE: 手动模式。
- i_bojog_transfer_positive(PI80.1): 轴 Axis_Transfer 正向点动按钮。
- i_bojog_transfer_negative(PI80.2): 轴 Axis_Transfer 反向点动按钮
- i_bojog_filler_positive(PI80.3): 轴 Axis_Filler 正向点动按钮。
- i_bojog_filler_negative(PI80.4): 轴 Axis_Filler 反向点动按钮。
- i_boauto_run(PI80.5):自动运行按钮。上升沿开始自动运行。下降沿停止自动运行。
- i_bosensor(PI80.6):检测瓶子的传感器。TRUE: 检测到瓶子。FALSE: 没有检测到瓶子。

- i_boemergencystop(PI80.7): 急停按钮。TRUE: 急停按钮没有被按下。FALSE: 急停按钮被按下。

5.3.5.2 手动模式

1.轴 Axis_Transfer 的点动程序

（1） 创建 MCC Unit，将其命名为 MCC_Unit_1.

（2）创建 MCC chart，将其命名为 MT_Jog_Transfer。

（3）在 MT_Jog_Transfer 中创建如下程序。

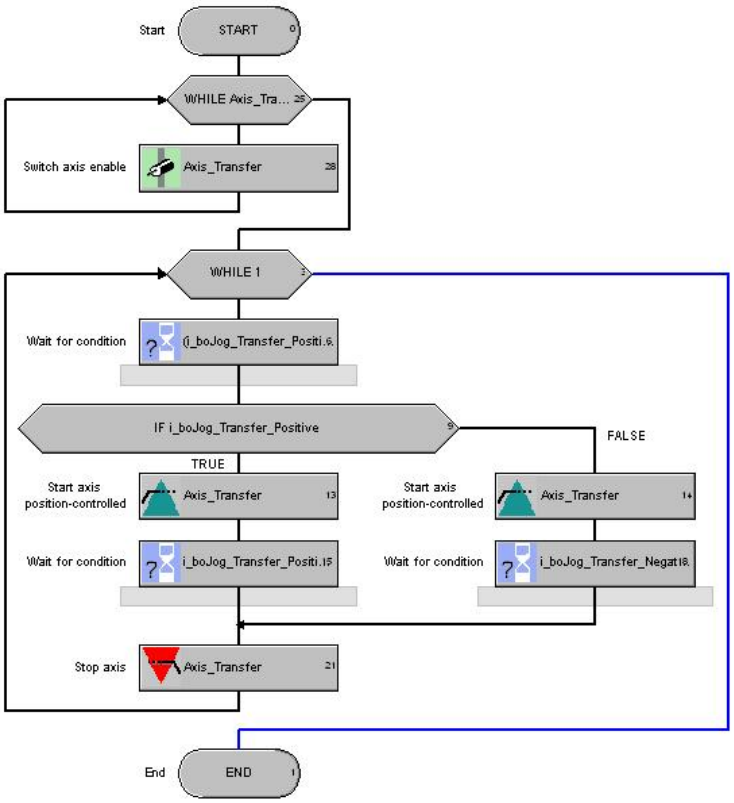


图 5.73 Axis_Transfer 的点动程序

当检测到 i_bojog_transfer_positive（PI80.1）的上升沿时，轴 Transfer 会正向运动（速度为 10mm/s），当检测到 i_bojog_transfer_positive（PI80.1）的下降沿时，轴 Transfer 会停止。

当检测到 i_bojog_transfer_negative（PI80.2）的上升沿时，轴 Transfer 会反向运动（速度为 10mm/s），当检测到 i_bojog_transfer_negative（PI80.2）的下降沿时，轴 Transfer 会停止。

2. 轴 Filler 的点动程序

（1）在 MCC_Unit_1 中创建 MCC chart，将其命名为 MT_Jog_Filler。

（2）参考轴 Transfer 的点动程序可以很容易的编写出轴 Filler 的点动程序，如图 5.74 所示。

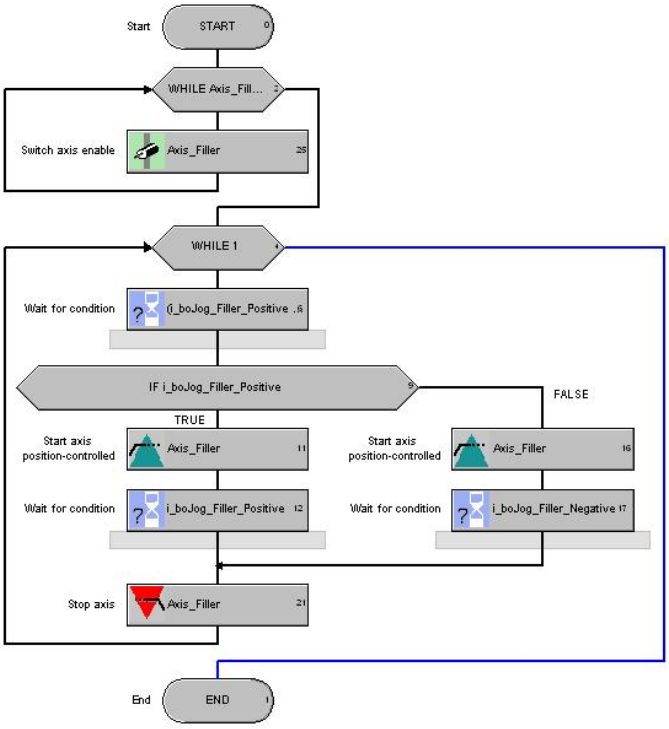


图 5.74 Axis_Filler 的点动程序

当检测到 i_bojog_Filler_positive（PI80.3）的上升沿时，轴 Filler 会正向运动（速度为 10mm/s），当检测到 i_bojog_filler_positive（PI80.3）的下降沿时，轴 Filler 会停止。

当检测到 i_bojog_filler _negative（PI80.4）的上升沿时，轴 Filler 会反向运动（速度为 10mm/s），当检测到 i_bojog_filler_negative（PI80.4）的下降沿时，轴 Filler 会停止。

3. 回零

（1）在 MCC_Unit_1 中创建 MCC chart，将其命名为 MT_Homing。

（2）MT_Homing 中创建如下程序。

为了简化程序，两轴的回零方式都选择了 Set home position，零点坐标为 0。

第一个条件判断：如果轴 Axis_Transfer 没有回零，则“set home position”Axis_Transfer 到 0 位置。

第二个条件判断：如果轴 Axis_Filler 没有回零，则“set home position”Axis_Filler 到 0 位置。

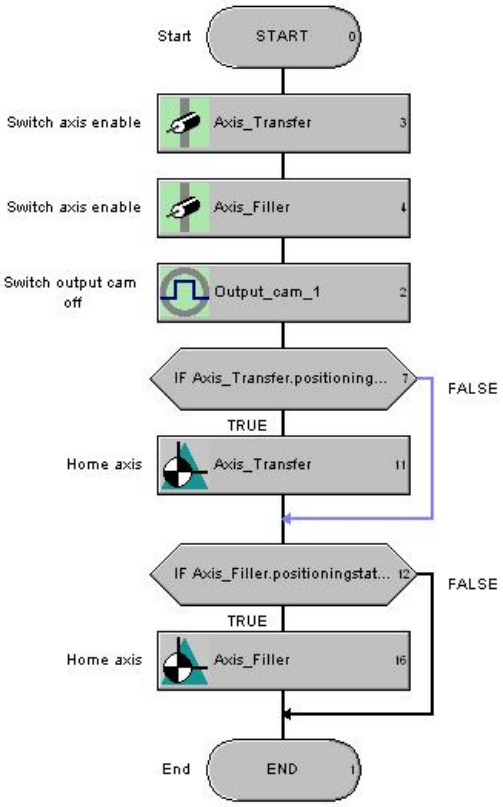


图 5.75 回零程序

5.3.5.3 自动模式

1. 自动运行程序

（1）在 MCC_Unit_1 中创建 MCC chart，将其命名为 MT_Auto_Run。

（2）在 MT_Auto_Run 中编写如下程序。

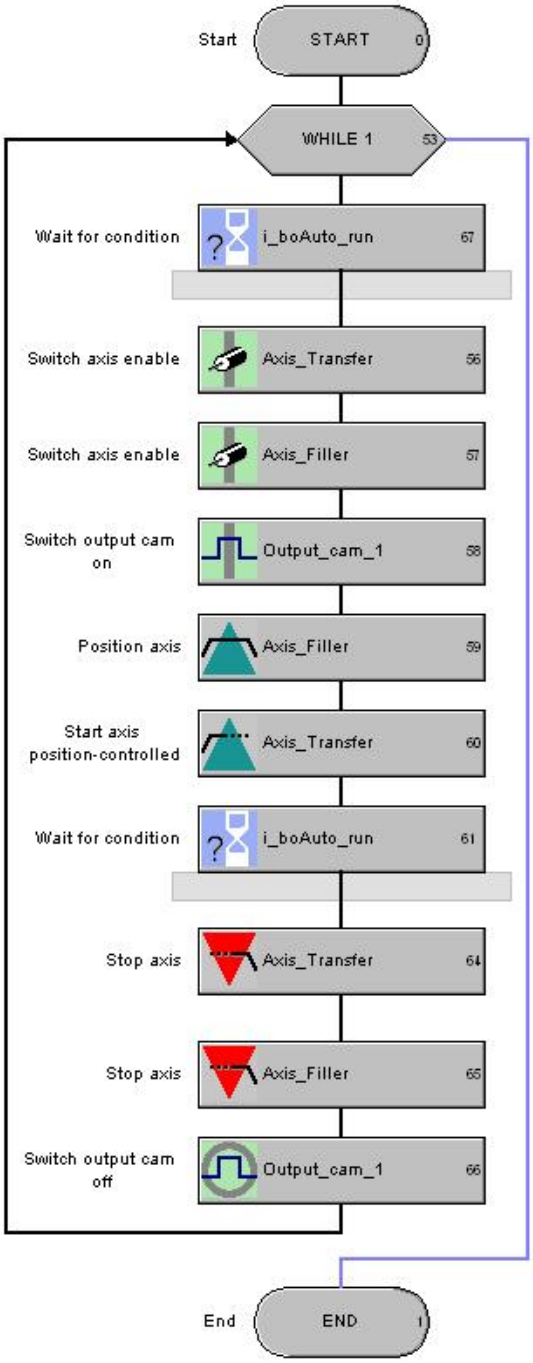

Start
START
0
WHILE 1
53
Wait for condition
i_boAuto_run
67
Switch axis enable
Axis_Transfer
56
Switch axis enable
Axis_Filler
57
Switch output cam on
Output_cam_1
58
Position axis
Axis_Filler
59
Start axis position-controlled
Axis_Transfer
60
Wait for condition
i_boAuto_run
61
Stop axis
Axis_Transfer
64
Stop axis
Axis_Filler
65
Switch output cam off
Output_cam_1
66
End
END
1

图 5.76 自动运行程序

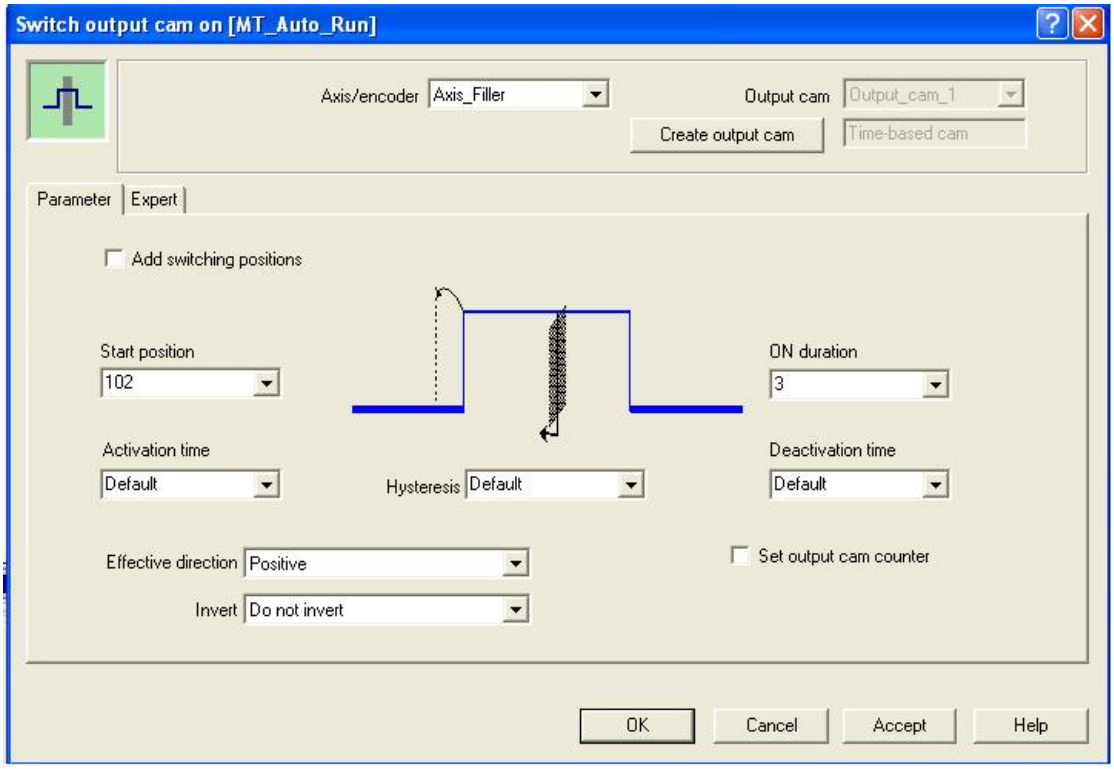


图 5.77 Switch output cam on 命令参数设置

2. 同步程序

（1） 在 MCC_Unit_1 中创建 MCC chart，将其命名为 MT_Filler_Synchronization。

（2）在 MT_Filler_Synchronization 创建如下程序。

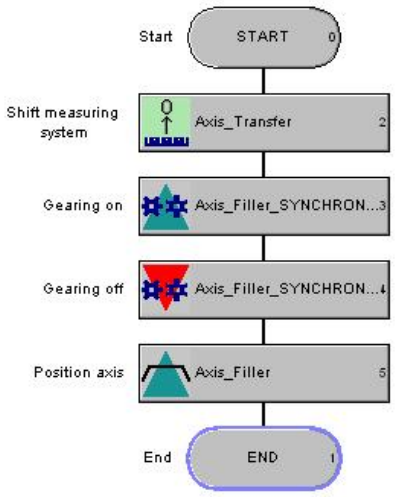


图 5.78 同步程序

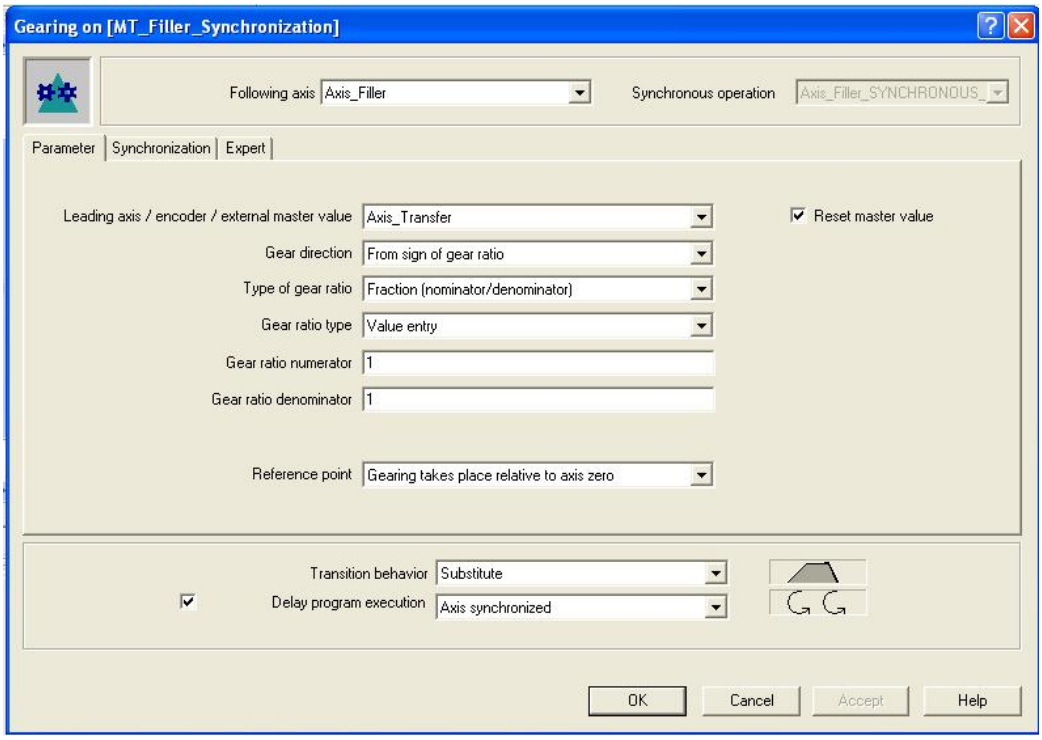


图 5.79 使能齿轮同步参数设置 1

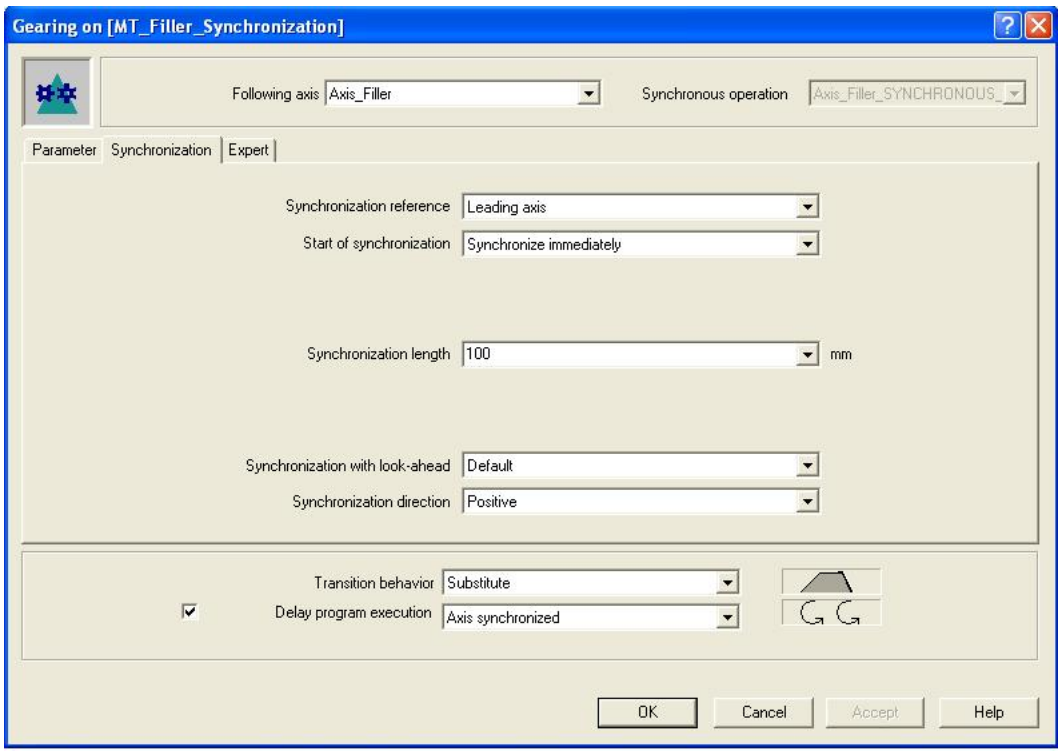


图 5.80 使能齿轮同步参数设置 2

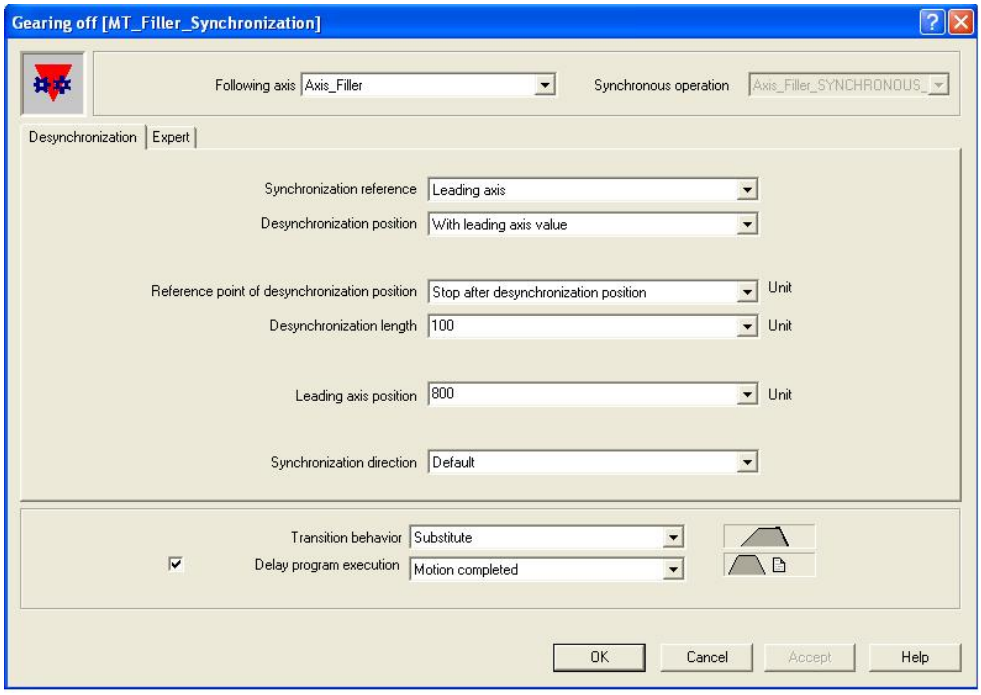


图 5.81 解除齿轮同步参数设置

5.3.5.4 急停程序

（1）在 MCC_Unit_1 中创建 MCC chart，将其命名为 MT_EmergencyStop。

（2）在 MT_EmergencyStop 中创建以下程序。

第一个条件判断是否按下急停按钮。

第二个条件判断轴 Axis_Filter 的急停命令是否激活。如果没被激活，则以最大加速度停止轴 Axis_Filter。

第三个条件判断轴 Axis_Transfer 的急停命令是否激活。如果没被激活，则以最大加速度停止轴 Axis_Transfer。

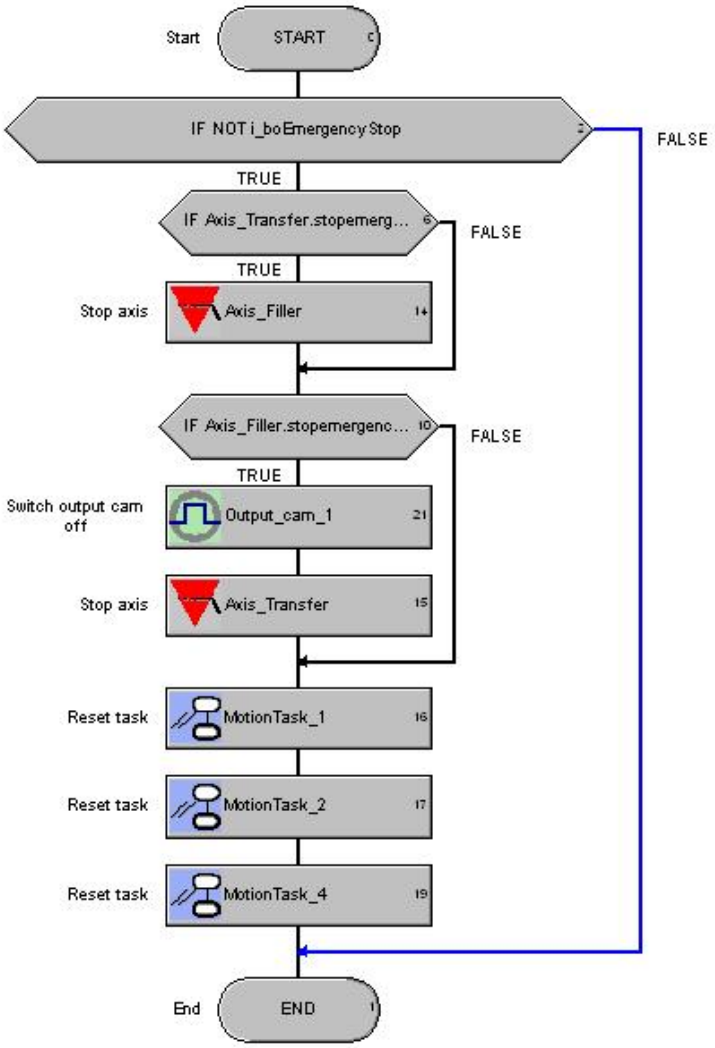


图 5.82 急停程序

5.3.5.5 错误处理程序

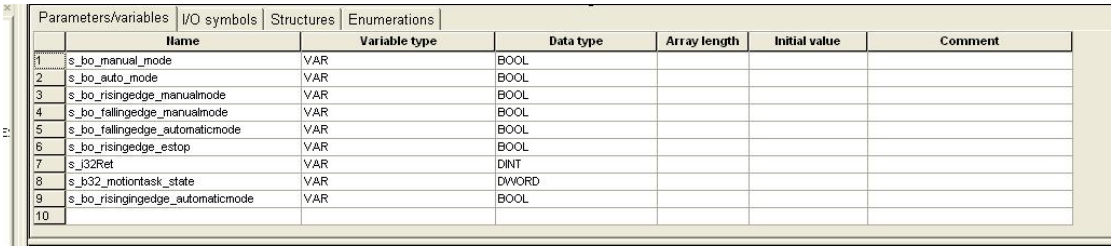
（1）在 MCC_Unit_1 中创建 MCC chart，将其命名为 MCC_Fault。

（2）为了简化程序，MT_Fault 中无任何程序。

5.3.5.6 主程序

（1）创建 LAD/FBD Unit，LAD/FBD program，将其分别命名为 Unit_Main 和 BG_Main。创建方法同 MCC Unit 和 MCC chart 类似。

（2）创建局部变量



Parameters/variables | I/O symbols | Structures | Enumerations

	Name	Variable type	Data type	Array length	Initial value	Comment
1	s_bo_manual_mode	VAR	BOOL			
2	s_bo_auto_mode	VAR	BOOL			
3	s_bo_risingedge_manualmode	VAR	BOOL			
4	s_bo_fallingedge_manualmode	VAR	BOOL			
5	s_bo_fallingedge_automaticmode	VAR	BOOL			
6	s_bo_risingedge_estop	VAR	BOOL			
7	s_i32Ret	VAR	DINT			
8	s_b32_motiontask_state	VAR	DWORD			
9	s_bo_risingingedge_automaticmode	VAR	BOOL			
10						

图 5.83 局部变量列表

- s_boManual_mode: 手动模式标志
- s_boAuto_mode: 自动模式标志
- s_bo_risingedge_manualmode: 手动模式标志的上升沿
- s_bo_fallingedge_manualmode: 手动模式标志的下降沿
- s_bo_risingedge_automaticmode: 自动模式标志的上升沿
- s_bo_fallingedge_automaticmode: 自动模式标志的下降沿
- s_bo_risingedge_estop: 急停按钮的上升沿
- s_i32Ret: 轴命令_Stop 的返回值。
- s_b32_motiontask_state: 功能块_restartTaskId 的返回值。

（3）在 BG_Main 中创建如下程序。

Network1: 如果 i_bomanual_auto 为 FALSE，则置位 s_boManual_mode(手动模式)，复位 s_boAuto_mode(自动模式)。

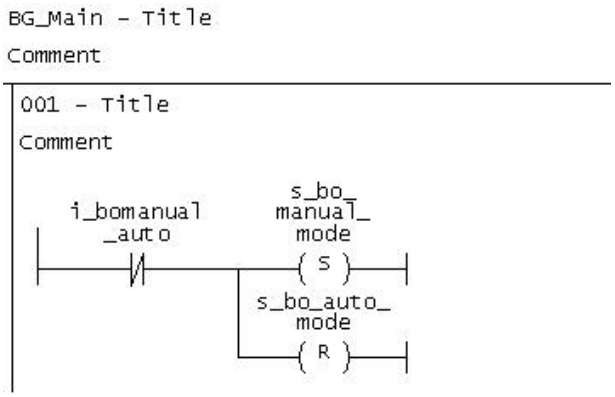


图 5.84 BG_Main 程序的 Network1

Network2: 如果 i_bomanual_auto 为 TRUE，则置位 s_boAuto_mode(自动模式)，复位 s_boManual_mode(手动模式)。

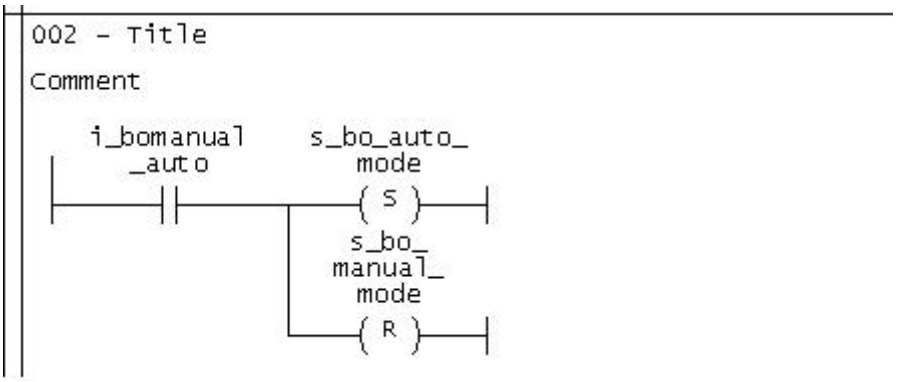


图 5.85 BG_Main 程序的 Network2

Network3: 如果检测到 s_boManual_mode 的上升沿，则启动 MotionTask_1(轴 Axis_Transfer 的点动程序),MotionTask_2（轴 Axis_Filler 的点动程序）。

_restartTaskID()命令位于 Command Library 的 Task system->_restartTaskID。

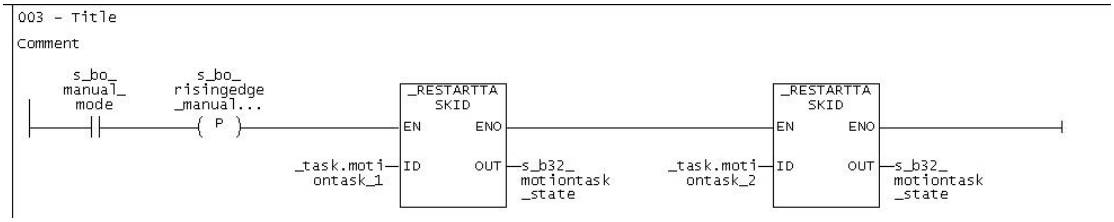


图 5.86 BG_Main 程序的 Network3

Network4: 如果检测到 s_boManual_mode 的下降沿，则复位 MotionTask_1(轴 Axis_Transfer 的点动程序)，停止轴 Axis_Transfer,MotionTask_2（轴 Axis_Filler 的点动程序），停止轴 Axis_Filler。

_resetTaskID ()命令位于 Command Library 的 Task system->_resetTaskID。

_stop()命令位于 Command Library 的 Technology->Positioning Axis->Motion ->_stop()。

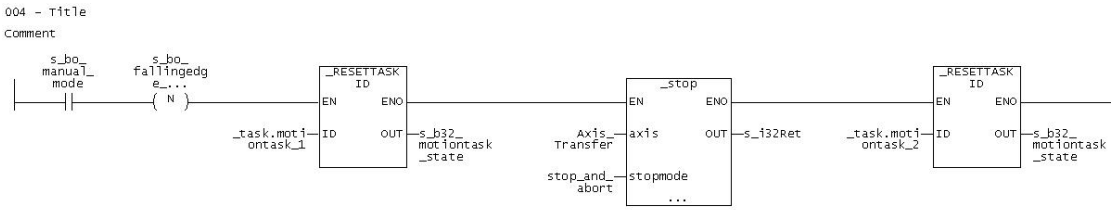


图 5.87 BG_Main 程序的 Network4

Network5: 如果检测到 s_boAuto_mode 的上升沿，则启动 MotionTask_4(自动运行程序)。

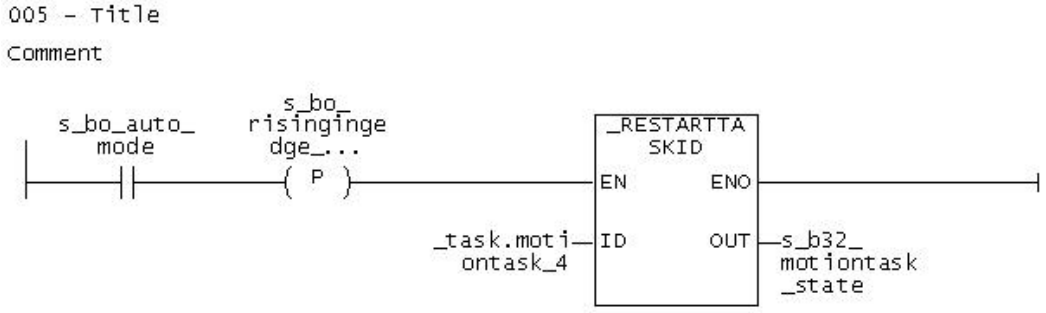


图 5.88 BG_Main 程序的 Network5

Network6: 如果检测到 s_boAuto_mode 的下降沿，则复位 MotionTask_4(自动运行程序)，停止轴 Axis_Transfer，Axis_Filler。

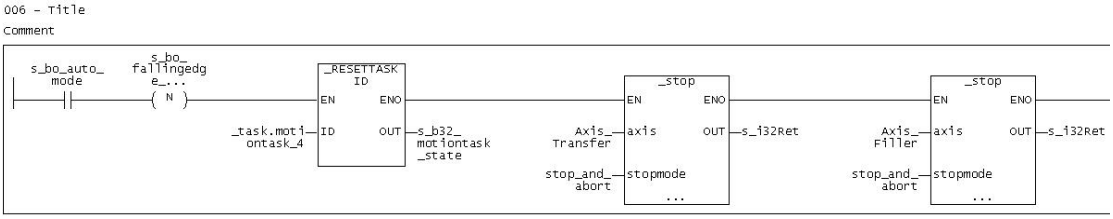


图 5.89 BG_Main 程序的 Network6

Network7: 如果检测到 i_boEmergencyStop 的上升沿，则使用 resetAxis 命令复位急停命令。

_resetAxis()命令位于 Command Library 的 Technology->Positioning Axis->Object and alarm handling->_resetAxis。

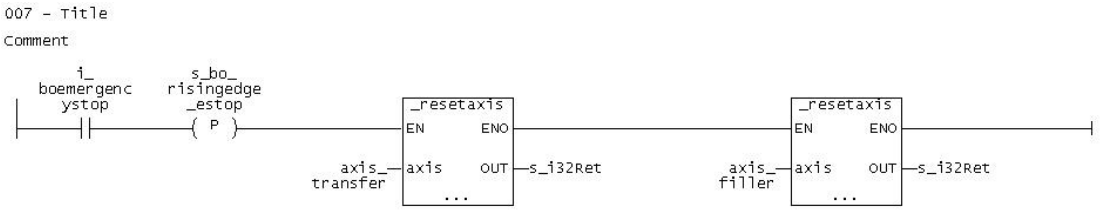


图 5.90 BG_Main 程序的 Network7

5.3.5.7 执行系统分配

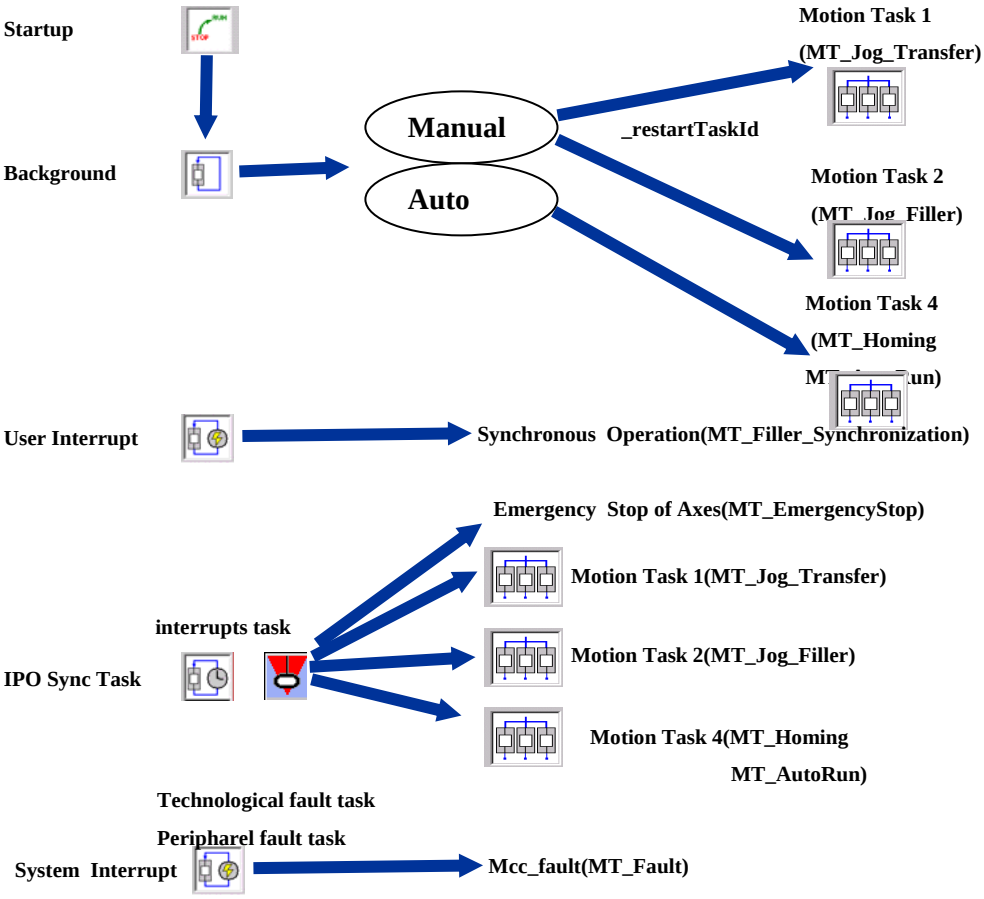


图 5.91 执行系统分配 1

按照图 5.92 进行执行系统分配。

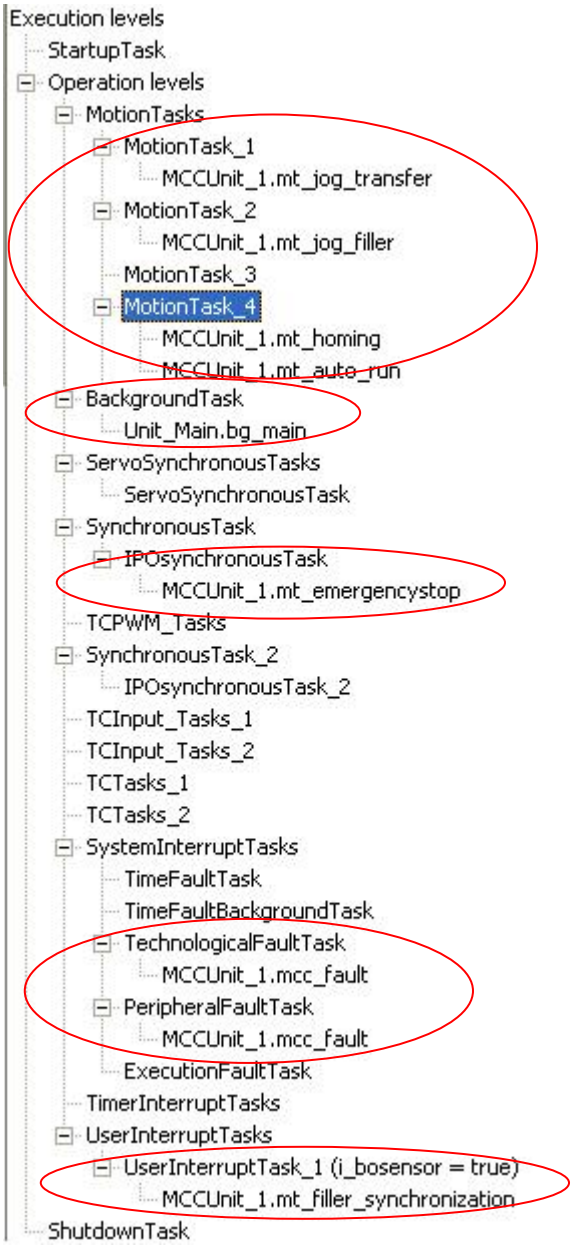


图 5.92 执行系统分配 2

5.3.6 测试

在线并下载程序，进行测试。

闭合 i_boemergencystop(PI80.7)，如果 i_bomanual_auto(PI80.0)为 FALSE，则程序进入手动模式，闭合 i_bojog_transfer_positive(PI80.1)或者 i_bojog_transfer_negative(PI80.2)，Axis_Transfer 将会正向点动或者反向点动（速度为 10mm/s），闭合 i_bojog_filler_positive(PI80.3) 或者 i_bojog_filler_negative(PI80.4)，Axis_Filler 将会正向点动或者反向点动（速度为 10mm/s）。

如果 i_bomanual_auto(PI80.0)为 TRUE，则程序进入自动运行模式，闭合 i_boauto_run(PI80.5)，轴 Axis_Transfer 将会以设定的速度沿正向运行，如何闭合 i_bosensor(PI80.6)，轴 Axis_Filler 则会立即与轴 Axis_Transfer 齿轮同步，同步一段距离后开始脱离同步，当同步解除后快速回到起点位置，等待下一次运行。

如果 i_boemergencystop(PI80.7)断开则进入急停模式。

参考文献及联系人

[1] SIMOTION D 系统组态及调试入门 西门子运动控制部

[2] SIMOTION SCOUT Configuration Manual 西门子运动控制部

[3] SIMOTION MCC Programming and Operating Manual 西门子运动控制部

[4] SIMOTION LAD/FBD Programming and Operating Manual 西门子运动控制部

[5] SIMOTION Easy Handbook 西门子运动控制部

如有任何问题，请联系：

西门子（中国）有限公司

工业业务领域，驱动技术集团

运动控制部，应用中心

解决方案小组

地址：上海市中山南二路 1089 号徐汇苑大厦 22 层

电话：021 - 54108666，传真：021 - 64579199

电子信箱：huijuan.li@siemens.com

公司网址：www.siemens.com.cn