运动控制产品

应用说明

AC500/ABB 运动驱动器 - EtherCAT®入门指南

AN00205

Rev I (CN)

使用 AC500 PLC 作为 EtherCAT®主设备来控制
MicroFlex e190 和 MotiFlex e180 伺服驱动器

介绍

可以使用 AC500 PLCs（PM585 和 PM59x）来控制支持 EtherCAT 协议的伺服驱动器 MicroFlex E190 和 MotiFlex E180 用于执行实时的运动控制功能。

本应用说明详细说明了如何使用 Automation Builder 来配置单台 MicroFlex e190 或 MotiFlex e180 的 EtherCAT 运动控制的硬件设置，以及如何编写简单的 AC500 PLC 程序控制这些驱动器执行运动。

前提条件

你需要配备以下项目以完成本应用说明：

- 版本为 5852 或更新的 Mint Workbench（参见 new.abb.com/motion 了解更新的下载和支持信息）
- 固件版本为 5867 或更新的 MicroFlex e190 或 MotiFlex e180 驱动器（推荐至少使用 5863 及以上的版本，因为该版本已经增强了与 CiA Ds402 规范的兼容性 - 如果你的驱动器在开始时运行的固件比 5860 老，在更新到较新版本前确保先更新到 5860）
- 运行 Automation Builder 1.2 或更新版本的 PC 或笔记本电脑
- 已安装（并且已授权）的 ABB PLCopen 运动控制库（PS552-MC-E v3.2.0 或更新）。更早版本的库同样有效，但在 ECAT_CiA402_CONTROL_APP 功能块中有一些可以设法解决的功能差异 - 如有必要，请联系 cn-motionsupport@cn.abb.com （亚洲）或 motionsupport.uk@gb.abb.com（世界其它地区）。
- AC500 PLC 处理器 PM585、PM590、PM591、PM592 或 PM595 中的一种（PLC 处理器应该运行 2.5.1 或更新版本的固件）。PM595 提供集成 EtherCAT 连接器（它应该运行 4.2.32.2 或更新版本的固件）。所有其它处理器需要 CM579-ECAT 通讯模块（必须运行 2.6.9 或更新版本的固件，但最好是 4.3.0.2 或更新的版本）。联系你当地的 ABB PLC 支持团队详细了解如何检查这些要求并在必要时更新，或访问 http://new.abb.com/plc/programmable-logic-controllers-plcs 并选择“软件”链接。在本应用说明的正文中，我们假设使用的是带 CM579-ETHCAT 连接器的 PM591 PLC。
- 用于连接 PLC EtherCAT 连接器到驱动器的以太网电缆

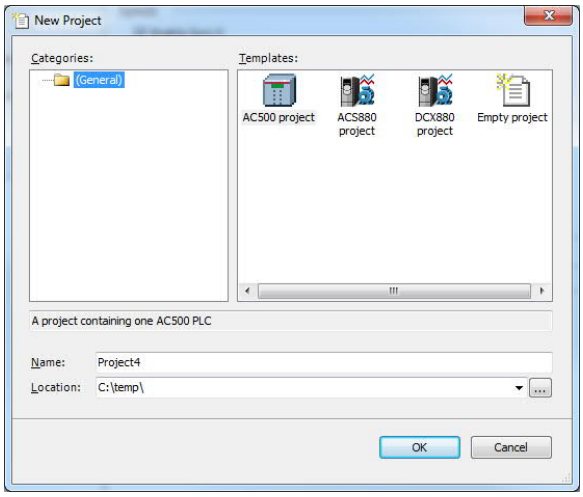
本应用说明假设读者应该已经具备基本的 Automation Builder、CoDeSys 和 AC500 PLC 操作知识，所连接的驱动器已经做过必要的调试/微调，作好通过 EtherCAT 控制的准备。

驱动器设置

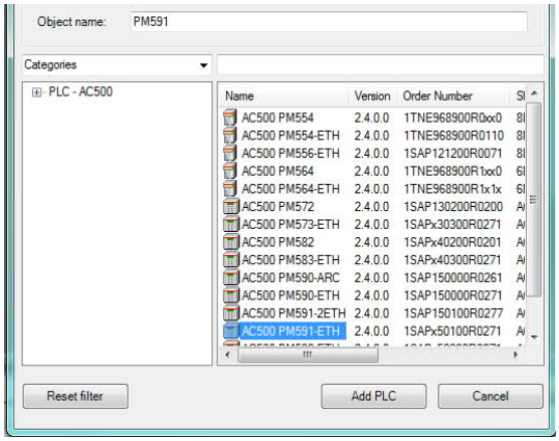
本应用说明假定你已经调试过驱动器。即，你已经通过 Mint Workbench 调试向导定义了电机和应用设置，并完成了控制环的自整定（如有必要，可进行微调）。可在相关的驱动器安装手册中找到关于驱动器调试的详细信息，也可以参考应用说明 AN00250。驱动器的初始操作控制给定值信号源（CONTROLREFSOURCESTARTUP）可设置为 Direct 或 Real time Ethernet。任一设置均可，因为 EtherCAT 在启动时始终会把驱动器切换到 Real time Ethernet。但是，最好是选择 Direct 作为驱动器的默认信号源。因为在这种情况下，当 PLC 切换到“ STOP ”状态时，都能够通过 Workbench 直接控制轴。

Automation Builder – 开始新 PLC 项目

如果还没有打开 Automation Builder，启动它并开始新的项目（File>New Project…）。根据你的 Automation Builder 安装情况，在开始新项目时可从几种模块中选择 - 选择 AC500 项目，为项目命名，并为项目指定一个位置（在下文的例子中，我们把该项目称为“ AN00205 ”，与本应用说明的编号一致）。通常，一个好的做法是创建一个新的文件夹来保存你的项目，因为最后你会有多个与该项目相关联的文件。

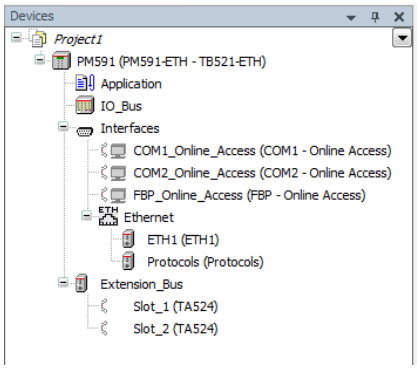


单击“ OK ”。现在，Automation Builder 会要求你选择 PLC。展开“ PLC ”图标选择你正在使用的处理器类型。点击“ Add PLC ”把 PLC 添加到项目中。

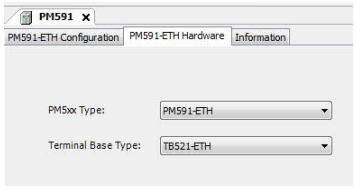


在选中并添加到设备树中后，Automation Builder 将按照选择的处理器类型自动给设备分配一个名称。你可以按照自己的意愿点击树形列表中的设备名称覆盖它，并把它命名为你想要的任何名称（我们把它称为 PM591）。

现在，Automation Builder 将显示 PLC 项目的硬件结构。

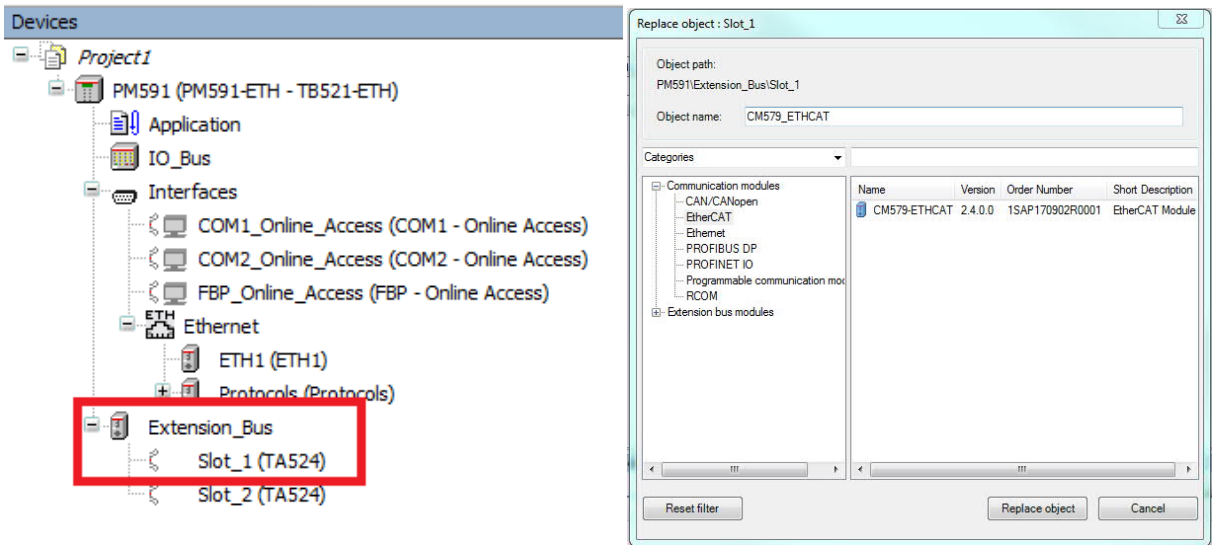


此时，你需要选择正确的端子底座。要完成本操作，双击 PM591-ETH 图标打开 PM591 字段。点击“ PM5xx-ETH Hardware ”选项卡，选择正确的端子底座。如果需要，你也可以在这里修改处理器类型。



Automation Builder – 添加 EtherCAT 硬件

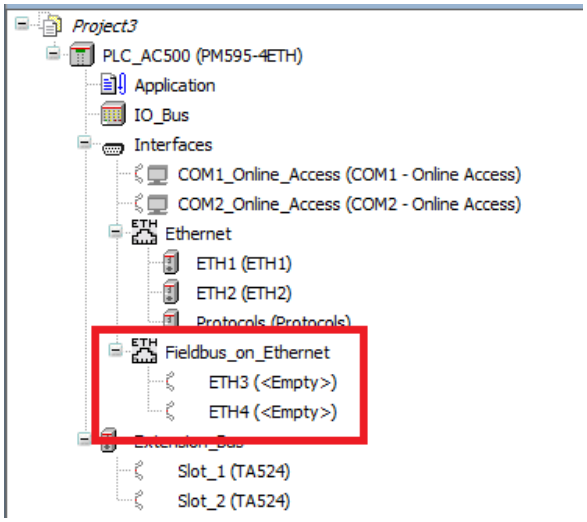
如果使用的是 PM585、PM590、PM591 或 PM592 处理器，在处理器左侧的扩展总线插槽按 1 到 4 的升序编号（你拥有的插槽数量取决于上文中选择的端子底座），其中 1 最接近处理器。CM579-ECAT 必须放在其中一个插槽中。最常见的是把 CM579-ECAT 模块插入第一个插槽“ Slot 1 ”中。要选择它，右击“ Slot_1 (TA542) ”图标并选择“ Add Object ”。Automation Builder 现在会显示一个对话框，让你选择与该插槽连接的通讯模块。展开“ Communication modules ”，然后点击“ EtherCAT ”，直到可以选择 CM579-ECAT 模块。

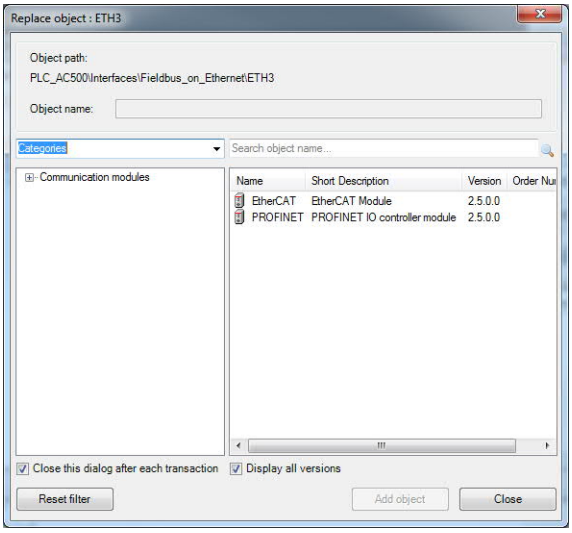


在对话框中，点击“ Replace Object ”把该模块添加到硬件系统中。

与之前一样，Automation Builder 会自动为设备命名。但是，你可以给它提供一个自己想要的任何名称（我们接受默认名称“ CM579-ECAT ”）。

如果使用带有集成 EtherCAT 连接器的 PM595，不需要 CM579-ETHCAT 模块。在设备树中右击其中一个“ Fieldbus_on_Ethernet ”条目（比如，ETH3），并选择“ Add object ”。



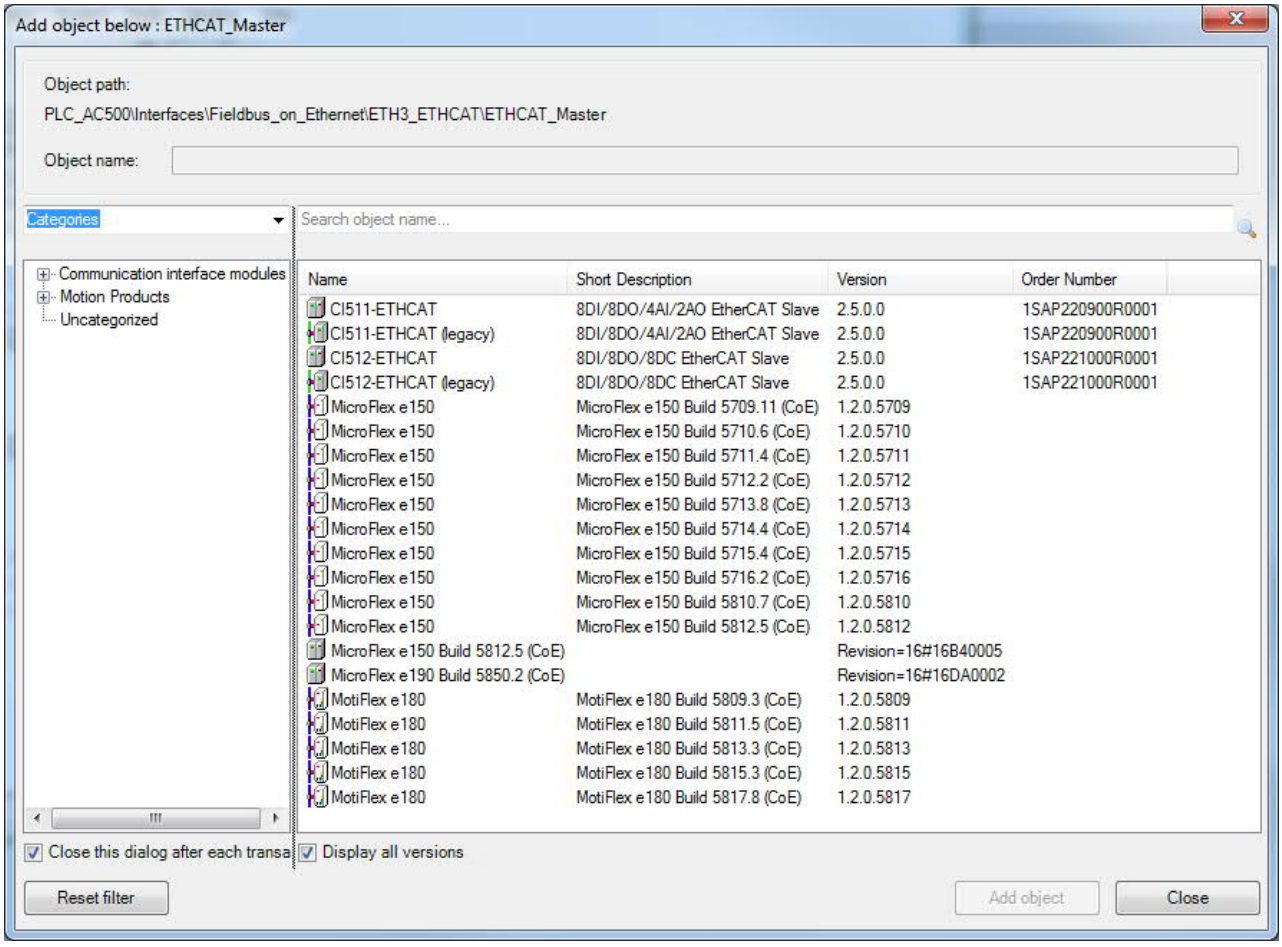


然后，你可以从可用模块列表中选择“ EtherCAT ”，把它添加到选定的以太网接口。

现在 EtherCAT 主设备已经在设备树中，我们作好了添加 EtherCAT 从设备的准备（即，e190 或 e180 驱动器）。

有两种方法可以把 ABB MicroFlex e190 或 MotiFlex e180 添加到 Automation Builder 项目。选择通过 Mint 伺服驱动器包安装的对象，或选择从 EtherCAT ESI 文件创建的对象。本文会在接下来的章节中讨论这两种方法。

右击树形列表中的 EtherCAT 设备，选择“ Add object ”。Automation Builder 会显示一个对话框，列出所有可用的 EtherCAT 从设备（比如， EtherCAT 驱动器和 EtherCAT I/O 模块）。



添加 EtherCAT 驱动器（Mint 伺服驱动器包的方法）

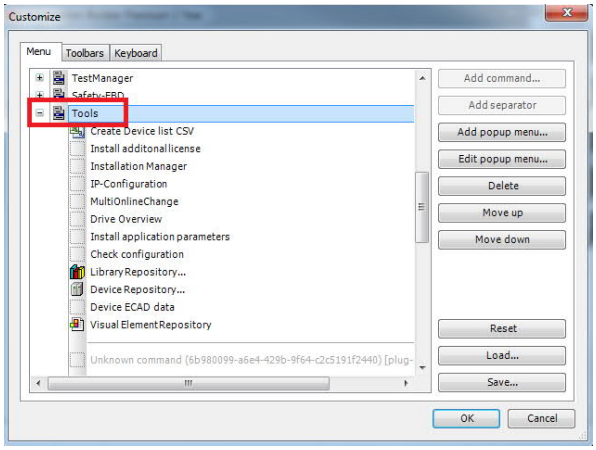
这不是首选的方法，因为驱动器固件的更新频率可能高于伺服驱动器包的发布频率。但是，出于介绍的完整性，我们会在本文中详细说明该方法。带有白色“网络驱动器样式”图像的设备图标作为 Mint 伺服驱动器包的一部分被添加进来。Mint 伺服驱动器包可作为 Automation Builder 的一部分安装。

Mint/伺服驱动器包包含 Automation Builder 需要的关于 e190/e180 从设备的所有信息，对特定的固本版本是唯一的。Mint/伺服驱动器包允许用户查看 Mint Workbench 项目信息，在 Automation Builder 项目中包括参数表等文件，以及从 Automation Builder 启动 Mint Workbench。始终可以从 new.abb.com/motion 免费下载最新版本的伺服驱动器包（在 MicroFlex e190 或 MotiFlex e180 驱动器的支持页面上，标签为“伺服驱动器包”）。

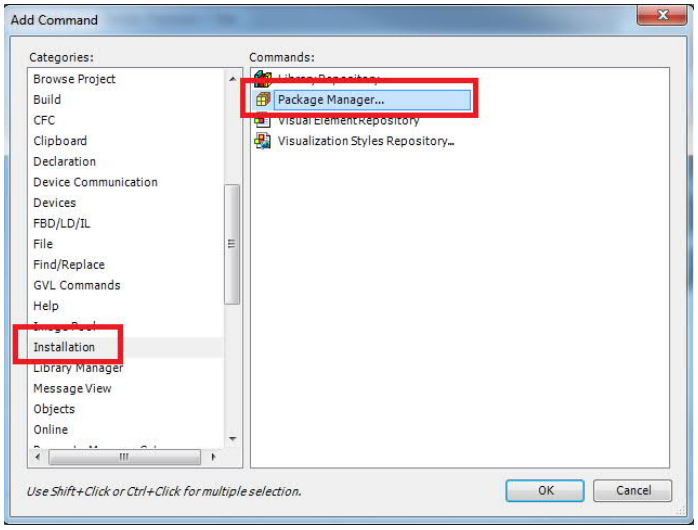
参考 http://www.baldormotion.com/support/SupportMe/productsupport.asp?ID=MTE180 为例。

Mint/伺服驱动器包是 Automation Builder 的一个可选安装组件。对于 Automation Builder 1.2，伺服驱动器包最高支持到 5817 固件版本（对较新的版本，除非你正在使用的较新版本的 Automation Builder 已经包含较新版本的包，否则你可能需要从运动支持网站下载和安装新的伺服驱动器包，或使用本应用说明包括的文件）。

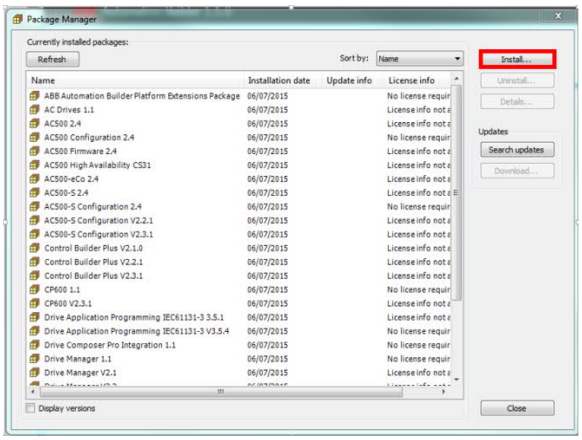
如果你要下载和安装新/较新的 Mint 伺服驱动器包，那么你首先需要访问 Automation Builder 工具中的“Package Manager”。为实现这种访问，保存并关闭当前的 Automation Builder 项目。在没有项目打开的情况下，点击 Tools > Customize 菜单。会弹出警告消息，但你可以忽略它-只需要点击 OK。然后会弹出定制对话框。向下滚动并展开“Tools”。



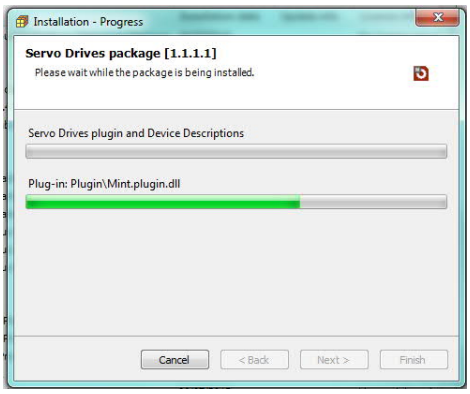
现在选择可用工具列表中的任何项目（Package Manager 最后出现在你选择的项目旁边的 Tools 菜单中）。现在点击“Add command…”。之后会出现 Add Command 对话框。



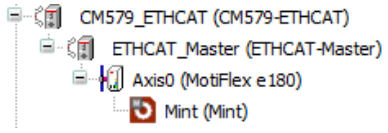
点击 Categories 列表中的“ Installation ”，从右侧的窗格中选择 Package Manager，点击“ OK ”。Package Manager 现在会出现在工具菜单中，作好安装 Mint 伺服驱动器包的准备。假设你已经从运动支持网站下载了最新的伺服驱动器包，现在应该在 Automation Builder 菜单中选择 Tools > Package Manager。这会打开 Package Manager。点击 Install。



浏览到你保存的包，点击“ Open ”。在下一个对话框中选择“ Typical setup ”，点击“ Next ”。



安装完成后，Automation Builder 立即自动重启，使安装生效。重启完成后，你可以重新打开原来的 Automation Builder 项目。在向 EtherCAT 主设备添加对象时，你会看到可添加到设备树的最新伺服驱动器列表。点击合适的 Mint 伺服驱动器包对象，把它添加到设备树中。



与树中的所有其它对象一样，可为它指定一个用户名称（在上例中，我们把驱动器重命名为“ Axis0 ”-如果被询问是否要自动重新分解/重命名所有的引用，选择“ No ”）。注意，现在 Mint 符号已经与驱动器对象关联在一起（上文例子中的 MotiFlex e180）。双击该符号，打开右侧窗口中的 Mint 选项卡，可从中启动 Mint Workbench。如果你已经使用该方法向设备树添加驱动器，你可以跳过第 8 页的“ Automation Builder – 配置 EtherCAT 设置 ”一节。

添加 EtherCAT 驱动器（ESI 方法）

这是把驱动器添加到硬件树的首选方法，因为可以从驱动器（或从支持网站）直接提取 ESI 文件 – 因此，始终可以获得最新的 ESI / 设备。带“ PLC 模块样式 ”图像的设备图标是导入 EtherCAT ESI 文件后添加的条目。

EtherCAT 从属设备信息文件（ESI 文件）是一种 XML 格式的文件。它描述了传动的标识信息和 EtherCAT 特性，可供 Automation Builder 等配置工具使用，用于向管理器（AC500 PLC）描述 EtherCAT 从属设备的信息。ESI 文件对特定的驱动器固件版本来说是

唯一的。如果你随后更新驱动器到之前没有使用过的固件版本，你需要更新基于新固件版本的 ESI 文件用于更新 Automation Builder 设备树（比如，如果你需要使用随新固件引入的新 PDO 可映射对象，可能有必要更新）。

这些项目允许用户把驱动器添加作为 EtherCAT 从设备，配置 PDO 映射和其它设置以通过 AC500 PLC 控制驱动器，但不允许作为 Automation Builder 项目的一部分添加 Mint Workbench 项目和相关文件。如果正在使用版本非常新（或 WIP）的固件，并且还没有为该固件创建 Mint 伺服驱动器包，则必须使用 ESI 方法导入设备。

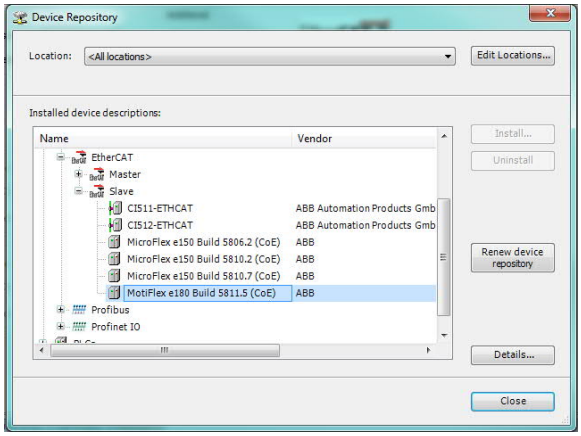
如果 Automation Builder 还没有列出与你正在使用的驱动器固件版本匹配的 ESI 文件，则可通过以下三种方式之一获取 ESI 文件。

1. 从 ABB 运动支持网站的产品支持页面获取 - 网站提供了 ESI/XML 文件的链接
2. 使用 Mint Workbench 连接到驱动器，选择“ EtherCAT ”页，在“ Summary ”选项卡下点击“ Save As… ”按钮从驱动器提取 ESI 文件
3. 使用 Mint Sidebar 应用程序。点击按钮把你的互联网浏览器连接到驱动器的主页。你可以在主页上找到 EtherCAT ESI 文件的链接

无论如何，你可以提取文件，给文件指定一个包括驱动器固件版本的名称（比如，ABB MicroFlex e190 Build 5867.4 (CoE).xml），并把文件保存到你想要的任何位置（只要你能够记住它！）。
在 Automation Builder 能够使用新的 ESI 文件前，我们需要安装 ESI 文件。在顶部菜单栏内的工具菜单下选择“ Device Repository… ”

现在，在 Device Repository 对话框中展开 Fieldbuses>EtherCAT>Slave 树形列表。现在，Automation Builder 将显示它已经有 ESI 文件的 EtherCAT 从属模块的列表。如果已经有一个条目的驱动器固件与你正在使用的固件版本匹配，则不需要安装 ESI 文件。但是，Automation Builder 默认不包括 ABB 伺服驱动器条目。如果你需要添加驱动器，点击“ Install… ”按钮。

浏览到你之前保存的 ESI 文件并点击“ 打开 ”。现在 Automation Builder 将导入 ESI 文件。在完成后，将在从属设备列表中出现代表驱动器的图标。

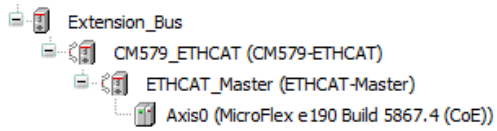


关闭对话框。现在我们可以把驱动器添加为 EtherCAT 从设备。

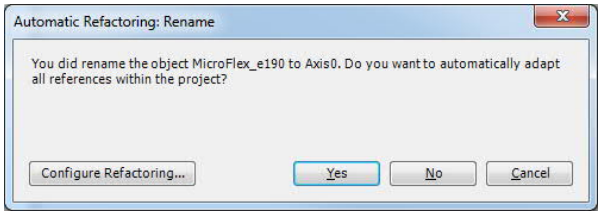
右击 EtherCAT 主设备的图标（在你添加 CM579-ECAT 模块，或在 PM595 上设置集成 EtherCAT 连接器时，它会自动添加），并选择“ Add Object… ”。

在可用 Mint 伺服驱动器包对象和 ESI 安装生成的对象列表中，选择与你当前在驱动器上使用的固件版本匹配的 ESI 对象，点击“ Add object ”。

现在，Automation Builder 将显示被选择的驱动器是连接到 EtherCAT 主设备的从设备。如果你想要修改驱动器名称（比如，Axis0、Infeed、Unwind 等），你可以点击硬件树中图标旁边的文字并输入一个新的名称（在下面的截图中，我们把它称为“ Axis0 ”）。



推荐使用这种方法，因为这会使之后查找与该驱动器相关的 PDO 对象更容易。
如下文所示，在为驱动器输入新的名称时，可能会询问你是否想要自动修改项目中的所有引用。

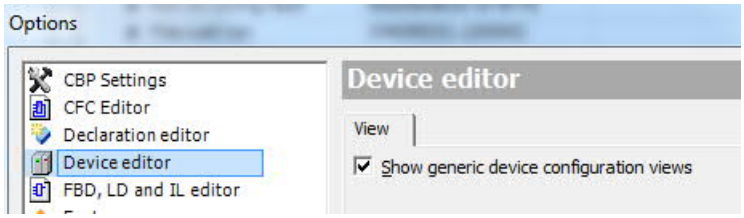


点击 NO（如果选择任何其它选项，Automation Builder 1.2 中的一个问题会导致程序崩溃）。

Automation Builder – 配置 EtherCAT 设置

现在，我们需要配置我们的 EtherCAT 主设备和从设备的运行方式（比如，ABB EtherCAT 伺服驱动器同时支持 Distributed Clock 和 Sync Manager 型同步模式。我们要选择 DC – Synchronous，因此我们必须设置它并配置一个 EtherCAT“ 总线周期时间 ”）。

首先，我们必须启用通用设备配置视图（在使用 PM595 PLC 处理器时，需要它来设置短于 1 ms 的周期时间）。在 Tools 菜单中选择 Options...，然后在左侧的窗格中选择“ Device editor ”。然后，右侧的窗格会显示一个复选框，允许你选择“ Generic device configuration view ”。确保它已经勾选并点击 OK。



现在双击设备树中 CM579_ECAT 模块的图标（在使用 PM595 时，双击集成 EtherCAT 连接器的图标）。Automation Builder 将显示该模块的当前设置列表。这些一般可保留其默认值设置，但重要的是要确保“ Distributed clocks ”的设置为“ Active ”。

Parameter	Type	Value	Default Value	Unit	Description
Run on config fault	Enumeration of BYTE	No	No		Start PLC program even on configuration fault
Max wait run	DWORD(0..120000)	30000	30000	ms	Max wait time for valid inputs
Min update time	DWORD(0..20000)	10	10	ms	Cycle time for data exchange to IEC program
Broken slave behaviour	Enumeration of DWORD	Leave all broken slaves down	Leave all broken slaves down		Behaviour of broken slaves
Distributed clocks	Enumeration of DWORD	Active	Active		Distributed clocks inactive or active
BootUpTime	DWORD(0..120000)	100000	100000	ms	Max wait time for booting of slaves

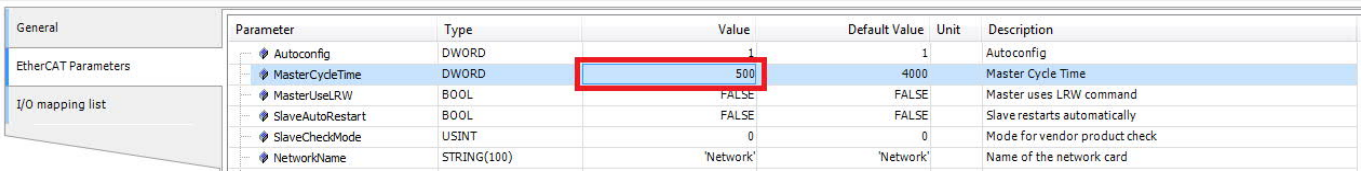
现在双击标签为“ (ETHCAT-Master) ”的设备树中 EtherCAT 连接器下方的图标。现在，Automation Builder 在右侧的窗格中显示三个选项卡（General、EtherCAT Parameters 和 I/O mapping list）。

在 General 选项卡上，多数设置可保留默认值。但是，我们必须确保一项设置正确，适合我们的应用。

Cycle time - 这是指 EtherCAT 总线周期时间。它是以微秒为单位的时间。举例来说，把它设置为 2000 代表 2ms 的周期时间。注意，如果正在使用 PM595，并且你希望设置 500 µs 的最小周期时间，如果使用的是 v1.2 或更早版本的 Automation Builder，则无法通过设置完成。后文将介绍如何设置 500 µs 周期时间，在这里这个设置可以先保持不变。

现在选择“ EtherCAT Parameters ”选项卡。

除非你正在使用 PM595 并且需要使用 500 µs 的 EtherCAT 周期时间，否则不需要对该选项卡做任何修改。如果你需要设置 500 µs 周期时间，你应该按下文所示相应编辑“ MasterCycleTime ”的值。



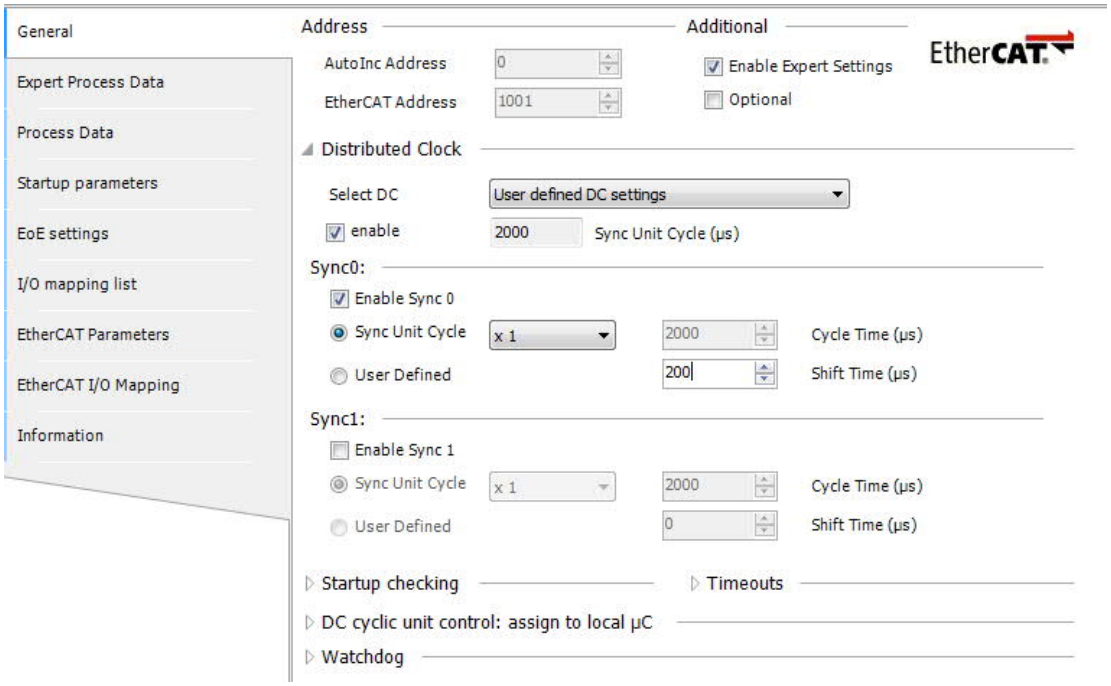
在 EtherCAT 周期时间大于等于 1 ms 时，这里的值应该与 General 选项卡上之前设置的值匹配。

重要提示：运动驱动器只支持该周期时间的双倍数（比如，500 µs、1ms、2ms、4ms、8ms 等）

现在双击 e190 或 e180 驱动器的图标（我们之前已经重新命名为 Axis0）。然后，Automation Builder 会在屏幕的右侧显示几个选项卡（General、Process data、Startup parameters、I/O mapping list、EtherCAT Parameters、EtherCAT I/O Mapping 和 Information）。

选择 General 选项卡。本选项卡将显示驱动器的 EtherCAT 地址（默认 1001）。它无法修改。在添加其它设备到网络时，它们的地址将自动设置（1002、1003，等）。需要注意的是驱动器在网络上的物理位置很重要。一旦配置好网络，不要交换任何 EtherCAT 电缆的顺序，因为这会实际影响从设备的寻址（进而导致其配置与你的预期设置不匹配）。

在 General 选项卡上，我们必须设置/检查的主要设置是 Distributed Clock 方案（设置为“ DC-Synchronous ”）和“ Shift Time (us) ”的值。Shift Time 一般应设置为 EtherCAT 周期时间的 10%（比如，在周期时间为 2 ms 时，设置 Shift Time 为 200 µs）。但是，这个设置默认为灰色。通过选择“ Enable Expert Settings ”以能够编辑这个值（它同时会增加两个新的选项卡，Expert Process Data 和 EoE settings）。



现在点击 Expert Process Data 选项卡。e190 和 e180 驱动器的 ESI 文件详细说明控制驱动器（使用 Cyclic Synchronous Position）只需要四个过程数据对象（PDO）- 从 PLC 到驱动器的控制字和目标位置，从驱动器到 PLC 的状态字和实际位置。POD 映射自动包括在项目中。在本简易“ 入门 ”指南中，不需要修改这些内容或添加额外的 PDO 映射。比如，在应用需要快速位置锁存（即，Touchprobe 对象）时，可能需要额外的 PDO 映射。在这种情况下，可以使用 Expert Process Data 页面来添加额外的输入和输出对象。这不在本应用说明的范围以内，在其它应用说明内涉及（比如，可参考 AN00220，它描述了如何添加 Touchprobe 对象）。

现在选择“ Startup parameters ”选项卡。同样，这些设置我们不需要修改，但将做详细说明以供参考。
“ Give EtherCAT control ”设置为 1。它可以保证在 PLC 程序启动时（以及 EtherCAT 配置数据被发送到驱动器时），驱动器的控制给定值信号源被自动切换为“ Real-time Ethernet ”。
ModesOfOperation 设置为 8（对 Cyclic Synchronous Position） - 这是 PLC 进行运动曲线控制的模式（每个总线周期发送新的目标位置）。还可使用 Cyclic Synchronous Velocity (CSV/9)和 Cyclic Synchronous Torque (CST/10)模式，但我们不会在本应用说明中详细描述这些内容。

现在选择“ I/O mapping list ”选项卡。本选项卡允许你定义变量名称。PLC 程序使用这些变量名称来对驱动器的输入和输出 PDO 寻址。我们需要为之前映射的四个 PDO 输入变量名称。双击变量栏下表格内的区域，输入之后要用到的有意义的变量名称。下面的截图显示了我们是如何确定为我们的变更命名的。

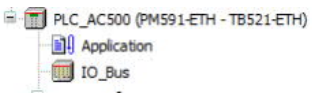
Object Name	Variable	Channel	Address	Type	Description
Axis0	wAxis0ControlWord	AX0_ControlWord_U16	%QW1.0	UINT	AX0_ControlWord_U16
Axis0	diAxis0TargetPos	AX0_TargetPosition_I32	%QD1.1	DINT	AX0_TargetPosition_I32
Axis0	wAxis0StatusWord	AX0_StatusWord_U16	%IW1.0	UINT	AX0_StatusWord_U16
Axis0	diAxis0ActualPos	AX0_ActualPosition_I32	%ID1.1	DINT	AX0_ActualPosition_I32

我们使用符合对象数据类型的前缀（按照 AN00244 的定义），后接“ Axis0 ”，因为这是我们的从属驱动器设备的名称，后面加上 PDO 功能的描述文字。

在我们之后启动 CoDeSys 时（PLC 编程环境），Automation Builder 将自动创建包括这些变量名称的全局变量模块。现在，我们已经做好建立 PLC 配置并启动 CoDeSys 编程环境的准备。但是，首先需要保存项目-点击屏幕顶部的软驱图标（或按下 CTRL+S）保存项目。

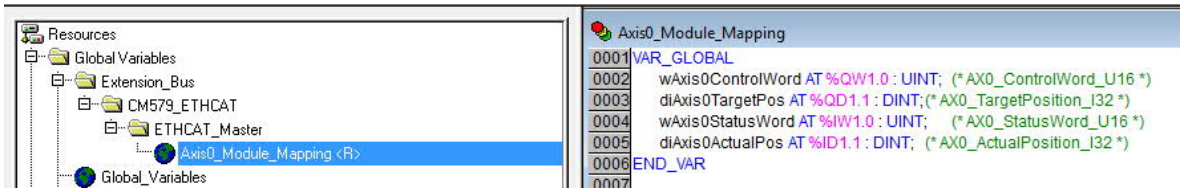
Automation Builder – 建立配置

在 Automation Builder 设备树视图中，有代表与项目关联的 PLC 程序的图标。该图标的默认名称为“ Application ”。如果需要，你可以选择该图标然后再次单击以重命名（在本例中，我们将保持其名称 Application）。



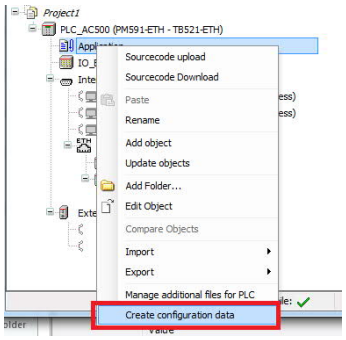
双击程序图标，Automation builder 根据我们已经添加的所有设备建立硬件配置。如果硬件配置有错误，在屏幕的底部将显示错误消息。你需要修复错误才能继续操作（如果需要更多信息，请参考 Automation Builder 帮助文件）。如果没有错误，一旦 Automation Builder 建立好配置，它将启动 CoDeSys 编程环境（记住，前提条件是已经安装好 ABB PS552-MC-E v3.1.0 PLCopen 运动控制库 - 如果你没有这些，请联系你当地的 ABB 销售办公室）。

要检查我们为驱动器添加的变量映射是否已经自动包括在 PLC 程序中，选择 CoDeSys 项目浏览器内的“ Resources ”选项卡，然后展开 Global variables 树形列表（其结构与 Automation Builder 中的硬件树一致），直到找到“ xxxx_Module_Mapping ”条目（其中，xxxx 是你之前为设备指定的名称 - 比如，Axis0_Module_Mapping）。双击该图标，右侧的窗格将显示声明的变量及其关联地址，如下文所示：



如果映射的变量不存在，你应该关闭 CoDeSys，返回 Automation Builder 并检查你的伺服驱动器设备是否已经正确添加到设备树，并且你已经按照上文的描述添加变量映射。

如果数据正确但没有正确连接到 CoDeSys，那么请打开 Automation Builder，然后右击“ Application ”图标，并选择“ Create configuration data ”，以重新创建配置数据。

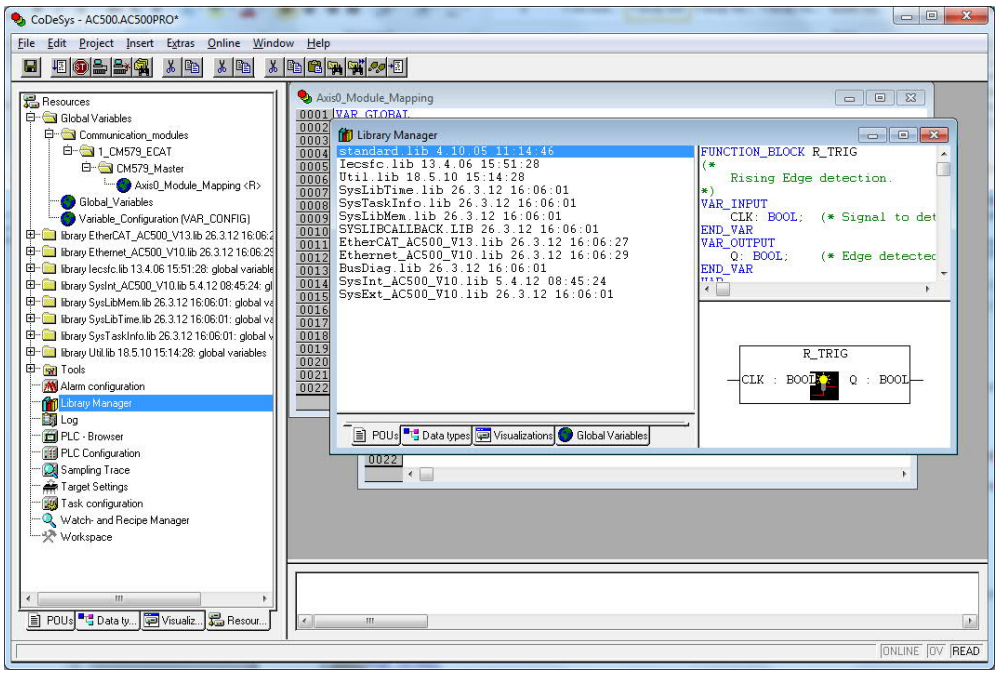


此时，已经做好使 EtherCAT 网络运行的一切准备（假设你的 EtherCAT 电缆连接到 PLC 和驱动器的正确以太网端口上 - 如果对这些连接不确定，请查阅用户手册）。如果你愿意，可以从 CoDeSys 登录 PLC（即下载 PLC 程序）并运行代码以检查网络运行情况（如果你不了解如何在线进入 PLC 和下载程序，请参考第 23 页的说明 - 测试 PLC 程序）。如果项目建立失败，在 CoDeSys 建立窗口中将出现错误 - 如有必要，参考 PLC 文档/帮助系统进一步了解如何解决这些错误。一旦程序下载完成并开始运行，驱动器上的绿色“ Net Run ” LED 灯将闪烁数次，最后变为固定的绿色表示 EtherCAT 可使用。如果没有出现这种情况，重新执行上述描述的步骤。如果你无法找到错误，联系你当地的 ABB 办公室获得进一步的支持。如果 LED 变为固定的绿色，那么你可以继续完成本应用说明的剩余部分。

使用 CoDeSys 编写简单的程序

基本的硬件设置和相关变量的创建已经完成。现在，我们已经准备好开始编写一些 PLCopen 代码。我们可以使用这些代码使能我们的驱动器并执行一些简单的运动。

首先，我们需要把 PS552-MC-E 库文件包括在我们的 PLC 项目中。点击 CoDeSys 项目浏览器内的“ Resources ”选项卡。然后双击“ Library Manager ”图标。现在，右侧的窗格将显示项目中已经包括的库。



右击 Library Manager 已经包括的.lib 文件列表下方的区域，并选择“ Additional Library… ”。将出现对话框让你浏览到新的.lib 文件，可点击“ Open ”按钮选中它们（你可以按照需要使用标准的 Windows CTRL 和 SHIFT 按钮来添加多个文件）。浏览到 PS552-MC-E 库文件的安装目录 （对 Win7 系统，它一般为 c:\Program Files(x86)\Common Files\CAA-Targets\ABB_AC500\AC500_V12\Library\PS552-MC）。

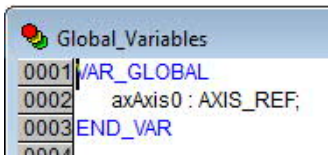
在定位 PS552-MC-E 库文件后，添加所有以下.lib 文件：

- CompactMotionControl_AC500_V12.lib
- ECAT_AC500_APPL_V21.lib
- MC_Base_AC500_V11.lib
- MC_Blocks_AC500_V11.lib

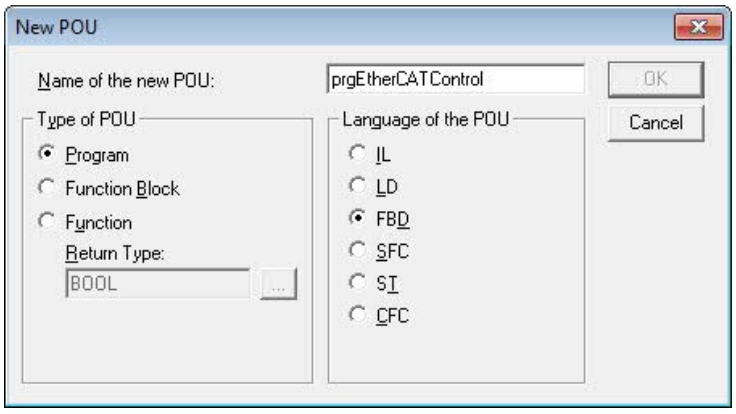
库管理器对话框将更新，显示这些文件现在已经包括在你的项目库中。

下面我们需要创建一个轴结构与我们的硬件设备相关联。

双击“ Resources ”选项卡上的“ Global Variables ”图标打开全局变量编辑器窗口。
声明一个 AXIS_REF 类型的变量（配合 PLCopen 运动控制功能使用的预定义数据结构）。我们把我们的变量命名为“ axAxis0 ”，以匹配与轴关联的设备名称和 AN00244 定义的前缀（但是你可以任意命名）。

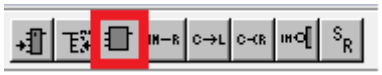


现在点击 CoDeSys 项目浏览器的“ POU ”选项卡。你可以看到，其中已经包括了一个默认的程序（名为 PLC_PRG）。对使用 EtherCAT 的 PLCopen 运动项目，最好是只为控制轴所需要的核心功能块创建新程序。可使用任意 IEC61131 语言，但作为示例，最简单的是使用功能块图（FBD）。在 POUs 选项卡中，右击 POUs 文件夹，并选择“ Add object ”。设置 POU 类型为 Program，为编程语言选择 FBD。你可以为 POU 任意命名，但在本例中，我们把它称为“ prgEtherCATControl ”，并假定你已经这样完成命名。



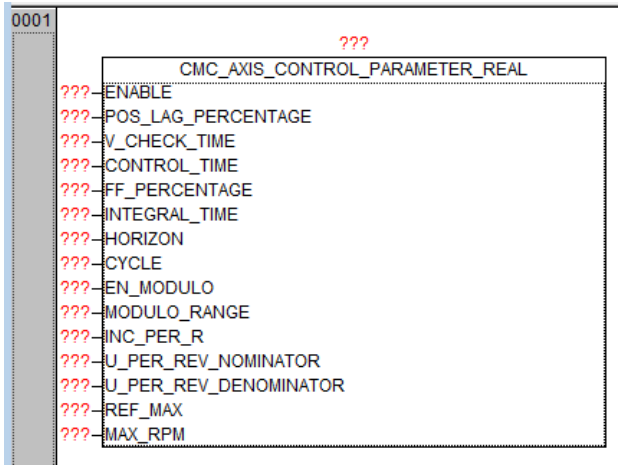
现在，因为不再需要默认程序，你可以右击 POU 浏览器中的 PLC_PRG 图标，选择“ Delete Object ”来删除它。

双击 prgEtherCATControl 图标打开本模块的代码。程序编辑器将在编码对话框的顶部显示本代码的变量声明区域及下方的程序段 1。点击程序程序段 1 内的空白虚线框，然后点击 CoDeSys 屏幕顶部的“ Box ”工具栏按键。

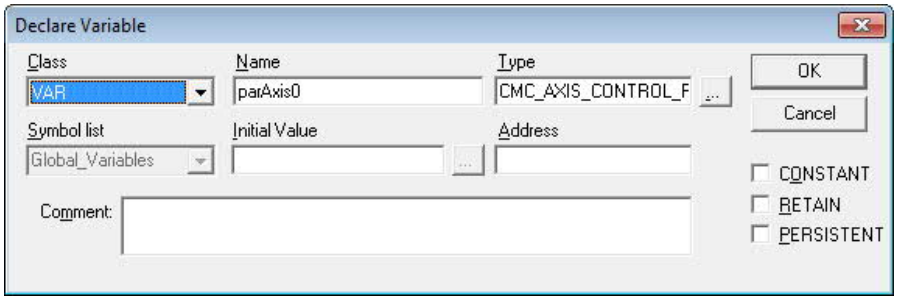


默认情况下，现在程序程序段 1 内将出现代表 AND 功能的 FBD（带高亮 AND 文字）。但是，我们要创建的是 CMC_AXIS_CONTROL_PARAMETER_REAL 功能实例（用于设置轴换算比例），因此在现有 AND 文字的区域输入 CMC_AXIS_CONTROL_PARAMETER_REAL（或按下 F2 使用输入助手，并从“ Standard Function Blocks ”部分选择这个功能块）。

如果你的拼写正确，或者在输入助手中正确选择它（同时已经正确添加 PS552-MC-E 库文件），则程序段 1 当前将显示我们的控制参数功能块的实例。



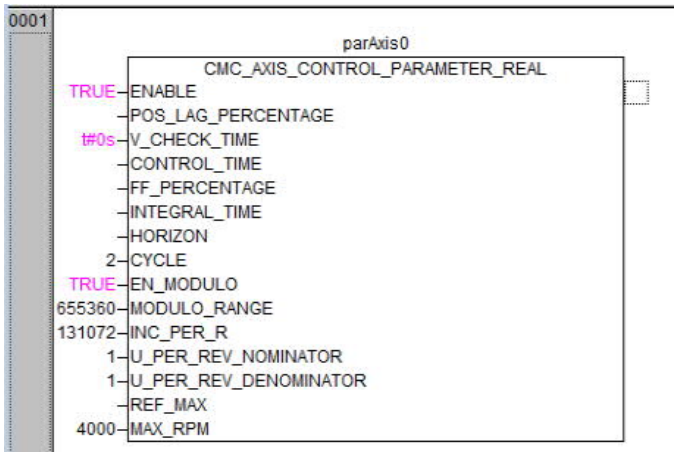
点击功能块上方的“ ??? ”并输入文字，为这个功能的实例提供一个名称（我们在这里把它称为 parAxis0）。在按下键盘上的回车键后，CoDeSys 将弹出对话框，要求你声明该变量的类型（它会自动选择 CMC_AXIS_CONTROL_PARAMETER_REAL，所以你只需要点击 OK）。



下表显示了必须输入的输入参数值（必须删除所有其它输入参数-即选择???并删除它）。

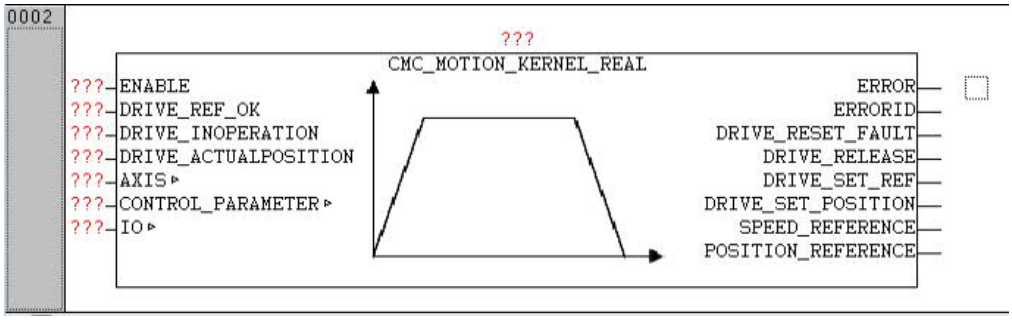
输入参数	说明	值	数据类型
ENABLE	确定 PLC 是否应该在每个总线周期内都处理轴（为了修改缩放值，必须设置为 FALSE。但在本例中，我们将使用固定缩放，因为可以把它永久性设置为 TRUE）	TRUE	BOOL
V_CHECK_TIME	用于设置抽样时间以获得实测速度（在使用 EtherCAT 时不会实际使用，但在当前必须强制设置为零）	t#0s	TIME
CYCLE	设置总线周期时间 – 这必须与我们之前通过 Automation Builder 设置的 EtherCAT 总线周期时间匹配	2	LREAL
EN_MODULO	如果轴需要被作为旋转轴（即轴位置在预先定义的范围内有循环-对可在一个周期内确定绝对位置的转盘/工具快换盘来说，这可能有用），或者轴位置有可能超过 32 位位置边界（比如连续旋转的轴），本参数应该设置为 TRUE。如果轴只在 32 位位置范围内向前/后移动，并且没有模数功能的要求，则将本参数设置为 FALSE。在本应用说明中，我们把它设置为 TRUE（因为我们可能保持无限止的向前移动）。	TRUE	BOOL
MODULO_RANGE	结合 EN_MODULO 使用。设置一个模数循环内有多个次编码器计数。在我们的例子中，我们假设我们的 131072 次计数的电机安装到 5:1 变速器上。因此，我们把 MODULO_RANGE 设置为 655360（因此，读取位置时将返回变速器输出的绝对位置）如果你对一个周期内的绝对位置不感兴趣（比如一台持续运行的输送机），则把 MODULO_RANGE 设置为 2147483647。	655360	DINT
INC_PER_R	设置所连接的电机旋转一周时传动收到的编码器计数（在本例中，我们把 BSM60R-240MT 电机连接到我们的驱动器。因此，电机旋转一周计数为 131072-设置与你的电机匹配的参数）	131072	DWORD
U_PER_REV_NOMINATOR	本参数（结合分母参数）允许你配置轴换算比例。注意，在数学中，术语“ NOMINATOR ”比“ NUMERATOR ”更正确。你必须指定电机旋转一周有多少个用户单位（为表示分数值，有必要使用分子和分母）。在我们的例子中，我们将使用用户单位“ 转 ”。因此电机旋转一周有一个用户单位（因此，分子和分母均可设置为 1）	1	DINT
U_PER_REV_DENOMINATOR	见上文	1	DINT
MAX_RPM	“ 校准 ” PLC 以匹配驱动器上配置的“ DRIVESPEEDMAX ”（如果你还不知道驱动器上设置的最大应用速度，通过 Workbench 查看驱动器上的 DRIVESPEEDMAX 设置-或使用 Workbench 的 Operating Mode 页来确定以 RPM 为单位的应用程序最大速度）。在我们的例子中，我们为驱动器配置的最大速度为 4000rpm	4000	WORD

在输入这些值后，程序段看起来应该像这样…



此时，有必要保存 PLC 程序（点击 CoDeSys 顶部的软驱图标保存文件）。

右击程序段 1 的灰色区域，选择“ Network(after) ”向 PLC 代码添加第二个程序段。再次点击虚线框（这一次是在程序段 2 中），然后点击工具栏中的“ Box ”图标插入另一个功能块。这次为 CMC_MOTION_KERNEL_REAL 类型，或再次使用输入助手（这将添加一个浮点运动控制器的实例）。如果你已经正确完成本操作，程序段 2 的外观如下。

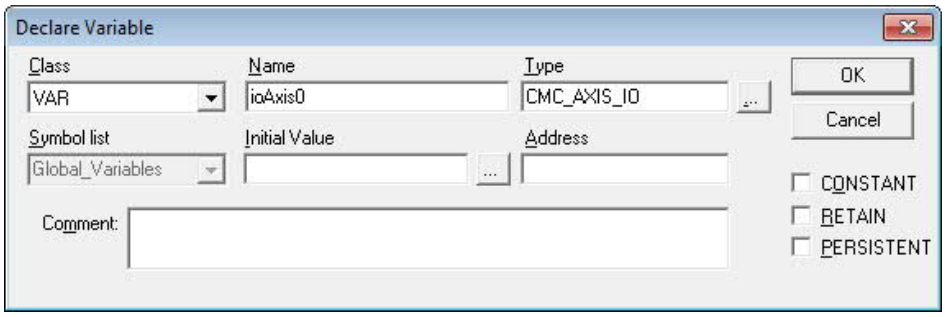


点击功能块上方的???并输入文字，为这个实例命名（我们在这里把它称为 kerAxis0）。Codesys 将显示一个对话框让你指定变量类型， 并自动声明为 CMC_MOTION_KERNEL_REAL 类型。因此，只需要接收该设置。

我们还必须为本功能块输入一些输入参数。同时，必须把控制器输出指向驱动器。因为我们之前已经为“ Cyclic Synchronous Position ”操作配置好驱动器，核心的“ POSITION_REFERENCE ”输出必须指向驱动器。下表显示了必须输入的参数（只需要删除任何不需要的???）。多数输出参数并不需要，但把这些输出分配给变量可能有助于诊断。如果你决定把变量分配给其中一些输出（比如，ERRORID），你可以右击功能块，并选择“ ZOOM ”查看包括参数变量类型在内的详情的块声明，以确定所需的变量类型。

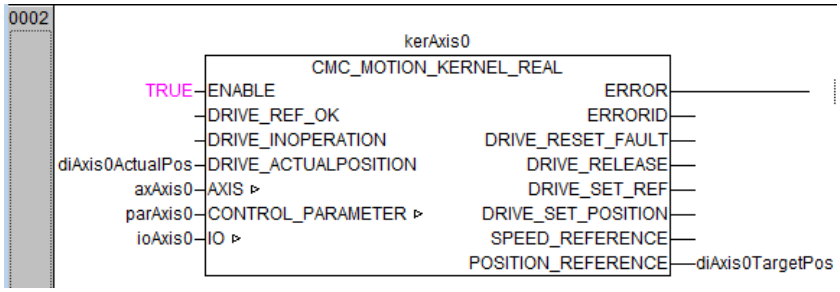
输入参数	描述	值	数据类型
ENABLE	确定 PLC 是否应该在每个总线周期内都处理控制器，可永久设置为 TRUE	TRUE	BOOL
DRIVE_ACTUALPOSITION	与驱动器连接的电机的位置反馈（以脉冲数表示）。这是我们之前在 Automation Builder 中设置的 PDO 对象之一，我们为此添加了一个映射变量-diAxis0ActualPos	diAxis0ActualPos	DINT
AXIS	要描述的轴的数据结构。在本例中，我们使用我们之前声明的 AXIS_REF 型变量-axAxis0	axAxis0	AXIS_REF
CONTROL_PARAMETER	对轴正在使用的控制参数块的引用。我们在这里输入之前对 CMC_CONTROL_PARAMETER_REAL 块的命名。它把控制器与轴参数链接在一起。	parAxis0	CMC_CONTROL_PARAMETER_REAL
IO	某些与轴关联的 IO 的数据结构（比如，寻零传感器、正向和负向限幅开关）	ioAxis0	CMC_AXIS_IO （见下面的注释）

注意：在添加 ioAxis0 作为 IO 输入参数的名称时，CoDeSys 将询问这个变量的数据类型（因为它是新的变量，到目前为止还没有声明）。如下文所示，输入 CMC_AXIS_IO 作为数据类型 - 这是适合轴 I/O 结构的预定义数据类型。



输出参数	描述	值	数据类型
POSITION_REFERENCE	由 PLC 控制器输出、必须发送到驱动器的位置目标。这是我们之前映射的 PDO 之一。我们为其添加了一个映射变量- diAxis0TargetPos	diAxis0TargetPos	DINT

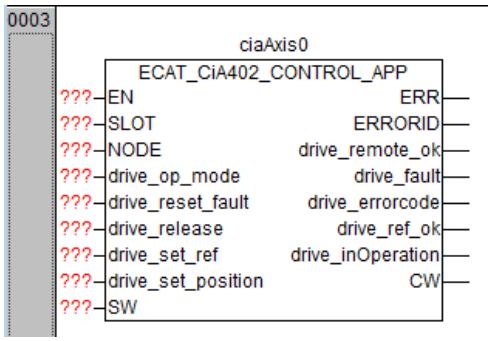
在输入这些值后，程序段看起来应该像这样…



现在添加第三个程序段，这次添加一个 ECAT_CiA402_CONTROL_APP 类型的框（我们把它命名为 ciaAxis0 。与你的预期一致，它是一个 ECAT_CiA402_CONTROL_APP 类型的变量）。

注：ECAT_CiA402_CONTROL_APP 块同时用于 MicroFlex e190 和 MotiFlex e180 的编程。还有一个名为 ECAT_CiA402_CONTROL_e150_APP 的较旧的块。它也可用于两种驱动器类型，但最好是选择上面推荐的块。

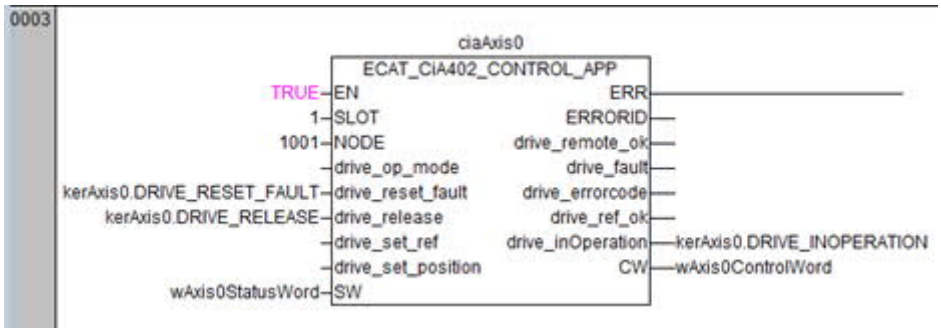
这个功能块封装了用于 MicroFlex e190 和 MotiFlex e180 驱动器的 CiA Ds402 驱动器控制协议，允许通过 EtherCAT 操作该协议。我们需要把这个功能块链接到驱动器本身（通过我们之前看到的驱动器 EtherCAT 地址。该地址由 Automation Builder 自动设置为 1001）。我们还需要把这个功能块链接到轴的核心（通过核心的某些参数）。最后，我们需要把相关 DS402 控制字和状态字与这个功能块关联起来。它们都是我们之前已经添加映射变量的 PDO（连接到/来自驱动器）。



下表显示了我们的 ciaAxis0 功能块需要的输入和输出参数：

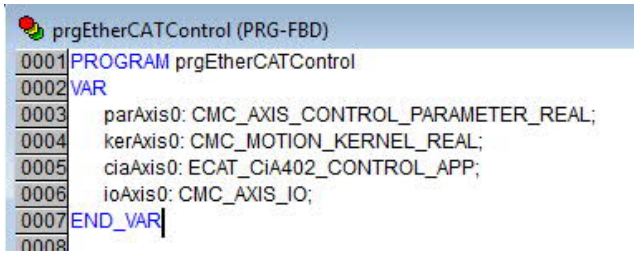
输入参数	描述	值	数据类型
EN	决定 PLC 是否应该在每个 EtherCAT 周期处理 DS402 PDO。在我们的例子中，我们希望始终保持对驱动器的控制，因此我们把它永久设置为 TRUE。	TRUE	BOOL
SLOT	CM579-ECAT 模块插入哪一个插槽？在本例中，我们把它插入到处理器左侧的第一个插槽内，也即是插槽 1。对 PM595，如果使用 ETH3，这个值是 5；如果使用 ETH4，这个值是 6。	1	BYTE
NODE	我们在添加驱动器时 Automation Builder 自动分配的 EtherCAT 地址（1001）	1001	DWORD
Drive_reset_fault	链接到相关轴核心的故障复位请求	kerAxis0.DRIVE_RESET_FAULT	BOOL
Drive_release	链接到相关轴核心的驱动器释放请求	kerAxis0.DRIVE_RELEASE	BOOL
SW	来自相关驱动器的 DS402 状态字（我们之前已经把 wAxis0StatusWord 变量映射到本驱动器 PDO）	wAxis0Statusword	WORD
输出参数	**描述**	**值**	**数据类型**
Drive_inOperation	链接到相关轴核心的 DRIVE_INOPERATION 输入参数的输出参数	kerAxis0.DRIVE_INOPERATION	BOOL
CW	连接到相关驱动器的 DS402 控制字（我们之前已经把 wAxis0ControlWord 变量映射到本驱动器 PDO）	wAxis0ControlWord	WORD

在输入这些值后，程序段看起来应该像这样…



与上一个程序段一样，多数输出参数并不需要，但把这些输出分配给变量可能有助于诊断。如果你决定把变量分配给其中一些输出（比如，drive_errorcode），你可以右击功能块，并选择“ ZOOM ”查看包括参数变量类型在内的详情的块声明，以确定所需的变量类型。

如果你已经决定不分配任何额外变量给添加的三个功能块，那么此时你应该只有自动添加的变量声明。

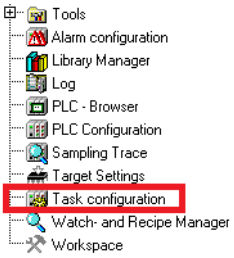


同样，此时可能有必要保存 PLC 程序（和 Automation Builder 项目）。

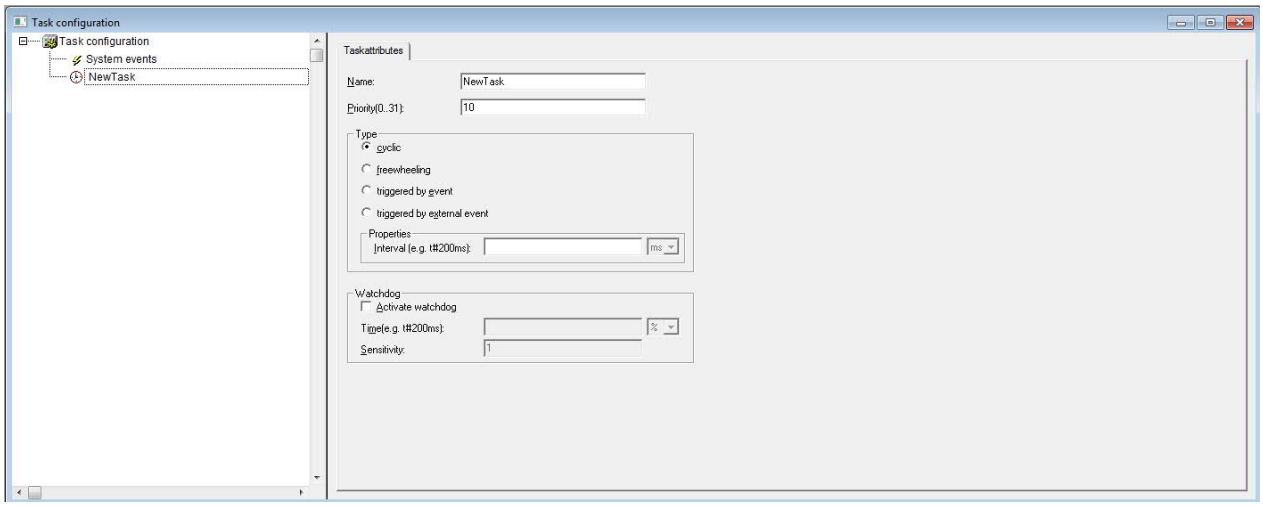
到目前为止我们添加的三个程序段构成了通过 EtherCAT 控制 ABB 伺服驱动器“ 核心 ”。现在，我们已经准备好从 PLC 任务调用该程序，并确保它与 EtherCAT 网络之间的数据传输同步。

任务配置

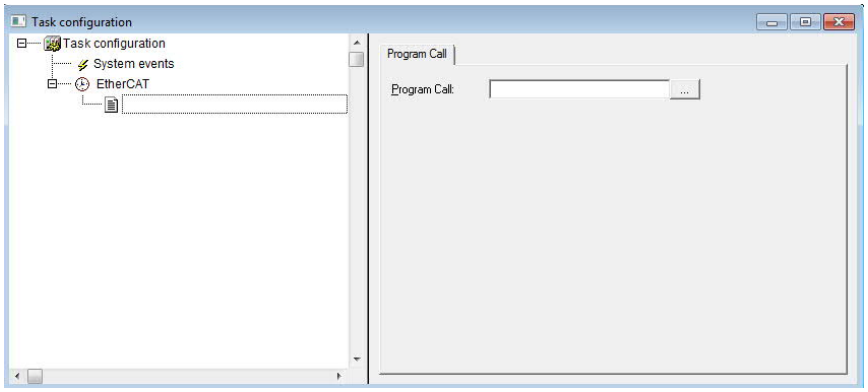
点击 CoDeSys 项目浏览器的“ Resources ”选项卡。然后双击‘ Task configuration ’图标。



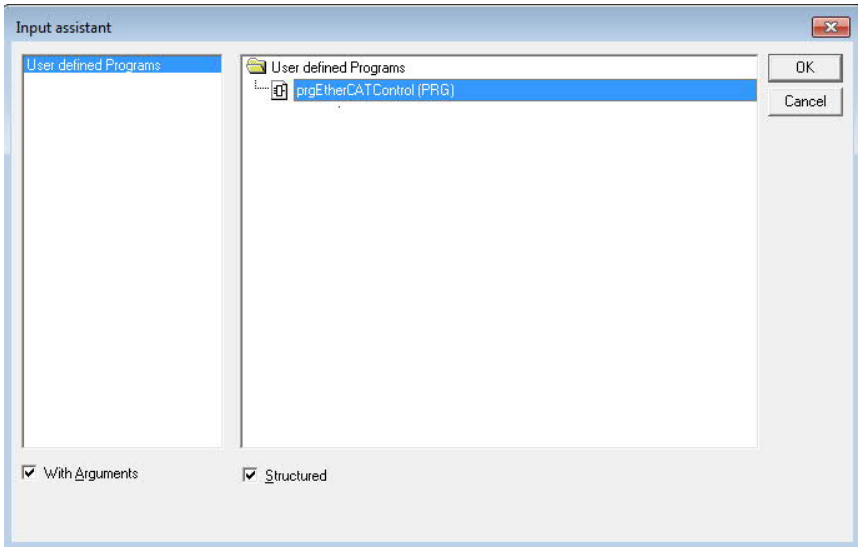
CoDeSys 的右侧窗格现在将显示 Task Configuration 对话框。右击右侧窗格中对话框内的 Task Configuration 图标并选择“ Append Task ”。对话框的外观应该如下所示。



我们需要配置一项任务，即按照 EtherCAT 连接器生成的事件执行我们的程序（prgEtherCATControl）。点击“ NewTask ”文字，并将其重新命名为“ EtherCAT ”。现在右击 EtherCAT 图标并选择“ Append Program Call ”。此时，你的任务配置对话框将如下所示。



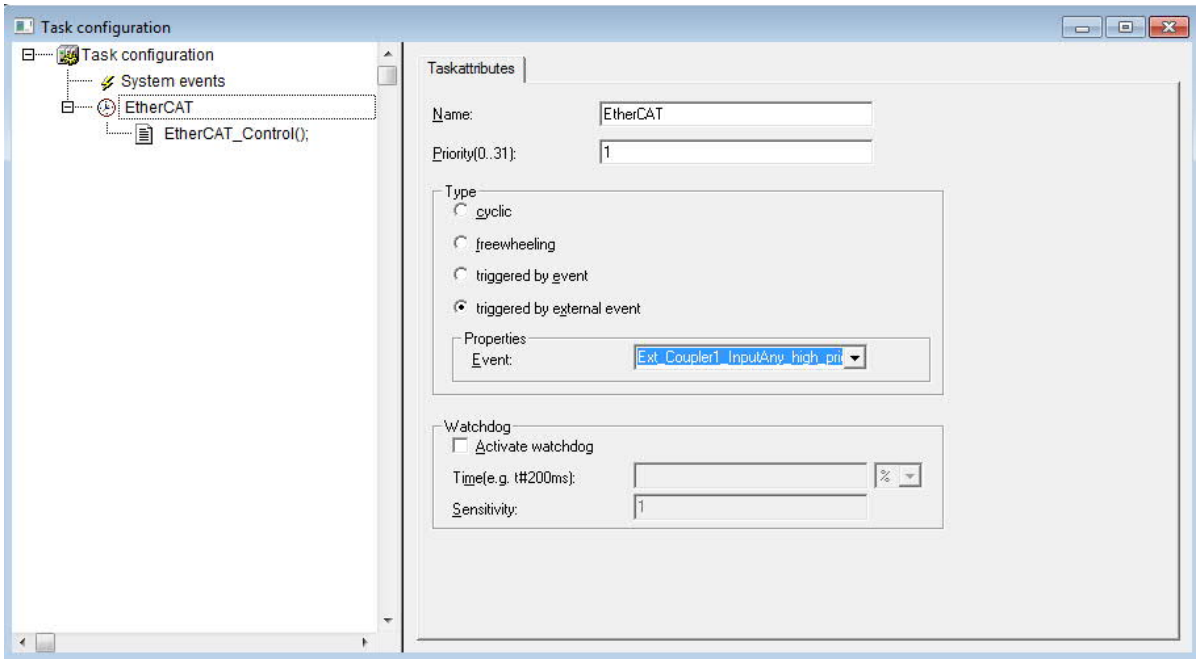
在右侧窗格内点击有“ … ”的按钮，并选择 prgEtherCATControl（我们项目中唯一的用户程序）。



点击 OK。

现在我们需要设置 EtherCAT 任务，使其被外部事件触发（即保证任务的处理与 EtherCAT 周期相关）。点击“ EtherCAT ”图标，并在右侧窗格内首先设置 Priority 为 1，然后选择“ triggered by external event ”选项。

然后，事件属性下拉菜单将提供一系列选项供选择…点击“ Ext_Coupler1_InputAny_high_priority ”（如果你正在使用 PM595，并且已经设置 ETH3 用作 EtherCAT 连接器，则点击 Ext_Coupler5_InputAny_high_priority）。



通过选择“ Ext_Coupler1_**InputAny**_high_priority ”，我们保证在收到每份 EtherCAT 报文后 EtherCAT 任务都得到处理，使可能的反应时间最短。“ Ext_Coupler1_**Input2Any**_high_priority ”允许任务在 EtherCAT 报文被发送时得到处理，它的反应时间要长一个周期。对每种输入类型，还提供一个正常优先级版本的事件。它为相关任务的执行赋予较低的优先权。但对运动控制来说，最好是始终使用最高优先权的事件。

现在你可以关闭本对话框，并在 CoDeSys 中保存 PLC 程序（同时保存 Automation Builder 项目）。

现在，你可以试着在 CoDeSys 菜单选项中选择 Project > Build。如果你正确执行了到目前为止的所有说明，项目在编译时不应该产生任何错误和警告。否则，请重新阅读之前的篇章，并检查到目前为止的工作。

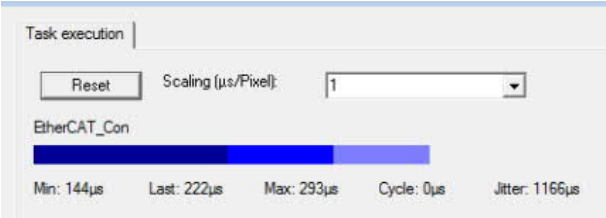
如果程序编译没有出现错误，那我们就可以开始向项目添加一些简单的运动命令。

添加简单的运动

到目前为止，我们编写的运动代码仅限于控制我们的 EtherCAT 轴所需的基本功能块。这些块始终应该位于由任务调用的 POU 中。任务由 EtherCAT 连接器事件触发（在本文的剩余部分中，我们将称之为“ 实际 POU ”）。本实时 POU 中应包括的其它代码/逻辑例子可以是...

- Touchprobe 功能块
- Touchprobe 功能块寻零
- 非常准确地检测轴位置或编码器阈值的逻辑
- 需要由位置或正在达到的编码器值触发的运动块
- 与通过 CD522 编码器模块处理主设备编码器相关的功能块

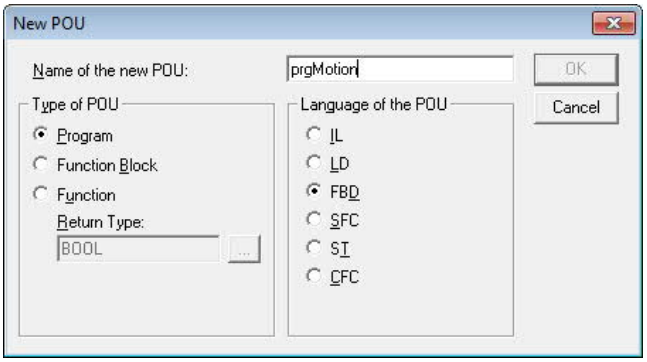
对许多应用（和本快速入门指南），没有必要在实时 POU 中包括一般运动逻辑。实时 POU 中包括的代码越多，本代码的处理时间越长。重要的是，实时 POU 的处理时间要低于或等于 EtherCAT 周期时间减去 300 µs - 在在线连接到正在运行的 PLC 程序时，可通过 CoDeSys 中的“ Task Configuration ”对话框对其进行监控。



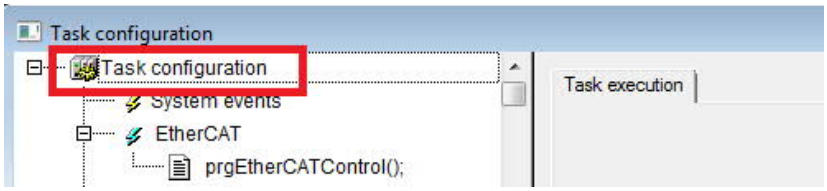
在下面来自应用程序的截图中，实时 POU 的最大执行时间是 293 µs。我们必须在其中考虑执行中的“ 延时抖动 ”。在 CoDeSys 中有一个错误，即会给显示的延时抖动加上 1000（它也不会显示可用周期的实际值）。因此，在我们的例子中，我们的 POU 的总执行时间（最坏情况）是 293 µs + 166 µs = 459 µs。它刚好位于我们的 2000 µs – 300 µs (即 1700 µs)限值以内。因此，如果我们有需要，还有大量的空间可向实时 POU 添加额外的代码。

现在，我们将向 Task Configuration 添加新的任务，并通过该任务调用新的 POU/程序。这项新的任务可以是 Cyclic。在我们的例子中，我们把这项任务称为“ Background ”。我们将通过该任务调用名为“ prgMotion ”的程序。

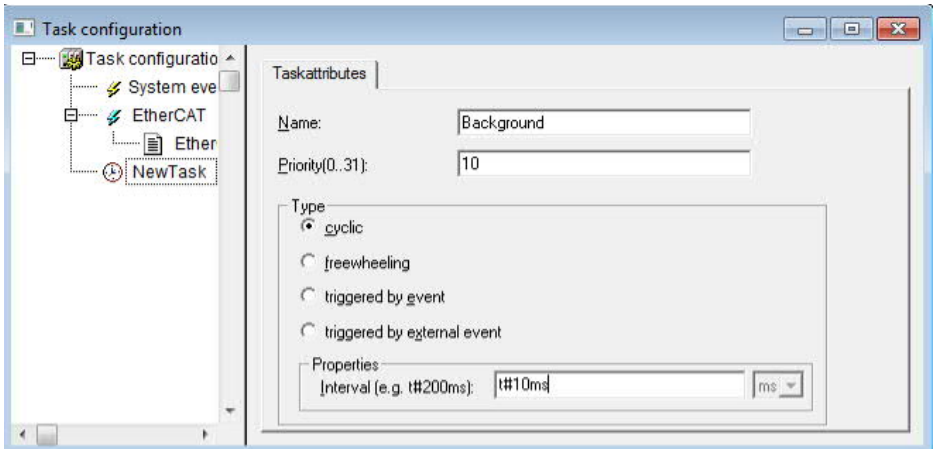
在 POU 浏览器中，右击 POUs 文件夹，并选择“ Add object ”。我们将添加一个使用功能块图（FBD）编程的“ prgMotion ”程序项目，因为 FBD 在非常简单的应用中经常被使用到。按照下文所示配置生成的对话框。



点击 OK，这个程序单元将被添加到 POU 树。现在切换到 Task configuration 对话框（如果窗口还没有关闭，它可能仍然在编码器的某个位置打开；否则，再次双击“ Resources ”选项卡上的“ Task configuration ”。）。右击对话框中的“ Task configuration ”图标。

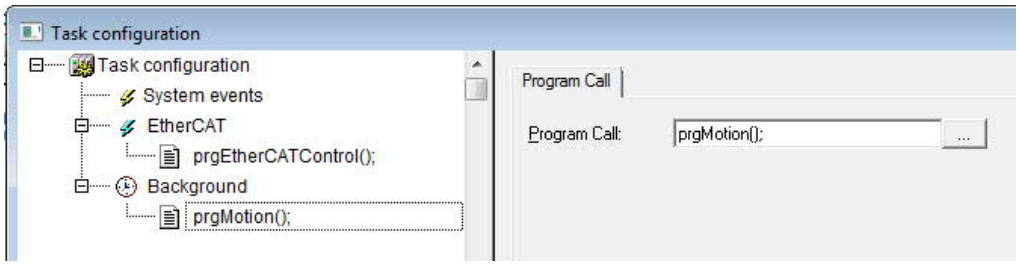


并选择“ Append Task ”-树形列表中将出现一项新的任务。选择这项新任务，然后按照下文所示编辑对话框，以创建一项名为“ Background ”的 Cyclic 任务，周期时间为 10ms。



注意，在关闭本对话框或点击树形视图的其它位置后，任务名称将从“ NewTask ”变为“ Background ”。

现在，我们需要添加对我们的 Motion 程序的调用。因此，右击“ NewTask ”图标（如果已经强制修改名称，则点击“ Background ”图标），并选择“ Append Program Call ”。点击有三个点的按钮（...），并使用输入助手（键盘快捷键是 F2）来选择 Motion 程序单元，或输入程序名称（即 prgMotion();），如下文所示：

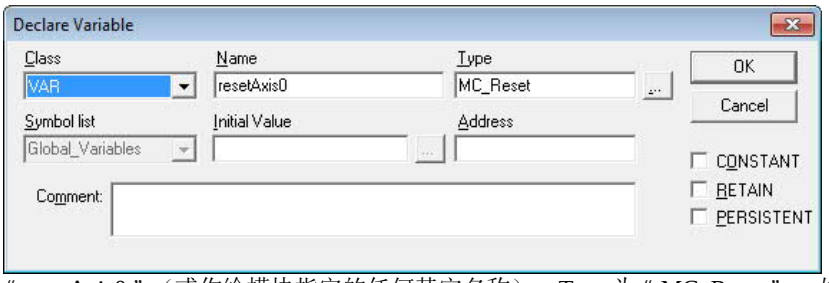


现在你可以关闭这个对话框。我们已经准备好向运动程序添加运动块。切换回 POU 浏览器，并双击“ prgMotion ”程序图标打开程序。你会看到空白网络 0001 已经准备好接受代码。到现在为止，你应该已经了解如何向程序添加功能块（选择新的框，使用 F2/输入助手来选择功能块名称或输入名称等...）。因此，我们只需要列出你需要添加到代码的功能块，并描述必要的输入和输出参数。在本例中，你可以按任何顺序添加这些块，但我们假定你按照下文所述的顺序添加。

MC_Reset

如果传动处于故障状态，则无法使能轴和/或发出任何运动请求。添加本功能块允许我们复位可能已经发生的任何驱动器错误（比如，因为要求过度加速导致的“ 跟随误差 ”，或把 e180 驱动器切换到用于微调的直接模式导致的“ PDO 数据丢失 ”）。如果驱动器有错误，将在七段显示屏上显示故障代码（比如，1 0 0 0 5 表示跟随误差）。如果 PLC 程序正在运行，则 ciaAxis0 功能块的 drive_errorcode 输出也将指示故障代码。

如果这是你正在添加的第一个块/网络，点击网络 0001 中的虚线框边缘（或右击你程序中的最后一个程序段，选择“ Network (after) ”添加另外一个程序段）。添加一个 MC_Reset 功能块，并为它提供一个相关的名称（我们的命名为 resetAxis0）。CoDeSys 会要求你声明这个变量（即，你刚刚命名的 MC_Reset 功能块的实例）。将弹出对话框，自动显示 Class 为“ VAR ”，Name 为



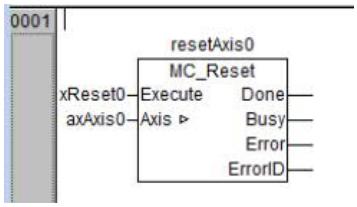
“ resetAxis0 ”（或你给模块指定的任何其它名称），Type 为“ MC_Reset ”， 如下文所示。

只需点击 OK 接受它，就可以把变量声明添加到程序文件的顶部。

本 PLCopen 功能块只采用两个输入参数：

- Execute：本输入的上升沿使功能块处理必要的动作，以尝试重置伺服驱动器上的错误。应该为本输入分配一个 BOOL 型变量（我们的命名为 xReset0）。只需要点击 Execute 输入框旁边的???，然后输入变量的名称。CoDeSys 将显示对话框，让你配置范围和数据类型（默认为 BOOL，你只需点击 OK 确认该对话框）
- Axis：这是与功能块相关的轴结构（在我们的例子中为 axAxis0）。

在这个简单的示例中，我们不必把变量分配给任何 MC_Reset 输出参数。以下是我们完成后的程序段的样子。



在之后对我们的程序进行测试时，我们双击输入变量（xReset0），然后按下 CTRL+F7 强制设置我们已经选择的变量值（比如设置为 TRUE 以重置错误）。

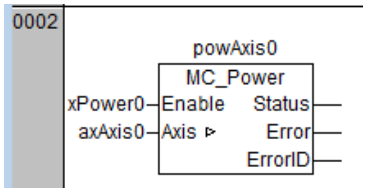
MC_Power

本 PLCopen 功能块允许用户使能和禁用相关的轴。右击你程序中的最后一个程序段，并选择“ Network (after) ”添加另一个程序段。这一次添加一个 MC_Power 功能块，并为它提供一个相关的名称（我们的命名为“ powAxis0 ”）。如上文所示，每次在你创建变量时，都会出现 Declare Variable 对话框。除非我们有其它命名，只需要点击 OK 创建变量声明。

本 PLCopen 功能块只采用两个输入参数：

- Execute：本输入的上升沿使功能块处理必要的动作，以使能轴。输入的下降沿导致功能块禁用轴。应该为本输入分配一个 BOOL 型变量（我们的命名为 xPower0）。
- Axis：这是与功能块相关的轴结构（在我们的例子中为 axAxis0）。

以下是添加 MC_Power 功能块后我们的另一程序段的样子。



在这个简单的示例中，我们没有必要向任何 MC_Power 输出参数分配变量，但把 Status 输出（我们示例中的 powAxis0.Status）用作运动功能上的联锁可能有用处（即，只允许在状态指明驱动器被启用时发出运动命令）。

在之后对我们的程序进行测试时，我们双击输入变量（xPower0），然后按下 CTRL+F7 强制设置我们已经选择的变量值（比如设置为 TRUE 以启用驱动器）。

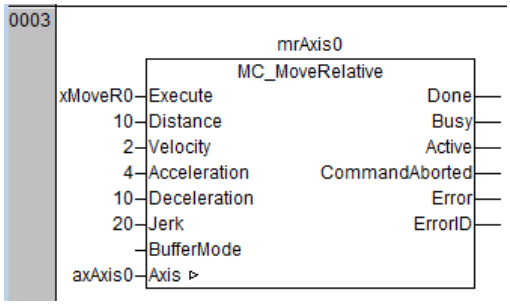
MC_MoveRelative

本 PLCopen 功能块允许用户把轴自当前位置开始移动一段相对的距离。右击你程序中的最后一个程序段，并选择“ Network (after) ”添加另一个程序段。这一次添加一个 MC_MoveRelative 功能块，并为它提供一个相关的名称（我们的命名为“ mrAxis0 ”）。

本 PLCopen 功能块只采用几个输入参数：

- Execute：本输入的上升沿使功能块处理必要的动作，以尝试移动指定的轴一段已定义的距离。应该为本输入分配一个 BOOL 型变量（我们的命名为 xMoveR0）。
- Distance：在执行输入激活时所需移动的量（+/-）。你可以为这个输入定义一个变量，使距离可调。但在我们的例子中，我们“ 强制给定 ”了 10 个单位的距离（你可以回想我们之前使用我们的 Control Parameters 功能块的 U_PER_REV_NOMINATOR 和 U_PER_REV_DENOMINATOR 输入参数来设置我们的用户单位，使其等于电机转数）
- Velocity：移动时的旋转速度，以每秒的用户单位数表示（在我们的例子中为转/秒）。你可以为这个输入定义一个变量，使速度可调节，但在我们的例子中，我们“ 强制给定 ”了 2 转/秒的速度。
- Acceleration：移动时的加速速率，以每秒的每秒用户单位数表示（在我们的例子中为转/秒 ^{2}）。你可以为这个输入定义一个变量，使加速度可调节，但在我们的例子中，我们“ 强制给定 ”了 4 转/秒 ^{2} 的加速速率。
- Deceleration：移动时的减速速率，以每秒 ^{2} 的用户单位数表示（在我们的例子中为转/秒 ^{2}）。你可以为这个输入定义一个变量，使减速度可调节，但在我们的例子中，我们“ 强制给定 ”了 10 转/秒 ^{2} 的减速速率。
- Jerk：以每秒 ^{3} 的用户单位表示的加速度/减速度的变化率。为 Jerk 添加一个值形成“ S 曲线 ”型运动轨迹。设置 Jerk 为 0 形成梯形运动轨迹。你可以为这个输入定义一个变量，使 Jerk 的量可调节，但在我们的例子中，我们“ 强制给定 ”了 20 转/秒 ^{3} 的值。
- BufferMode：ABB运动库当前不支持除mcAborting以外的任何缓冲模式（mcAborting是库中包含的预定义常量）。在我们的示例中，我们将删除本输入参数中的???字符，把Buffermode设置为mcAborting的默认设置，即non-buffered。
- Axis：这是与功能块相关的轴结构（在我们的例子中为 axAxis0）。

以下是添加 MC_MoveRelative 功能块后我们的另一程序段的样子。



我们没有添加任何输出参数，但可按需要添加（比如，可向“ Done ”输出分配变量以指示移动完成的时间，或向“ ErrorID ”输出分配变量以指示移动失败时的故障代码）。

现在，我们已经添加了足够的代码来使能/禁用驱动器和执行简单的相对移动。

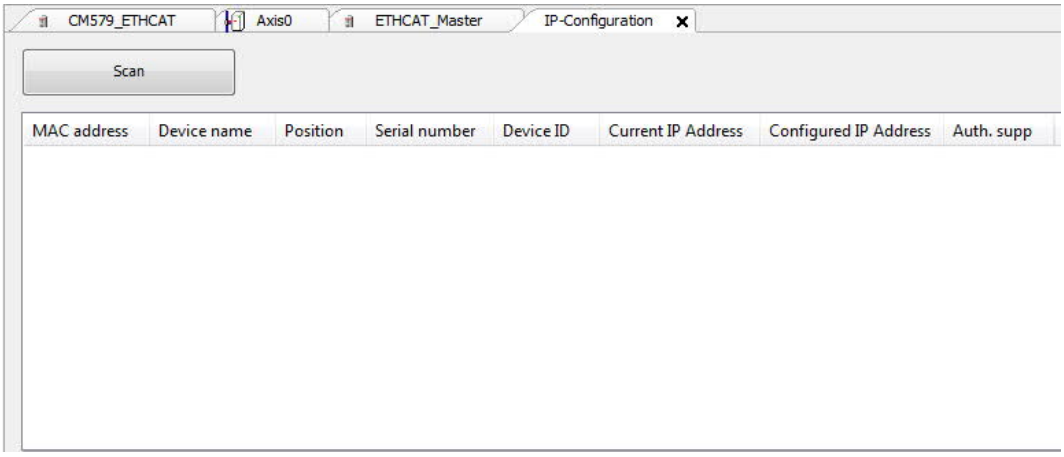
测试 PLC 程序

要测试 PLC 程序，假设驱动器已经调试和做好使能准备（比如，已经向驱动器施加交流电，STO 输入不激活） - 如需要，请参考驱动器的安装手册了解更多详情。

如果你还没有在线连接到 PLC，在 CoDeSys 中选择 Online>Communication Parameters 菜单，并确保你已经选择你正在使用的 PLC（如果你需要添加一个新的条目，选择“ New… ”，选择 3S TCP/IP device，输入你 PLC 的 IP 地址，选择 Port 为 1201 和 Motorola 字节顺序）。

如果你不确定你的 PLC 设置的 IP 地址，那么切换到 Automation Builder，选择 Tools>IP Configuration 菜单。注意，你需要禁用你已经配置好的任何防火墙本工具才能工作。

右侧的窗格将有如下显示。



点击“ Scan ”。这将扫描你的以太网端口，并尝试找到已连接的 PLC（以及与网络连接的所有 ABB MAC 地址）。出现的对话框将显示 PLC 的 IP 地址（并允许你作出修改）。你需要确保你的 PC 的网络适配器位于同一子网内（比如，如果发现 PLC 位于 192.168.0.10，则我的 PC 适配器需要位于 192.168.0.x。其中，x 是除 0、10 或 255 以外的值，且不能被同一网络中的任何其它以太网设备使用）。

只要 CoDeSys 通讯参数正确，就可以编译你的项目（Project > Build）。查看屏幕底部的对话框，了解是否没有错误，编译正确。如果 CoDeSys 报告任何错误，在项目下载成功前你需要解决这些错误 - 如有必要，参考 PLC 编程文档获得更多帮助。

一旦项目下载完成，选择 Online>Run 菜单选项。PLC 程序现在应该运行（PLC CPU 的前端应该显示 RUN，CoDeSys 的右下角应该指示“ Running ”。同时，在 PLC 把 EtherCAT 配置传输到驱动器时，驱动器上的绿色网络状态 LED 灯应该闪烁。（在配置完成后立即保持常亮，驱动器进入运行状态）。
如果驱动器七段显示屏正在显示“ - ”，则表示驱动器当前无错误，我们可以尝试使能它。如果驱动器七段显示屏正在闪烁某种错误代码（按顺序重复的一系列数字），则我们需要复位错误。

向下滚动浏览整个 PLC 程序，直到找到包含 MC_Reset 功能块的程序段。双击“ xReset0 ”变量。每次双击该变量时，CoDeSys 将显示/切换变量名称旁边的强制值（在这种情况下，因为变量为 BOOL 类型，强制值将从 FALSE 切换到 TRUE 和反向切换）。设置强制值为 TRUE 并按下 CTRL+F7 把该值写入 PLC。MC_Reset 功能块的“ Execute ”输入的上升沿（从 FALSE 到 TRUE）应该使驱动器上的任何故障被重置（如果之前有错误代码闪烁，现在应该以实虚线代替）。

重复该过程，但需要把 xReset0 的值强制设置为 FALSE。

如果你不能重置驱动器错误，使用 Workbench 连接到驱动器，并进一步分析错误的原因（比如，可能因某些逻辑原因无法重置错误）。

一旦传动显示虚线，向下滚动浏览整个 PLC 程序，直到找到包含 MC_Power 功能块的程序段。
这时双击 xPower0 变量，直到它显示 TRUE。然后按下 CTRL+F7 把它写入 PLC。如果一切正常，驱动器应该被使能（同时，根据驱动器上当前使用的固件版本，七段显示屏应该显示 8 或 P）。
保持 xPower0 设置为 TRUE（将其设置为 FALSE 将导致驱动器停用）。

向下滚动浏览整个 PLC 程序，直到找到包含 MC_MoveRelative 功能块的程序段。双击 xMoveR0 变量，直到它显示 TRUE。然后按下 CTRL+F7 把它写入 PLC。如果一切正常，每次 bMoveR0 的值从 FALSE 变为 TRUE 时，PLC 应该执行一次相对移动（如果你此前一直按照本应用指南准确操作，移动量应为 10 转）。

如果你滚动到程序的顶部（到变量声明区域），你可以展开 mrAxis0 功能块等的定义，并检查输出参数的状态（比如 Done 和 Busy 输出）。选择 Resources 选项卡，展开 Global variables 树形列表，找到 Axis0_Module_Mapping 变量。你可以查看在 PLC 和驱动器之间传递的 PDO 数据。

如果你展开 Global variables 模块本身，你可以找到作为 AXIS_REF 的 axAxis0 的定义。同时你可以展开它查看轴可用的整个数据范围。

祝贺你！你已经用 AC500 PLC 通过 EtherCAT 成功控制了 ABB 伺服驱动器。请参考 PS552-MC-E 库文档了解更多信息。

其它资源

现在，你已经熟悉了使用 AC500 PLC 和通过 EtherCAT 对 ABB 伺服驱动器的基本操作，你可能想要探索其它的主题。你可以在运动支持网站上找到支持 EtherCAT 操作的其它应用说明。 http://new.abbmotion.com/support/SupportMe/ApplicationNotes.asp).

在编写本文件时，已经可以获取以下应用说明：

AN00203 – 使用 TwinCAT
AN00220 – EtherCAT 寻零方法
AN00221 – EtherCAT 快速位置捕捉
AN00234 – 用于简单运动的经由 EtherCAT 的通用驱动器接口
AN00239 – 使用 CD522 模块实现主设备编码器输入
AN00240 – 使用 CD522 模块快速锁存输入
AN00241 – 使用运动驱动器编码器通道实现主设备编码器输入

AN00242 – 通过 EtherCAT SDO 访问驱动器参数
AN00243 – 初始化 EtherCAT 网络
AN00244 – PLC 编码风格指南
AN00245 – 线性飞剪示例
AN00246 – 转位输送机示例
AN00252 – 通过 EtherCAT 访问驱动器错误数据
AN00253 – 圆盘刀示例

联系我们

要获得更多信息，请联系你的
当地的 ABB 代表，或以下一种方式：

new.abb.com/motion
new.abb.com/drives
new.abb.com/drivespartners
new.abb.com/PLC

EtherCAT®是由德国倍福自动化有限公司许可的注册商标和专利技术。