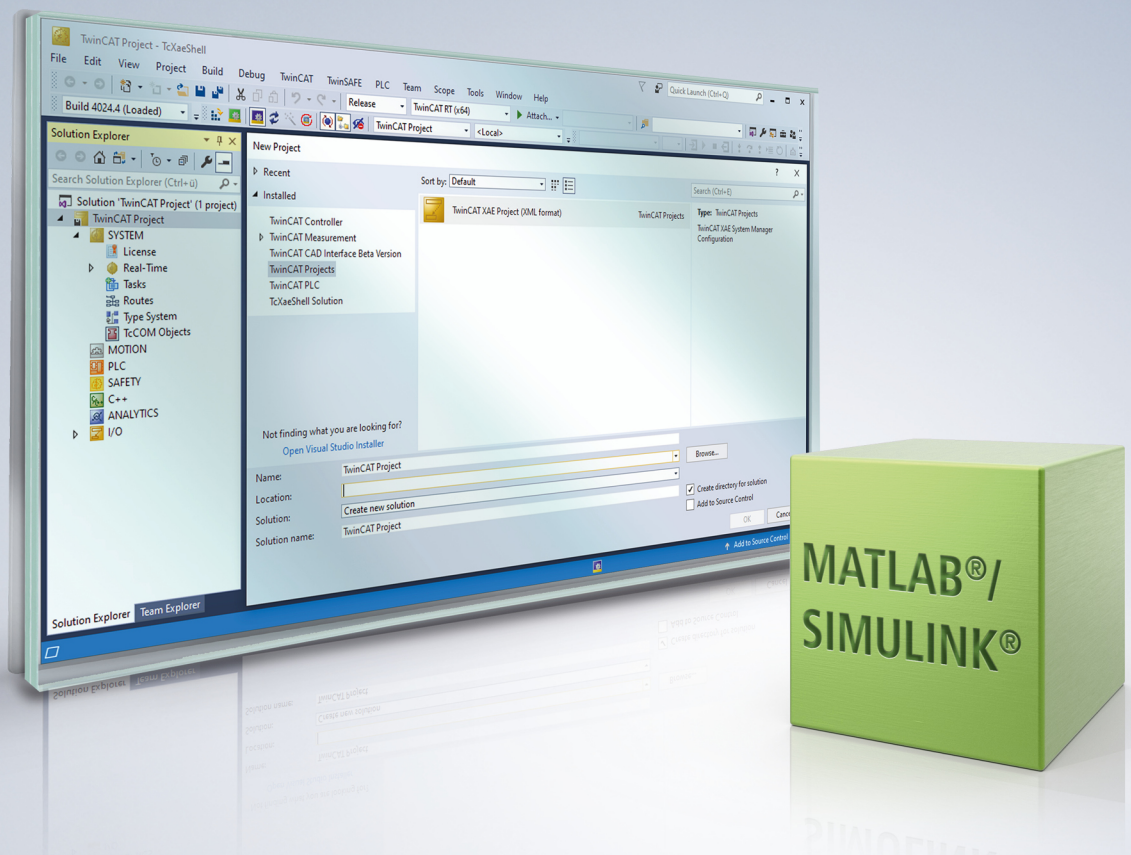
BECKHOFF New Automation Technology

手册 | ZH

# TE1400

## TwinCAT 3 | Target for Simulink®




2025-03-13 | 版本： 2.3.1

# 目录

# 1 前言

## 1.1 文档说明

本说明仅适用于熟悉国家标准且经过培训的控制和自动化工程专家。
在安装和调试组件时，必须遵循文档和以下说明及解释。
操作人员应具备相关资质，并始终使用最新的生效文档。

相关负责人员必须确保所述产品的应用或使用符合所有安全要求，包括所有相关法律、法规、准则和标准。

### 免责声明

本文档经过精心准备。然而，所述产品正在不断开发中。
我们保留随时修改和更改本文档的权利，恕不另行通知。
不得依据本文档中的数据、图表和说明对已供货产品的修改提出赔偿。

### 商标

Beckhoff®、TwinCAT®、TwinCAT/BSD®、TC/BSD®、EtherCAT®、EtherCAT G®、EtherCAT G10®、EtherCAT P®、Safety over EtherCAT®、TwinSAFE®、XFC®、XTS® 和 XPlanar® 是德国倍福自动化有限公司的注册商标并由其授权使用。
本出版物中所使用的其它名称可能是商标名称，任何第三方出于其自身目的使用它们可能会侵犯商标所有者的权利。

EtherCAT® 是注册商标和专利技术，由德国倍福自动化有限公司授权使用

### 版权所有



### 第三方商标

本文档可能使用了第三方商标。有关商标信息，可以访问：https://www.beckhoff.com/trademarks。

## 1.2 安全信息

### 安全规范

为了确保您的使用安全，请务必仔细阅读
并遵守本文档中每个产品的安全使用说明。

### 责任免除

所有组件在供货时都配有适合应用的特定硬件和软件配置。严禁未按文档所述修改硬件或软件配置，否则，德国倍福自动化有限公司对由此产生的后果不承担责任。

### 人员资格

本说明仅供熟悉适用国家标准的控制、自动化和驱动工程专家使用。

### 警示性词语

文档中使用的警示信号词分类如下。为避免人身伤害和财产损失，请阅读并遵守安全和警告注意事项。

### 人身伤害警告

| ⚠ 危险 |
| --- |
| 存在死亡或重伤的高度风险。 |

| ⚠ 警告 |
| --- |
| 存在死亡或重伤的中度风险。 |

| ⚠ 谨慎 |
| --- |
| 存在可能导致中度或轻度伤害的低度风险。 |

### 财产或环境损害警告

| *注意* |
| --- |
| 可能会损坏环境、设备或数据。 |

### 操作产品的信息

这些信息包括：
有关产品的操作、帮助或进一步信息的建议。

## 1.3 信息安全说明

Beckhoff Automation GmbH & Co.KG (简称 Beckhoff) 的产品，只要可以在线访问，都配备了安全功能，支持工厂、系统、机器和网络的安全运行。尽管配备了安全功能，但为了保护相应的工厂、系统、机器和网络免受网络威胁，必须建立、实施和不断更新整个操作安全概念。Beckhoff 所销售的产品只是整个安全概念的一部分。客户有责任防止第三方未经授权访问其设备、系统、机器和网络。它们只有在采取了适当的保护措施的情况下，方可与公司网络或互联网连接。

此外，还应遵守 Beckhoff 关于采取适当保护措施的建议。关于信息安全和工业安全的更多信息，请访问本公司网站 https://www.beckhoff.com/secguide。

Beckhoff 的产品和解决方案持续进行改进。这也适用于安全功能。鉴于持续进行改进，Beckhoff 明确建议始终保持产品的最新状态，并在产品更新可用后马上进行安装。使用过时的或不支持的产品版本可能会增加网络威胁的风险。

如需了解 Beckhoff 产品信息安全的信息，请订阅 https://www.beckhoff.com/secinfo 上的 RSS 源。

## 1.4 文档发行状态

| 版本 | 修订 |
|---|---|
| 2.3.x | 通过 软件保护 UI 界面设置驱动程序签名，请参阅：设置驱动程序签名。 |
| 2.2.x | TwinCAT 3.1 Build 4026<br>• 安装 [▶ 12]简介<br>• 对系统要求信息进行更新 |
| | • **新增：**对于初始化时的默认参数和 MATLAB 路径进行设置 [▶ 14]<br>• “设置驱动程序签名"部分的更新 |
| | 将 Simulink® 模型转换为 TwinCAT 对象：新示例 [▶ 21]（参见本章末尾），以进一步深化您的知识体系 |
| | 修订/扩展的配置参数 [▶ 43]表 |
| | **新增：**针对不同的平台进行生成 [▶ 54] |
| | **新增：**自动更新 TwinCAT 项目中的对象 [▶ 59] |
| | TcCOM 在线更改 [▶ 65]部分的修订 |
| | **新增：** 指令集扩展 [▶ 90] |
| | 修订/扩展的可用占位符表 [▶ 97] |
| | **新增：**部署后回调函数，参见使用回调参数 [▶ 108]。 |
| | 从 PLC 引用静态模块实例或进行动态实例化和引用的新代码示例，参见 应用 TcCOM Wrapper FB [▶ 134]。 |
| | **新增：**Simulink® Coder™ 可以将 Simulink® 字符串转换为数据类型 std::string，详情请参见常见问题解答/ 使用 Simulink® 字符串 [▶ 160]。 |
| | 关于"无法加载 DataExchange 模块 [▶ 163]"主题的常见问题解答/故障排除。 |

# 2 中文版手册的适用范围有限

本中文版手册可能与英文版内容不完全一致。中英文版本内容如有差异或不一致，请以英文版本为准。

# 3 概述

## TE1400 TwinCAT Target for Simulink®

使用 TwinCAT 3 Target for Simulink® 可以将在 Simulink® 中开发的模型应用于 TwinCAT 3 中。Simulink® 中也可以集成 SimScape™、Stateflow™ 或 DSP System Toolbox™ 等工具箱。它还支持嵌入式 MATLAB® 功能块。这些模型通过 Simulink Coder™ 自动转换成 C/C++ 代码，并可通过使用 TwinCAT 3 Target for Simulink® 转换为 TwinCAT objects。随后，这些 TwinCAT objects 可以在 TwinCAT Runtime 中的实时环境下执行。这些 TwinCAT objects 可以作为 TcCOM 对象直接实例化并连接实时任务，也可以作为功能块，在 PLC 项目中进行实例化和处理。

### 应用领域和应用举例

TwinCAT Target for Simulink® 的应用领域可以用以下关键词来概括：

- 快速控制原型
- 实时仿真
- SiL（**S**oftware**i**n the**L**oop，软件在环）仿真
- HiL（**H**ardware**i**n the**L**oop，硬件在环）仿真
- 基于模型进行设计
- 基于模型进行监控

以下应用举例旨在说明可以应用的领域：

- **示例 1：快速控制原型**
  在 Simulink® 的仿真开发阶段，控制器可以由 Simulink® 模型来实现构建，并通过*模型引用*集成到环路控制的仿真模型中。这样可以在仿真中设计和测试闭环控制（**M**ode**l**in the**L**oop simulation，模型在环仿真（MiL））。在通过鼠标单击将原本结构的控制器模型无需修改直接编译成 TwinCAT module之前，该模块作为实际系统的实时控制器运行。由于标准 Simulink® 功能块用作输入和输出，因此可用于上层 Simulink® 模型，并随后在 TwinCAT 中生成的模块中使用。
- **示例 1a：受控系统的实时仿真**
  受控系统也可以作为 Simulink® 模型实现，并通过*模型引用*集成到闭环控制的模型中。由此生成的 TcCOM 模块用于执行实时仿真，在仿真中可以测试以 IEC61131-3、C++ 或 Simulink® 模型构建的控制器。
- **示例 2：设备的实时仿真/虚拟调试**
  TcCOM 模块由在 Simulink® 中创建的设备模型生成。它可用于在连接实际设备之前实时测试 PLC 程序（虚拟调试）。视配置的不同，可以通过这种方式进行 SiL 或 HiL 仿真。另请参见。
- **示例 2a：设备组件的 SiL 仿真**
  根据 VDI/VDE 3693 第 1 部分，软件在环（SiL）被定义为 MiL 仿真之后的一个阶段，控制代码在该阶段可作为系列代码使用。系列代码可以在仿真控制器中执行，用于根据系统仿真模型进行测试。
  根据这一定义，使用 TwinCAT 对系统（组件）进行 SiL 仿真有两种选择：
  - 系统模型保留在 Simulink® 中，并使用 ADS 与在 TwinCAT Runtime 中执行的系列代码进行通信。另请参见。
  - 系统模型还被编译成 TcCOM 模块并实时执行（见示例 1a）。
- **示例 2b：系统组件的 HiL 仿真**
  根据 VDI/VDE 3693 第 1 部分，硬件在环（HiL）被定义为高级测试阶段，在该阶段，将实际目标控制代码运行在实际控制器上针对系统模型进行测试。系统模型在仿真工具中运行，并且以总线设备的状态进行运用，进而实现仿真工具使用自动化系统的实际通信网络与实际控制器进行通信。
  根据这一定义，系统模型或系统组件被转换为 TcCOM 模块，并在第二台工业 PC 上执行，同时兼顾实时性的要求。此功能用于配置该工业 PC，以便向实际控制器提供镜像的操作。这样，就可以使用实际控制器和实际配置与“仿真工业 PC”进行硬实时通信。
- **示例 3：基于模型监控系统组件**
  在许多情况下，所测量的变量无法直接获得，或者需要投入大量精力/成本才能获得。通过使用包含可测量输入变量且具有物理意义的模型，仍可用于确定不可测量的变量。例如，测量电机永磁体等无法获得的位置的温度。根据电机热模型，可通过电流、转速和冷却温度等辅助参数计算温度。

**更多信息**

**技术视频短片**

- TwinCAT Target for Simulink

**产品说明**

- https://www.beckhoff.com/TE1400

**客户应用视频**

- 客户应用概述
- Vintecc 有限公司成功案例
- Magway 的成功故事

**使用 TwinCAT 3 的 MATLAB® 和 Simulink®：** http://www.beckhoff.com/matlab

# 4 2.x.xxxx.x 及以上版本

- TE1400 Target for Simulink® 1.2.xxxx.x 以下版本支持 MATLAB R2010b 至 MATLAB R2019a。
- TE1400 Target for Simulink® 2.x.xxxx.x 以上版本支持 MATLAB R2019a 及更高版本。
- 两个版本可安装在同一个开发系统中并行使用。
- 已创建模块的兼容性：参见重新加载 TMI/TMC 时映射丢失 [▶ 156]。

## 4.1 安装

**系统要求**

下文介绍了开发用 PC 和运行时 PC 的区别。使用以下定义：

在**开发用 PC** 上，Simulink® 模型可通过使用 TwinCAT Target for Simulink® 转换为 TwinCAT 对象。所创建的 TwinCAT 对象可以在 TwinCAT 项目中实现实例化，从而集成到程序流中。

**可轻松转发生成的对象**

在开发用 PC（或 Build Server）上构建的 TwinCAT objects 可以轻松转发给其他人。他们只需要 TwinCAT XAE 开发环境就可以在 TwinCAT 解决方案中使用创建的对象（TcCOM 或 PLC 功能块）。

然后，将创建的 TwinCAT 解决方案从开发用 PC 加载到具有TwinCAT 运行时环境中的**运行时 PC** 中，用于执行项目。

**在开发用 PC 上的要求**

- MATLAB® R2019a 或更高版本，以及至少安装 Simulink® 和 Simulink® Coder™
- Visual Studio 2017 或更高版本（Professional、Ultimate 或同等版本）
  - 在安装过程中，必须手动选择 *Desktop development with C++* 选项。该选项也可以稍后安装。
  - 为支持 TwinCAT OS (ARMV8-A) 平台（用于 CX82xx 和 CX9240 的编译），还需要使用 MSBuild Support for llvm (clang-cl) 工具集组件。此项同样需要手动选择。
- TwinCAT 3.1.4024.35 或更高版本
- TwinCAT Target for Simulink®
  - TwinCAT-build 4024：使用倍福网站上的下载查找器，并安装用于 MATLAB® 和 Simulink® 的 TwinCAT 工具
  - TwinCAT-build 4026：为您安装相应的下载项（见下段）。

**在运行时 PC 上**

- 支持的操作系统
  - Windows Embedded Standard 7、Windows 10、Windows Server（32 位和64 位）
  - Beckhoff RT Linux®（仅适用于 TwinCAT Build 4026 以上版本）
  - TwinCAT/BSD®
- TwinCAT XAR 3.1.4024.35 或更高版本

**安装（TwinCAT 3.1 Build 4026）**

TE1400 | TwinCAT 3 Target for Simulink® 下载项安装了创建 TwinCAT 对象所需的所有依赖项。

- 如果尚未安装，请安装支持的 **Visual Studio** 版本之一。注意安装 *Desktop development with C++ 选项*。
- 安装 TwinCAT Package Manager。
- 安装以下下载项：
  - 用户界面中的名称：TE1400 | TwinCAT 3 Target for Simulink®
    命令行：`tcpkg install TE1400.TargetForSimulink.XAE`

- 在 Visual Studio 中执行集成，其中已安装 *Desktop development with C++* 选项。
- 以管理员身份启动 MATLAB® 并在 MATLAB® 中执行 *C:\Program Files (x86)\Beckhoff\TwinCAT\Functions\TE14xx-ToolsForMatlabAndSimulink\SetupTE14xx.p*。

如果您想创建 TwinCAT 项目并使用已经构建好的 TwinCAT 对象，请安装以下下载项：

- TwinCAT 标准版
  `tcpkg install TwinCAT.Standard.XAE`
- TwinCAT 经典框图
  `tcpkg install TwinCAT.XAE.BlockDiagramClassic`
- 在某些情况下，您还需要使用以下软件包：
  `tcpkg install TwinCAT.XAE.TMX.DataExchange`
  （请参见 DataExchangeModules [▶ 59]）
  `tcpkg install TwinCAT.XAE.TMX.MatSimUtilities`
  （用于将 ITcVnImage 转换为 ARRAY）

**安装（TwinCAT 3.1 Build 4024）**

✓ 如果尚未安装，请安装支持的 **Visual Studio** 版本之一。注意安装 *Desktop development with C++* 选项。

1. 启动 **TwinCAT 3 XAE 或 Full Setup**（如果尚未安装）。
   如果已经安装 Visual Studio 和 TwinCAT，但 Visual Studio 版本不符合上述要求（例如 TwinCAT XAE Shell 或不带 C++ 选项的 Visual Studio），则必须首先安装合适的 Visual Studio 版本（必要时安装 C++ 选项）。然后运行 TwinCAT 3 设置，将 TwinCAT 3 集成到新的（或修改后的）Visual Studio 版本中。
2. 如果系统中尚未安装 **MATLAB®**，请安装。MATLAB® 的安装顺序无关紧要。
3. 启动 **TwinCAT Tools for MATLAB® 和 Simulink®** 设置以安装 TE1400。
   ⇨ TE1400 安装在 TwinCAT 文件夹结构中，即与 MATLAB® 安装相互独立。
4. 以管理员身份启动 MATLAB® 并运行 *%TwinCAT3Dir%。\Functions\TE14xx-ToolsForMatlabAndSimulink\SetupTE14xx.p* in MATLAB®。

## 4.2 许可证

使用 TE1400 TwinCAT Target for Simulink® 的完整功能需要两个许可证。一个是用于从 Simulink® 创建 TwinCAT objects 的 TE1400 开发环境许可证；另一个是用于在 TwinCAT 运行时期间执行这些对象的运行时许可证。

**开发环境许可证**

需要 *TE1400 Target for Simulink®* 许可证（对于**开发系统**），以便从 Simulink® 创建 TcCOM 和 PLC 功能块。如果是出于测试目的，产品可以无需许可证，作为演示版本在演示模式下使用。

本产品不提供完整功能的 7 天试用许可证。

**演示版本的限制**

在没有有效 TE1400 许可证情况下，模型的要求如下：

- Simulink Coder™ 的所有 cpp 和头文件（包括相关的库）总大小不得超过 100 kB。
- 至多 5 个输入信号
- 至多 5 个输出信号

使用演示许可证创建的模块只能用于非商业用途!

**运行时许可证**

如果要启动的 TwinCAT 配置包含一个或多个由 Simulink® 生成的 TwinCAT objects，则需要运行时许可证。

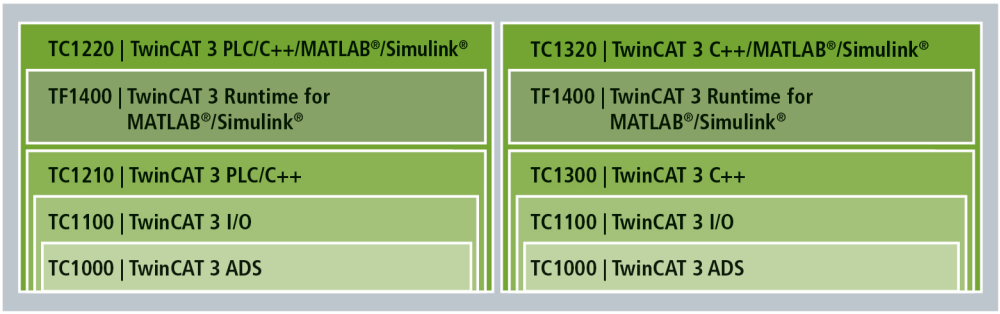
执行所需的许可证包括：
TF1400 TwinCAT 3 Runtime for MATLAB®/Simulink® **和** TC1300 TwinCAT 3 C++。这两种许可证都包含在 TC1320 和 TC1220 **许可证绑定包**中。TC1320 绑定 TC1300 和 TF1400许可证（C++ 和 MATLAB®/Simulink® Runtime）。TC1220 绑定 TC1200、TC1300 和 TF1400 许可证（PLC、C++ 和 MATLAB®/Simulink® Runtime）。

**TF1400 许可证仅适用于 TwinCAT 3.1.4026.0 及以上版本**

仅 TwinCAT 3.1.4026.0 及以上版本支持 TF1400 许可证。更早的版本只能使用许可证捆绑包 TC1220 或 TC1320。Build 4026 及更高版本仍支持许可证捆绑包。早期版本只能使用 TC1220 或 TC1320 许可证捆绑包。只有当目标系统已获得 TwinCAT 3 C++ 许可证时，才需要 TF1400 许可证。

如果没有激活的许可证，则无法启动模块，进而无法启动 TwinCAT 系统。

您可以为已命名的运行时许可证生成 7 天试用许可证，这样就可以在不购买许可证的情况下进行初始测试。




# 4.3 软件初始设置

## 4.3.1 对一些默认值和 MATLAB® 路径进行设置

**软件初始设置**

**设置 MATLAB® 路径**

如安装 [▶ 12]部分所述，软件安装后必须执行 SetupTE14xx.p 文件。该脚本会将必要的路径添加到 MATLAB® 路径中。保存 MATLAB® 路径需要 MATLAB® 管理员权限。

**每次产品升级后额外运行 SetupTE14xx.p**

为确保在 MATLAB® 路径中设置所有路径，另请在更新产品后运行 p 文件。

**设置默认设置**

该脚本还会打开一个对话框，您可以在其中保存 TwinCAT Target 的常规默认设置。这些设置适用于整个系统，即*所有*安装的 MATLAB® 版本。

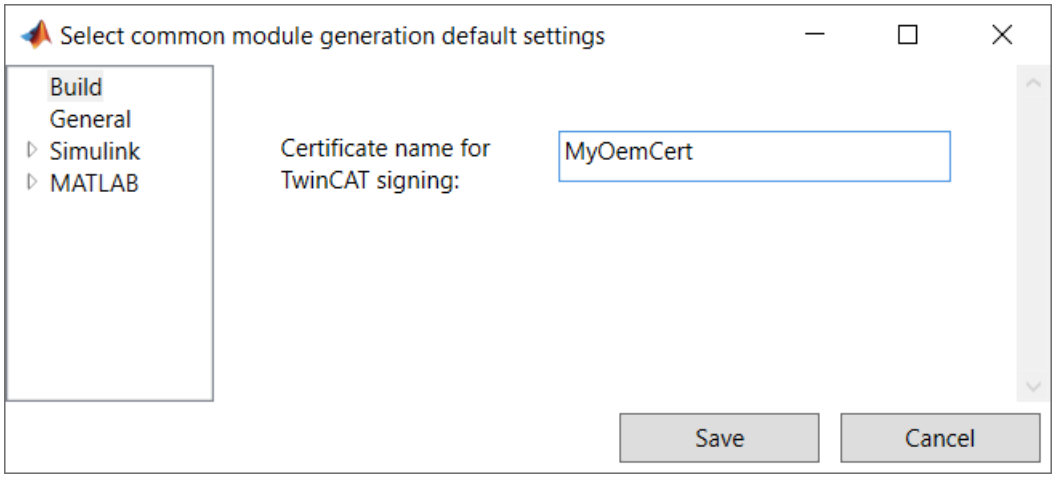
如果想不通过该对话框执行 p 文件，可以使用以下命令：

```
SetupTE14xx('Silent', true);
```

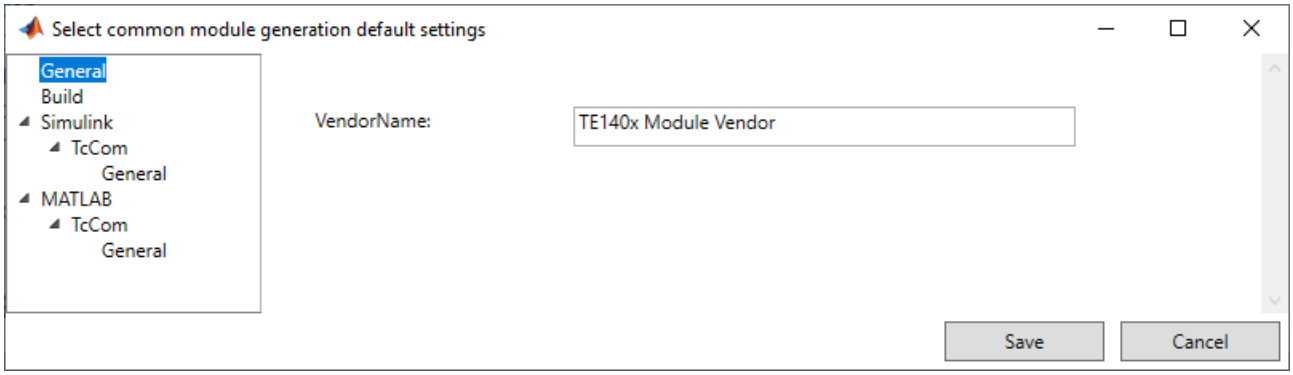
设置对话框的默认值。

可以此时进行设置，也可以稍后再进行设置/更改。

在**构建**（Build）选项卡下的对话框中，可以存储用于驱动程序签名的默认证书。在设置驱动程序签名 [▶ 17]一章中详细说明驱动程序签名的设置选项。




生成的 TwinCAT 对象的层次结构也会受到对话框的影响。

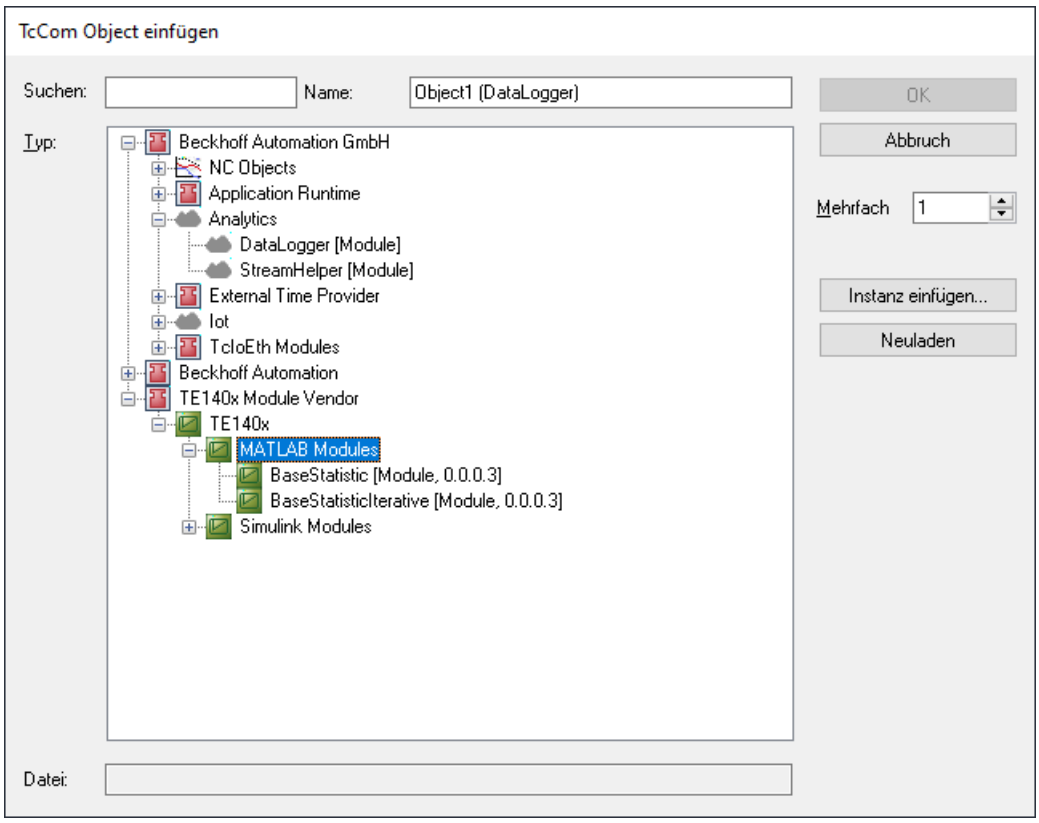



对话框中提供以下设置选项：

- *VendorName*
- *GroupName*（MATLAB®）和
- *GroupName*（Simulink®）

在 TwinCAT 中通过以下示例显示层次结构：

- VendorName “TE140x Module Vendor”
- GroupName “TE140x”
  - “MATLAB Modules” for MATLAB® 和
  - “Simulink Modules” for Simulink®

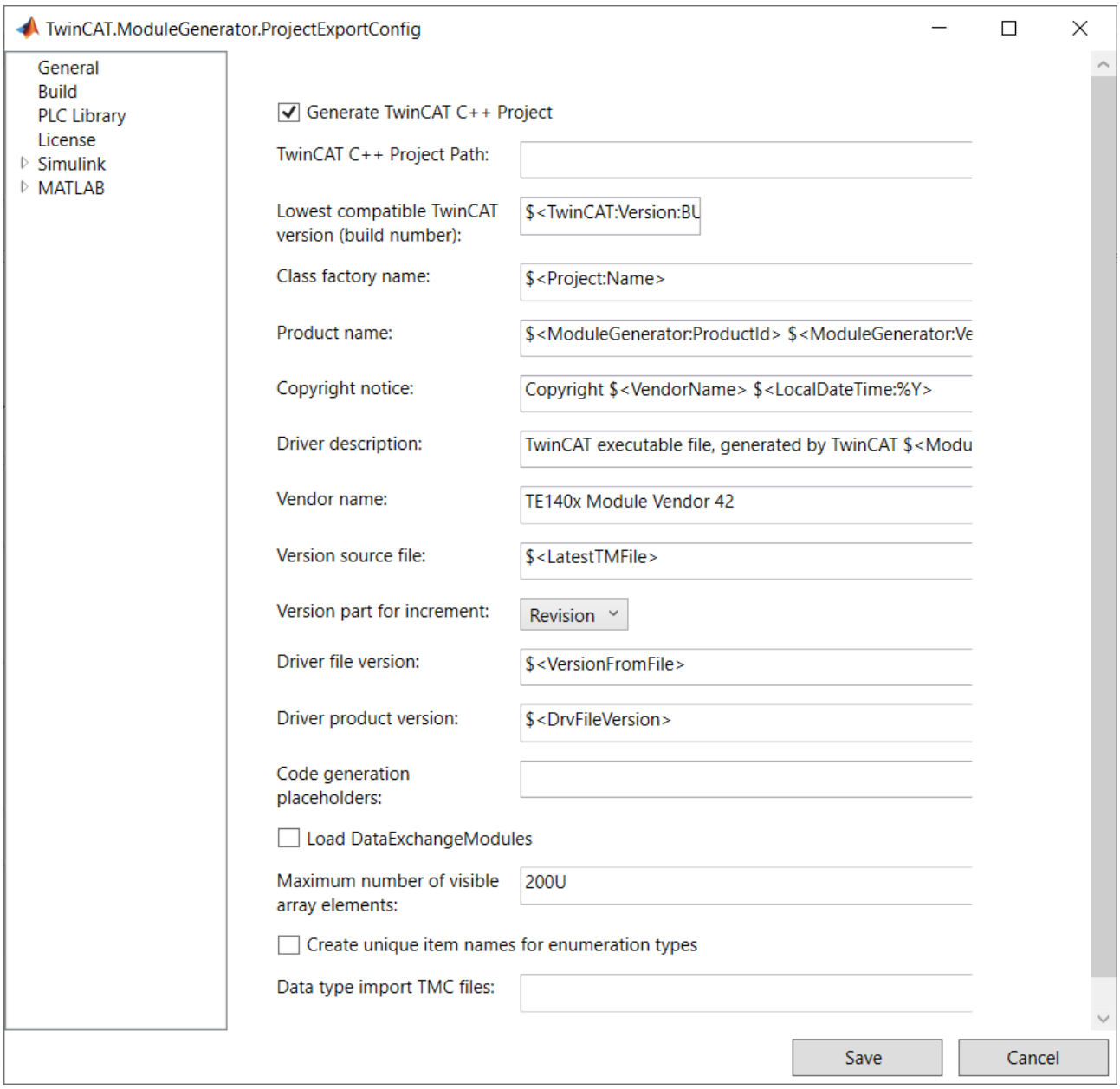



**更改软件设置**

要更改 TwinCAT Target 的默认设置，可以访问 MATLAB® 控制台中的对话框，如下所示：

```
TwinCAT.ModuleGenerator.Settings.Edit
```

这里提供了各种您可以将其存储为默认值条目。




**应用更改**

1. 在对话框中输入新的默认设置。
2. 使用**保存**（Save）按钮确认。
3. 重新启动 MATLAB®。

⇨ 更改即被应用。

## 4.3.2 设置驱动程序签名

**创建 OEM 2 级证书**

从 MATLAB® 或 Simulink® 生成的 TwinCAT 对象与 TwinCAT C++ 对象一样，都基于 tmx 驱动程序（TwinCAT Module Executable）。这些驱动程序必须使用 OEM 2 级证书进行签名，才能在 TwinCAT 开发环境加载到运行时 PC 上。

关于如何创建 OEM 证书来对驱动程序进行签名，请参见以下链接中的详细文档资料：

- 关于 OEM 证书的一般文档
- tmx 驱动程序签名的应用相关文档

**重点总结如下：**

- 创建自己的证书。为此，进入 Visual Studio，点击
  **菜单栏 > TwinCAT > 软件保护.....**（Software Protection...）
  - 您需要 OEM 加密版本 2 证书（选项：*签名 TwinCAT C++ 可执行文件 (*.tmx)*）。
  - 系统会提示您为证书创建密码。
- 创建驱动程序时可以先不签名，随后再签名。
- 在开发阶段出于测试目的，可以使用没有会签的证书。
- 中国地区需要从倍福订购会签证书（TC0008）。

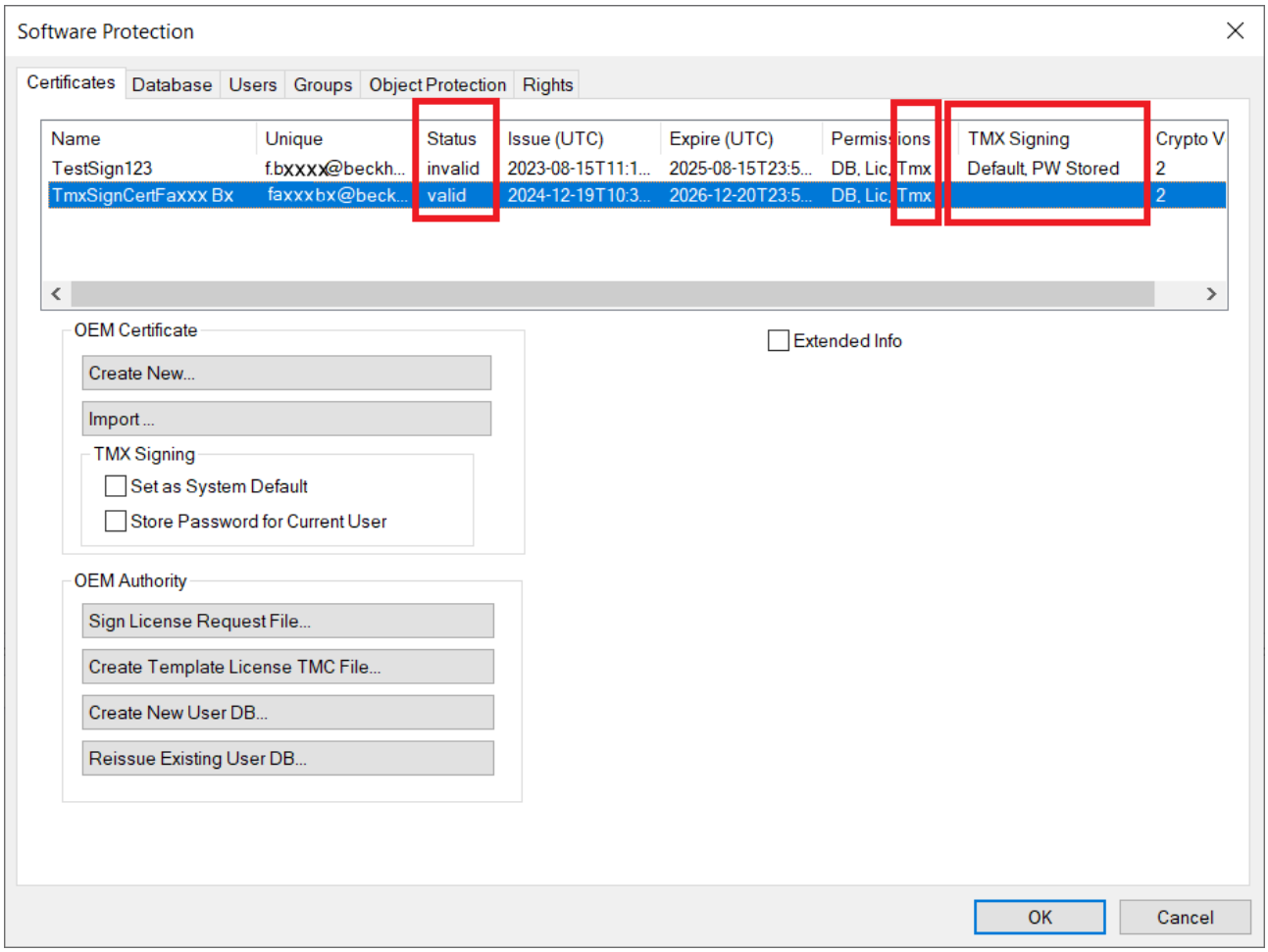
**在软件保护下设置 OEM 2 级证书**

**TwinCAT Build 4026：对设置对话框的要求**

以下有关软件保护的信息仅适用于 TwinCAT 3.1 Build 4026。至少需要 TwinCAT Standard 4026.14 下载项。如果您使用的是旧版本，请继续阅读下一节中的“设置 OEM 2 级证书以在没有软件保护对话框的情况下进行驱动程序签名”。

在软件保护界面（**菜单栏 > TwinCAT > 软件保护.....**（Software Protection...））中，您既可以创建证书（创建新证书.....（Create New...）），也可以：

- 将证书设置为用于签署 tmx 文件的全系统默认证书（可选）。
- 为每个证书存储登入 Windows 用户的相应密码（必填）。




**上述概述包含两个证书示例。**

第一个证书“TestSign123”未经倍福会签，因此在状态中被归类为*无效*。未经会签的证书仍可用于签名。然后必须将目标系统设置为测试模式 - 参见TwinCAT Runtime 的行为 [▶ 20]章节。另一方面，“TmxSignCertFaxxxBx”证书经过会签，因此被归类为*有效*证书。这两种证书都适用于签名 tmx 文件，请参见“*权限*”部分。在“TMX 签名”栏中，“默认”表示证书是否被设为全系统默认证书。注释“PW 已存储”表示证书的密码可供登入的 Windows 用户使用/存储。

**将证书设置为全系统默认证书（可选）**

您可以在工程 PC 上设置默认证书，该证书始终用于 TwinCAT C++、Target for MATLAB®、Target for Simulink® 等，除非您明确指定不同的证书。

从*软件保护*（Software Protection）对话框的列表中选择要用作默认证书的证书，并选择“设置为系统默认值”（Set as System Default）复选框。

然后会创建一个名称为 *TcSignTwinCatCertName* 的环境变量。在 Windows 中，在进程启动时会告知环境变量。因此，如果已经在运行该进程，则请重新启动 MATLAB®。

本章稍后将介绍使用证书的更多选项。

**存储证书密码（必填）**

出于安全考虑，不得在 Simulink® 模型的项目或源代码或 MATLAB® 代码中输入证书的密码。通过“为当前用户存储密码”（Store Password for Current User），可以在系统中存储证书的相应密码。

密码以混淆方式存储在 Windows 操作系统的注册表中。这意味着特定证书的密码在操作系统中（对于当前用户）是已知的，并且会自动使用。

在*软件保护*（Software Protection）对话框中选择要存储相应密码的证书。选择“为当前用户存储密码”（Store Password for Current User）。系统会要求您输入密码。如果已成功检查并输入，则会在“TMX 签名”（TMX Signing）下显示注释“PW 已存储”。

存储密码的**替代方式**是使用命令提示符以及 TcSignTool（C:\Program Files (x86)\Beckhoff\TwinCAT\3.1\SDK\Bin）。

密码通过以下调用存储：

```
tcsigntool grant /f "C:\TwinCAT\3.1\CustomConfig\Certificates\MyCertificate.tccert" /p MyPassword
```

经过混淆处理的密码在注册表中存储于以下项下：

```
HKEY_CURRENT_USER\SOFTWARE\Beckhoff\TcSignTool\
```

通过以下调用删除密码：

```
tcsigntool grant /f "C:\TwinCAT\3.1\CustomConfig\Certificates\MyCertificate.tccert" /r
```

**在没有软件保护对话框的情况下，为驱动程序签名设置 OEM 2 级证书**

要对 tmx 文件进行签名，需要证书和与证书相关的密码。

可用证书可在以下网址找到：

Build 4026: C:\ProgramData\Beckhoff\TwinCAT\3.1\CustomConfig\Certificates
Build 4024: C:\TwinCAT\3.1\CustomConfig\Certificates

**证书的处理**

对 tmx 驱动程序进行签名有四种不同的方式。

**第 1 种方式：用于 TwinCAT C++ 和 TE14xx 的全系统默认证书**

该方式与通过“软件保护”（Software Protection）>“设置为系统默认设置”（Set as System Default）的路径相同。

或者，您也可以为该方法手动创建一个 Windows 环境变量。在**用户 > 变量**（User > Variables）下使用以下变量和值创建一个新的**环境变量**：

变量：*TcSignTwinCatCertName*

值：证书的完整路径

**第 2 种方式：用于 MATLAB® 的全系统默认证书**

您可以在 MATLAB® 环境中设置默认证书，除非明确指定不同的证书，否则该证书始终用于 Target for MATLAB® 和 Target for Simulink®（**而非** TwinCAT C++）。

通过 *TwinCAT.ModuleGenerator.Settings.Edit*（MATLAB® 命令行）打开“通用设置”（Common Settings）对话框，并在 **构建 > TwinCAT 签名证书名称**（build > Certificate name for TwinCAT signing）下输入所需的默认证书。该证书默认存储在用户目录中，系统上所有 MATLAB® 版本默认使用该证书。

**第 3 种方式：Simulink® 模型配置证书**

您可以为每次构建操作明确命名一个证书。对于第 3 种方式，您无需事先进行任何设置。在每个构建过程之前，您都可以为该构建过程明确定义自己选择的证书。

Target for Simulink®：**TC Build > 用于 TwinCAT 签名的证书**

Target for MATLAB®：属性`SignTwinCatCertName`

**第 4 种方式：在没有证书的情况下构建，稍后使用 TcSignTool 签名**

您可以在没有证书的情况下进行构建，然后使用 **TcSignTool** 进行签名。

TcSignTool 是命令行程序。例如，打开命令提示符并执行 `tcsigntool sign /?` 以显示帮助。可在以下位置找到该程序：

Build 4026: C:\Program Files (x86)\Beckhoff\TwinCAT\3.1\SDK\Bin

Build 4024: C:\TwinCAT\3.1\SDK\Bin

**从 MATLAB® 操作 TcSignTool**

在 MATLAB® 中，可使用 `system()` 或 ! 命令启动该工具。

请求为 TwinCAT 进行 tmx 驱动程序签名的示例：

```
TcSignTool sign /f "C:\TwinCAT\3.1\CustomConfig\Certificates\ MyCertificate.tccert" /p MyPassword
"C:\TwinCAT\3.1\Repository\TE140x Module Vendor\ModulName\0.0.0.1\TwinCAT RT (x64)\MyDriver.tmx"
```

**TwinCAT 运行时的行为**

如果在 TwinCAT 解决方案中使用的 TwinCAT object 是从 MATLAB® 或 Simulink® 使用签名的驱动程序创建，并使用**激活配置**（Activate Configuration）将该对象加载到目标系统上，则必须遵循以下规定：

**非会签证书的测试模式**

如果使用未会签的 OEM 证书进行签名，必须将目标系统设置为测试模式。为此，请在目标系统上以管理员身份运行以下命令：

```
bcdedit /set testsigning yes
```

如果使用的是会签的 OEM 证书，则无需此步骤。

**目标系统证书白名单**

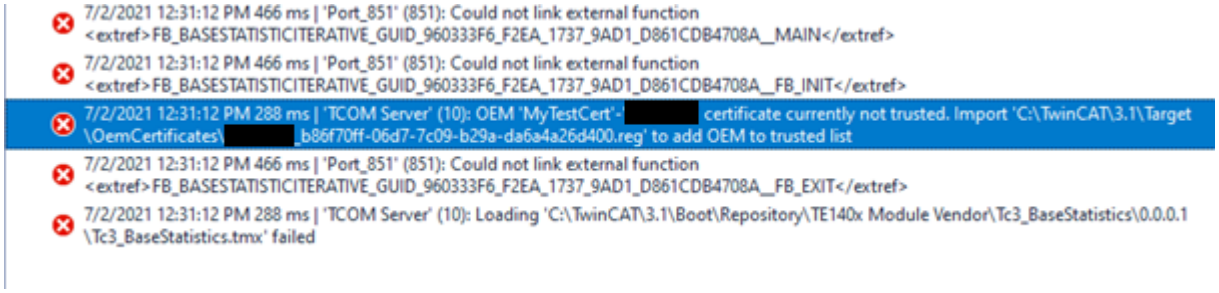
每个 TwinCAT 运行时 (XAR) 都有自己的受信任证书白名单。

**使用 TwinCAT Build 4026 时的行为**

TwinCAT-XAE 会检查激活配置所需的所有证书是否都在运行时系统的白名单中。如果不是这种情况，则会显示弹出窗口。您可以用它来设置白名单条目。

**使用 TwinCAT Build 4024 时的行为**

如果用于签名的证书不包含在此白名单中，则不会加载驱动程序。TwinCAT Engineering（XAE）中会输出相应的错误信息。




错误信息中包含以管理员身份在目标系统上执行注册表文件的指令，该文件是在目标系统上自动创建的。在该过程中，会将使用的证书添加到白名单中。

**注册表文件只取决于 OEM 证书**

注册表文件也可在其它目标系统上使用。它只包含所用 OEM 证书的相关信息，与目标系统无关。

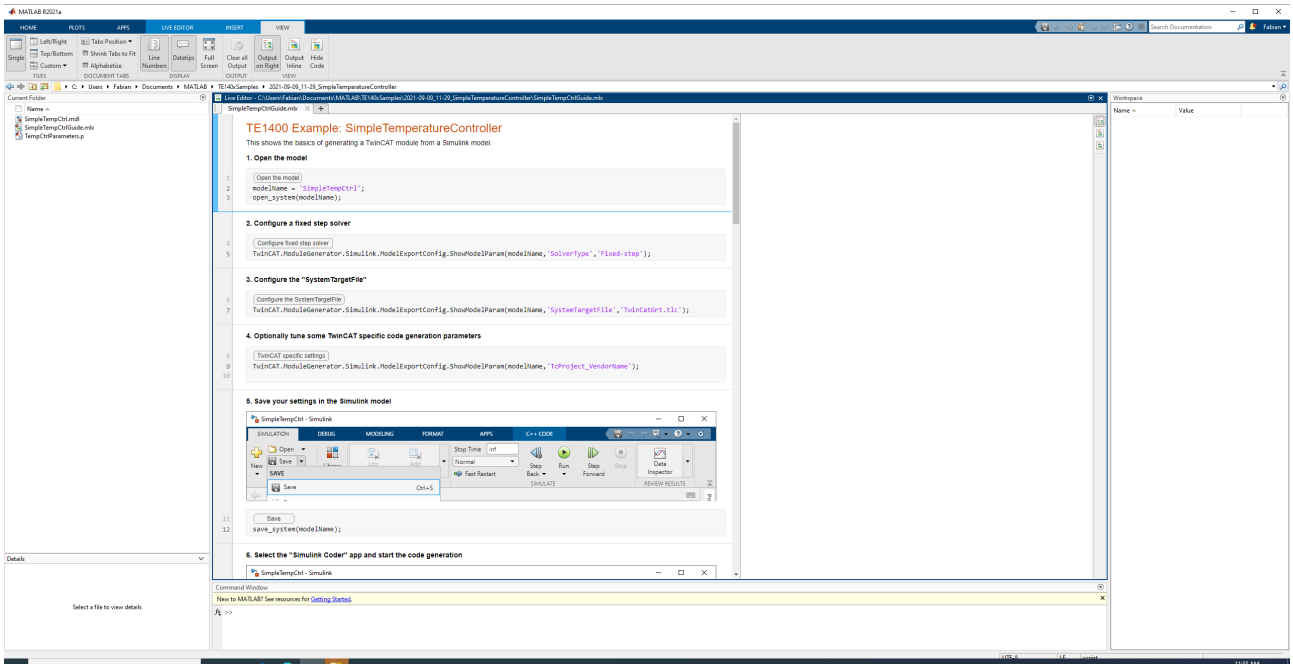
## 4.4 快速入门

下方快速入门会向您展示如何将 Simulink® 模型转换为 TwinCAT 对象。如果您对如何将创建的 TwinCAT 对象集成到 TwinCAT XAE 中感兴趣，请参见 亲自试用创建的 TwinCAT 对象 [▶ 167]。

### 从简单的 Simulink® 模型开始

✓ 请随意使用我们的内置示例作为使用 TwinCAT Target for Simulink® 的第一步。MATLAB® 命令窗口通过以下代码提供可用示例列表 `TwinCAT.ModuleGenerator.Samples.List`

1. 例如，选择示例“从 Simulink 模型生成 TwinCAT 类”，并通过命令窗口中的“开始”链接启动示例。`TwinCAT.ModuleGenerator.Samples.Show('Generate TwinCAT Classes From Simulink Models')`




⇨ 在下文中，将按照该示例执行快速启动。

2. 首先，在 Live Script 中选择**打开模型**（Open the model）按钮。

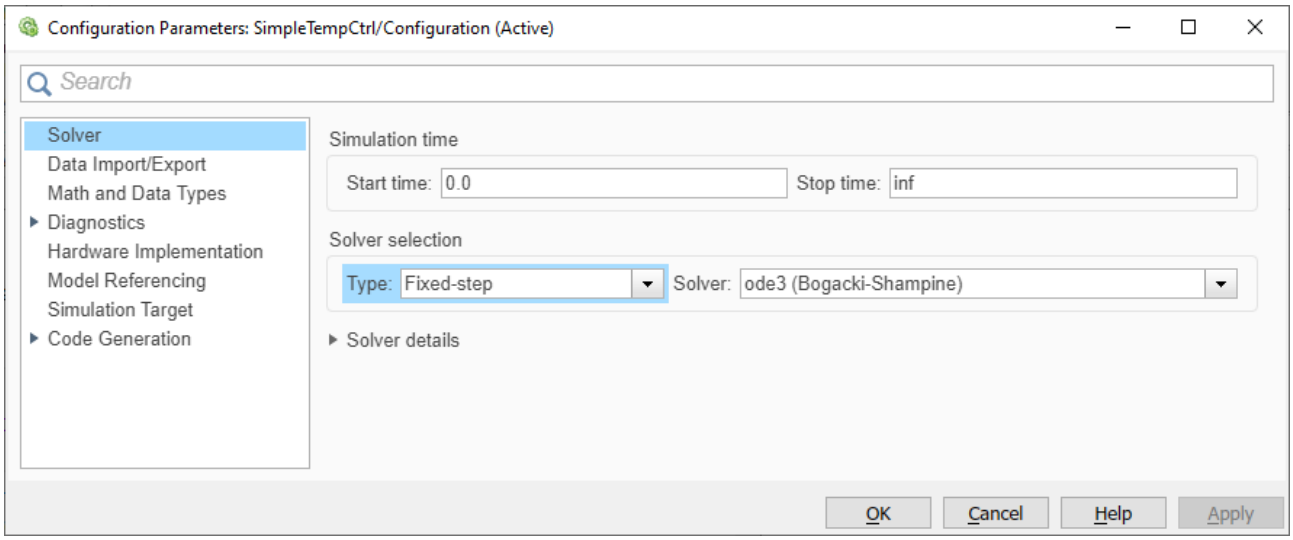
⇨ 只需点击 Live Script 中的下一步按钮，便可进入 Simulink® 接下来的配置步骤。

**面向初学者的视频**

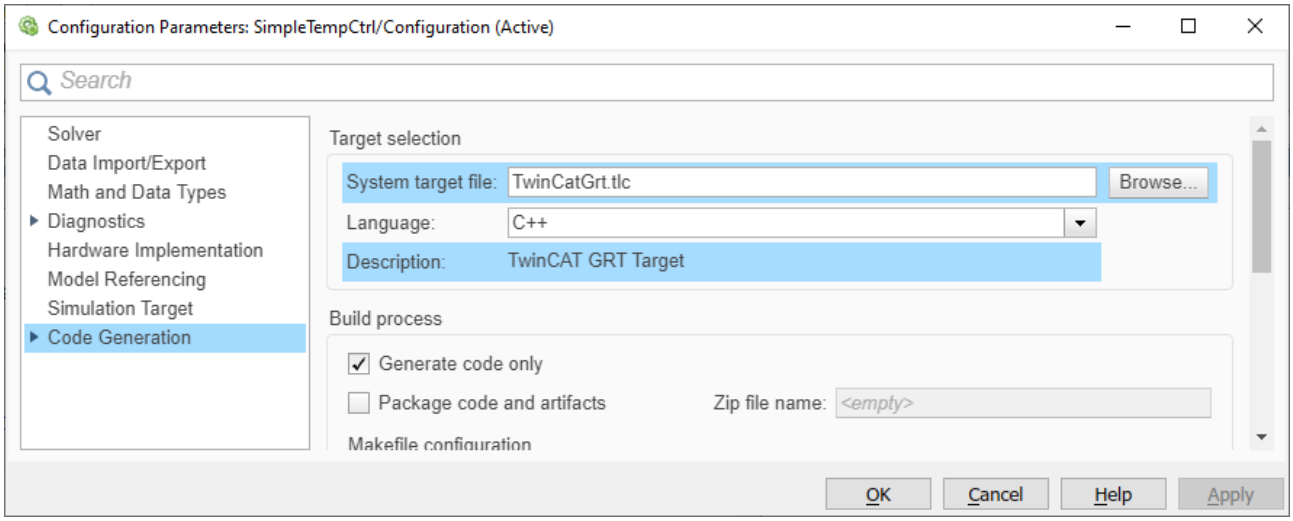
以下视频（仅提供英文版）也可用作介绍说明：TwinCAT Target for Simulink®

### Simulink® 中的配置步骤

1. 选择固定步长求解器。为此，进入模型的**配置参数**（Configuration Parameters）。

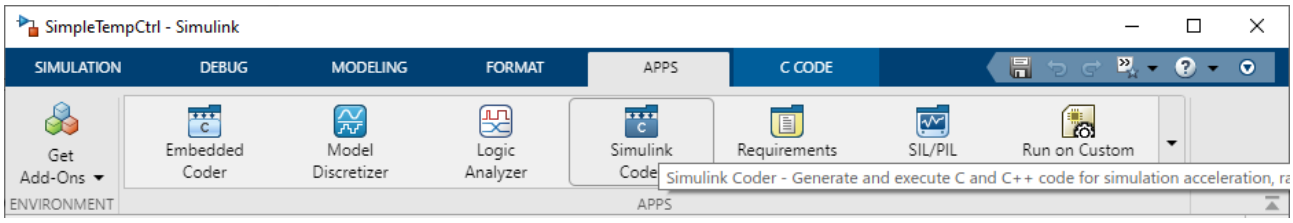


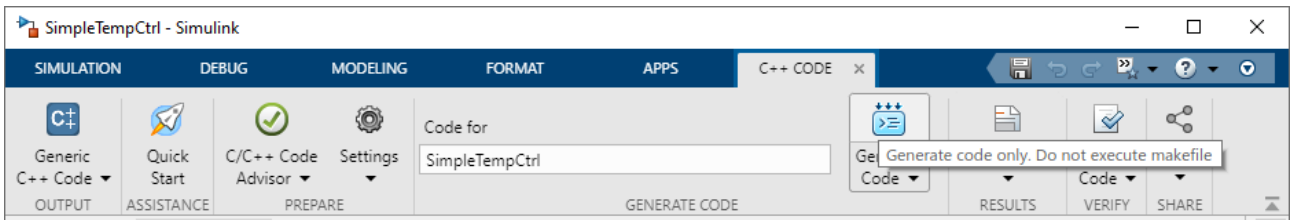

2. 选择系统目标文件为“TwinCatGrt.tlc”。




可选：在*优化*（Optimization）下，将参数*默认参数行为*（Default parameter behavior）设置为“可调”（Tunable），以便继续在 TwinCAT 中更改模型参数，另请见 模块实例的参数设置 [▶ 111]。

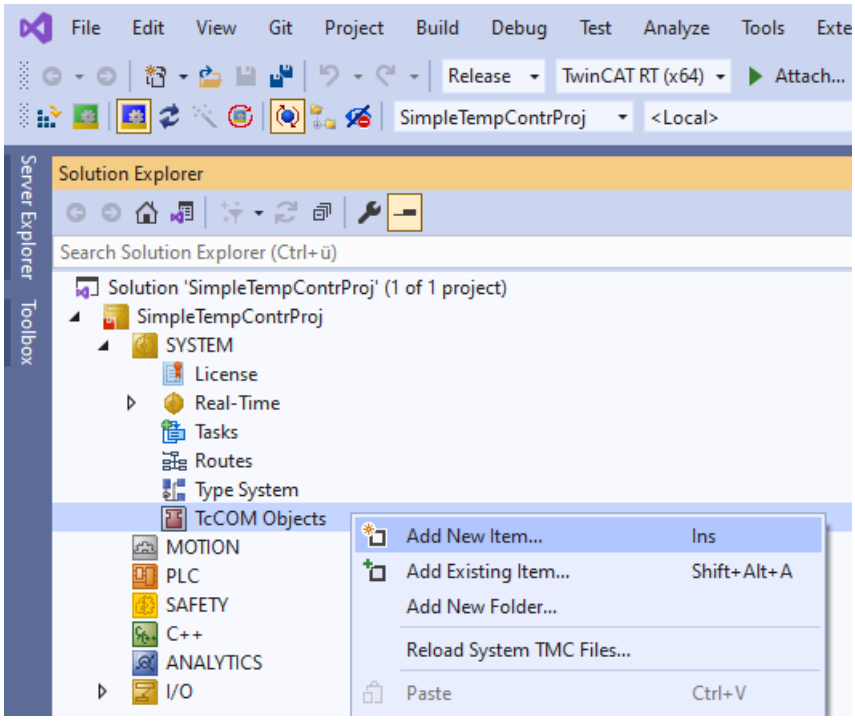
3. 在 Simulink® 模型中保存更改。
4. 通过 Simulink® Coder™ 应用启动代码生成。



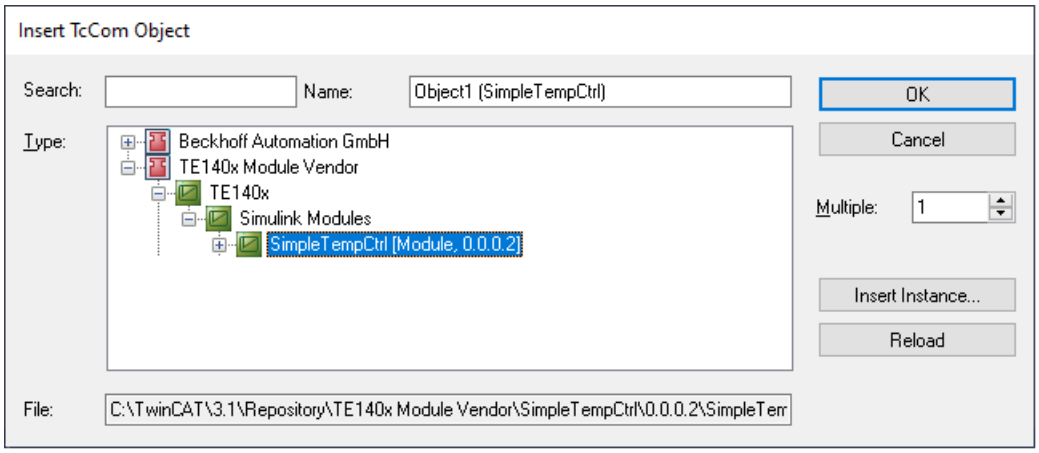



**在 TwinCAT 中添加 TcCOM**

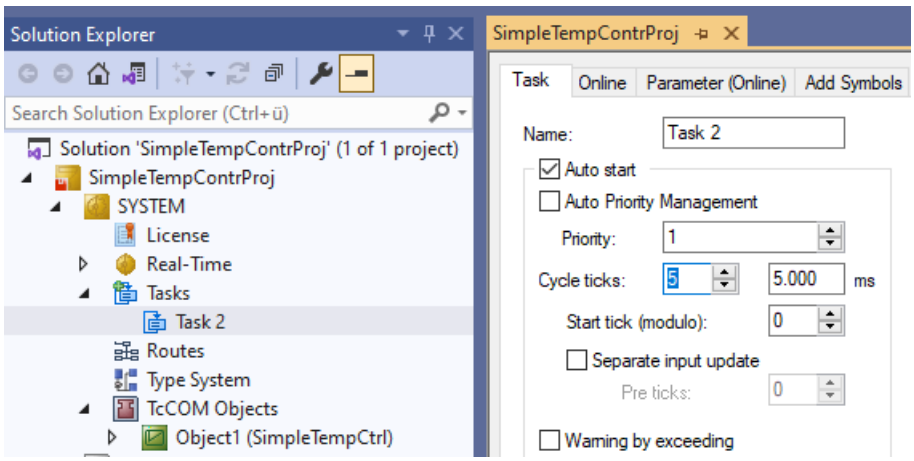
1. 打开 TwinCAT（TwinCAT XAE 或 Visual Studio 环境中的 TwinCAT）。
2. 实例化一个新的 TcCOM 对象。




3. 选择所需的对象。

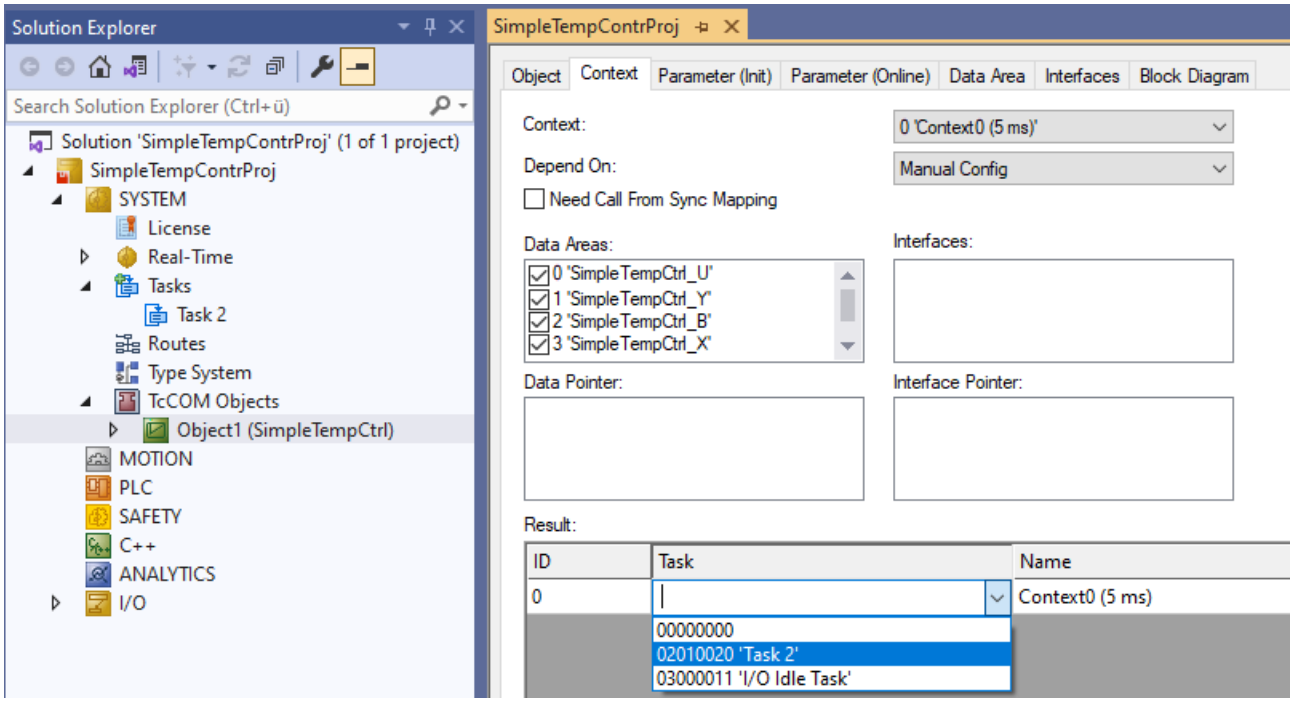



4. 创建一个周期性任务。

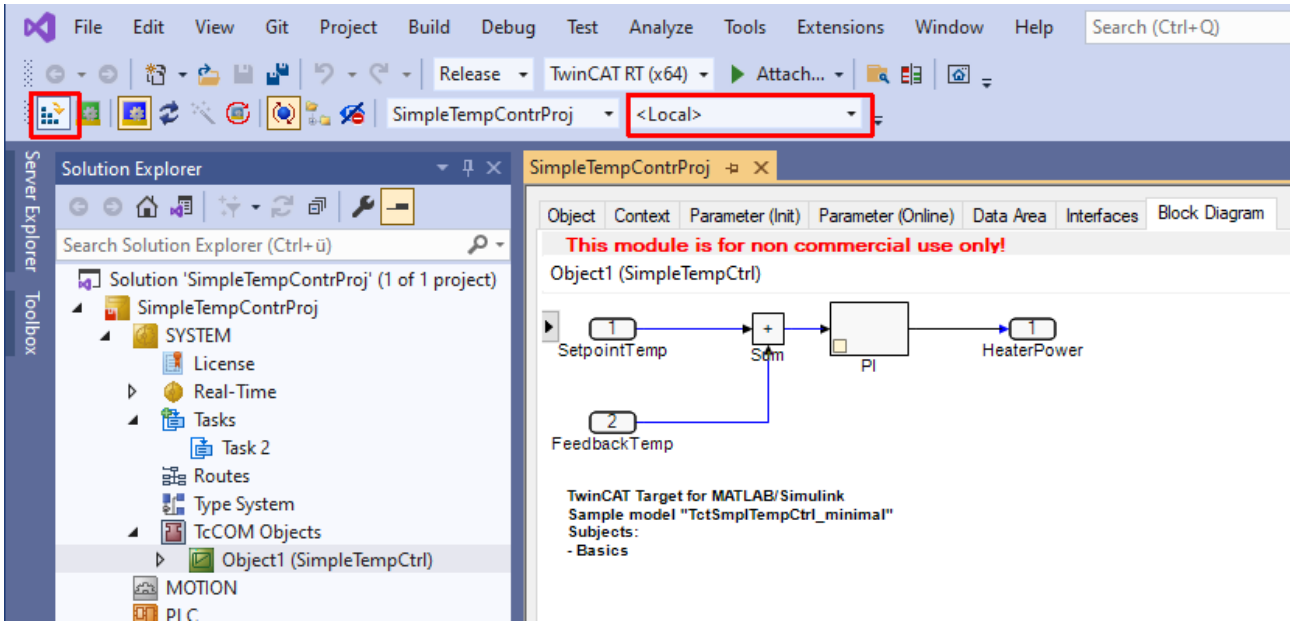



5. 将创建的任务分配给 TcCOM 实例。

请注意，任务的周期时间和 Simulink® 中的 SampleTime（此处为 5 ms）相符。



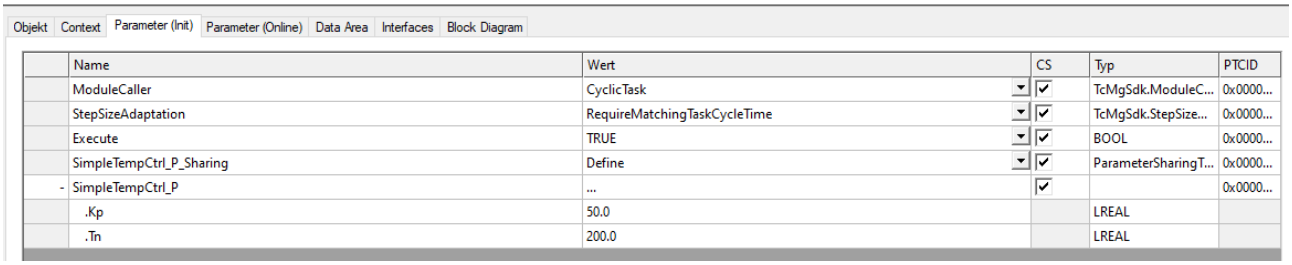

6. 激活配置。



### 配置和链接 TcCOM 实例

TcCOM 实例的数据交换通过过程映像的映射实现。Simulink® 输入和 Simulink® 输出会自动映射为过程映像中的输入或输出，并可链接到 I/O 或其它对象。

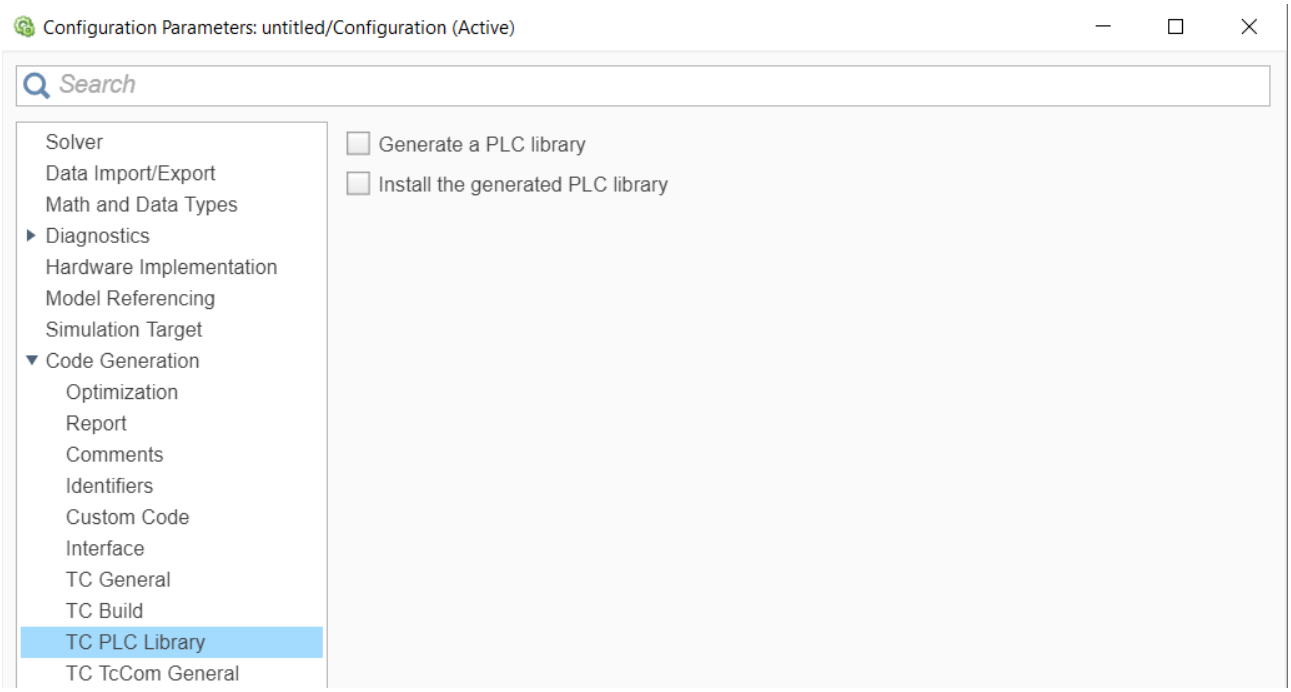
在 TcCOM 实例的“参数（初始值）”（Parameter (Init)）区域，可以选择以不同于从 Simulink® 创建实例时指定的方式配置该实例。




例如，可在此处将参数 Kp 设置为“52”。然后，TcCOM 模块将使用该值作为该实例的默认初值。

### 作为 PLC 功能块插入

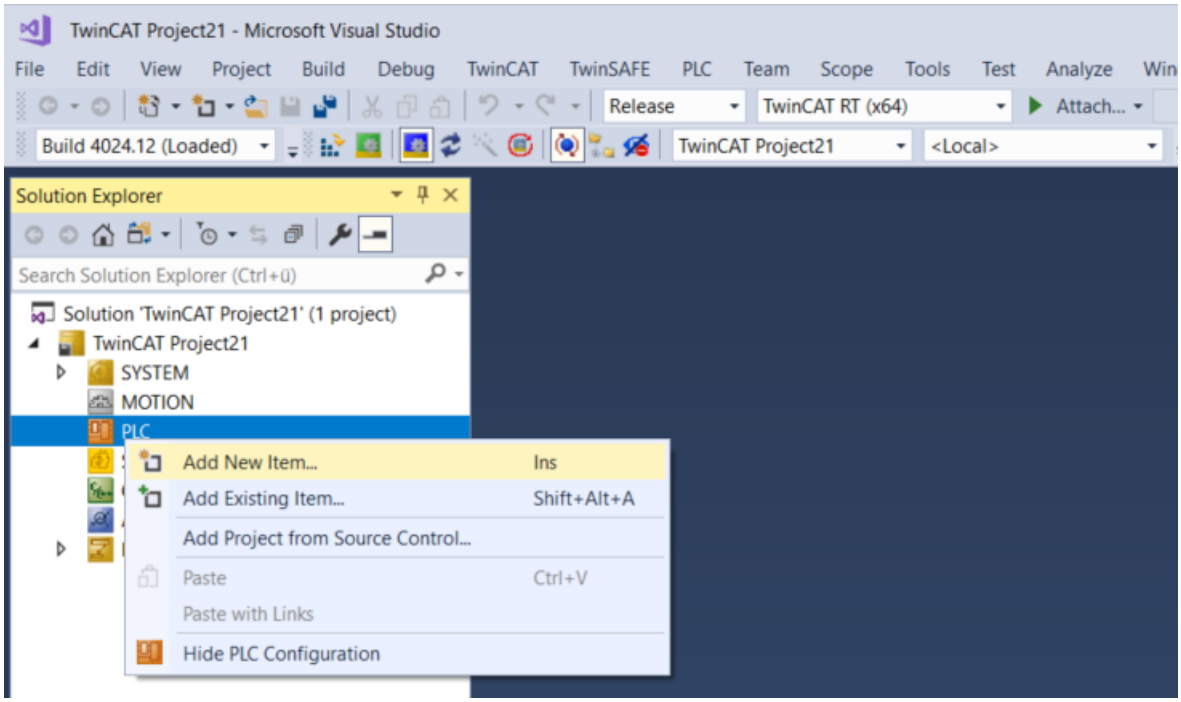
只有在 Simulink® 中设置了以下参数，才能使用下文中提到的 PLC 库：



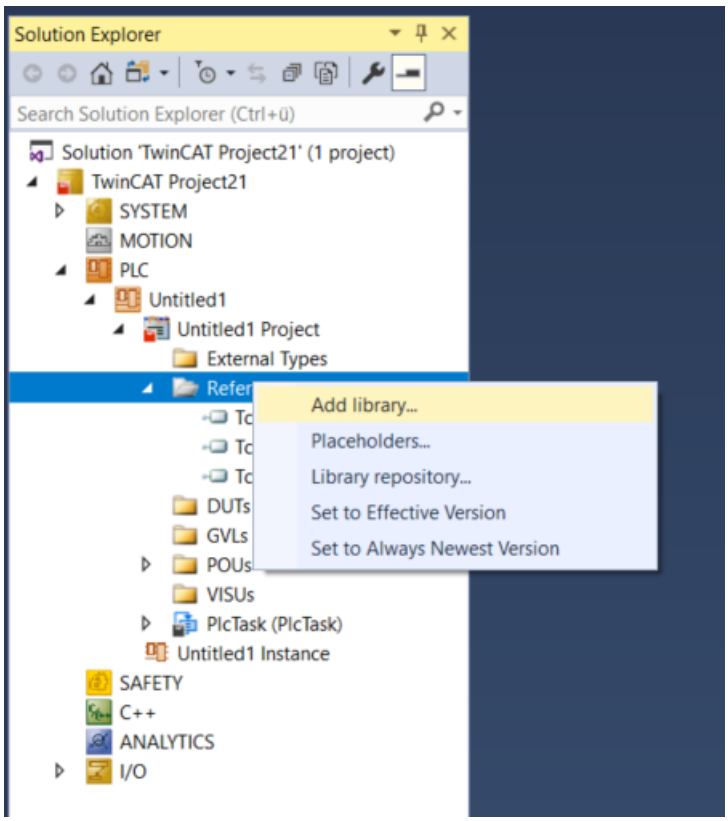

如果尚未设置这些选项，可以随后在不使用 Simulink® 和 TwinCAT Target for Simulink® 的情况下进行设置，请参见 创建并安装 PLC 库 [▶ 131]。

### PLC 库/功能块操作步骤概述

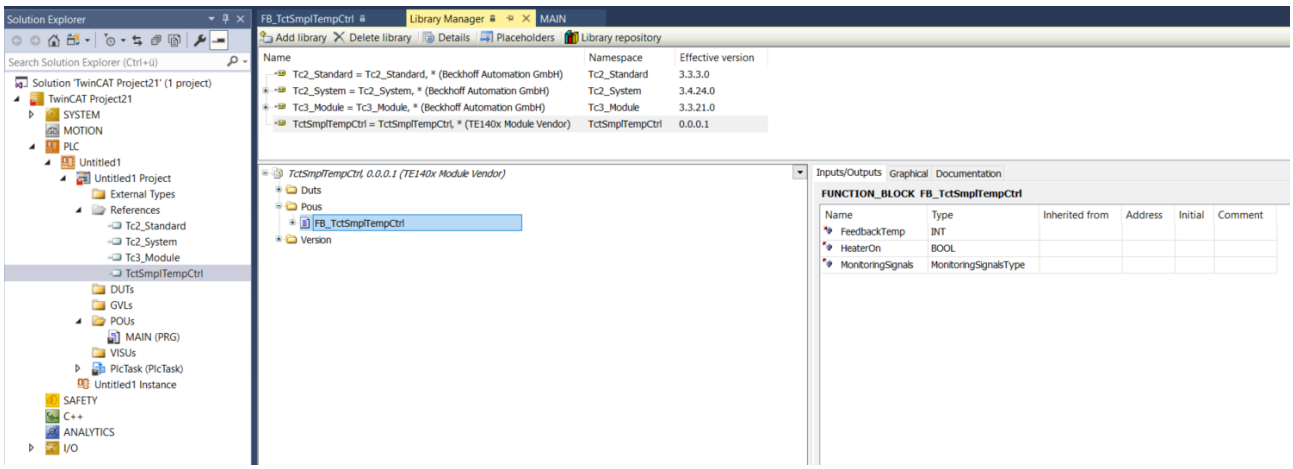
- 在 TwinCAT 中创建 PLC：



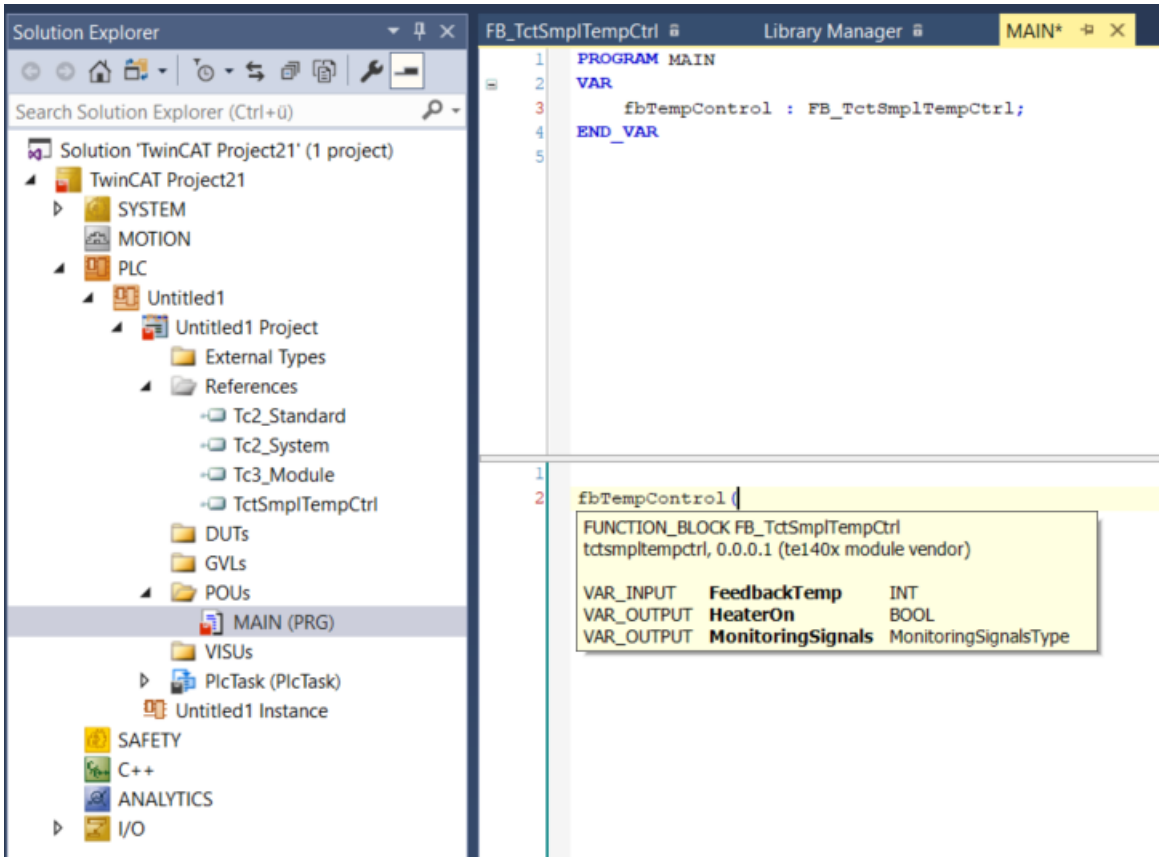

- 添加 PLC 库：




- 选择 PLC 库并查看内容：




- 使用 PLC 库中的功能块：




**也可从 PLC 调用 TcCOM 对象**

除了此处描述的版本外，还可以从 PLC 调用 TcCOM 实例，请参见 应用 TcCOM Wrapper FB [▶ 134]。

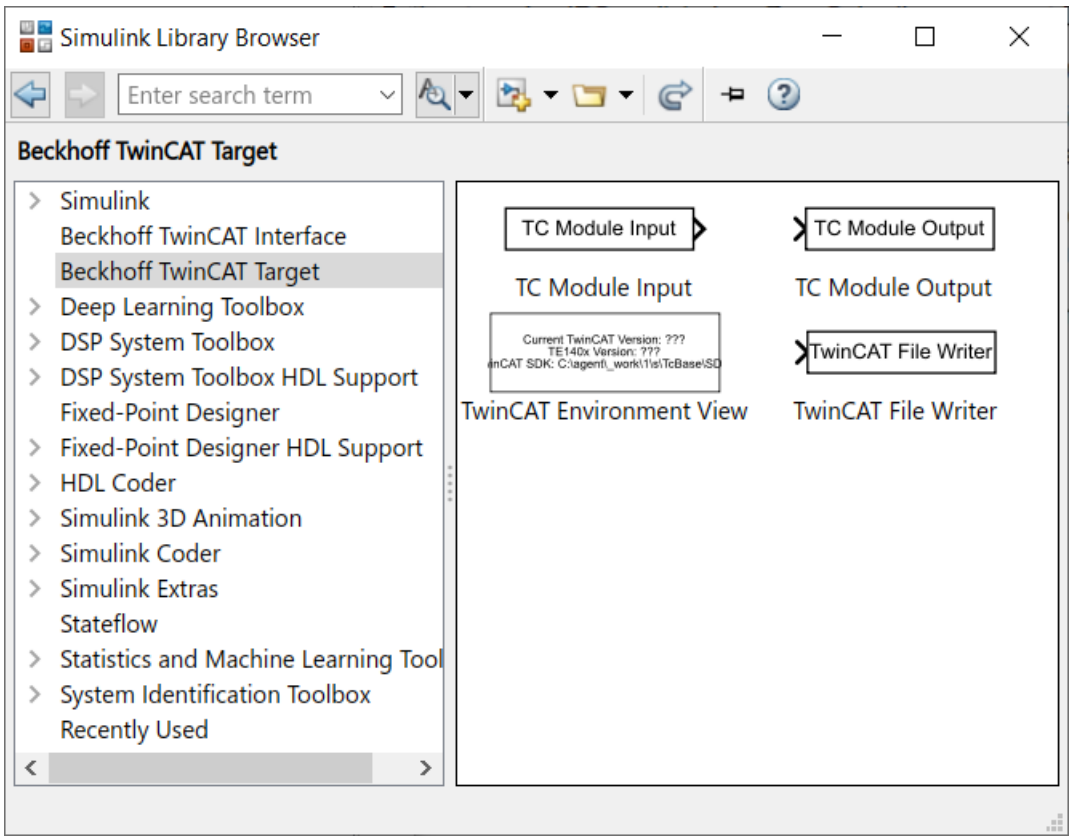
**更多示例**

在成功完成初级示例后，我们建议您阅读以下示例，以加深理解：

```
TwinCAT.ModuleGenerator.Samples.Show("Generate TwinCAT Classes From Simulink Models - Extended")
```

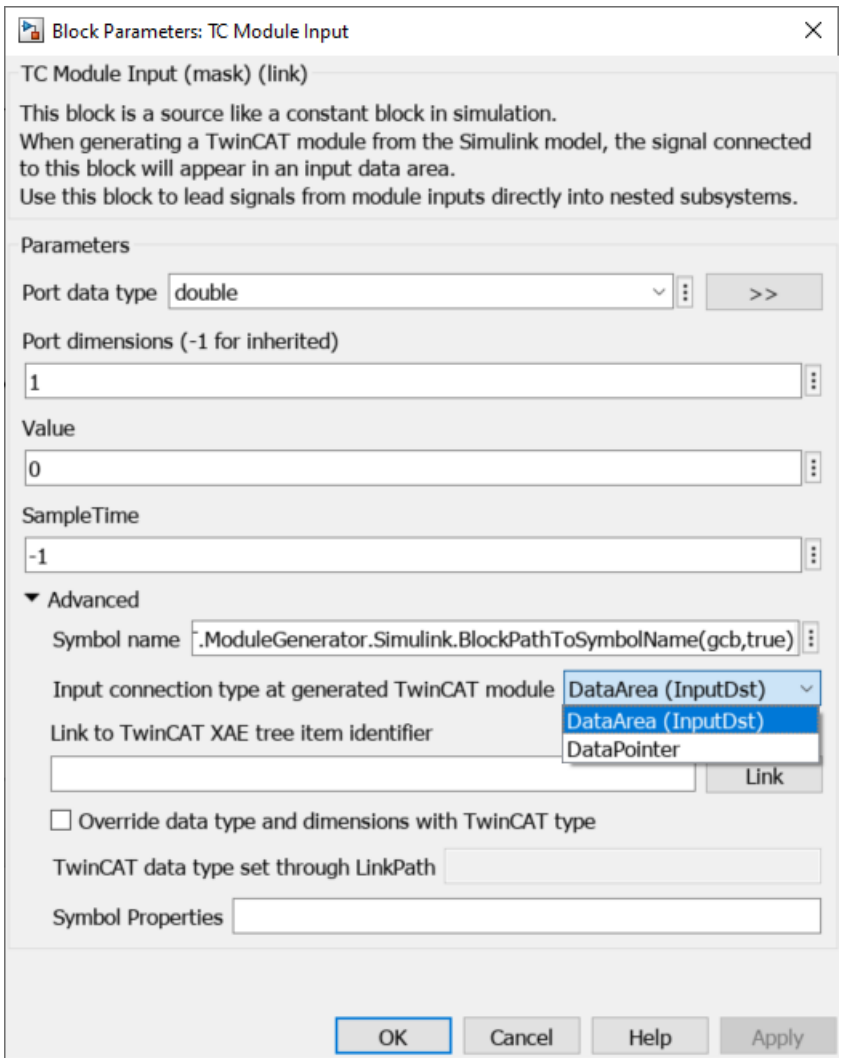
## 4.5 Simulink® 中的 TwinCAT 库

用于 TwinCAT Target for Simulink® 的特定功能块位于**库浏览器**（Library Browser） > **Beckhoff TwinCAT Target** 区域。




## 4.5.1 TwinCAT 输入和输出模块

在 Simulink® 中可选择使用 *TwinCAT 特有的*输入和输出功能块。另一种有效方法是使用 Simulink® 的标准输入端口（In）和输出端口（Out）。这通常也是*最佳做法*，除非需要使用下文所述的 TwinCAT 输入和输出模块的附加功能。




### TC 模块的其他功能

如果使用倍福提供的输入功能块（TC 模块输入）和输出功能块（TC 模块输出），与标准 Simulink® 输入和输出端口相比，您将受益于以下附加功能：

- 您还可以将**子系统中的信号和总线直接定义为 TcCOM 的输入或输出**，而无需先将子系统中的信号/总线传输到顶层系统。示例：子系统输入和输出 [▶ 30]。
- 您可以选择在块参数中存储与其他 TcCOM 或 I/O 的自动映射，以便在 TcCOM 实例化时直接**自动执行映射**，请参见自动映射 [▶ 30]。
- 可以分别选择映射或 **DataPointer** 作为每个 TC 模块输入/输出端口的*连接类型*。例如，您可以通过映射访问所有标准输入端口，也可以通过 DataPointer 访问其他 TC 模块输入端口，请参见DataArea 或 DataPointer [▶ 32]。
  示例：TcCOM 实例之间的共享内存 [▶ 74]。
- 您可以影响**符号名称**（Symbol Name），即更改流程图像中输入或输出的名称和层次，例如，请参见符号名称 [▶ 33]。
- 您可以为信号和总线配备特定的**符号属性**（Symbol Properties），请参见符号属性 [▶ 33]。
- 您可以使用**初始值**（initial values）进行输入。为此，请将 TC 模块输入值设置为任意值，请参见初始值 [▶ 34]。

> 也可为标准输入端口执行初始值，请参见 输入选项：TC TcCom 接口下的初始值 [▶ 67]（Option Input: Initial values under TC TcCom Interfaces）。

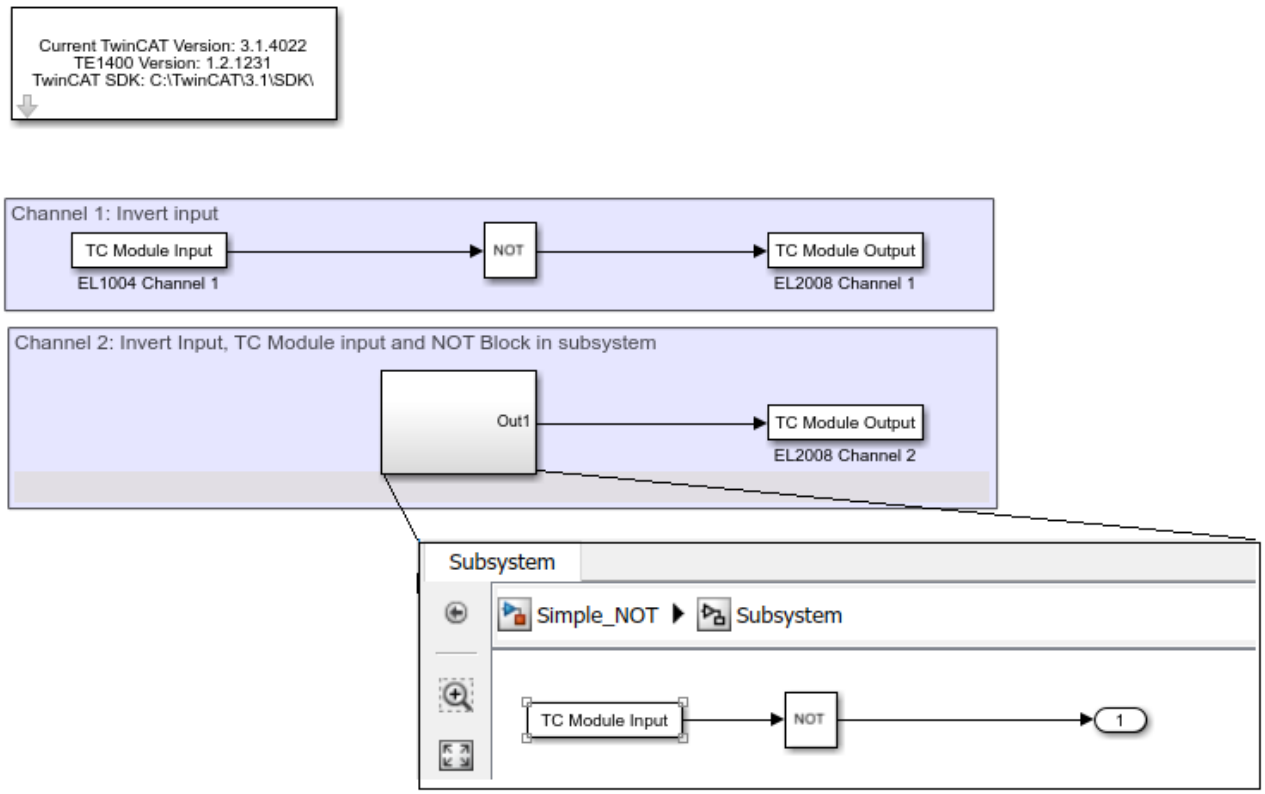
在使用自动映射时，请注意如果 TcCOM 在 TwinCAT 中被实例化不止一次，就会出现映射冲突，必须通过手动映射来解决。因此，不建议在多个实例中使用该选项。

### 4.5.1.1 子系统输入和输出

**从 Simulink® 子系统创建 TcCOM 输入/输出**

创建一个 Simulink® 模型，输出两个负输入。输入被置于一个子系统中，见下图。

根据所描述的 TC 模块输入/输出功能块属性，TwinCAT 的 Simulink® 子系统中的 TC 模块输入也包含在 TcCOM 的过程映像中。因此，无需像标准输入端口或输出端口那样，布线到 Simulink® 模型的顶层。




### 4.5.1.2 自动映射

Simulink® 模型的输入和输出会自动映射为下列数字输入和输出。这意味着在 TwinCAT 项目中实例化 TcCOM 后，映射将自动创建。不再需要手动链接。

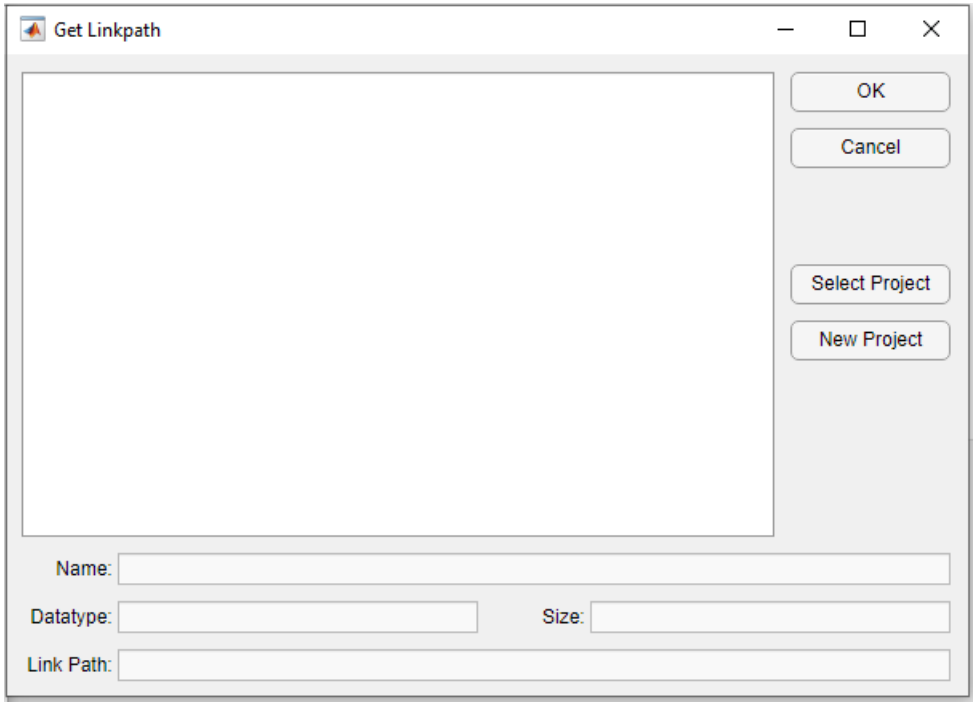
导航至**高级**（Advanced）下 Tc Module In 或 Tc Module Out 块参数中的参数 **Link to TwinCAT XAE 树状项标识符**（Link to TwinCAT XAE tree item identifier）。您可以手动输入树状项标识符和数据类型，也可以通过浏览器从 Simulink® 中选择。

选择**链接**（Link）按钮，会打开一个新对话框。现在，您可以加载一个现有的 TwinCAT 项目（**选择项目**（Select Project））并浏览现有的输入或输出，也可以创建一个新项目（**新项目**（New Project））并自动扫描新项目中的 EtherCAT 现场总线，将其与检测到的 I/O 连接起来。

✓ 创建新项目，显示并选择目标输入和输出：

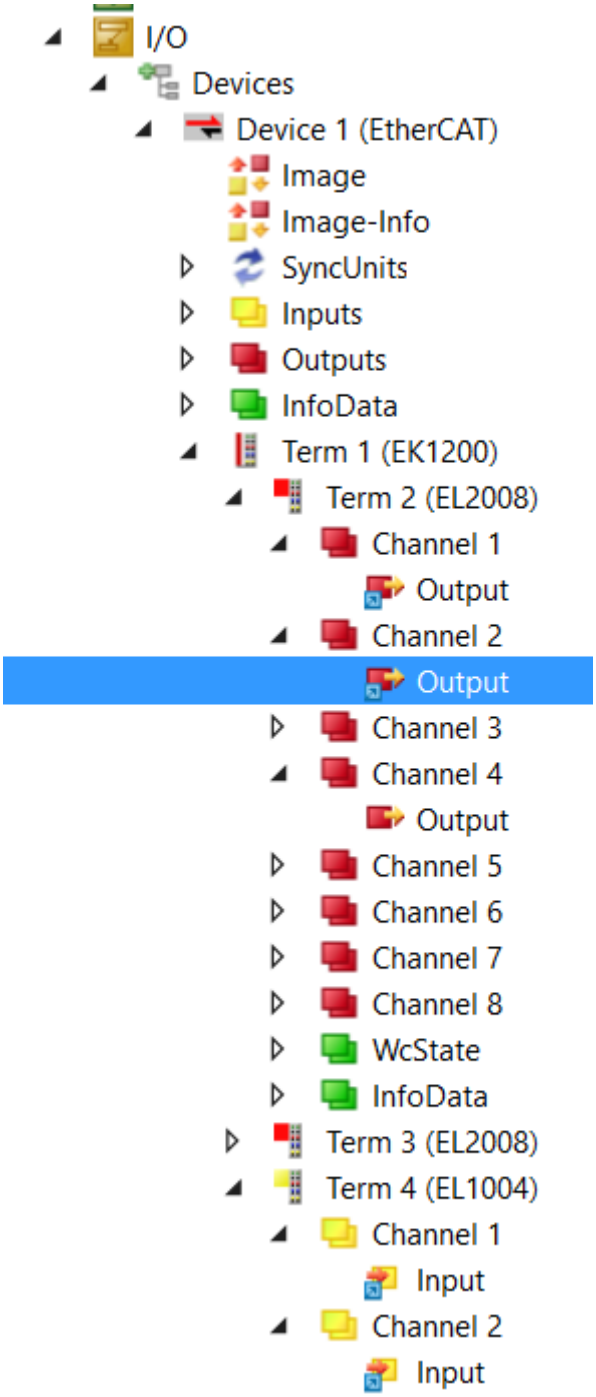
1. 创建一个新项目。
2. 为此，请选择要创建的新 TwinCAT 项目的内存路径。
3. 选择要下载项目的目标。
4. 自动扫描目标的 I/O 树。

⇨ 显示并选择目标的所有输入和输出。




选择要链接的输入或输出后，树状项标识符会自动设置，相应的数据类型也会自动输入 Simulink®。

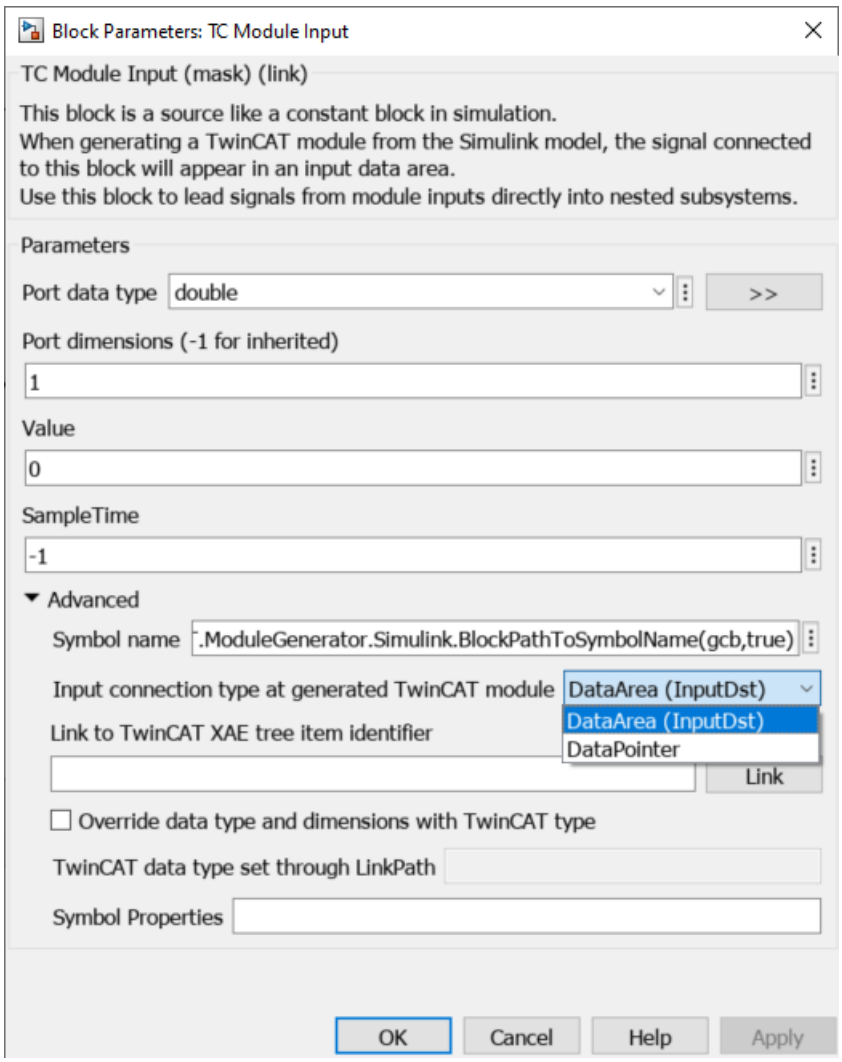
如果将上述 Simulink® 模型编译成 TcCOM 并集成到 TwinCAT 3 解决方案中，则会自动创建与 Simulink® 中所选输入和输出的映射。
区分符号的颜色：
自动生成的映射使用*蓝色*符号，而手动映射符号显示为*白色*。




### 4.5.1.3 DataArea 或 DataPointer

默认情况下，“DataArea（InputDst）”被选为**生成的 TwinCAT 模块的输入连接类型**（Input connection type at generated TwinCAT module）。这意味着 TC Module Input 参数创建为*Input Destination DataArea*，TC Module Output 相应地创建为*Output Source DataArea*。

或者，也可以在此处选择“DataPointer”，请参见TcCOM 实例之间的共享内存 [▶ 74]。




如果上图所示的 Simulink® 模型是使用 **Target for Simulink®** 创建的，且**生成的 TwinCAT 模块中的输入连接类型**（Input connection type at generated TcCOM module）中选择“mapping”，则在 TcCOM 实例的过程映像中会出现 2 个输入和 2 个输出。

### 4.5.1.4 符号属性

可以为 TC 模块输入/输出块定义特定的符号属性，请参见符号特性和编译指令的赋予 [▶ 91]。

示例：`OPC.UA.DA.=1`

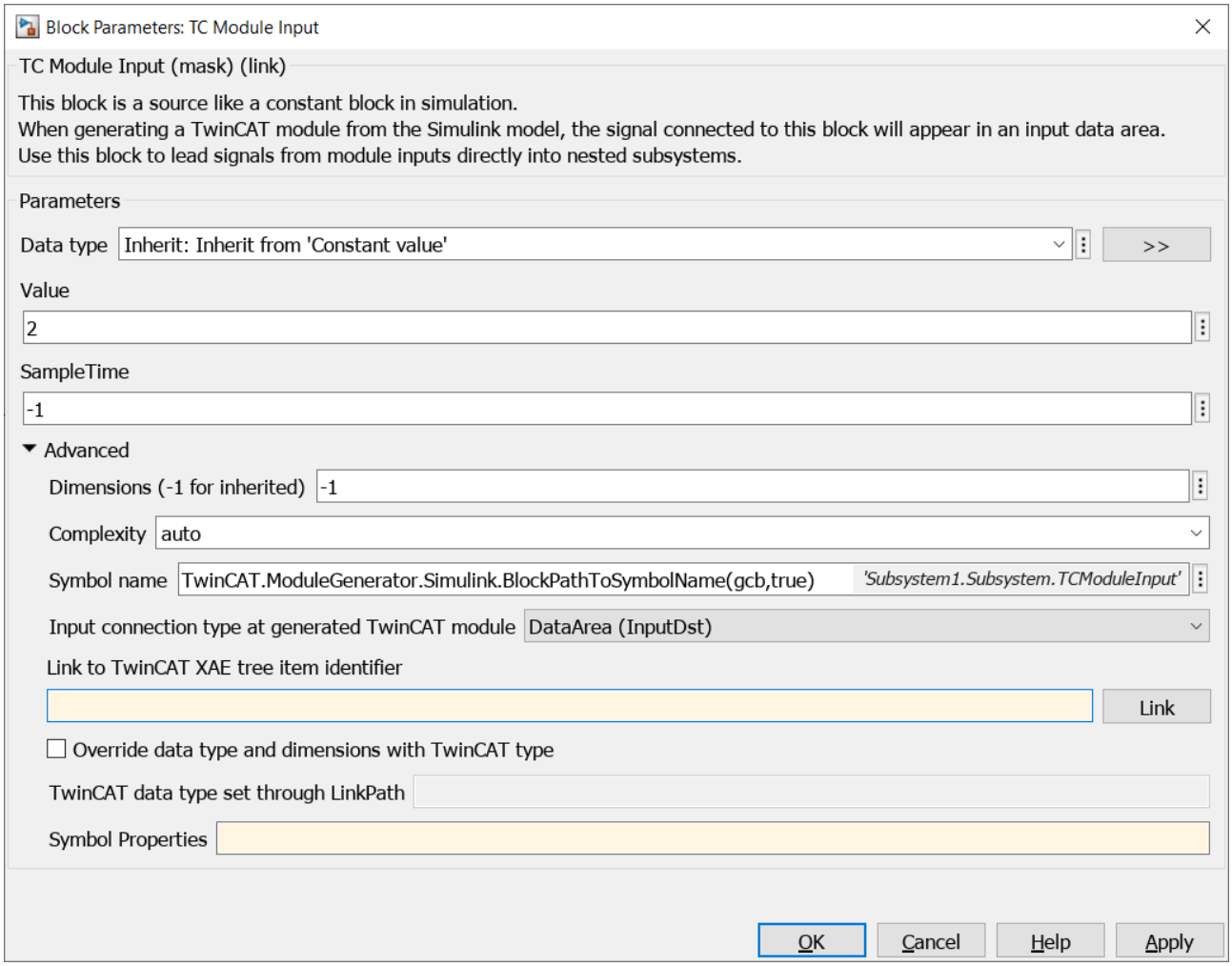
**限制**

使用符号属性时，符号名称不得嵌套。

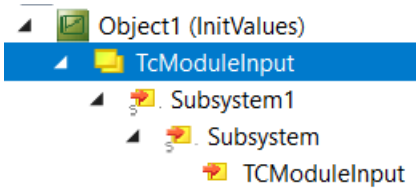
### 4.5.1.5 符号名称

您可以通过*符号名称*更改名称，从而更改流程图像中的符号显示。

在下面的示例中，一个 TC 模块输入嵌套在两个子系统中。默认的名称解析是（如名称符号右侧所示）<子系统>.<子系统>.<TcModuleName>。



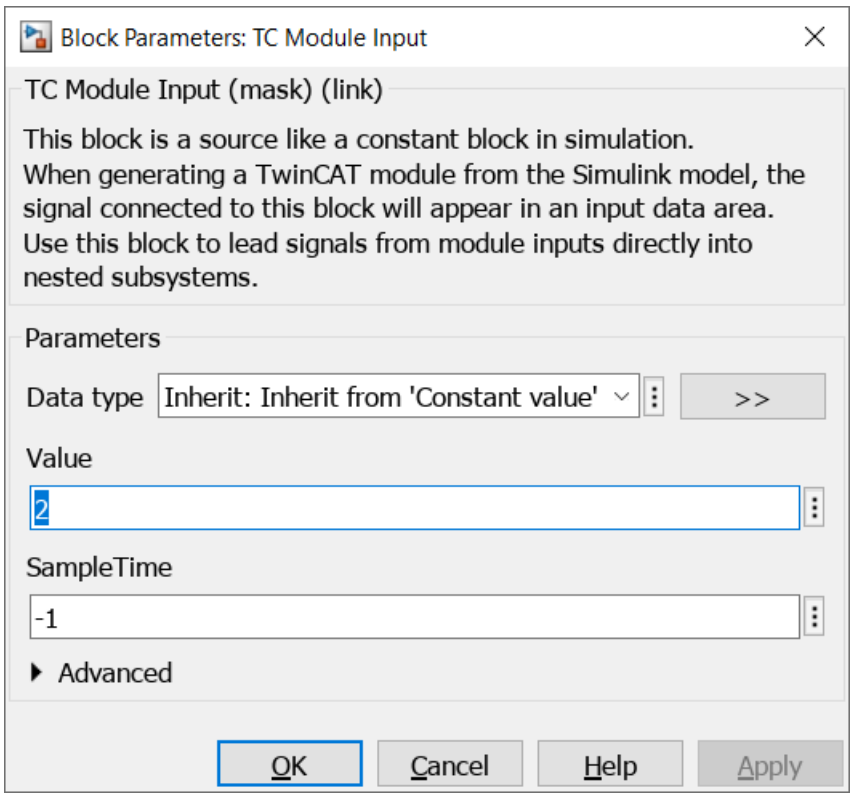

流程图像看起来是这样的：




特别是在嵌套很深的情况下，最好缩短符号名称。因此，您可以选择以文本形式手动输入符号名称，例如 `'Subsystem.InputValue'`。

### 4.5.1.6 初始值

除了数据类型、维度和采样时间外，还可编辑可选的 **Value** 字段，以及标准 Simulink® in-port 和 out-port。该字段允许您指定输入的初始值（仅适用于 TC 模块输入）。如果 TwinCAT 中的输入未链接到相应的输出，则此处输入的值将用作输入值。

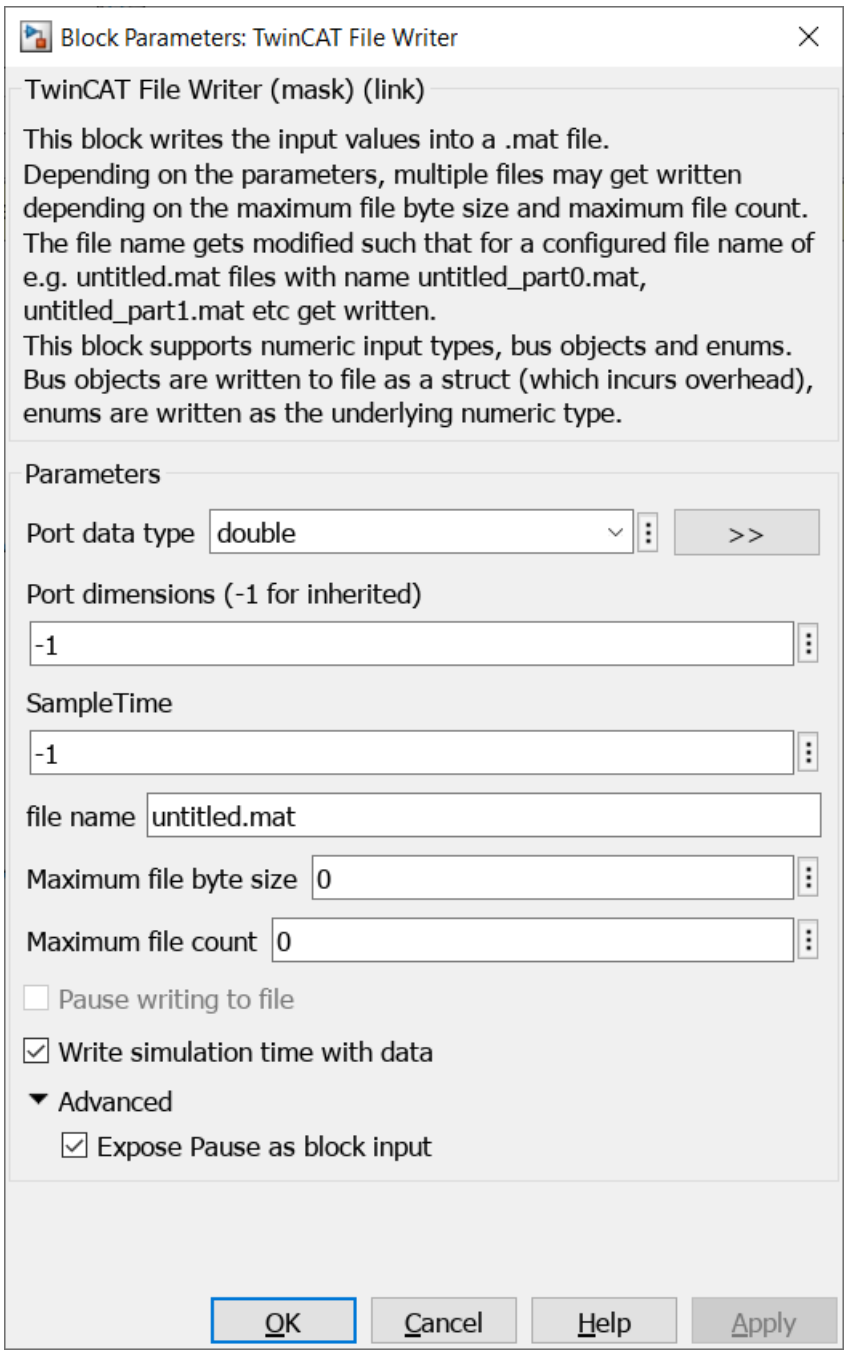



### 4.5.2 TwinCAT 环境视图

将 TwinCAT 环境视图拖入 Simulink® 环境，可直接显示可用的 TwinCAT XAE 版本和当前安装的 TE1400 版本，例如用于支持案例。

### 4.5.3 TwinCAT File Writer

TwinCAT File Writer 功能块从 TwinCAT 环境中写入 .mat 文件。只有 Simulink® 模型已被传输到 TcCOM 或 FB 并在 TwinCAT Runtime 中执行时，该功能块才能实现这一功能。如果 Simulink® 模型在 Simulink® 中执行，则此功能块无效。




| 参数 | 描述 | 说明 |
|---|---|---|
| 端口数据类型 | 输入信号的数据类型 | 为以下提供支持：<br>• 整数类型（Integer Types）<br>• 单精度浮点型（float）<br>• 双精度浮点型（double）<br>• 布尔类型（boolean）<br>• 枚举类型（enums）<br>• 总线对象（bus objects） |
| 端口尺寸 | 输入信号的维度 | -1 -> Inherit<br>否则 [1,2]、[1,5].....，等 |
| 采样时间（SampleTime） | 块采样时间（秒） | -1 -> Inherit |
| 文件名称 | .Mat 文件的文件名 | 可使用完整路径或相对路径。相对于 *TwinCAT\|3.1\|Boot* 的相对路径。 |

| 参数 | 描述 | 说明 |
|---|---|---|
| 最大文件字节大小 | .mat 文件的最大大小（单位：字节）。当达到此大小时，当前文件将被关闭，并启动一个新文件。 | 0 -> 基于文件格式的最大值 |
| 最大文件数 | 要写入的 .mat 文件的最大数量。 | 0 -> 不限数量的文件。如果达到最大值，旧文件将被覆盖，从 `_part0.mat` 开始。 |
| 暂停写入文件 | 暂停写入 | TcCOM 上的参数 |
| 用数据写入模拟时间 | 为每个日期写入一个包含“时间”和“数据”两个字段的结构。 | |
| 将“暂停”（Pause）作为程序块输入进行呈现 | 创建一个可暂停 TwinCAT 文件写入器的输入。 | |

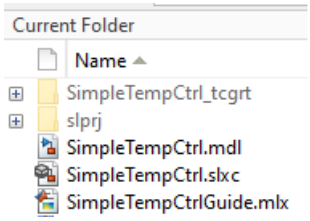
## 4.6 自动生成文件概述

启动构建程序时，会自动创建一些文件和文件夹。“文件在哪里？”“怎么处理它们？”以及“这些文件意味着什么？”下文将回答这些问题。

自动生成的文件有哪些类别?

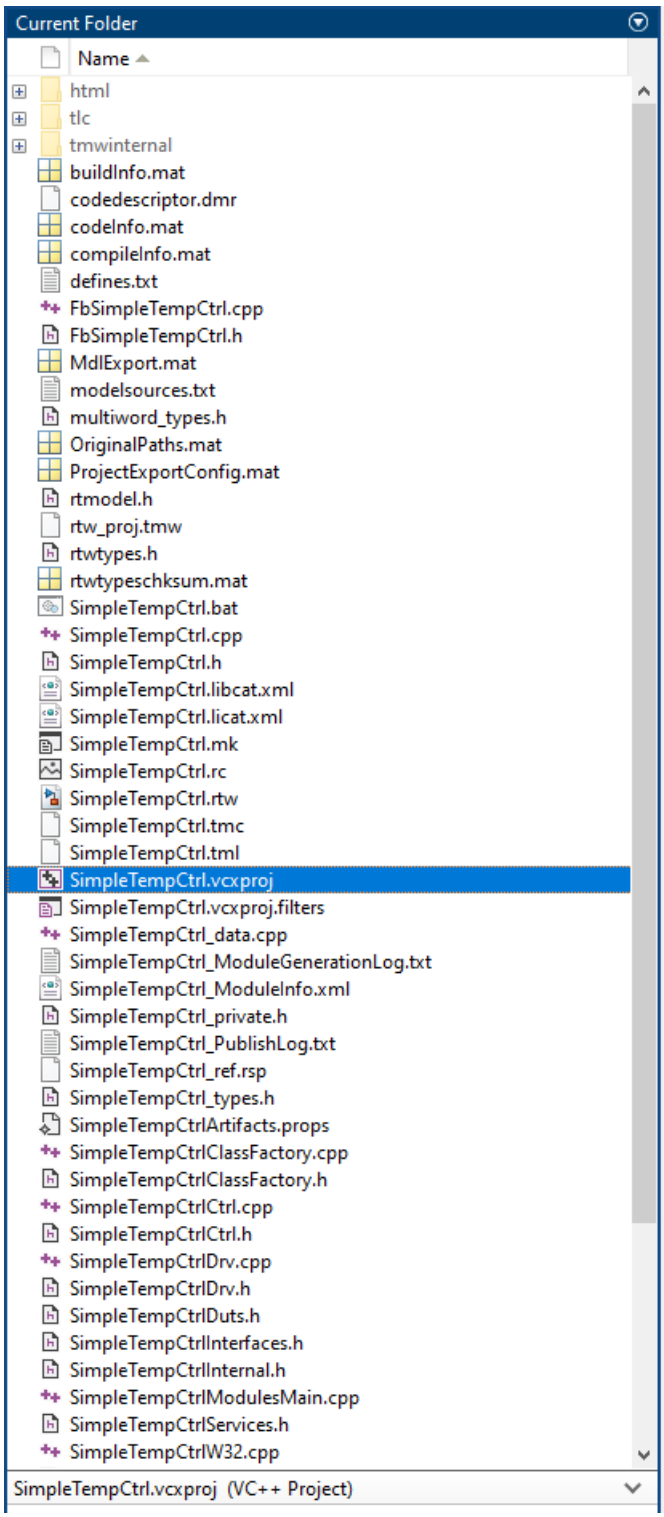
- 生成源代码 [▶ 37]。
- 生成日志文件 [▶ 39]。
- 创建 TwinCAT 对象、驱动程序（*.tmx）和说明文件（*.tmc、*.library...） [▶ 40]。

TwinCAT Target 创建的所有文件都汇总在当前 MATLAB® 路径下的文件夹 *<SimulinkModelName>_tcgrt* 中。该文件夹位于 MathWorks® 生成的 slprj 文件夹旁边。



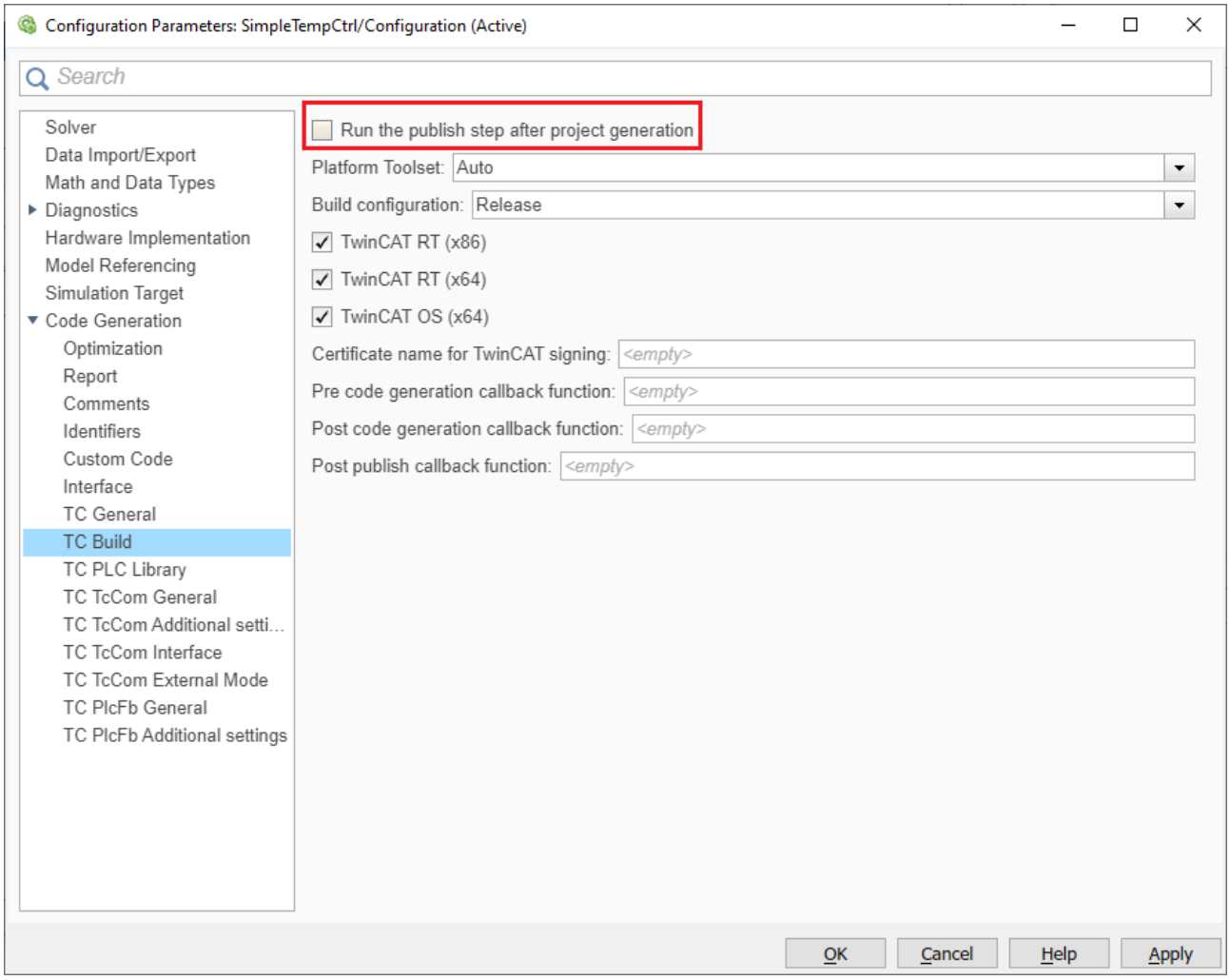

### 生成的源代码

源代码的中心文件是 *<SimulinkModelName>.vcxproj*，位于 <SimulinkModelName>_tcgrt 文件夹中。该文件将打开一个 TwinCAT C++ 项目，可用于检查生成的源代码，随后构建 TwinCAT 对象，或在 TwinCAT 中进行调试 [▶ 139]。




关于生成的 TwinCAT C++ 项目的构建选项，值得注意的是，您可以在 Simulink® 中关闭发布步骤，即为配置的平台构建。

您可以在 Simulink® 中取消选择*项目生成后运行发布步骤*，从而实现代码生成而无需构建。*发布步骤*包含所选平台（TwinCAT RT x86、x64 ...）的 TwinCAT 对象的*构建*。

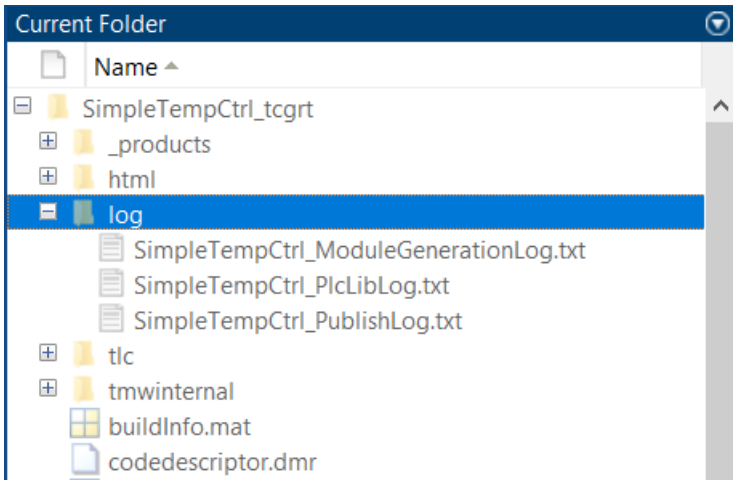



更多实用信息：

- 利用 Target for Simulink，打包即用
- 持续集成

### 生成的日志文件

生成的日志文件汇总在 *log* 子文件夹中。调试时首先要查看创建的日志文件。




该文件夹最多包含三个日志文件。主要联系点是文件 *<ModelName>_ModuleGenerationLog.txt* 中的所有日志摘要。

**倍福支持团队需要日志文件**

如果您需要我们支持团队的帮助，请至少发送日志文件夹中的以下文件：
*<ModelName>_ModuleGenerationLog.txt*

模块生成日志（ModuleGenerationLog）的结构分为几个部分，代表 TwinCAT Target 的不同步骤。这些步骤在日志中显示为 ###。

示例：

```
2023-10-18 14:48:37: ### Starting build procedure for: SimpleTempCtrl
2023-10-18 14:48:43: ### Save TLC export
2023-10-18 14:48:47: ### Export block diagram
2023-10-18 14:48:47: Block diagram export succeeded
2023-10-18 14:48:48: ### Export TwinCAT C++ project
2023-10-18 14:48:53: ### Save project
2023-10-18 14:48:53: The TwinCAT C++ project "C:
\Users\xyz\Documents\MATLAB\TE14xxSamples\2023-10-18_13-58_SimpleTemperatureController\SimpleTempCtr
l_tcgrt\SimpleTempCtrl.vcxproj" was generated successfully
```

如果步骤中没有出现警告或错误，则通常不对所执行的步骤进行记录。在某些情况下，会明确引用已创建的文件，如上述示例中创建的 vcxproj 文件。还可参考其他日志文件获取更多详细信息，请参见以下示例：

```
2023-10-18 14:48:53: ### Publish TMX
2023-10-18 14:48:53: Configuration: "Release"
2023-10-18 14:48:53: Platform(s): "TwinCAT RT (x86);TwinCAT RT (x64);TwinCAT OS (x64)"
2023-10-18 14:48:53: TwinCAT SDK: "C:\TwinCAT\3.1\SDK\" (Version 3.1.4024.50)
2023-10-18 14:48:53: Platform Toolset: V142 (Automatically selected)
2023-10-18 14:48:53: Microsoft (R) Build Engine version 16.11.2+f32259642 for .NET Framework
2023-10-18 14:48:53: Copyright (C) Microsoft Corporation. All rights reserved.
2023-10-18 14:49:06: Publish procedure completed successfully
2023-10-18 14:49:06: See log file "C:
\Users\xyz\Documents\MATLAB\TE14xxSamples\2023-10-18_13-58_SimpleTemperatureController\SimpleTempCtr
l_tcgrt\log\SimpleTempCtrl_PublishLog.txt" for details
```

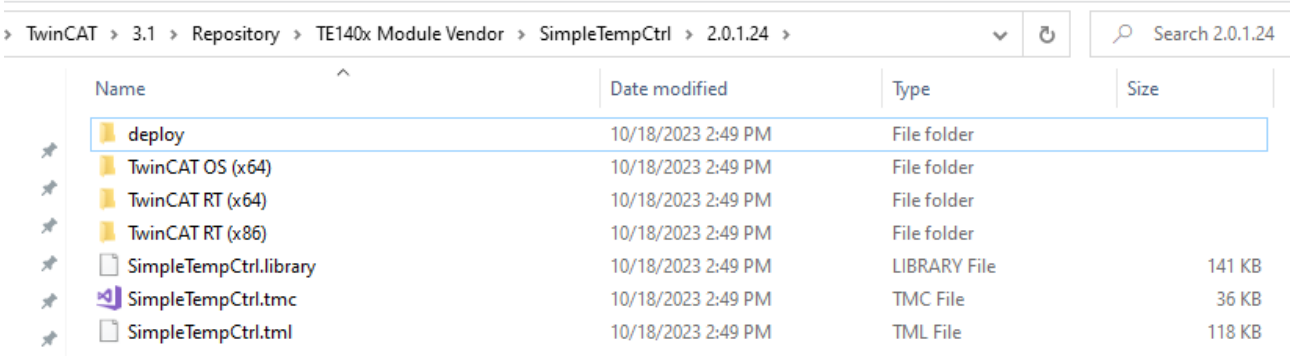
如果出现警告或错误，将显示在模块生成日志中。而详细日志则包含所执行步骤的所有输出。例如，在创建的 tmx 文件的签名验证区域就会出现警告：

```
2023-10-18 14:49:06: ### Publish summary
2023-10-18 14:49:06: Configuration: "Release"
2023-10-18 14:49:06: Platform(s): "TwinCAT RT (x86);TwinCAT RT (x64);TwinCAT OS (x64)"
2023-10-18 14:49:06: TwinCAT SDK: "C:\TwinCAT\3.1\SDK\" (Version 3.1.4024.50)
2023-10-18 14:49:06: Platform Toolset: V142 (Automatically selected)
2023-10-18 14:49:06: Vendor name: TE140x Module Vendor
2023-10-18 14:49:06: Library name: SimpleTempCtrl
2023-10-18 14:49:06: Library version: 2.0.1.24
2023-10-18 14:49:06: Local installation folder: "C:\TwinCAT\3.1\Repository\TE140x Module
Vendor\SimpleTempCtrl\2.0.1.24"
2023-10-18 14:49:06: TMX archive: -
2023-10-18 14:49:06: Signatures:
2023-10-18 14:49:06: File 'C:\TwinCAT\3.1\Repository\TE140x Module
Vendor\SimpleTempCtrl\2.0.1.24\TwinCAT RT (x86)\SimpleTempCtrl.tmx' has signature.
2023-10-18 14:49:06: issuer TestSign123 (x.yz@beckhoff.com), certificate expires on 08/15/2025
2023-10-18 14:49:06: Warning: Signature found, but OEM certificate was not signed by Beckhoff.
Driver can only be used in test mode.
2023-10-18 14:49:06: File 'C:\TwinCAT\3.1\Repository\TE140x Module
Vendor\SimpleTempCtrl\2.0.1.24\TwinCAT RT (x64)\SimpleTempCtrl.tmx' has signature.
2023-10-18 14:49:06: issuer TestSign123 (x.yz@beckhoff.com), certificate expires on 08/15/2025
2023-10-18 14:49:06: Warning: Signature found, but OEM certificate was not signed by Beckhoff.
Driver can only be used in test mode.
2023-10-18 14:49:06: File 'C:\TwinCAT\3.1\Repository\TE140x Module
Vendor\SimpleTempCtrl\2.0.1.24\TwinCAT OS (x64)\SimpleTempCtrl.tmx' has signature.
2023-10-18 14:49:06: issuer TestSign123 (x.yz@beckhoff.com), certificate expires on 08/15/2025
```

### 创建的 TwinCAT 对象

在成功*构建*后，所创建的二进制文件和描述文件可在 TwinCAT XAE 中重复使用，这些文件被存储在*开发环境存储库*（Engineering Repository）中，即开发环境 PC 上，地址为：

*%TwinCATInstallDir% \3.1|Repository|<Vendor>|<ModelName>|<Version>|*




- <ModelName>.tmc：TcCOM 对象（对象类）的描述文件
- <模型名>.tml：库文件：可以像 .library 文件一样使用。
- <模型名>.库：TwinCAT PLC 库文件
- Deploy\<ModelName>_ModuleInfo.xml：模块信息，例如框图
- <Platform>\<ModelName>.tmx：驱动程序文件

向其他 XAE 系统分发 TwinCAT 对象：如果将 <ModelName> 级别的文件夹复制到本地*开发环境存储库*（Engineering Repositories）中装有 TwinCAT XAE 的其他 PC 上，其用户就可以在其 TwinCAT 解决方案中使用创建的 TwinCAT 对象。

另请比较共享已创建的 TwinCAT objects [▶ 57]。

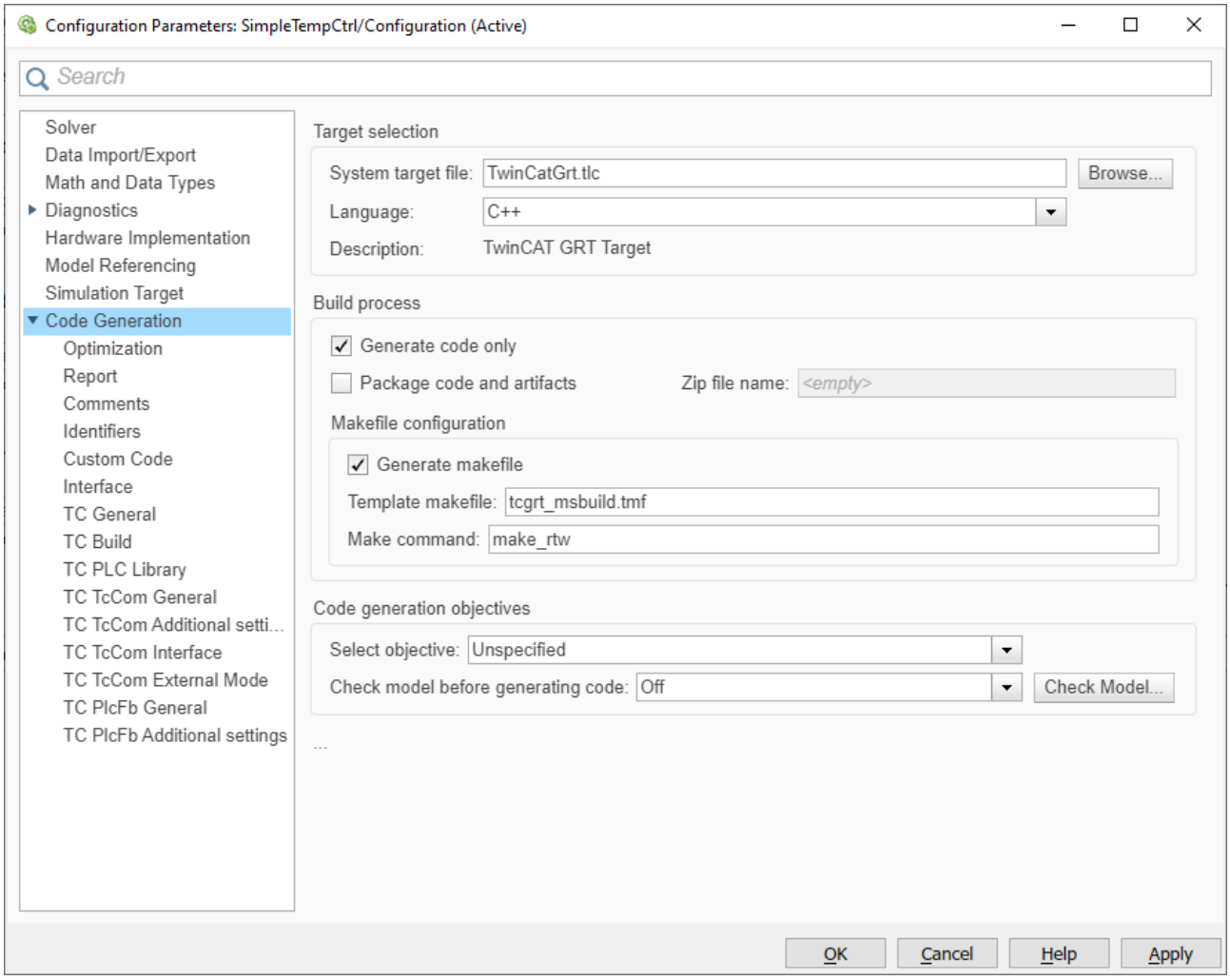
**补充说明**

生成的 C++ 文件和二进制文件的说明

版本控制的 C++ 项目

## 4.7 在 Simulink® 中生成代码的参数设置

在 Simulink® 中，配置要生成的 TwinCAT objects（TcCOM 和功能块）时可以进行大量设置。只要选择 **TwinCatGrt.tlc** 作为系统**目标文件**，代码生成下的树形结构中就会增加几项条目（参见以 TC 开头的条目）。




### 配置视图

由于配置选项范围广泛，因此可以切换视图。安装后，配置级别设为*标准*（Standard），只显示最常用的参数。还可以将配置级别提高到*高级*（Advanced），以便进行更多设置。

使用 MATLAB® 命令窗口可以设置视图：

```
TwinCAT.ModuleGenerator.Settings.Change('ConfigurationLevel', 'Advanced')
TwinCAT.ModuleGenerator.Settings.Change('ConfigurationLevel', 'Standard')
```

初始的设置只是暂时的。如要保存，请使用保存命令：

```
TwinCAT.ModuleGenerator.Settings.Save;
```

如果在不同的系统上运行代码生成和构建过程，请确保两个系统上的配置级别相同。

### 配置参数概述

关于所有配置参数的概述，请参见所有配置参数的概览表 [▶ 43]章节的文档资料。

**阅读工具提示**

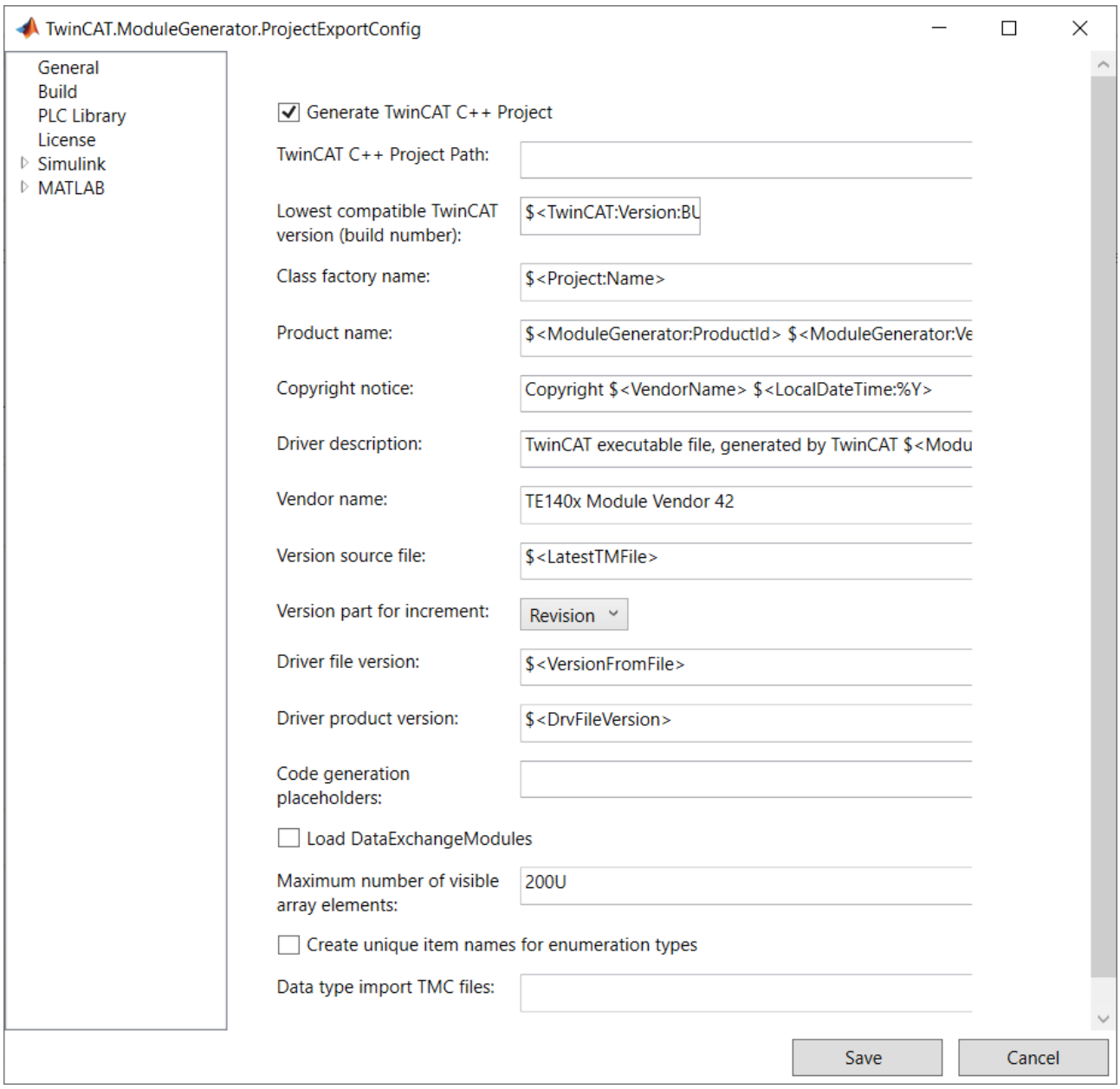
将光标悬停在对话框的文本字段上，就会以工具提示的形式显示该选项的详细说明。

### 更改软件设置

要更改 TwinCAT Target 的默认设置，可以访问 MATLAB® 控制台中的对话框，如下所示：

```
TwinCAT.ModuleGenerator.Settings.Edit
```

这里提供了各种条目，您可以将其存储为默认值。




**接受更改**

1. 在对话框中输入新的默认设置。
2. 使用**保存**（Save）按钮确认。
3. 重新启动 MATLAB®。

⇨ 更改即被采纳。

**构建后更改配置**

在*配置参数*（Configuration parameters）中选择的许多设置可以在 TwinCAT 3 的 TcCOM 实例层级再次更改，例如，可以定义通过周期性任务调用的一类模型，但也可以随后将个别实例模型配置为从 PLC 调用。

## 4.7.1 所有配置参数的概览表

以下表格中对所有配置参数进行了概述。配置参数是否显示取决于 Simulink® 用户界面中的配置级别。如果通过 m 文件配置模块生成器 [▶ 52]，配置级别不会产生影响。

要在用户界面中查看下方描述的所有参数，请切换到高级配置级别：

```
TwinCAT.ModuleGenerator.Settings.Change('ConfigurationLevel', 'Advanced')
```

为每个参数都提供了简要说明，在某些情况下，更详细的信息请参考本文档的其它章节。基于用户界面的表示方式进行分类。在用户界面中，使用的是显示名称。如果通过 m 文件使用模块生成器，“名称”（Name）一栏至关重要。

| Category | Name | Displayname | Default | Description |
|---|---|---|---|---|
| TC General | Generate | Generate TwinCAT C++ Project | TRUE | Generate a TwinCAT C++ project. If unset, only code artifacts will be generated which can get used to generate C++ projects later [▶ 52]. |
| | ProjectDir | TwinCAT C++ Project Directory | | Full path to the directory with the VCXPROJ file (e. g. "C:\Temp") |
| | ProjectName | TwinCAT C++ Project Name | | Name to the generated VCXPROJ file (e. g. "MyGeneratedProject.vcxproj") |
| | LowestCompatibleTcBuild | Lowest compatible TwinCAT version (build number) | $<TwinCAT:Version:BUILD> | The lowest TwinCAT build number the generated C++ project and its modules and POUs are to be compatible with. |
| | ClassFactoryName | Class factory name | $<Project:Name> | Name of the generated C++-Project, Name of the TcCOM classfactory and tmx-file name |
| | ProductName | Product name | $<ModuleGenerator:ProductId> $<ModuleGenerator:Version:MAJOR.MINOR> | Product name, used e.g. for the module driver description and the module TMC description [▶ 83]. |
| | Copyright | Copyright notice | Copyright $<VendorName> $<LocalDateTime:%Y> | Copyright notice of the generated module driver file [▶ 83] |
| | Description | Driver description | TwinCAT executable file, generated by TwinCAT $<ModuleGenerator:ProductId> | Driver description [▶ 83] |
| | VendorName | Vendor name | TE140x Module Vendor | Module vendor name, used as the company name of the generated executables in the repository [▶ 40] and the major module group as shown in the TwinCAT XAE module dialog. |
| | VersionSrc | Version source file | $<LatestTMFile> | Path to an existing TMC, TML or XML file containing the previous version value [▶ 61]. |
| | IncrementVersion | Version part for increment | Revision | The part of the version number that is to be incremented [▶ 61]. |
| | DrvFileVersion | Driver file version | $<VersionFromFile> | Executable file version and library version. [▶ 61] |

| Category | Name | Displayname | Default | Description |
|---|---|---|---|---|
| | DrvProductVersion | Driver product version | $<DrvFileVersion> | Product version [▶ 61] |
| | CodeGenPlaceholders | Code generation placeholders | | Define custom placeholders |
| | UseDataExchangeModules | Load DataExchangeModules | 0 | Manually set DataExchangeModule [▶ 59] dependency (currently no need to set manually) |
| | MaxVisibleArrayElements | Maximum number of visible array elements | 200U | Specifies the maximum number of array elements to be displayed in the TwinCAT XAE. In the TwinCAT XAE, larger arrays cannot get expanded and linked to by its individual items |
| | PackOutputPath | Pack output path | | Path to pack the generated TwinCAT C++ project. Optional. Can be a directory or .zip file. See Sample "*Use Continuous Integration Principles with Code Generation*" |
| TC Build | PreferToolArchitectureX64 | Prefer X64 build tools | TRUE | Prefer X64 compiler and linker. Useful for complex source files, where X86 tools may run out of heap space. |
| | CppLanguageStandard | Specify C++ language standard version | Default | Enable supported C++ language features from the specified version of the C++ language standard |
| | Verbosity | Codegeneration and build verbosity | Normal | Verbosity level of code generation and build output messages. Silent and Detailed are other possible values. |
| | Publish | Run the publish step after project generation | TRUE | Start the build procedure after code generation for all selected platforms. The generated module binaries and module descpription files will get copied to the "publish folder". Published modules are automatically located by the XAE and can get instantiated in all TwinCAT 3 projects. If unset, the module generation process will be stopped after code generation. To instantiate in a TwinCAT3 project, the generated C++ project needs to be inserted and built from. |
| | PublishPlatformtoolset | Platform Toolset | Auto | Choose Platform Toolset to build binaries. |
| | PublishConfiguration | Build configuration | Release | Build configuration to build binaries. |
| | PublishTcRTx86 | TwinCAT RT (x86) | TRUE | Publish binaries for platform 'TwinCAT RT (x86).' [▶ 54] |

| Category | Name | Displayname | Default | Description |
|---|---|---|---|---|
| | PublishTcRTx64 | TwinCAT RT (x64) | TRUE | Publish binaries for platform 'TwinCAT RT (x64).' [▶ 54] |
| | PublishTcOSx64 | TwinCAT OS (x64) | TRUE | Publish binaries for platform 'TwinCAT OS (x64)' (e.g. TwinCAT/BSD) [▶ 54] |
| | PublishTcOSARMv8A | TwinCAT OS (ARMV8-A) (TwinCAT XAE >= 3.1.4026) | FALSE | Publish binaries for platform 'TwinCAT OS (ARMV8-A)' (requires TwinCAT XAE >= 3.1.4026) [▶ 54] |
| | ForceRebuildForPublish | Always rebuild all source files on publish | FALSE | Always rebuild all source files on publish |
| | PublishParallel | Build parallel to publish | TRUE | Build parallel on multiple cores to publish |
| | SignTwinCatCertName | Certificate name for TwinCAT signing | | Certificate name for TwinCAT signing with OEM Certificate level 2. [▶ 14] |
| | TmxInstall | Install TMX | TRUE | Install all generated TwinCAT Objects on local XAE (fill local Engineering Repository [▶ 40]). |
| | TmxArchive | TMX Archive | | Name of an optional archive containing all files required to use the generated TwinCAT Objects on another TwinCAT development system. [▶ 57] |
| | MsBuildPublishProperties | MsBuild publish properties | | Set additional MsBuild publish properties. |
| | MsBuildProjProperties | MsBuild project properties | | Set additional MsBuild project properties. |
| | PreCodeGenerationCallbackFcn | Pre code generation callback function | | The defined MATLAB® function is called before code generation [▶ 108]. |
| | PostCodeGenerationCallbackFcn | Post code generation callback function | | The defined MATLAB® function is called after code generation [▶ 108]. |
| | PostPublishCallbackFcn | Post publish callback function | | The defined MATLAB® function is called after publish [▶ 108]. |
| | DeploymentPath | Deployment project | | Full path to a TwinCAT project (.tsproj). Instances of the generated TcCOM Module in the specified project will be upgraded to the newly generated version [▶ 59] |
| | DeployRestart | Activate and restart deployment project | FALSE | If set the specified TwinCAT project will be activated and restarted on the configured target system [▶ 59] |
| | PostDeployCallbackFcn | Post deploy callback function | | The defined MATLAB function is called after deployment [▶ 108] |
| TC PLC library | LibCatPath | PLC library category description file | $<ProjectDir>\$<Name>.libcat.xml | Path to the PLC library category description file |

| Category | Name | Displayname | Default | Description |
|---|---|---|---|---|
| | LibraryCategories | PLC library categories | $<VendorName> | Define PLC library category hierarchy. Default only one hierarchy level = vendor. List separated with \| possible: <MainCategory>\| <SubCategory1>\|… |
| | GeneratePlcLibrary | Generate a PLC library | FALSE | Generate a PLC library with POUs. [▶ 131] Define containing POUs with parameter TcComWrapperFb and PlcFb>General>Generate. |
| | InstallPlcLibrary | Install the generated PLC library | FALSE | Install the generated PLC library for use in the local TwinCAT XAE/PLC [▶ 131]. |
| | PlcTypePrefixes | Type Prefixes | | Define custom type prefixes |
| | PlcVarPrefixes | Variable Prefixes | `PVOID=p \| BOOL=b \| BOOL32=b \| DATE=d \| TIME_OF_DATE=td \| TIME=t \| LTIME=t \| GUID=n` | Define custom variable prefixes. |
| TC License | OemId | ID of OEM | | ID of OEM. Required for OEM Licence checks [▶ 80] |
| | OemLicenses | IDs of OEM Licenses | | IDs of OEM Licenses. Multiple IDs may be inserted as a comma separated list. "{GUID},{GUID}" [▶ 80] |
| TC TcCom General | Generate | Generate TcCOM Module (TwinCAT Module Class) | TRUE | Generate a TcCOM module class for the model. |
| | OnlineChange | Online change support | FALSE | Allow to switch between different TcCOM module versions without switching TwinCAT runtime to config mode. |
| | ModuleProperties | TMC Properties | | Additional properties added to the module description in the TMC file: Name1=Value1\| Name2=Value2\|... |
| | GroupName | GroupName | TE140x\| Simulink Modules | Minor module group name in the TwinCAT XAE module dialog |
| | GroupDisplayName | GroupDisplayName | $<GroupName> | Minor module group description in the TwinCAT XAE module dialog |
| | GroupIcon | GroupIcon | $<TE140x:Icon> | Optional module group icon in the TwinCAT XAE module dialog |
| | ModuleIcon | ModuleIcon | $<TE140x:Icon> | Optional module icon in the TwinCAT XAE module dialog |

| Category | Name | Displayname | Default | Description |
|---|---|---|---|---|
| | InitExceptionHandling | Floating point exception handling during initialization | CallerExceptions | Configures how to throw, suppress or handle floating point exceptions during initialization [▶ 145]. |
| | UpdateExceptionHandling | Floating point exception handling during update | CallerExceptions | Configures how to throw, suppress or handle floating point exceptions during cyclic execution [▶ 145]. |
| | AdditionalIncludeFiles | Additional include files | | Additional files required to be included after rtwtypes.h |
| TC TcCom License | OemLicenses | IDs of OEM License | $<Project:OemLicenses> | IDs of OEM Licenses. Multiple IDs may be inserted as a comma separated list. "{GUID},{GUID}" [▶ 80] |
| TC TcCom Wrapper | TcComWrapperFb | TcCom Wrapper FB | FALSE | Generate a PLC Functionblock simplifying the interaction between a PLC and an instance of the generated TcCOM module [▶ 134] |
| | TcComWrapperFbProperties | TcCom Wrapper FB properties | FALSE | Generate properties for accessible data in the referenced TcCOM object [▶ 134] |
| | TcComWrapperFbPropertyMonitoring | TcCom Wrapper FB property monitoring | NoMonitoring | NoMonitoring: Online values of properties are not monitored in the PLC online view, CyclicUpdate: Update property values in the PLC online view cyclically, ExecutionUpdate: Update property values in the PLC online view when the property getter or setter is called [▶ 134] |
| TC TcCom Additional settings | ModuleCaller | Default module caller | CyclicTask | CyclicTask: Call module via TwinCAT Task. Module: Cal module from another TwinCAT module (see e.g. TcCOM-Wrapper-FB). |
| | CallerVerification | Verify caller | Default | Verify the caller context to prevent concurrent execution of the model code and corresponding DataArea mappings. Skip verification to reduce the execution time. |
| | StepSizeAdaptation | Default StepSize adaptation mode | RequireMatchingTaskCycleTime | Configure how to handle differences between the default model step size(s) and the cycle time of the assigned task(s). |
| | ExecutionSequence | Default execution sequence | UpdateBeforeOutputMapping | Configure the execution order of input mapping, model code execution and output mapping. |
| | ExecuteModelCode | Execute model code after startup | TRUE | Start cyclic execution of the model code after startup by default. If FALSE, Module |

| Category | Name | Displayname | Default | Description |
|---|---|---|---|---|
| | | | | Parameter Execute needs to be set to TRUE to start execution of code. |
| | BlockDiagramExport | Export BlockDiagram | TRUE | Export graphical block diagram information for monitoring and optional debugging on the generated TwinCAT module in TwinCAT XAE [▶ 118] |
| | ResolveMaskedSubsystems | Resolve Masked Subsystems | FALSE | Resolve masked subsystems in the block diagram |
| | ExtendSignalResolution | Extended resolution of signals in block diagram | FALSE | Intensified search for assignments of variables and block diagram signals (blue signals). This option increases the build time. [▶ 159] |
| | BlockDiagramVariableAccess | Access to VariableGroup not referenced by any block | AssignToParent | Variables from a block within an unresolved subsystem are either assigned to the next higher visible block or hidden in the block diagram. |
| | BlockDiagramDebugInfoExport | Export BlockDiagram debug info | TRUE | Export additional information required to debug the module using the block diagram [▶ 122]. |
| TC TcCom Interfaces | ExecutionInfoOutput | Create ExecutionInfo output | FALSE | Create additional output DataAreas containing execution and exception information [▶ 145]. |
| | MonitorExecutionTime | Monitor execution time | FALSE | Calculate and expose the execution time of the module as an ADS variable for monitoring purposes. |
| | InputDataAccess | Input: Data Access | Input Destination DataArea | Defines how the input variables are exposed in TwinCAT [▶ 67]. |
| | InputCreateSymbols | Input: Create ADS Symbols | TRUE | Create ADS symbol information for the input variables [▶ 67] |
| | InputInitValues | Input: Initial values | FALSE | Create module parameters for the input variables to allow definition of initial values [▶ 67] |
| | InputProperties | Input: TMC Properties | | Additional properties added to the Input symbol description in the TMC file. [▶ 91] |
| | OutputDataAccess | Output: Data Access | Output Source DataArea | Defines how the output variables are exposed in TwinCAT [▶ 67]. |
| | OutputCreateSymbols | Output: Create ADS Symbols | TRUE | Create ADS symbol information for the output variables [▶ 67]. |
| | OutputProperties | Output: TMC Properties | | Additional properties added to the Output symbol description in the TMC file. [▶ 91] |
| | ParametersDataAccess | Parameters: Data Access | Internal DataArea | Defines how the model parameter variables are exposed in TwinCAT [▶ 67] |

| Category | Name | Displayname | Default | Description |
|---|---|---|---|---|
| | ParametersCreateSymbols | Parameters: Create ADS Symbols | TRUE | Create ADS symbol information for the model parameter variables [▶ 67]. |
| | ParametersInitValues | Parameters: Initial values | TRUE | Create module parameters for the model parameter variables to allow definition of initial values [▶ 67]. |
| | ParametersProperties | Parameters: TMC Properties | | Additional properties added to the Parameters symbol description in the TMC file. [▶ 91] |
| | BlockIoDataAccess | BlockIO: Data Access | Internal DataArea | Defines how the BlockIO variables are exposed in TwinCAT [▶ 67] |
| | BlockIoCreateSymbols | BlockIO: Create ADS Symbols | TRUE | Create ADS symbol information for the BlockIO variables [▶ 67]. |
| | BlockIoProperties | BlockIO: TMC Properties | | Additional properties added to the BlockIO symbol description in the TMC file. [▶ 91] |
| | ContStateDataAccess | ContState: Data Access | Internal DataArea | Defines how the continuous state variables are in TwinCAT [▶ 67] |
| | ContStateCreateSymbols | ContState: Create ADS Symbols | TRUE | Create ADS symbol information for the continuous state variables [▶ 67]. |
| | ContStateProperties | ContState: TMC Properties | | Additional properties added to the ContState symbol description in the TMC file. [▶ 91] |
| | DWorkDataAccess | DWork: Data Access | Internal DataArea | Defines how the DWork variables are exposed in TwinCAT [▶ 67] |
| | DWorkCreateSymbols | DWork: Create ADS Symbols | TRUE | Create ADS symbol information for the DWork variables [▶ 67]. |
| | DWorkProperties | DWork: TMC Properties | | Additional properties added to the DWork symbol description in the TMC file. [▶ 91] |
| | DataStoreDataAccess | DataStore: Data Access | None | Defines how the DataStore variables are exposed in TwinCAT [▶ 67] |
| | DataStoreCreateSymbols | DataStore: Create ADS Symbols | TRUE | Create ADS symbol information for the DataStore variables [▶ 67]. |
| | DataStoreReadOnly | DataStore: Read Only | FALSE | Restrict ADS access to be read only for the DataStore variables [▶ 67]. |
| | DataStoreProperties | DataStore: TMC Properties | | Additional properties added to the DataStore symbol description in the TMC file. [▶ 91] |
| | SymbolProperties | Additional TMC Symbol Properties | | Additional properties added to specific symbol descriptions in the TMC file. [▶ 91] |

| Category | Name | Displayname | Default | Description |
|---|---|---|---|---|
| | VariableSymbolMapping | Mapping between variable names and ADS symbol names | Identical | Defines the TwinCAT symbol names for the generated C/C++ variables. 'Identical': Symbol name equals variable name, 'Classic': Use symbol names known from TE1400 Release 1.2.x.x [▶ 67] |
| TC TcCom External Mode | ExtModeRtAllowExecutionCommands | Allow RealTime execution commands via External Mode | FALSE | Allow to start and stop model code execution via External Mode [▶ 142]. |
| | ExtModeRtWaitForStart | Wait for RealTime execution start command via External Mode | FALSE | Wait for External Mode [▶ 142] connection before starting model code execution. |
| | ExtModeRtAllowForParameterChange | Allow to change parameters via External Mode | FALSE | Allow to change parameter online values via External Mode [▶ 142]. |
| TC PlcFb General | Generate | Generate TwinCAT PLC Function Block | TRUE | Generate a PLC-FB for the model [▶ 138]. |
| | InitExceptionHandling | Floating point exception handling during initialization | CallerExceptions | Configures how to throw, suppress, or handle floating point exceptions during initialization [▶ 145]. |
| | UpdateExceptionHandling | Floating point exception handling during update | CallerExceptions | Configures how to throw, suppress, or handle floating point exceptions during cyclic execution [▶ 145]. |
| TC PlcFb License | OemLicenses | IDs of OEM License | $<Project:OemLicenses> | IDs of OEM Licenses. Multiple IDs may be inserted as a comma seperated list. "{GUID},{GUID}" [▶ 80] |
| TC PlcFb Additional settings | MonitorExecutionTime | Monitor ExecutionTime | FALSE | Calculate and expose the execution times of TwinCAT modules as an ADS variable for monitoring purposes. |
| PlcFb->Interface | InputAttributes | Input variables: PLC Attributes | | Additional attributes added to the PLC FB Input variables. |
| | OutputAttributes | Output variables: PLC Attributes | | Additional attributes added to the PLC FB Input variables. |
| TC PlcFb External Mode | ExtModeRtAllowExecutionCommands | Allow RealTime execution commands via External Mode | FALSE | Allow to start and stop model code execution via External Mode [▶ 142]. |
| | ExtModeRtWaitForStart | Wait for RealTime execution start command via External Mode | FALSE | Wait for External Mode connection before starting model code execution [▶ 142]. |
| | ExtModeRtAllowForParameterChange | Allow to change parameters via External Mode | FALSE | Allow to change parameter online values via External Mode [▶ 142]. |

## 4.7.2 通过 m 文件为代码生成设置参数

通过 m 文件（或 mlx 文件）可以使用两种方法对代码生成进行参数化：

- 通过特定模型的 Simulink® 参数与 `set_param`
- 通过模块生成器的实例 `TwinCAT.ModuleGenerator`

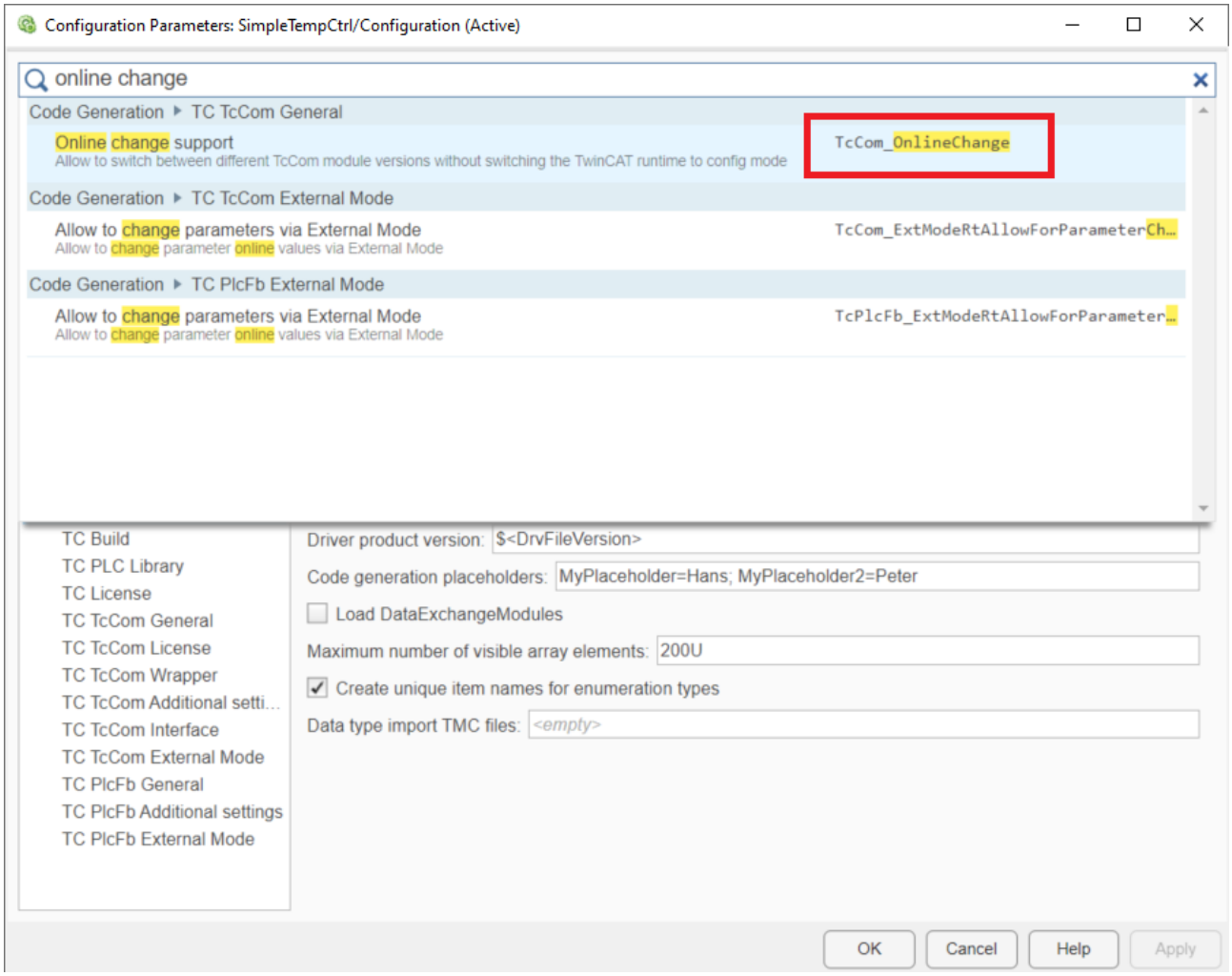
### 通过 Simulink® 参数进行配置

可以使用 `set_param` 为 Simulink® 中的对象分配特定参数。如果使用 `set_param` 配置模块生成器，它将相应地存储在 Simulink® 模型中。

如要规划配置参数 [▶ 43]的结构，这里不能使用命名空间，因此要为配置参数设置前缀。根据级别的不同，参数名称（配置参数 [▶ 43]表中的“名称”列）的前面会加上 `Project_`，`TcCom_` 或 `TcPlcFb_`。

- `Project_` 前缀：用于分组在 TC General、TC Build、TC PLC Library 和 TC License 选项卡的配置参数中显示的所有参数。
- `TcCom_` 前缀：用于分组在 TC TcCom 选项卡的配置参数中显示的所有参数。
- `TcPlcFb_` 前缀：用于分组在 TC PlcFb 选项卡的配置参数中显示的所有参数。

还可以使用 Simulink® 模型中的搜索功能找出准确的参数名称。




### 示例

```
% load Simulink model
controller = load_system('TempCtrl.mdl');

% configure TwinCatGrt
set_param(controller,'SystemTargetFile','TwinCatGrt.tlc');

% set project specific parameters
set_param(controller,'TcProject_VendorName','CompanyName');
```

```
set_param(controller,'TcProject_GeneratePlcLibrary','on');
set_param(controller,'TcCom_OnlineChange','on');
set_param(controller,'TcPlcFb_MonitorExecutionTime','on');

% build the model
slbuild(controller);

% save and close the model
close_system(controller,1);
```

### 通过模块生成器进行配置

通过模块生成器进行上述示例中的参数设置。因此，这些设置不会保留在 Simulink® 模型中，而是保留在模块生成器的实例中。

为此，仅为 Simulink® 模型定义 SystemTargetFile，并关闭参数 TcProject_Generate。这样只会生成*代码结果*，但不会生成 TwinCAT C++ 项目，也不会进行编译。代码工件存储在当前 MATLAB® 路径下的文件夹 *<ModelName>_tcgrt* 中。

**导出框图**

在 Simulink® 模型层，还应决定是否要导出框图。以下步骤中使用的是*代码结果*，而不再是 Simulink® 模型。

之后可以指定*代码结果*文件夹，将 ProjectExport 配置加载到 `TwinCAT.Modulgenerator`。然后，可在此进行设置并编译项目。

### 示例

```
% load Simulink model
controller = load_system('TempCtrl.mdl');

% configure TwinCatGrt
set_param(controller,'SystemTargetFile','TwinCatGrt.tlc');

% disable generation of C++ project files for each model (suppresses build)
set_param(controller,'TcProject_Generate','off');

% create code artifacts
slbuild(controller);

% save and close the model
close_system(controller,1);

% find the code artifacts in the existing code generation directories
controllerBuildDir = fullfile(pwd,'TempCtrl_tcgrt');

% load existing export configurations
controllerCfg = TwinCAT.ModuleGenerator.ProjectExportConfig.Load(controllerBuildDir);

% show complete configuration in MATALB Command Window
controllerCfg

% set project specific parameters
controllerCfg.Project.VendorName = "Company";
controllerCfg.Project.GeneratePlcLibrary = true;
controllerCfg.ClassExportCfg{1}.TcCom.OnlineChange = true;
controllerCfg.ClassExportCfg{1}.PlcFb.MonitorExecutionTime = true;

% set generate to true
controllerCfg.Project.Generate = true;

% instantiate and run the project exporter
TwinCAT.ModuleGenerator.ProjectExporter(controllerCfg);
```

### 应用举例

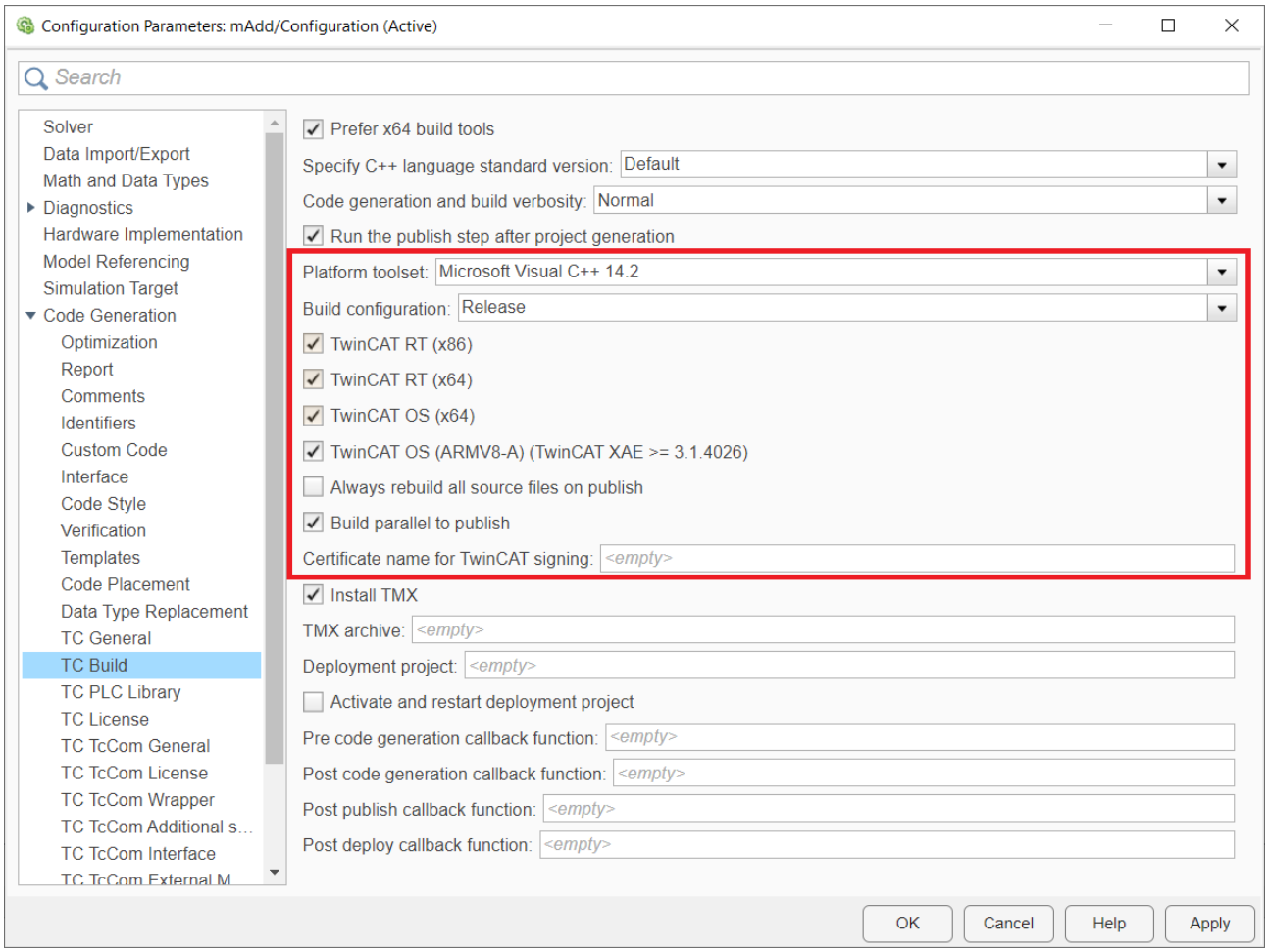
代码生成过程分为以下两个步骤：（1）创建*代码结果*；（2）配置模块生成器并创建 TwinCAT objects，在不同的应用场景下非常有帮助：

- 不希望在每个 Simulink® 模型中都保存模块生成器的设置。
- 希望将多个 Simulink® 模型合并到一个项目中。 [▶ 55]这样可以将所有模型整合到一个驱动程序和一个 PLC 库中。示例：`TwinCAT.ModuleGenerator.Samples.Start('Combine Multiple TwinCAT Classes In Libraries')`

- 您正在使用 Build Server 或 CI/CD 系统。示例：
  `TwinCAT.ModuleGenerator.Samples.Start("Use Continuous Integration Principles with Code Generation")`

## 4.7.3 为不同平台构建

在 **TC 构建** (TC Build) 选项卡下，您可以进行各种设置，以便指定目标系统平台应使用哪种编译器。




**平台工具集：**

默认“自动”选项始终使用最新版本。也可通过下拉菜单选择特定版本。

**相关版本**

Microsoft Visual C++ 14.1 => VS 2017
Microsoft Visual C++ 14.2 => VS 2019
Microsoft Visual C++ 14.3 => VS 2022

**构建配置：**

发布构建结果或调试驱动程序 请注意，调试驱动程序在目标系统上的执行时间通常要慢得多。

**始终在发布时重建所有源文件：**

Simulink® Coder™ 和 TwinCAT Target for Simulink® 会缓存一些文件，以加快重复构建的处理速度。选择此参数可始终重建所有源文件。

**构建并行发布：**

如果选择多个构建平台，则编译流程相互独立，依次执行。激活该选项后，所有构建流程都会在开发系统上并行执行。由操作系统执行并行化。为了避免不必要的等待时间，您应该只为实际使用的平台创建二进制文件。

**用于 TwinCAT 签名的证书名称**

如果未通过环境变量指定，请在此处输入 OEM 2 级证书，以签署所创建的二进制文件，请参见 设置驱动程序签名 [▶ 17]。

**TwinCAT RT（x86）**

选择该平台可为具有以下规格的目标系统构建二进制文件：Intel 或 AMD CPU、Windows 32 位操作系统（TwinCAT 平台级别 40 及以上）。

**TwinCAT RT（x64）**

选择该平台可为具有以下规格的目标系统构建二进制文件：Intel 或 AMD CPU、Windows 64 位操作系统（TwinCAT 平台级别 40 及以上）。

**TwinCAT 操作系统 (x64)**

选择该平台可为具有以下规格的目标系统构建二进制文件：Intel 或 AMD CPU、TwinCAT/BSD 操作系统（TwinCAT 平台级别 40 及以上）。

**TwinCAT 操作系统 (ARMV8-A)**

选择该平台可为具有以下规格的目标系统构建二进制文件：Arm® Cortex®-A CPU、Beckhoff RT Linux® 操作系统（TwinCAT 平台级别 20 和 30）。

**不需要对 Beckhoff RT Linux® 进行驱动程序签名**

Beckhoff RT-Linux® 不需要签署驱动程序。其他平台都必须进行签名。

为支持 TwinCAT OS (ARMV8-A) 平台（用于 CX82xx 和 CX9240 的编译），安装 Visual Studio 时需要使用 MSBuild Support for llvm (clang-cl) 工具集组件。同时必须在安装过程中进行手动选择。

TwinCAT OS (ARMV8-A) 平台仅支持 TwinCAT 3.1 Build 4026。

## 4.7.4 在一个 TwinCAT 驱动程序中捆绑多个模型

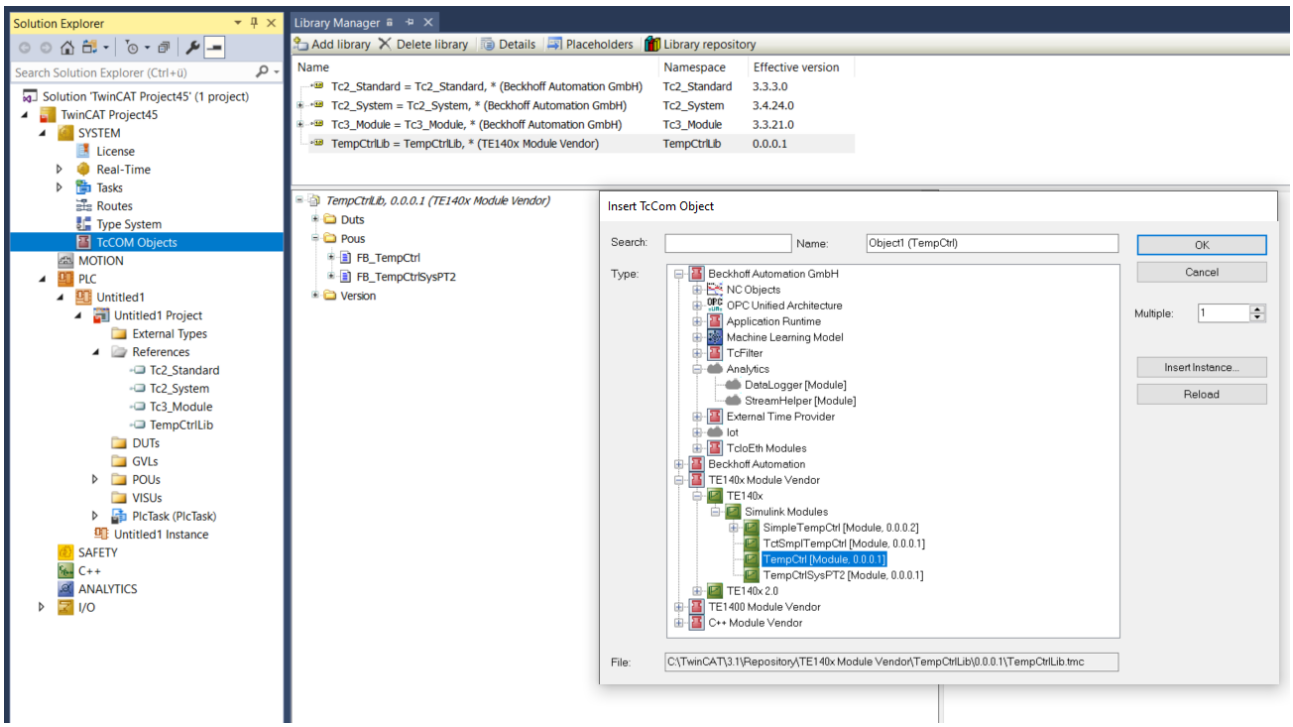
Simulink® 模型和 MATLAB® 功能自动生成的代码可以绑定在同一个 C++ 项目中。构建流程结束后，所有绑定对象都可以在一个驱动程序中使用。

**MATLAB® 中的示例代码**

通过以下命令打开相应的示例：`TwinCAT.ModuleGenerator.Samples.Start('Combine Multiple TwinCAT Classes In Libraries')`

**绑定在一起的优势**

创建 PLC 库后，所有创建的对象都将作为功能块（FB）在库中列出。虽然只创建了一个驱动程序和一个 tmc 文件，但仍可通过**系统 > TcCOM**（System > TcCOM）分别对所有模块进行实例化，也就是说，从用户的角度来看，在 TwinCAT XAE 中使用 TcCOM 对象没有任何变化。使用 PLC 库可以更加明确。




**绑定在一个驱动程序中的优势：**

- 资源库目录中的文件数量显著减少。这也意味着，为了能在多个工程系统中提供大量可用模块，需要复制到其他工程系统的文件数量会更少。
- 不同版本的管理也得到了简化，因为可以在一个捆绑包中交换交互的模块，这样就不会发生版本冲突。

**注意**

**Simulink® Coder™ 不支持命名空间**

如果在多个模型中定义了具有相同命名的数据类型或函数，构建过程就会失败，因为这些定义都在相同的命名空间中。

**操作步骤**

1. 在 Simulink® 中禁用“项目生成后运行发布步骤”。这样代码生成后就会终止，将不会编译创建的 C++ 项目。
2. 绑定多个模块时，只需使用生成的文件夹 *<modelname>_tcgrt*，这些文件夹位于当前 MATLAB® 路径下。
3. 使用 ModuleGenerator 在项目中加载、绑定和进行导出配置（*<modelname>_tcgrt*）。
4. 创建一个导出项目。

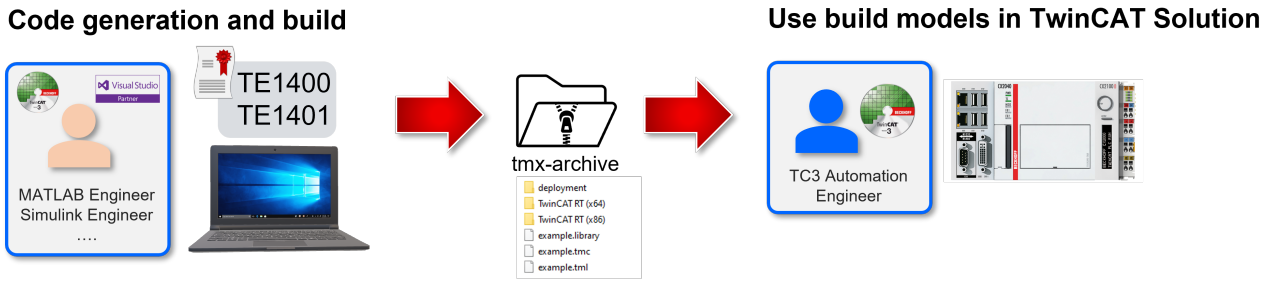
下文以绑定 2 个导出配置为例进行说明：

```
% find the code artifacts in the existing code generation directories
controllerBuildDir = fullfile(pwd,'TempCtrl_tcgrt');
ctrlsystemBuildDir = fullfile(pwd,'TempCtrlSysPT2_tcgrt');
% load existing export configurations
controllerCfg = TwinCAT.ModuleGenerator.ProjectExportConfig.Load(controllerBuildDir);
ctrlsystemCfg = TwinCAT.ModuleGenerator.ProjectExportConfig.Load(ctrlsystemBuildDir);
% create a new project export configuration
combinedCfg =
TwinCAT.ModuleGenerator.ProjectExportConfig('FullPath',fullfile(pwd,'TempCtrlLib','TempCtrlLib.vcxpr
oj'));
% add the loaded class export configurations to the new project configuration
combinedCfg.AddClassExportConfig(controllerCfg.ClassExportCfg{1});
combinedCfg.AddClassExportConfig(ctrlsystemCfg.ClassExportCfg{1});
% ...
% additinal class export configurations can be added here, loaded from
% - Simulink code generation directories (as described)
% - MATLAB code generation directories (from MATLAB Coder with TE1401) in the same way
% turn on generation and installation of the PLC library
combinedCfg.Project.GeneratePlcLibrary = true; % generate a PLC Lib true/false
combinedCfg.Project.InstallPlcLibrary = true; % install the PLC lib on local system true/false
% instantiate and run the project exporter
TwinCAT.ModuleGenerator.ProjectExporter(combinedCfg);
```

## 4.7.5 共享已创建的 TwinCAT objects

从 Simulink® 或 MATLAB® 创建 TwinCAT objects（TcCOM 和/或 PLC 库）的同事往往并不是将创建的模块实施到 TwinCAT 配置中并创建设备程序的人。

为了能够在 TwinCAT XAE 中使用创建的 TwinCAT objects，本地开发用 PC 的存储库文件夹中必须提供这些对象，而且 PLC 库必须安装在本地 PLC 库存储库中。手动复制到开发用 PC 容易出错。因此，强烈建议创建所谓的 *TMX 压缩包*，并与要使用已创建模块的同事共享。

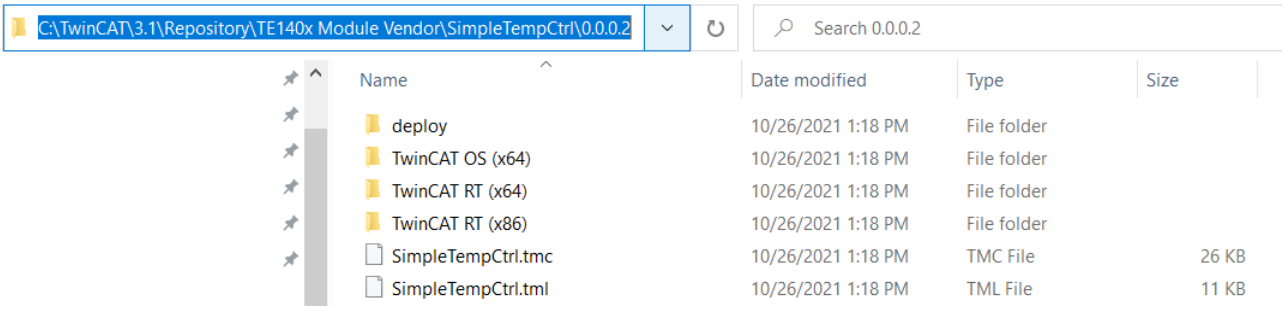



### 什么是 TMX 压缩包?

TMX 压缩包是一个包含所有必要文件的压缩包，在 TwinCAT Engineering 中使用 TcCOM 和 PLC 库需要这些文件。压缩包中包含所有已编译的驱动程序、描述文件（如 TwinCAT 中的框图）、用于 TcCOM 的 tmc 文件或用于 PLC 的 tml 或库文件。

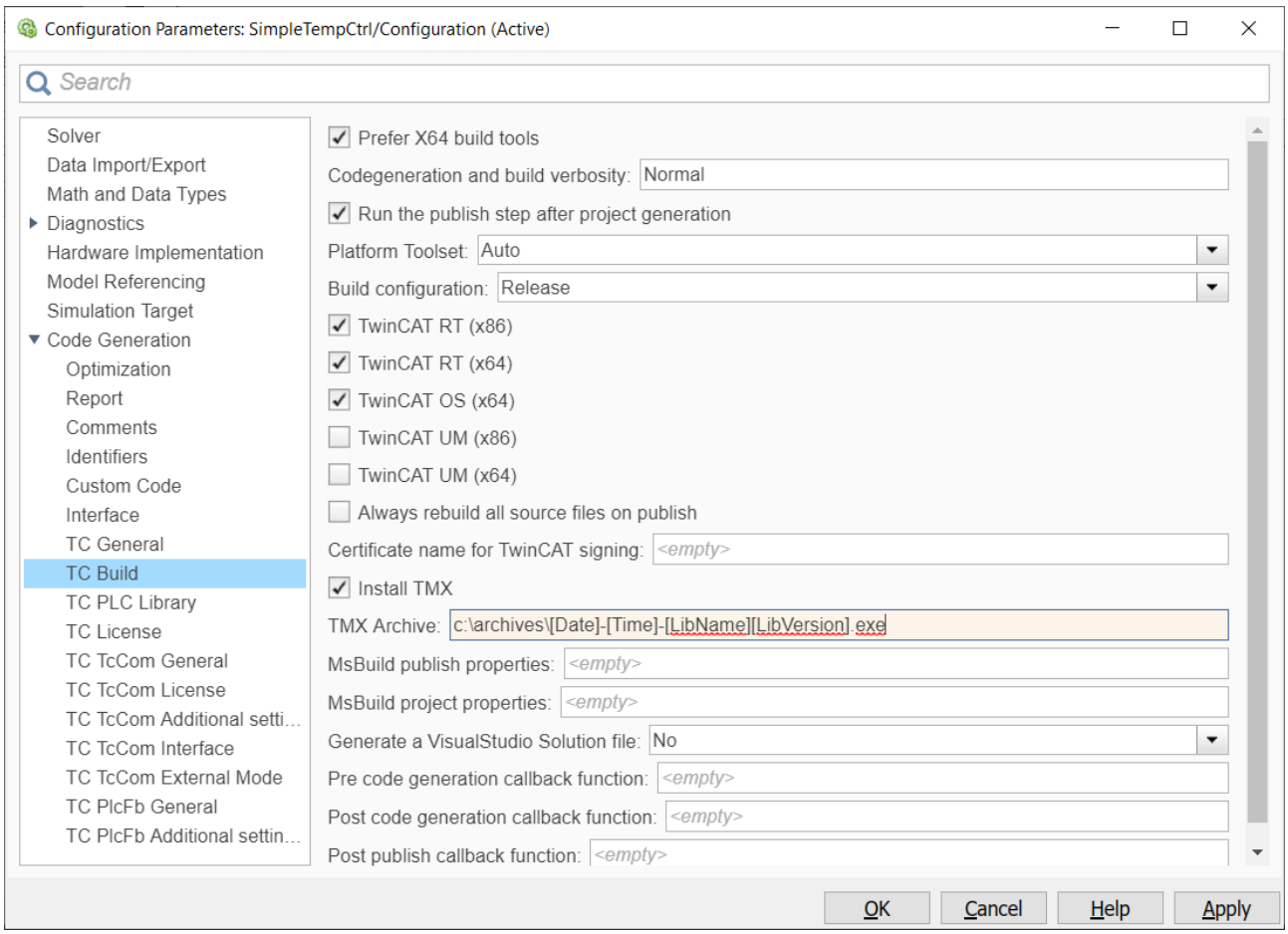
只需将 TMX 压缩包复制到开发用 PC 上的任意路径并执行即可。它是一个自解压文件包，随后会将所有文件自动复制到正确的定位。

例如，模型 *SimpleTempCtrl* 的文件（来自供应商 TE140x Module Vendor 的 0.0.0.2 版本）必须位于此位置：




### 如何创建 TMX 压缩包?

可以在 TwinCAT Build 下指定 TMX 压缩包的路径和名称，以便在下一次构建时创建该压缩。



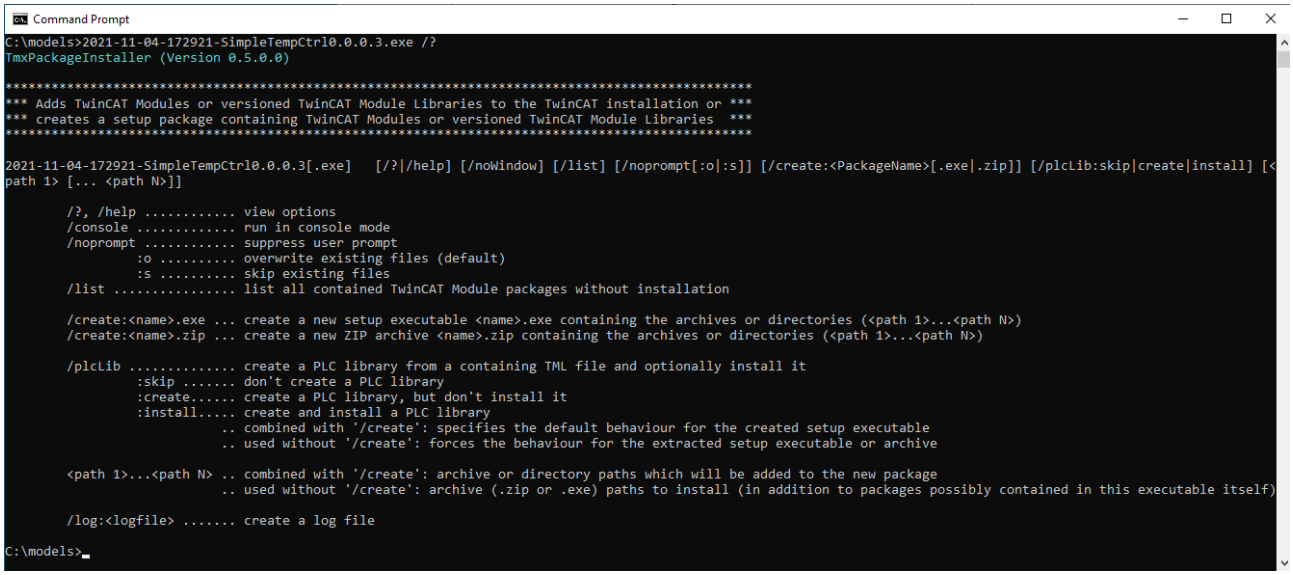

也可以如以上示例所示，使用占位符来表示路径和名称。例如，设置后生成 TMX 压缩包 *2021-11-4-172921-SimpleTempCtrl0.0.0.3.exe*（新的编译版本，因此修订版本号递增）。

### 如何使用 TMX 压缩包?

可将 TMX 压缩包复制到开发用 PC 上的任意路径并执行它。这会将压缩包中的文件复制到存储库中的正确位置。（自解压文件包）。如果不仅要在开发用 PC 上使用压缩包中的 TcCOM 模块，还要使用 PLC 库，则必须仍需通过 PLC 库资源库进行安装。

或者也可以使用**命令提示符**。通过以下命令调用 **TmxPackageInstaller**：

```
<tmxarchive>.exe /?
```



例如，要解压 TMX 压缩包并将其中包含的 PLC 库安装到本地 TwinCAT Engineering 中，应在命令行中执行以下命令：

```
<tmxarchive>.exe /plclib:install
```

**TwinCAT 开发用 PC 上需要哪些软件?**

在要使用已编译好的 TwinCAT objects 并将其应用于 TwinCAT 配置的开发用 PC 上，只需安装 TwinCAT XAE 即可。

这台开发用 PC 不需要完整版 Visual Studio（TwinCAT XAE 安装程序中的 XAE Shell 即可），也不需要安装 MATLAB®。

在某些情况下，需要在开发用 PC 上安装 **DataExchange modules**，以便使用在另一个系统上创建的 TwinCAT objects。

这些情况包括：

- 创建模块时使用了“External Mode”选项。
- 创建的模块使用 TwinCAT File Writer [▶ 35] 或 MAT 文件记录。
- 创建的模块包含来自 TE1410 TwinCAT Interface for MATLAB®/Simulink® 的功能块。

在其他情况下，创建的 TwinCAT 对象与 DataExchange 模块没有依赖关系。

**安装 DataExchange 模块**

TwinCAT 3.1 Build 4026

```
tcpkg install TwinCAT.XAE.TMX.DataExchange
```

TwinCAT 3.1. Build 4024

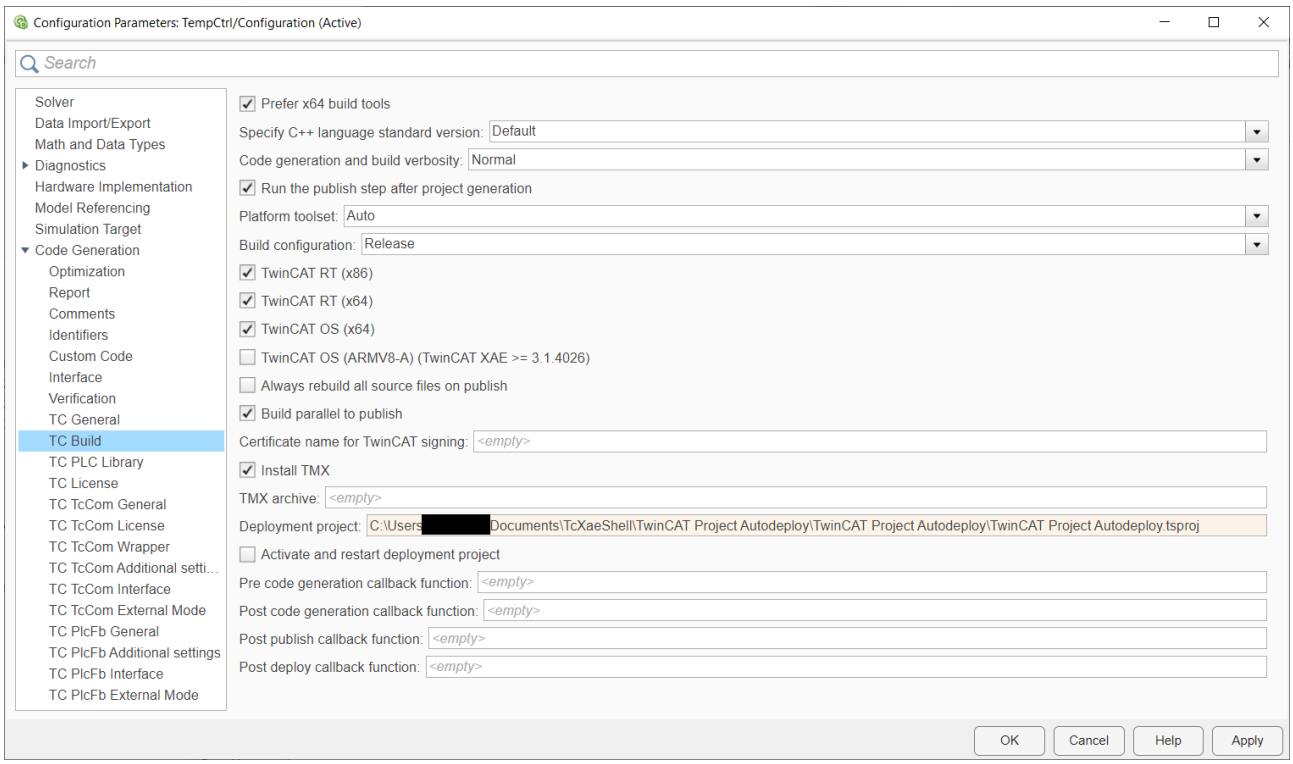
DataExchange-Modules 安装程序随着 MATLAB® 和 Simulink® 安装程序专用 TwinCAT 工具复制到以下文件夹，这样安装了 TE14xx-ToolsForMatlabAndSimulink 安装程序的员工便可将 DataExchange 模块安装程序分发给相关同事。

```
<TwinCATInstallDir>\TwinCAT\Functions\TE14xx-
ToolsForMatlabAndSimulink\TE140x\SDK
```

## 4.7.6 自动更新 TwinCAT 项目中的对象

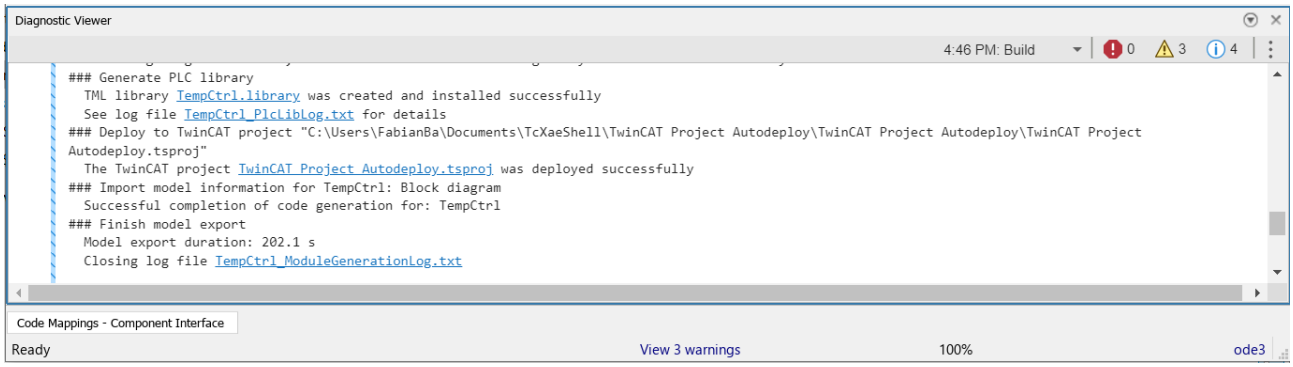
✓ 在工程设计阶段，通常要对 Simulink® 模型进行多次修改，模型建立后再在 TwinCAT 项目中进行更新。可使用**部署项目**参数实现此流程的自动化。

1. 在相应字段中输入 TwinCAT 项目（tsproj 文件）的完整路径。




2. 然后像往常一样构建 Simulink® 模型。

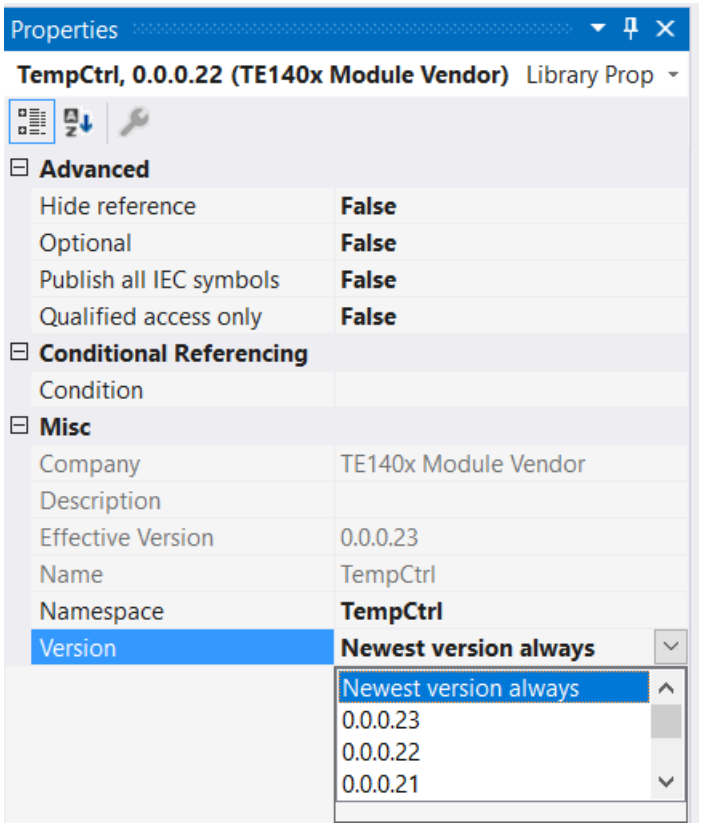
   ⇨ 日志文件中显示一个新的部分，可提供有关自动更新 TwinCAT 项目流程的信息。




   ⇨ 在指定地址的 TwinCAT 项目的更新过程中，所有新建 Simulink® 模型的现有 TcCOM 实例都会自动更新。保留所有现有的映射和任务分配。

3. 设置**激活并重新启动部署项目**参数，以便在更新对象后编译 TwinCAT 项目，并在设定的 TwinCAT 目标上激活该项目。

4. 如果您正在使用 PLC-FB 和/或 TcCOM-Wrapper-FB，请在相应 PLC 库的属性项目中将版本设置为“始终为最新版本”，以便自动更新。




## 4.7.7 创建版本控制的驱动程序

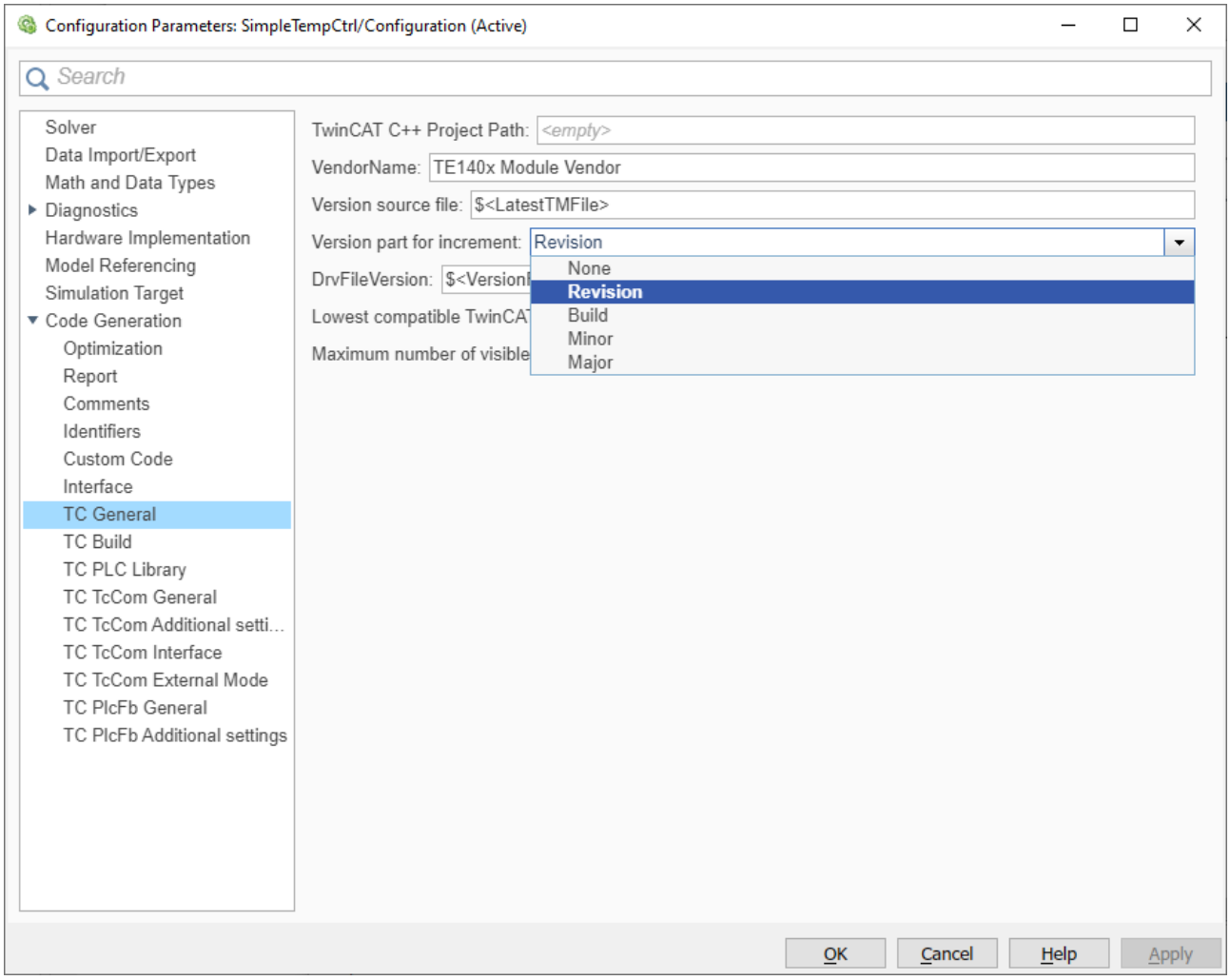
从 Simulink® 创建的每个对象都包含版本信息。可以相应地构建多个版本的 Simulink® 模型，并在 TwinCAT 中选择性地实例化所创建的模块版本。

### 在 Simulink® 中定义修订控制

切换到高级模式可显示以下参数。

```
TwinCAT.ModuleGenerator.Settings.Change('ConfigurationLevel', 'Advanced')
```

在创建 PLC 功能块或 TcCOM 对象之前，可在 TC General 选项卡下通过 Version source file 和 Version part for increment 条目定义 TcCOM 和所创建 PLC 库的版本。




**版本自动递增**

通过“Version source file”指定要创建版本更新的基本版本。在标准情况下，这里指定的是 $<LatestTMFile>。这将在本地开发用 PC 上搜索模型的最新可用版本，然后以此为基础进行版本递增。

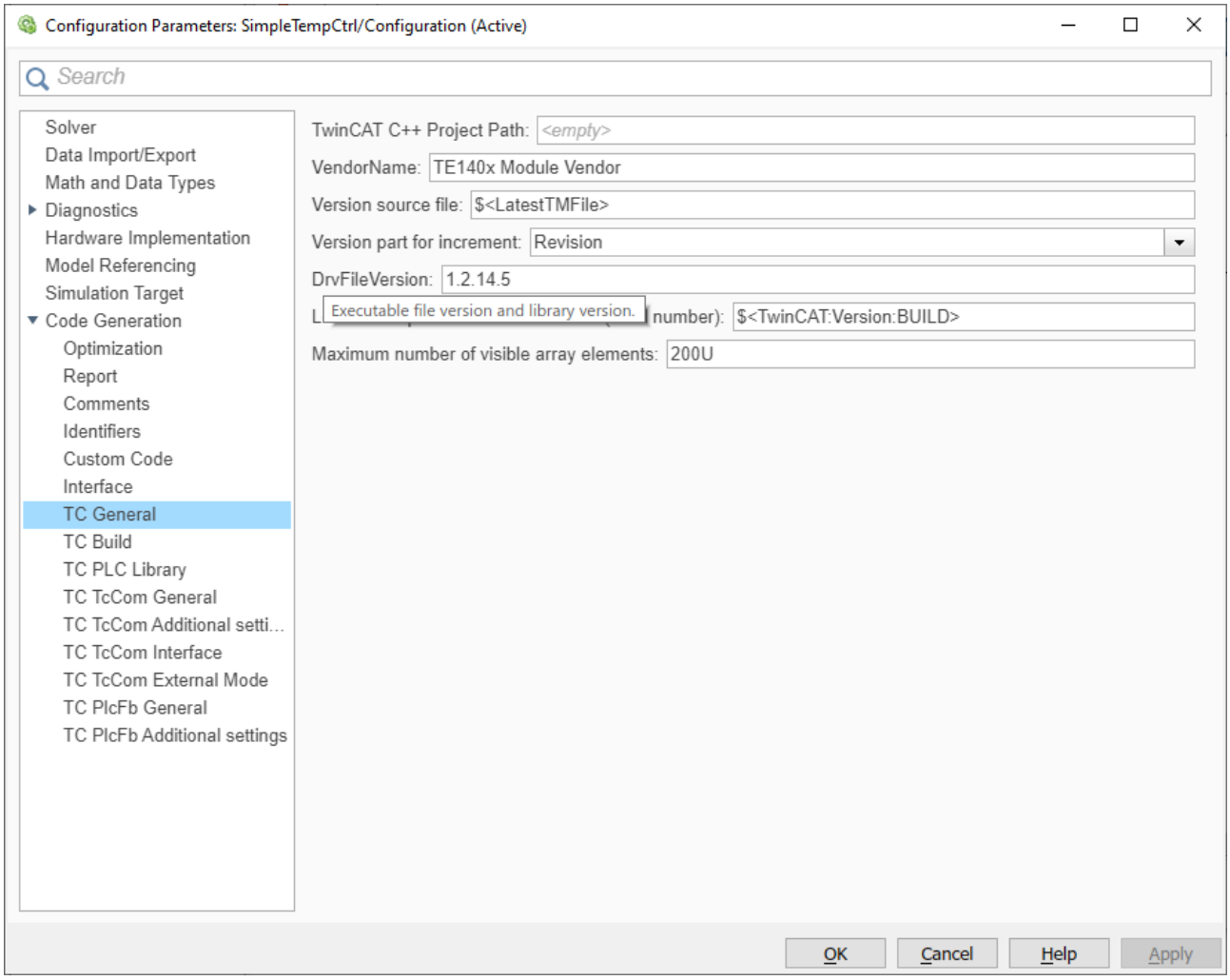
版本号由四位数字组成，如 1.0.3.2 或 2.12.123.14。每一位都可以按照以下方案单独递增：<主版本号>、<次版本号>、<内部版本号>、<修订版本号>

例如，如果发现开发用 PC 上名为“MyModelXY”的模型最新版本是 1.2.12.4，且设置为“修订版本”递增，则会创建版本 1.2.12.5。

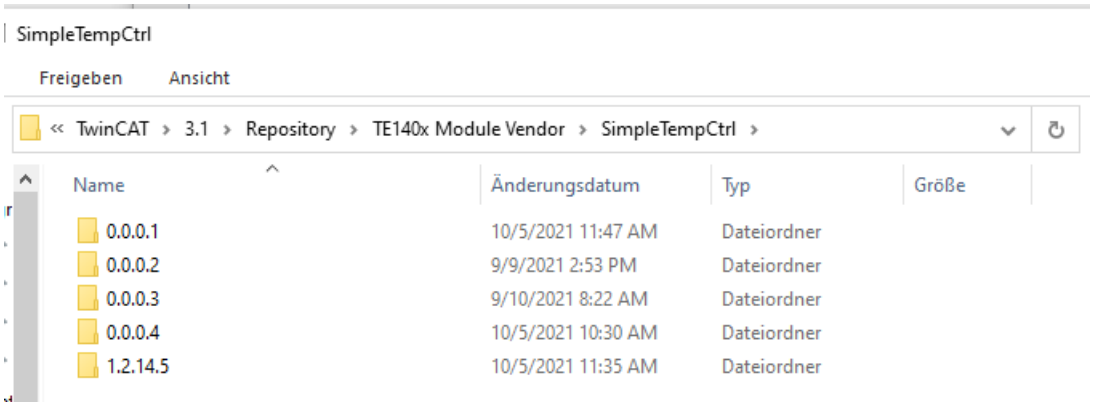
如果选择“None”，则不进行版本升级，开发用 PC 上的最新版本将被覆盖。

**固定版本号的默认值**

如果要在 Simulink® 中指定版本，可通过 **DrvFileVersion** 指定。只需在输入字段中输入目标版本即可。



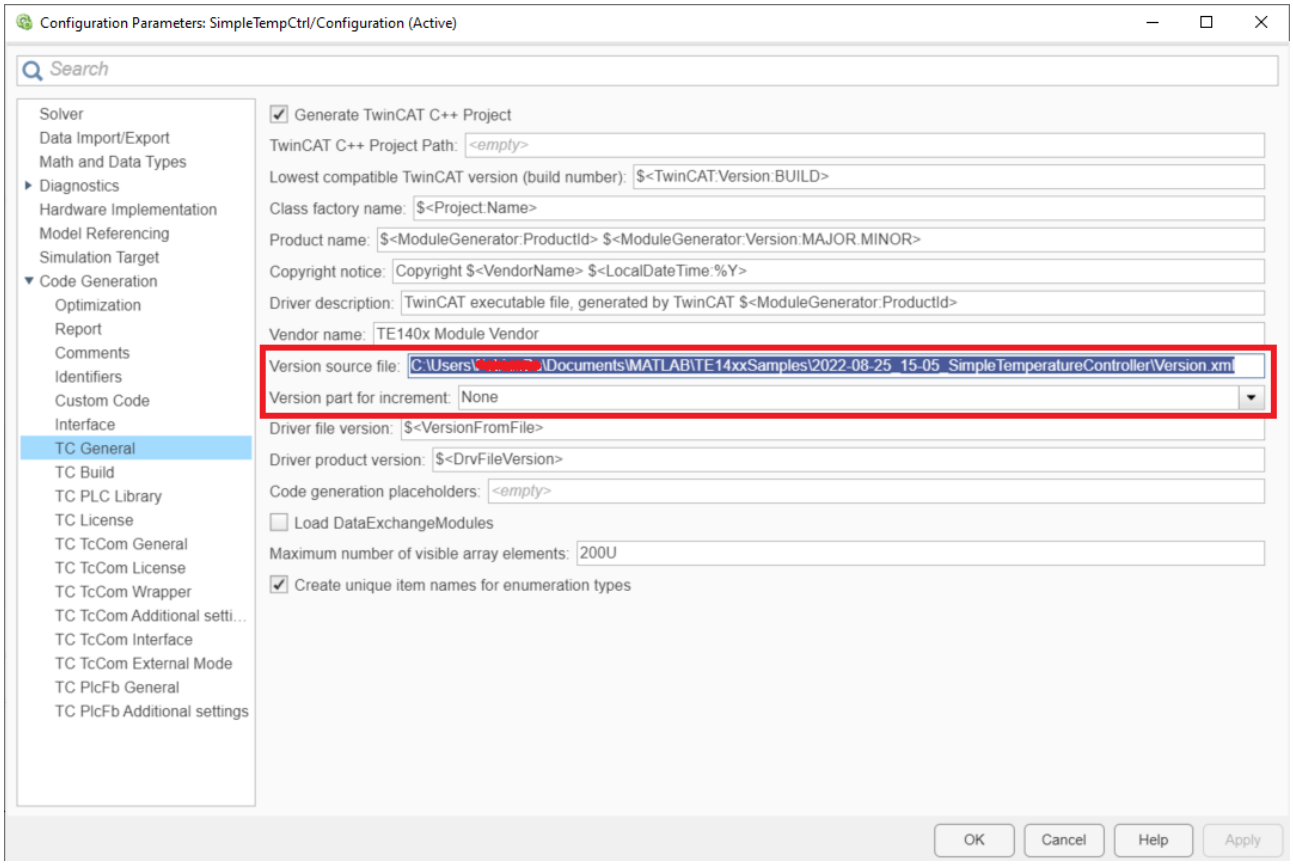

在工程存储库中，会为每个创建的版本在模型名称下创建一个文件夹。每个版本文件夹都包含相应的驱动程序和 TwinCAT 文件。另请参见 TwinCAT objects [▶ 40]。




**通过外部文件预设版本号**

也可以使用外部文件来指定版本。例如，在*持续集成*过程中使用构建代理时，这是一种常用方法。

在 Simulink® 中进行配置时，将 TC General 下的**版本递增部分**（Version part for increment）设为 None，并在**版本源文件**（Version source file）中指定版本文件的完整路径。



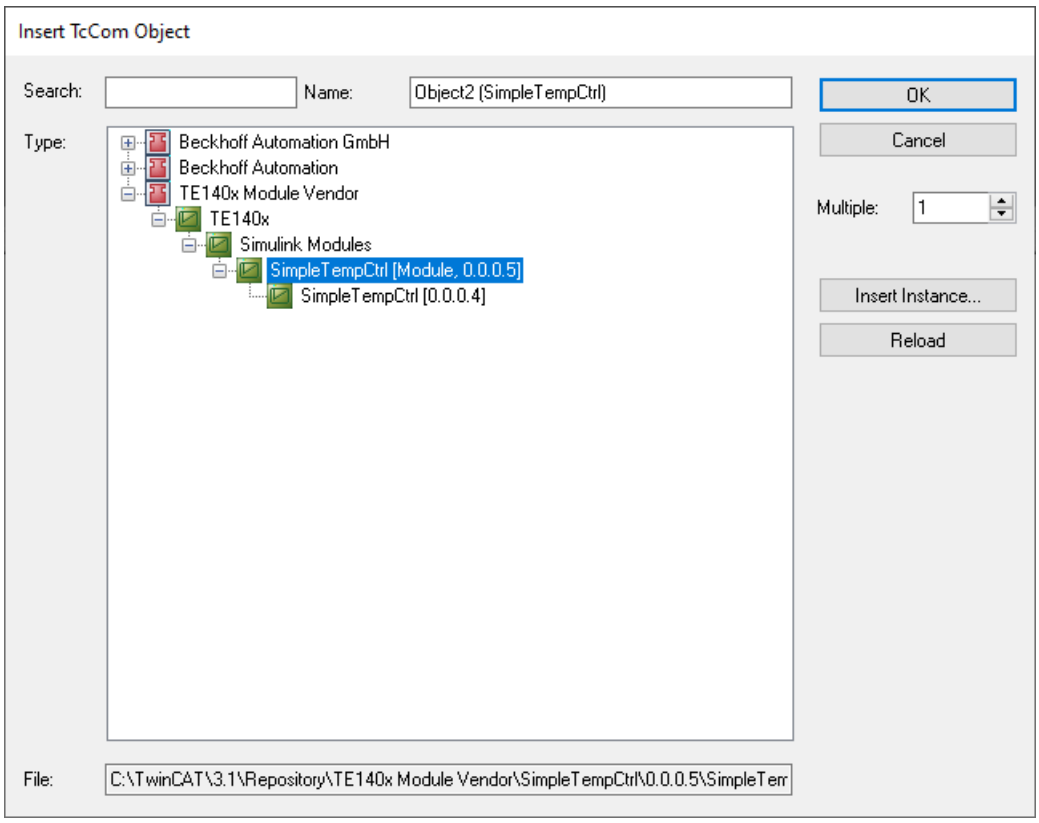

外部文件的结构：

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
  <appSettings>
    <add key="major" value="1" />
    <add key="minor" value="3" />
    <add key="build" value="1301" />
    <add key="revision" value="267" />
  </appSettings>
</configuration>
```

### 在 TwinCAT XAE 中使用版本控制模型

工程存储库中的所有可用模型都可以在任何可用版本中实例化。如要进行实例化，正常浏览到树形结构，找到您选择的 TcCOM。现在，还可以在最后一个层级选择 TcCOM 的版本。

此处提供两个版本（0.0.0.4 和 0.0.0.5）的 SimpleTempCtrl 为例：




## 4.7.8 在线更改 TcCOM

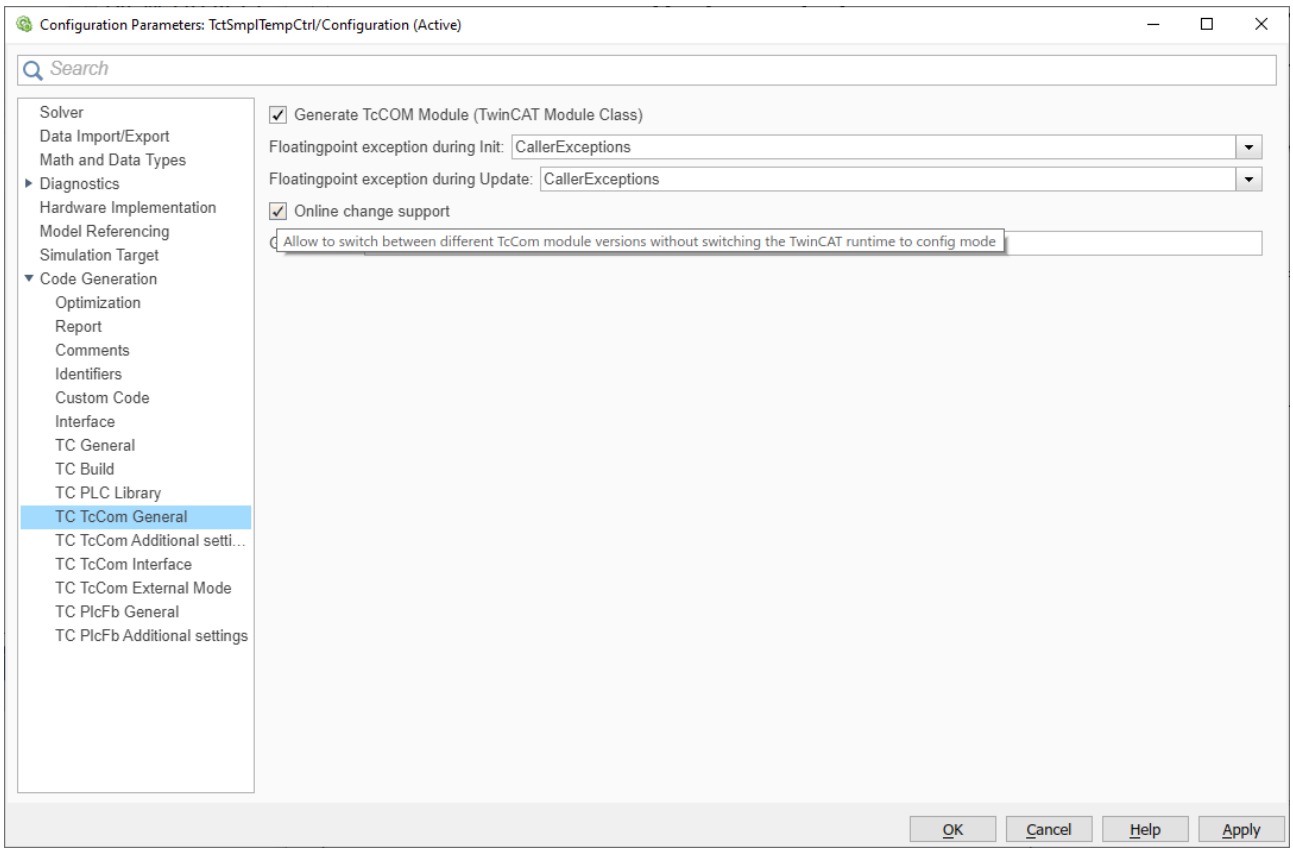
**在 TwinCAT Run 模式期间在线更改 TcCOM**

✓ 如要在运行期间切换不同版本的 TcCOM，必须实现相应的接口。

1. 为此，请激活 **TC TcCom 常规**（TC TcCom General）菜单下的**支持在线更改**（Online change support）复选框（在 Simulink® 中）。

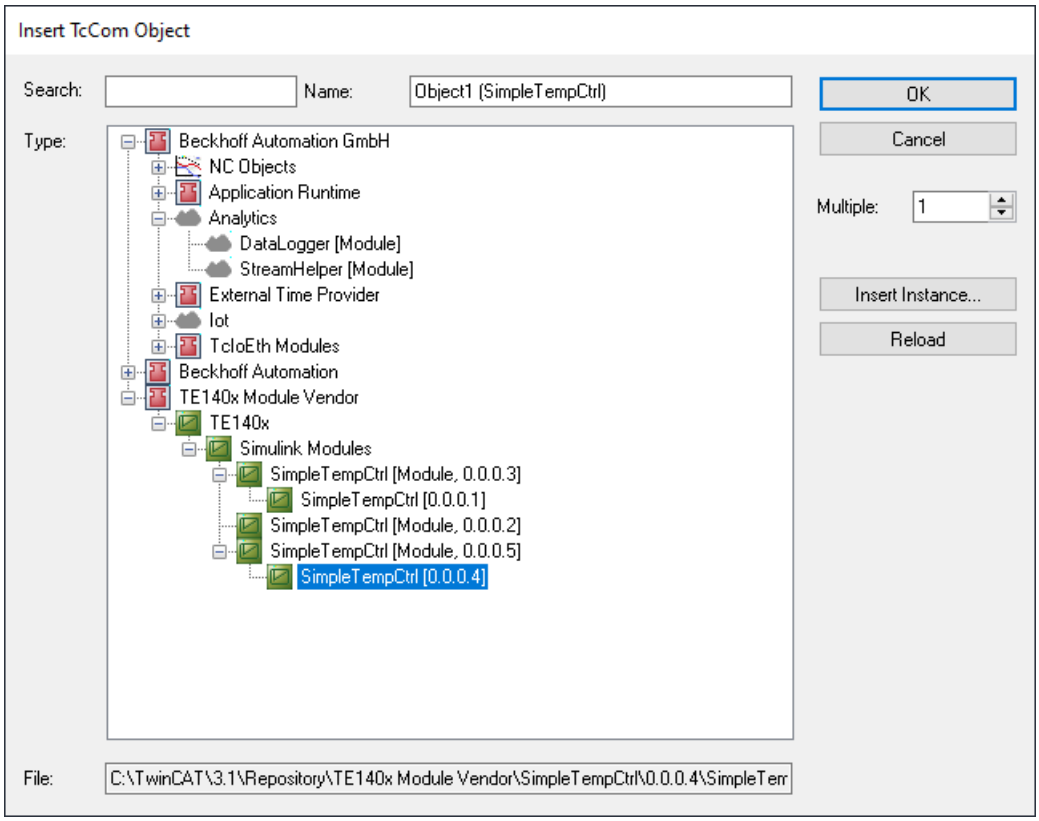
**在线更改 PLC 功能块**

如果在版本控制 PLC 库中使用功能块（PLC-FB），则不必选中复选框**支持在线更改**（Online change support）。PLC 特有的机制实现在线更改过程。

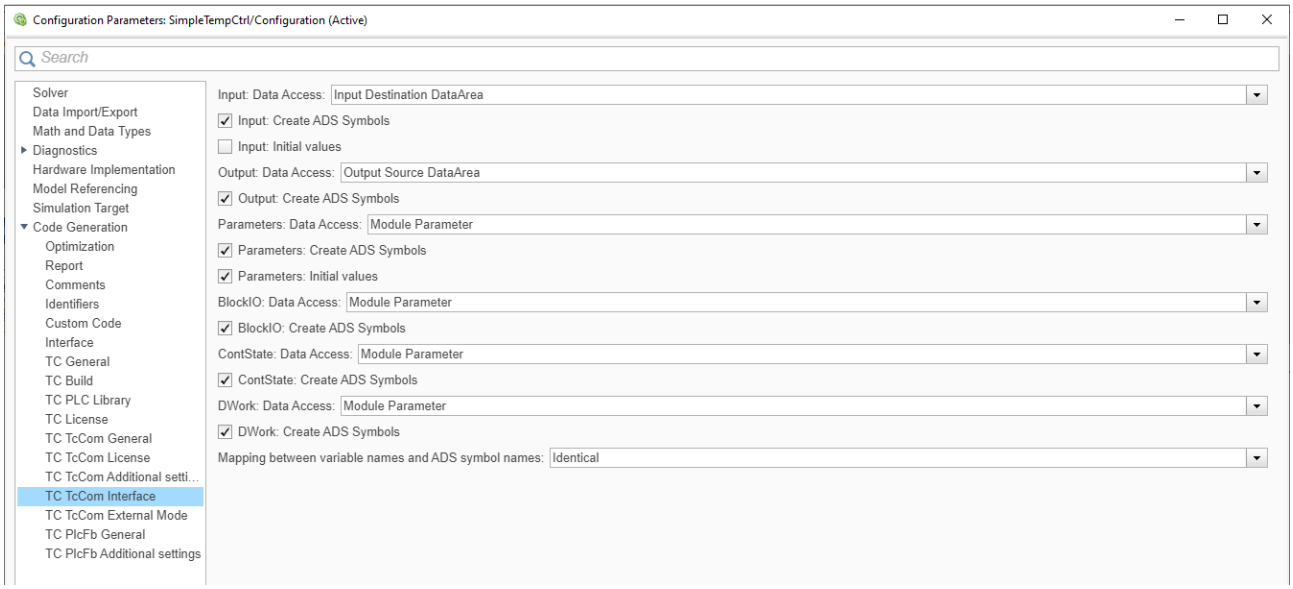



此外，创建的 TcCOM Data Area 必须相互兼容。如果激活了**支持在线更改**（Online change support），则在“插入 TcCom 对象”（Insert TcCom Object）对话框中会更严格地区分最后一个层级。只有可进行在线更改的 TcCOM 才能合并。

以下显示的版本均可进行在线更改：0.0.0.1 和 0.0.0.3 或 0.0.0.4 和 0.0.0.5。但 0.0.0.3 和 0.0.0.4 或 0.0.0.2 不兼容。



例如，为了更好地确保 Data Area 的兼容性，可以将模型的参数、Block I/O、ContState 和 DWork 不保留在内部 Data Area，而是设定为模块参数。这意味着，只有作为 Data Area 的输入和输出才与 TcCOM 版本的兼容性有关。




2. 如要在 TwinCAT XAE 中执行在线更改，请使用 **树项 TcCOM 模块**（TcCOM Modules）并导航至**可在线更改的对象**（Online Changable Objects）选项卡。

3. 从**在线版本**（Online Version）的下拉菜单中选择一个版本（只显示兼容版本）。
4. 右键单击对象行，选择**应用更改的在线对象版本**（Apply changed online object versions）激活 TcCOM 的新版本。

⇨ 详细信息请参见 TwinCAT C++ 文档。

## 4.7.9 配置对 TcCOM 对象数据的访问权限

在 *TwinCAT TcCom 接口*（TC TcCom Interface）区域，可以配置某些变量组数据的访问方式。可以根据需要配置 ADS 访问权限和过程映像类型。这些设置会影响组中变量与 TwinCAT 开发环境中其它过程映像的关联方式以及它们的数据交换方式。

### 变量组

除输入和输出变量外，还有几组内部变量，视 Simulink® 模型而定。

可以配置以下组：

| 组 | 描述 | DataArea 的命名（默认） | DataArea 的命名（“经典”（classic）选项） |
|---|---|---|---|
| 输入 | 模型输入 | <ModelName>_U | 输入 |
| 输出 | 模型输出 | <ModelName>_Y | 输出 |
| BlockIO | Simulink® 功能块的全局输出信号：已定义“测试点”的内部信号。 | <ModelName>_B | BlockIO |
| 参数 | 模型特定参数：Simulink® 功能块的*可调*参数。 | <ModelName>_P | 模型参数 |

| 组 | 描述 | DataArea 的命名（默认） | DataArea 的命名（“经典”（classic）选项） |
|---|---|---|---|
| ContState | 连续状态变量 | <ModelName>_X | ContStates |
| DWork | 离散状态变量 | <ModelName>_DW | DWork |
| 数据储存 | 数据存储内存 | <ModelName>_DW_<DataStoreName> | <ModelName>_DW_<DataStoreName> |

在 TC TcCom Interface 下，可使用**变量名与 ADS symbol 之间的映射**（Mapping between variable names and ADS symbols）选项来影响 DataArea 的命名。默认情况下，它们的名称与相关的 C++ 变量相同， 由 Simulink® Coder™ 指定。在“经典”（Classic）设置下，名称缩写为变量组的名称，与 TE1400 1.2.x 早期版本相同。然后将输入合并，例如合并到 DataArea“输入”（Input）而不是“<ModelName>_U”中。

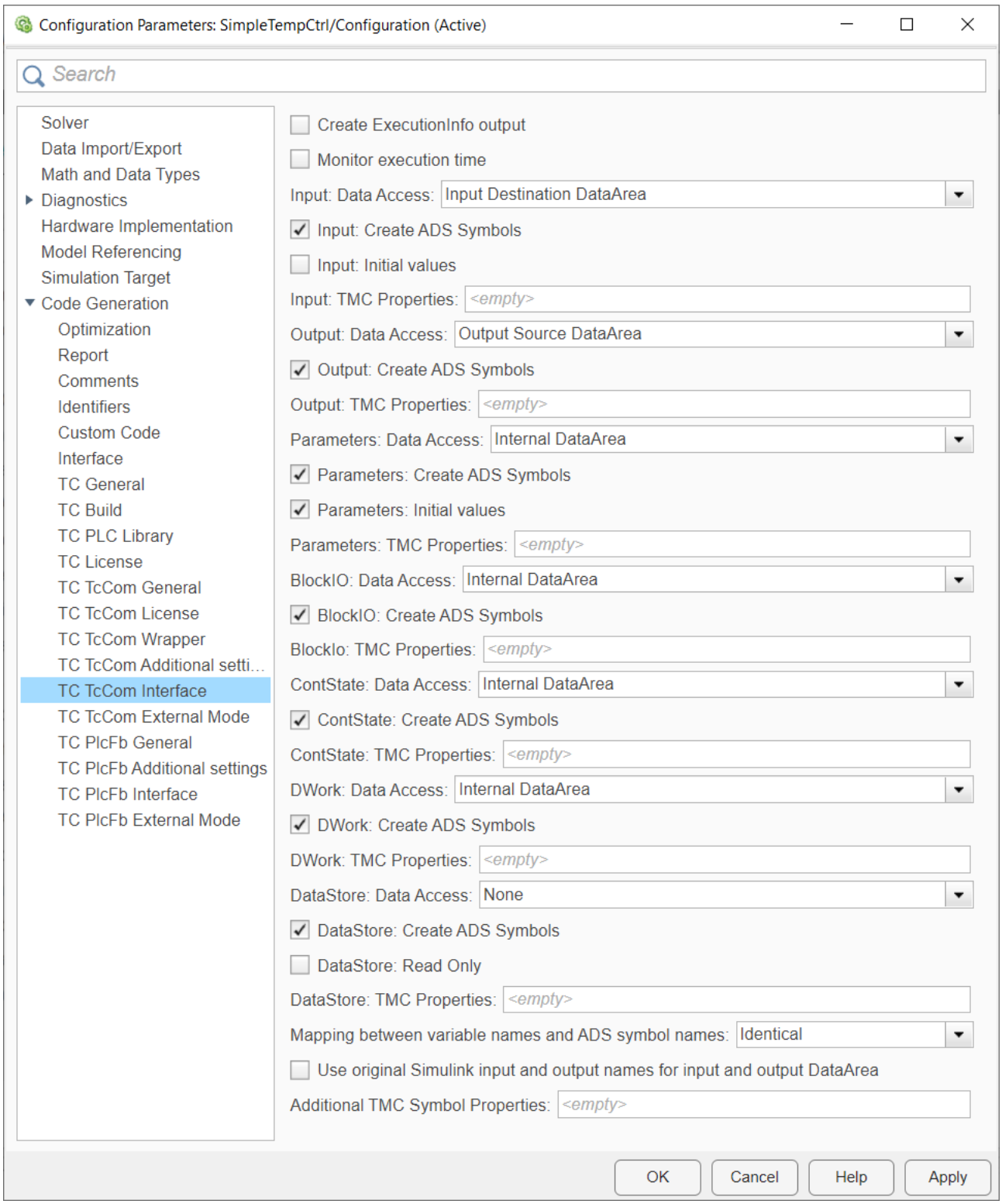
**具有多个任务上下文的 TcCOM**

如果创建的 TcCOM 有多个任务上下文（参见多任务、并发执行和 OpenMP [▶ 83]），则 DataArea 自动分隔。例如，有多个输入或 Dwork DataArea。

### Simulink® 中的配置界面

切换到高级模式可显示以下参数。

```
TwinCAT.ModuleGenerator.Settings.Change('ConfigurationLevel', 'Advanced')
```

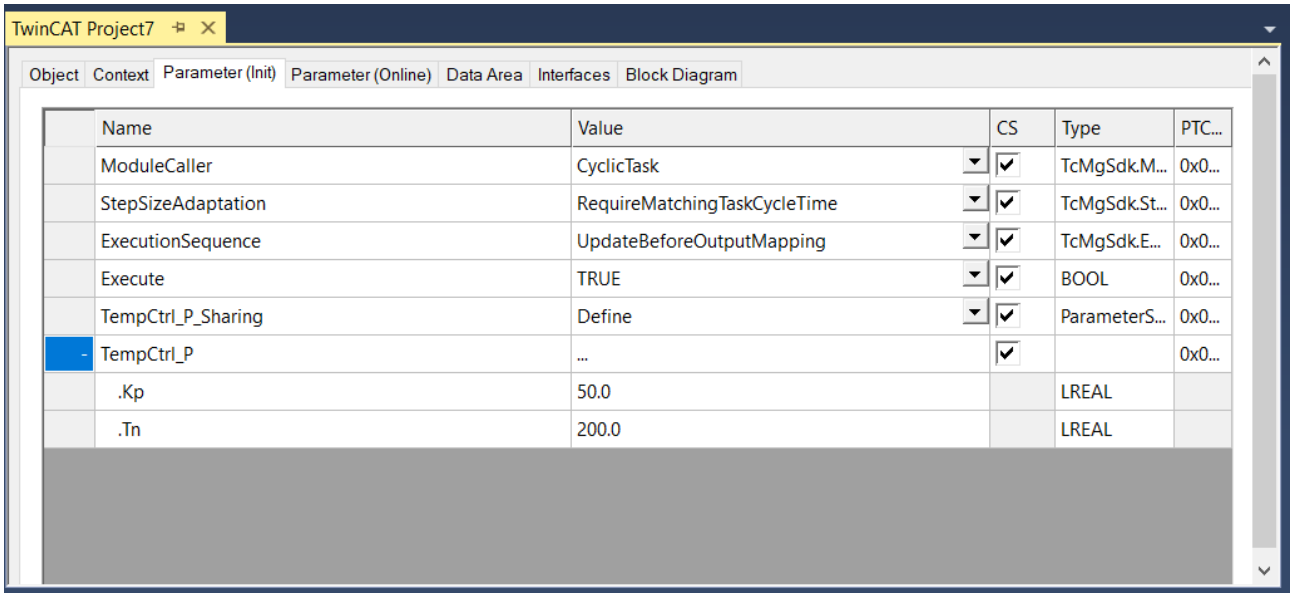



可以分别为上述每个变量组定义*数据访问*。下表汇总了 DataArea 和模块参数的选项：

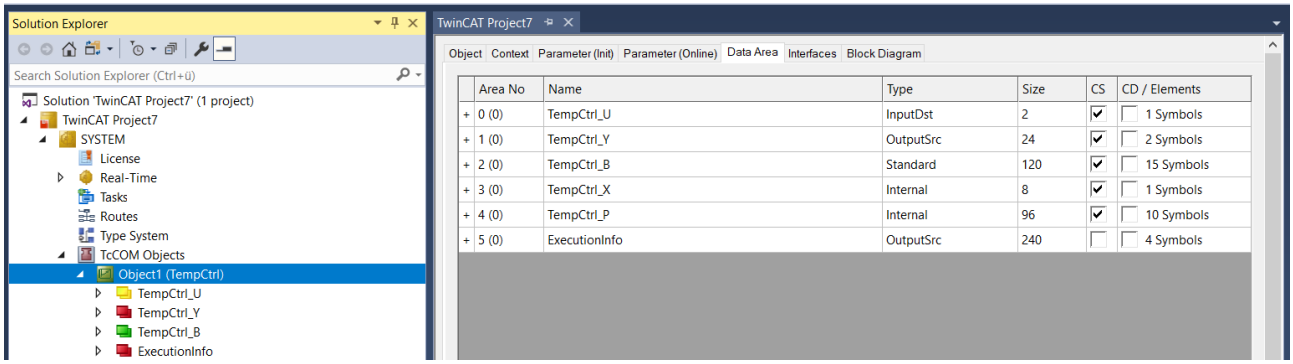
| 数据访问 | 通过 ADS 访问 | 通过映射访问 | 通过 Data Pointer 访问 |
|---|---|---|---|
| 无 | 否 | 否 | 否 |
| 模块参数 | 是 | 否 | 否 |
| 输入 DataArea | 是 | 是（带输出） | 是 |
| 输出 DataArea | 是 | 是（带输入） | 是 |
| 内部 DataArea | 是 | 否 | 否 |
| 标准 DataArea | 是 | 否 | 是 |

| 数据访问 | 通过 ADS 访问 | 通过映射访问 | 通过 Data Pointer 访问 |
|---|---|---|---|
| 保留 DataArea | 是 | 是（使用保留处理程序） | 否 |

**模块参数**与 DataArea 不同，不用于与其它模块或 I/O 进行周期性（过程）数据交换。它们通常是异步读取或写入的，例如通过 ADS。**参数（初始值）**（Parameters (Init)）选项卡上的参数具有初始值，该值可在项目中配置，并在启动 TcCOM 实例时写入。而**参数（在线）**（Parameter (Online)）选项卡上的参数则没有可配置的初始值，通常用于监控内部状态。




**DataArea** 可在 **DataArea** 选项卡上的 TcCOM 实例中找到。DataArea 的具体类型也显示在此处。内部 DataArea 不会在开发环境中创建为过程数据图像。




### 参数设置的特性

**通过 ADS 访问**通常为**只读**。只有 Inputs 组和 Parameters 组还可以通过 ADS 写入。

除此以外，还可以为每个组设置是否为相应组创建 ADS symbol 信息（**创建 ADS Symbol**）。如果没有 ADS symbol 信息，ADS 只能通过 IndexGroup 和 IndexOffset 访问数据。

通过“**输入：初始值**”（Input: Initial values）选项，除用于周期性数据交换的 DataArea 外，还可以在“参数（初始值）”（Parameter (Init)）选项卡下获得一个条目。这样就可以设定输入的初始值。如果相应的输入尚未关联，则这些值将被使用。如果没有这些参数，未关联的输入值始终为 0。

通过“**参数：初始值**”（Parameter: Initial values）选项，除 DataArea 设置外，Simulink® 模型的参数还可以创建为模块参数（结构 <ModelName>_P）。

**特殊情况：模型参数的 DataArea**

如果代码接口设置为可复用函数 [▶ 116]（默认设置），则不会为模型参数创建 DataArea，因为在这种情况下，参数保持全局，因此可在多个实例之间共享。如果选择“参数：初始值”（Parameter: Initial values），则除了**参数（初始值）**（Parameter (Init)）选项卡上的模型参数条目外，还会创建参数“<ModelParameterName>_Sharing”。只能为该 TcCOM 模块类的实例将该参数设置为“Define”。

所有其它实例都必须使用“Inherit”。在这种情况下，由于所有实例共享参数，因此它们都使用“Definer”中设置的参数。其它实例的模型参数设置将被忽略。
另请参见多个模块实例的参数设置 [▶ 116]。

**监控执行时间**：在**参数（在线）**（Parameter (Online)）选项卡和框图（右侧）中创建一个模块参数。可通过该参数读取 TcCOM 的执行时间。每次调用 TcCOM 时都会更新测量结果，因此可以通过 TwinCAT Scope 精确跟踪每次调用的执行时间。

**创建 ExecutionInfo 输出**：另外创建一个输出源 DataArea，其中包含模块调用的相关信息，请参见异常处理 [▶ 145]。

**DataStores**：数据存储记忆块中的变量默认存储在 DWork 中。为 DataStores 创建 DataArea 时，可以为这些变量创建额外的地址。默认情况下不创建 DataArea。示例请参见TcCOM 实例之间的共享内存 [▶ 74]。

**其它说明：**

- TcCOM 的结构：TcCOM 特性描述
- DataArea 的特性
- 通过 NOV-RAM 使用 RetainDataArea：RetainDataArea
- 可以通过 TC Module Input 和 TC Module Output 功能块等使用 DataPointer，请参见TcCOM 实例之间的共享内存 [▶ 74]。

### 4.7.9.1 最佳实践：访问 TcCOM 数据

**简介**

配置对 TcCOM 对象数据的访问权限 [▶ 67]章节介绍了如何在 Simulink® 模型的配置参数中设置对 TC TcCom 接口中 TcCOM 的数据访问。下文将介绍各种数据访问类型的**优缺点**、**应用场景** 以及哪种类型的 TcCOM 数据访问在什么情况下适用的**实用提示**。

**通过 ADS 访问**

**属性**

- 按需数据交换
- 异步通信
- 线程安全
- 无指定读写时间

**用例**

- 从 TwinCAT Runtime 以外（例如 HMI）读取或写入数据。读写模型参数或读取内部信号。
- 如果没有在同一任务上下文中调用编写软件模块（其它 TcCOM 或 PLC），则可从 TwinCAT Runtime 内部访问 TcCOM。因此，数据交换的线程是安全的。一个软件模块在不同情况下向不同的软件模块提供参数集时就可以使用。

**注意**

- 如果要同时更改或读取多个参数，则应使用 ADS sum 命令。否则无法保证在一个任务周期内对独立的 ADS 读取或写入命令全部进行统一处理。
- 如果通过 OPC UA 而不是 ADS 进行网络数据访问，这些特性也同样适用。

### 通过映射访问

**属性**

- 周期性数据交换
- 线程安全
- 确定的数据交换时间

**用例**

- 不同软件模块之间的周期性数据交换，尤其是每个任务周期都会变化的数据。

**注意**

- 在每个任务周期中，即使没有更改的值也会被复制。因此，建议将输入和输出映射 DataArea 保持较小的规模，将其限制在必要范围内即可。这里应避免将大型 Simulink® 总线结构作为其中只有少数元素会发生周期性变化的输入。
- 可在 Simulink® 中的“配置参数 -> TC TcCom 附加设置 -> 默认执行顺序”（Configuration Parameters -> TC TcCom Additional settings -> Default execution sequence）下确定/指定数据交换的时间。
- 只有在参数进行周期性变化（这种情况应首先检查参数是否无法更好地映射为模型输入），或参数数量很少以至于不影响周期性复制的系统开销的情况下，将模型参数定义为输入目的地 DataArea 才有意义。

### 通过 Data Pointer 访问

**属性**

- 按需数据交换
- 无线程安全
- 确定的读取或写入时间（共享内存区域）
- 每个实例中都有局部数据副本
- 在 XAE 中灵活链接 DataArea 和 Data Pointer

**用例**

- 多个 TcCOM 的共享内存区域，例如，用于 DataStore 中的变量。TcCOM 实例之间可以通过 Data Pointer 的数据互相共享一个公共内存区域。

**Data Pointer 示例**

示例请参见：TcCOM 实例之间的共享内存 [▶ 74]。

### 全局导出/外部导入

**属性**

- 按需数据交换
- 无线程安全
- 确定的读取或写入时间（共享内存区域）
- 每个实例中没有本地数据副本
- 只有绑定在同一驱动程序中 [▶ 55]的模块才能访问全局变量。

**用例**

- 多个 TcCOM 共享内存区域。也适用于大型变量。

**全局导出/外部导入的示例**

示例请参见：TcCOM 实例之间的共享内存 [▶ 74]。

**特殊情况：通过 TcCOM Wrapper FB 进行交互**

TcCOM Wrapper FB（参见应用 TcCOM Wrapper FB [▶ 134]）可以简化 PLC 与 TcCOM 之间的交互。这种 FB 几乎不需要编程就能循环执行 PLC 代码中的链接 TcCOM 对象，包括交换输入和输出数据。不过，它也允许对对象的参数进行简单的非周期性访问，如果需要，还可以灵活访问 DataArea。

Wrapper 一方面提供 ITcADI 接口（参见 ADI 接口 [▶ 137]），另一方面提供功能块特性（可选）（参见 FB 特性 [▶ 137]）。应在与 TcCOM 相同的上下文中调用 TcCOM Wrapper FB。

**功能块特性**

- 仅访问模块参数
- 不访问 DataArea。注意：默认情况下，模型参数既创建为 DataArea，也创建为模块参数。
- 按名称轻松访问模块参数
- 无线程安全

**ITcADI 接口**

- 高效灵活地访问所有 DataArea（也包括 DataArea 的子区域）
- 只能通过 DataArea 编号和 ByteOffset 访问
- 无类型信息（类型转换由用户实现）
- 必须在每个任务周期获取并释放指针
- 不访问模块参数
- 无线程安全

**修改保持**

上文介绍了如何在 TwinCAT 运行时更改 TcCOM 实例中的数据。如果重新启动 TwinCAT，但不采取进一步的恢复措施，这些更改将会丢失。启动 TcCOM 实例时，TcCOM 默认根据其**启动值**执行参数设置。

此外，还可定义在正常程序运行时之外仍保留其值的其余变量。其余变量可以声明为 **RETAIN** 变量，或者声明为 **PERSISTENT** 变量。

关于 TwinCAT PLC 中剩余数据（持久性和保留性）的更多信息：链接

下面介绍了重启 TwinCAT 后恢复数值的三种方法。

**1）启动值**

关于在线值、预备值、默认值和启动值的区别，请参见模块实例的参数设置 [▶ 111]。

**需要 TwinCAT 3 XAE**

启动值在 TwinCAT 配置中，即在开发中进行更改。可在此处输入更改内容。不过，必须对更改进行编译并下载到运行时系统中。

如果在线值在运行时期间发生变化，可以通过 ADS 等读取当前的在线值。例如，通过 TE1410 TwinCAT Interface for MATLAB®/Simulink® 来完成。随后，读取的值可以作为新的启动值输入 TwinCAT 配置中。

对此请使用自动化接口示例：

```
TwinCAT.ModuleGenerator.Samples.Show('Use the TwinCAT Automation Interface in MATLAB')
```

示例中使用一个实时脚本来展示不同的功能。*通过 ADS 读取和写入参数在线值*章节介绍了如何从 TcCOM 读取和写入在线值。*读取和更改参数启动值*章节介绍了如何读取和写入启动值。

新的启动值将被输入开发系统的 TwinCAT 配置中。如要在运行时系统上启用更改，必须在系统上激活配置。还可以通过自动化接口（`sysManager.ActivateConfiguration`）来实现这一点。可以通过自动化接口（`sysManager.StartRestartTwinCAT`）直接重启系统，也可以让系统在不重启的情况下运行。启动值在下次启动 TwinCAT 时设置，TcCOM 将以新的启动值启动。

**2）保留数据**

在 TC TcCom 接口 [▶ 67]（TC TcCom Interfaces）选项卡上，可以将模型参数（以及其它组）创建为保留源 DataArea 类型的 DataArea。然后，可以在 TwinCAT 配置中创建一个保留处理程序，并将其连接到 DataArea。详见 NOV-RAM 和保留处理程序。

**需要 NOV RAM**

如要使用保留处理程序，需要一个内置 NOV-RAM 的工业 PC/CX。不支持使用带有 NOV-RAM 的 EtherCAT 端子模块（如 EL6080）。

**3）持久性数据**

可以在没有 TwinCAT XAE 和 NOV-RAM 的情况下访问 TwinCAT 3 PLC 中的持久性数据。

**需要 TwinCAT 3 PLC Runtime**

如要使用持久性数据，必须在 PLC 中将变量声明为持久性变量。

持久性数据只能在 PLC 中创建。因此，请使用 PLC 来定义要持久保存的数据。例如，使用 TcCOM Wrapper FB [▶ 134] 从 TcCOM 读取数据，并通过持久性 PLC 变量持久保存这些数据。创建一个状态机，用于在 TwinCAT 重启后启动 TcCOM 对象，以便将持久性数据写回 TcCOM 对象，之后在代码中再次调用。

## 4.7.10 TcCOM 实例之间的共享内存

在某些应用中，TcCOM 实例共享一个内存区域可能是有利的，这样，某些结构/变量只需在一个对象中定义一次，所有其他对象都会引用该内存位置。

为此，可以在 TwinCAT 中选择两种不同的途径：

- 使用数据指针在 TwinCAT 中链接 TcCOM 实例 [▶ 74]
- 全局变量的使用（全局导出/外部导入） [▶ 78]

本章接下来将介绍这两种途径。

使用数据指针可以更灵活地链接数据，而使用全局变量的优势在于结构/变量在内存中只存在一次。如果使用数据指针，每个 TcCOM 实例都有一份本地数据副本。

***注意***

**数据传输无线程安全**

请谨慎使用数据指针和全局变量。数据交换不具有线程安全。因此，我们强烈建议在同一 task 中运行所有涉及的 TcCOM 实例。

### 使用数据指针在 TwinCAT 中链接 TcCOM 实例

下面将具体介绍如何通过数据指针在 TcCOM 实例之间使用共享内存区域。

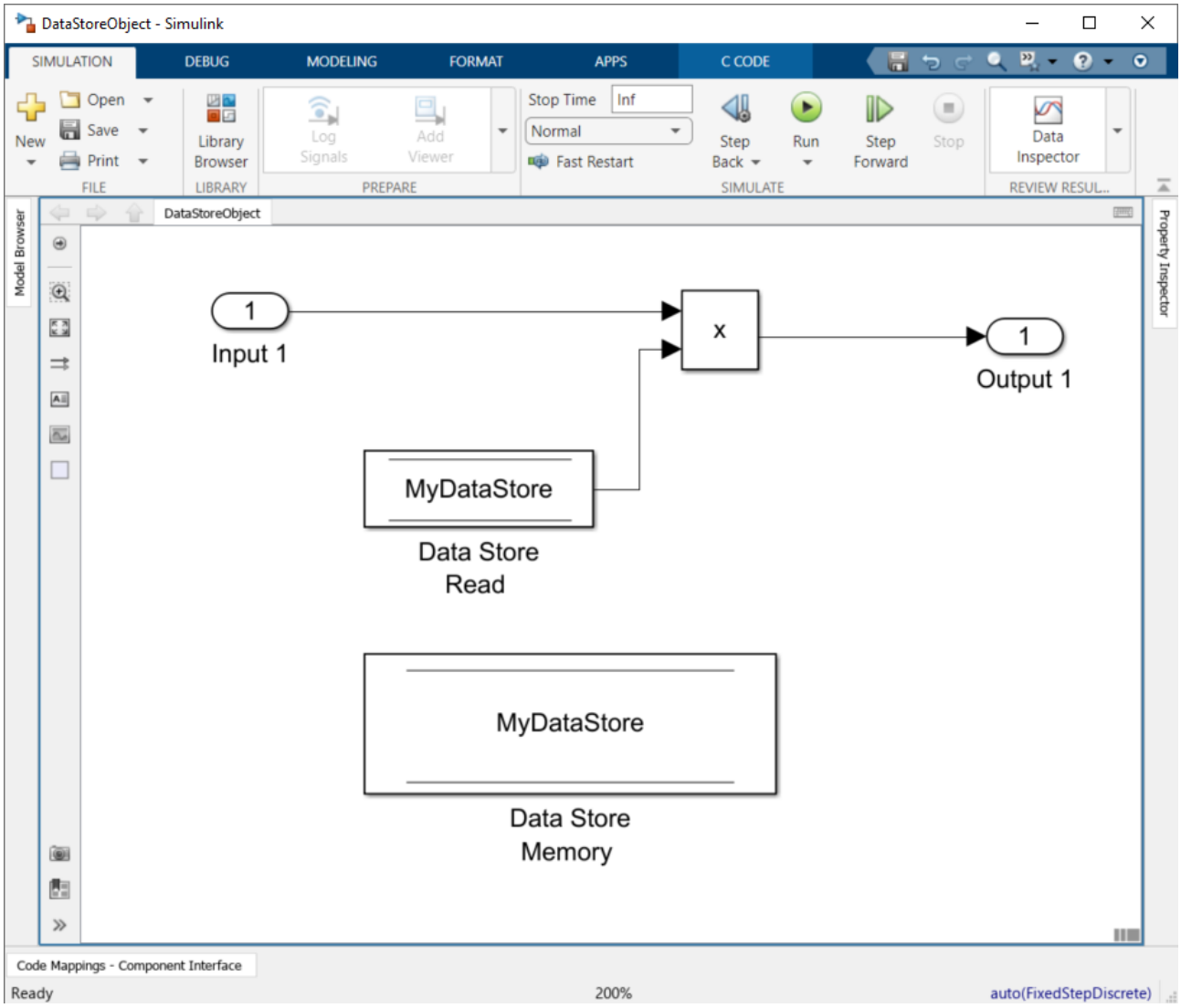
如配置对 TcCOM 对象数据的访问权限 [▶ 67]章节所述，可以通过 DataPointer 访问 DataArea。通过 DataPointer，可以访问输入目的地 DataArea、输出源 DataArea 和标准 DataArea。原则上可以将任何变量组、模型参数、DWork、BlockIO 等创建为标准 DataArea，从而使其可以作为 Data Pointer 访问。

#### 目标

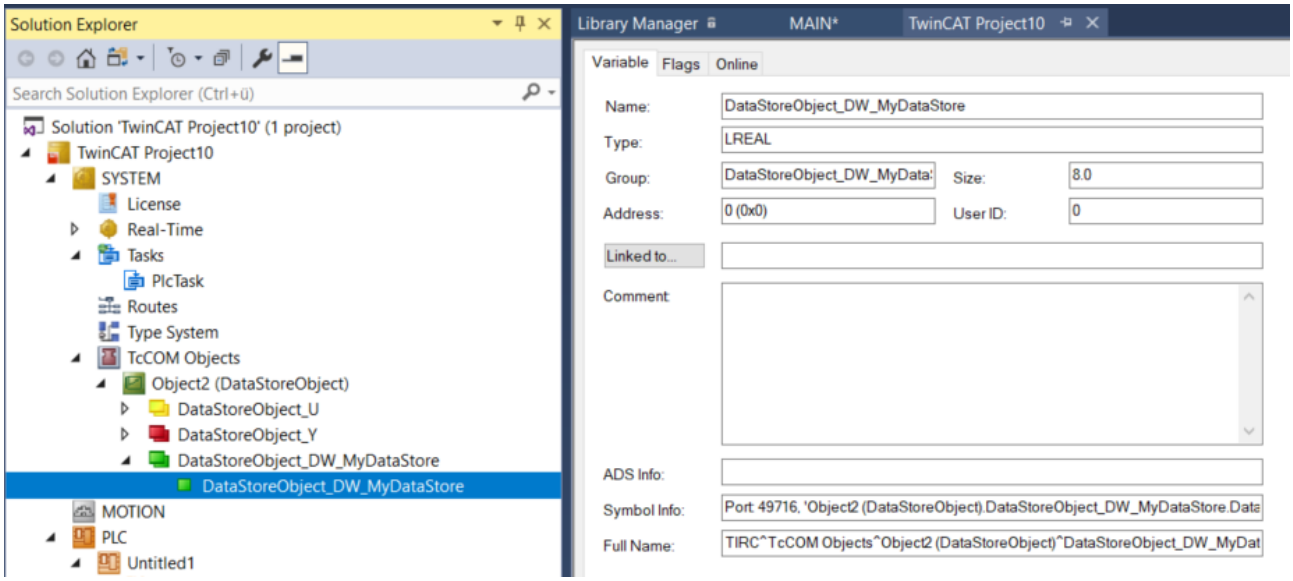
下文将为您展示如何通过 DataPointer 访问特定的变量子范围，而不是同时访问整个变量组范围。基本概念基于 Simulink® 中数据存储记忆块的使用。

#### 带 DataStore 的模型

在模型 *“DataStoreObject”* 中创建一个数据存储记忆块。在该示例中，只有一个双精度型变量。该变量通过 Data Store Read 读取，乘以输入值并设置为输出。



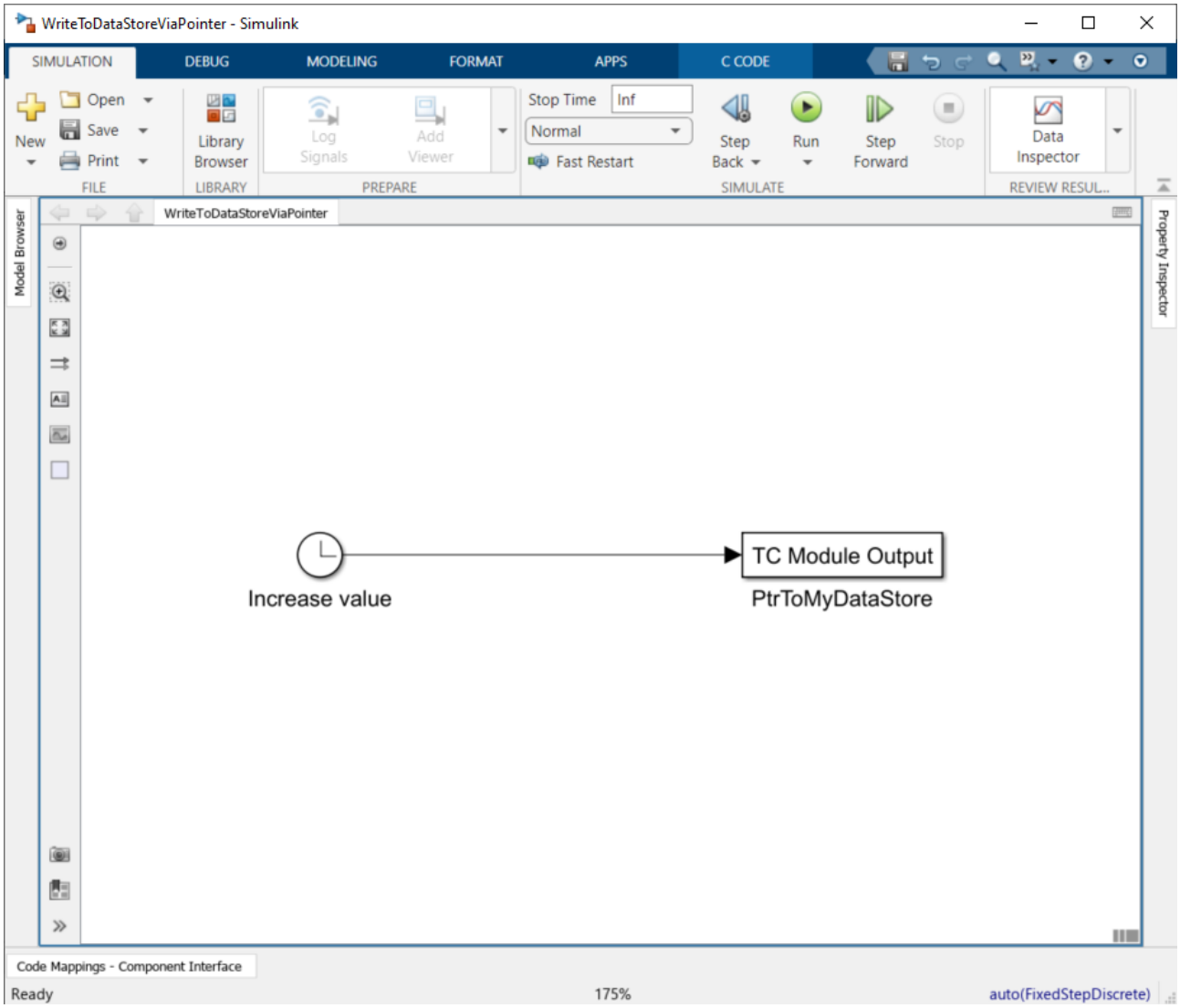

在 TC TcCom 接口，选项*DataStore：数据访问*（DataStore: Data Access）由“无”（None）更改为*标准 DataArea*（Standard DataArea），Simulink® 模型被编译成 TcCOM。如果在 TwinCAT 中对该对象进行实例化，显示如下：



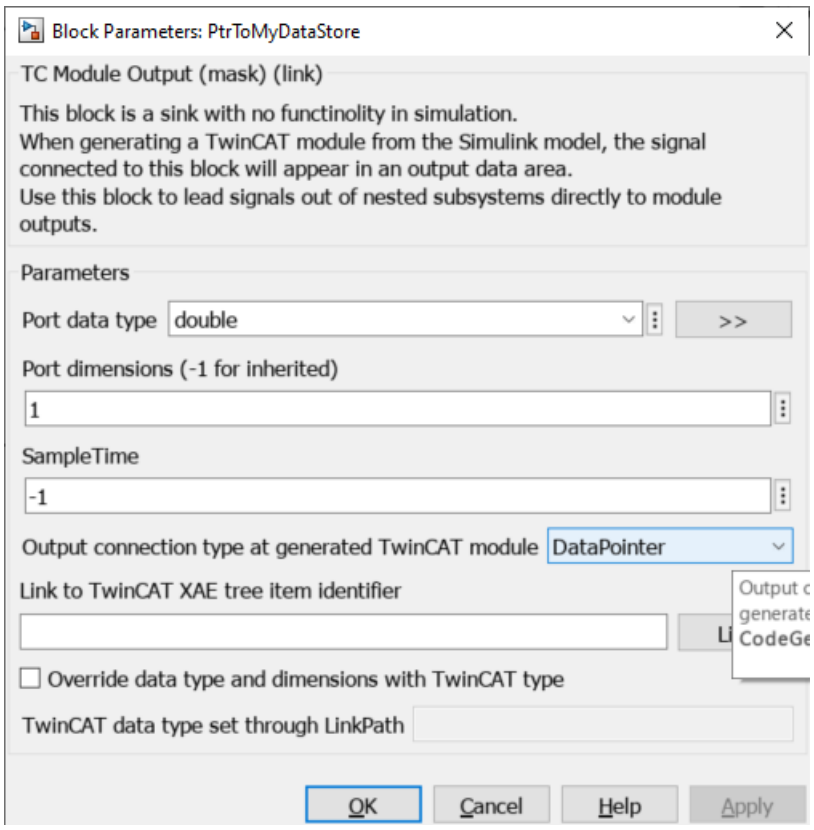

现在，DataStore DataArea 显示在过程映像中，包含一个 LREAL 型变量，可通过 DataPointer 访问该变量。

**带 DataPointer 的模型**

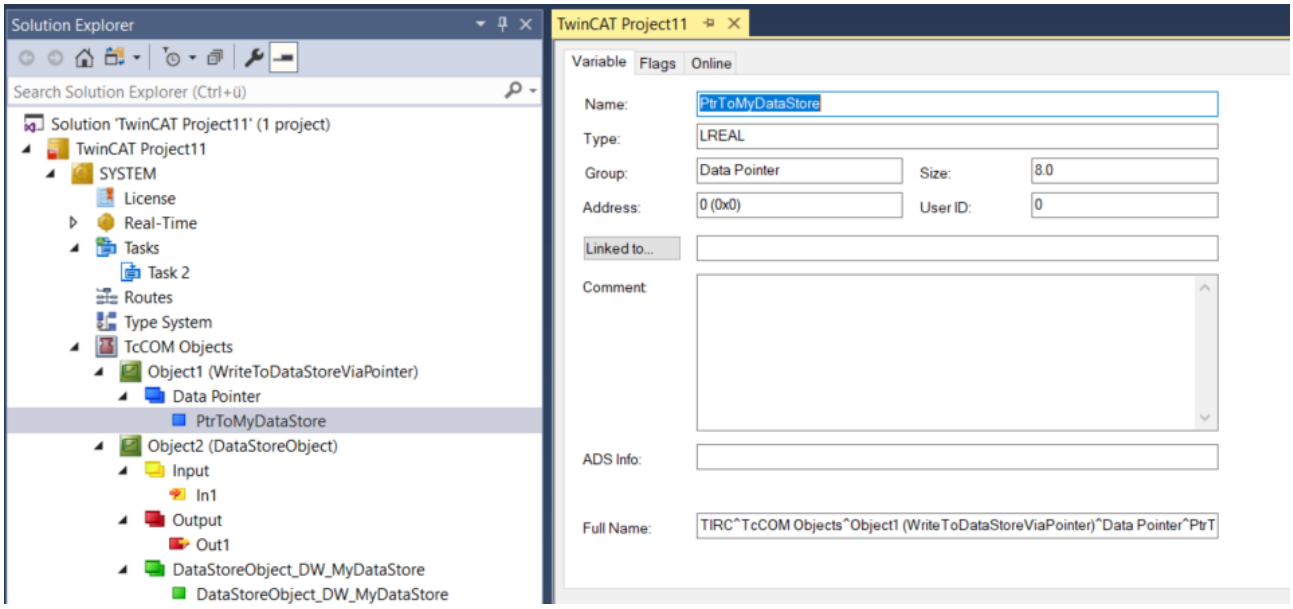
在第二个 Simulink® 模型中，时钟会产生一个双精度值，并将其设置为 TC Module Output [▶ 28]。




在 TC Module Output 的配置中，连接类型设置为 DataPointer，数据类型设置为双精度。




如果将该模型编译成 TcCOM 对象，TwinCAT 中显示如下：

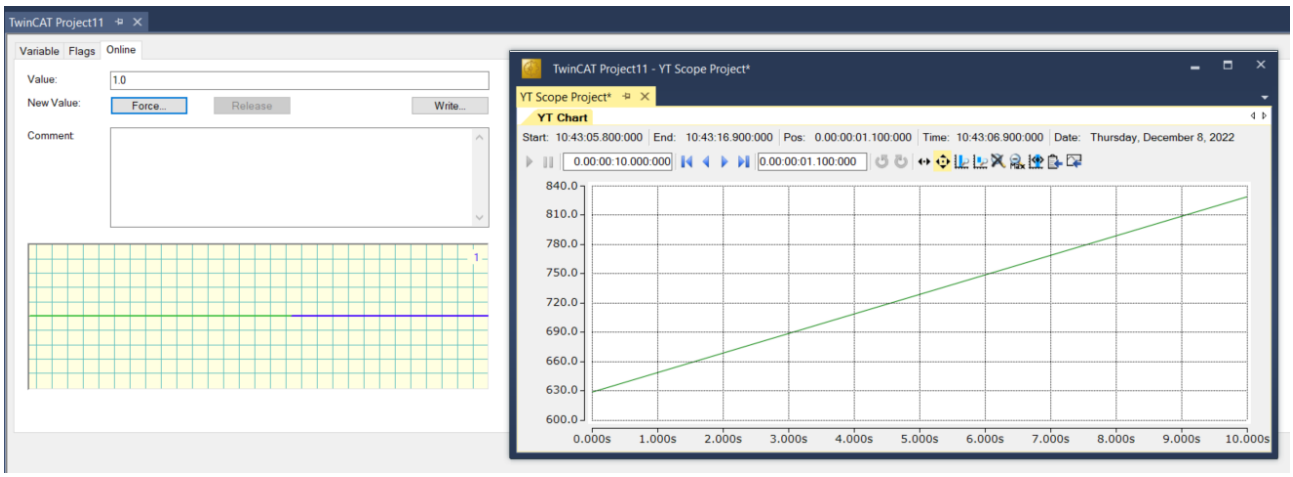



LREAL 型 TC Module Output 的名称“PtrToMyDataStore”现在显示在 Data Pointer 下。现在可以通过双击 DataStoreObject_DW_MyDataStore 进行链接。

两个 TcCOM 实例应在同一任务中调用，因为通信之间的线程不安全。

如果使用不同的任务，用户必须确保数据的一致性，并在适当的时间读取或写入。

此处示例中的行为如下图所示。
如果手动将 DataStore 模型的输入 In1 设置为 1，并通过 TwinCAT Scope 观察输出 Out1，就会看到一条不断增长的直线。这意味着时钟的增量值通过指针写入到 DataStoreObject-TcCOM 的 DataStore。




**通过 TC Module Input 进行读取访问**

使用 TC Module Input 通过 DataPointer 可以实现对 DataArea 的读取访问。

**每个模块中都有本地数据副本**

Simulink Coder™ 生成 C/C++ 代码会为TC Module Input 和 TC Module Output 创建局部变量，因此每个模块实例都包含数据的本地副本。因此，项目的内存需求会随着每个实例的增加而增加。

### 全局变量的使用（全局导出/外部导入）

下文介绍如何使用 Simulink® 中的存储类在 TcCOM 中创建全局变量，并在其它 TcCOM 中引用该变量。

### 目标

创建一个 Simulink® 参数 GlobalPrams，该参数的结构由 3 个元素组成。目标是通过一个 TcCOM 对象定义该结构，并对通过共享内存区域访问该结构的更多 TcCOM 对象进行实例化，因此定义的 TcCOM 中的更改可以直接到达所有连接的 TcCOM 实例。

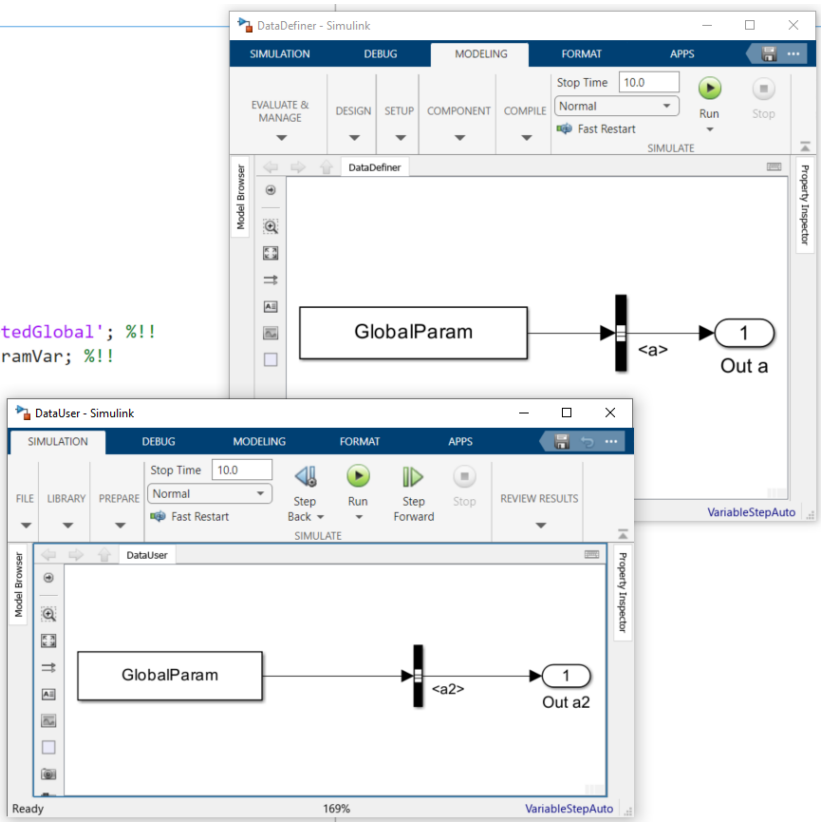
### 在 Simulink® 中建模

创建两个 Simulink® 模型：

1. “DataDefiner”模型
2. “DataUser”模型

此外，还定义了 Simulink® 总线，并将其创建为 Simulink® 参数，命名为 `GlobalParam`。DataDefiner 模型在最后决定 GlobalParam 的内容，而 DataUser 模型只读取其内容。

```
%% Global parameter variable
% parameter structure and appropriate bus
ParamStruct = struct('a',5,'a1',5,'a2',6);
struct2bus(ParamStruct,'ParamBus');

% global variable name
globalParamVar = 'GlobalParam';

% global parameter object
GlobalParam = Simulink.Parameter;
GlobalParam.Value = ParamStruct;
GlobalParam.Complexity = 'real';
GlobalParam.CoderInfo.StorageClass = 'ExportedGlobal'; %!!
GlobalParam.CoderInfo.Identifier = globalParamVar; %!!
GlobalParam.CoderInfo.Alignment = -1;
GlobalParam.Description = '';
GlobalParam.DataType = 'Bus: ParamBus';
GlobalParam.Min = [];
GlobalParam.Max = [];
GlobalParam.DocUnits = '';
```




请注意，以下 DataDefiner 和 DataUser 的 GlobalParam 存储类定义设置不同。DataDefiner 的存储类设置为**全局导出**（Exported Global），而 DataDefiner 的存储类设置为**外部导入**（Imported Extern）。

下面的脚本将所述的两个 Simulink® 模型绑定在一个驱动程序中，参见在一个 TwinCAT 驱动程序中捆绑多个模型 [▶ 55]。

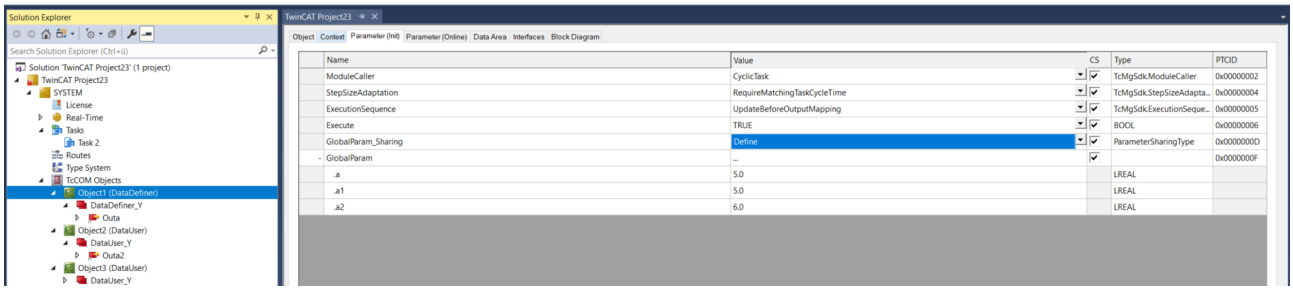
```
thisDir = fileparts(mfilename("fullpath"));

% model names
mdlNames = ["DataDefiner","DataUser"];
codeDirectories = fullfile(thisDir,strcat(mdlNames,'_tcgrt'));
isParamDefiner = [true,false];

% instantiate project configuration
projCfg = TwinCAT.ModuleGenerator.ProjectExportConfig('FullPath',fullfile(thisDir,'TcCppProj','DataS
haringModules.vcxproj'));

regenerate = false;
for i=1:length(mdlNames)
   % generate code for the models (only if the code directory doesn't exist) -
> remove the directory to rebuild specific models
   if regenerate || ~isfolder(codeDirectories(i))

      % load the model and apply basic settings
      mdl = load_system(mdlNames(i));
      set_param(mdl,'SystemTargetFile','TwinCatGrt.tlc');
      set_param(mdl,'CodeInterfacePackaging','C++ class');
      set_param(mdl,'TcCom_TcComWrapperFb','off');
      set_param(mdl,'TcProject_Generate','off');
      set_param(mdl,'SolverType','Fixed-step');

      % adapt the storage class of the shared parameter structure
      if isParamDefiner(i)
         GlobalParam.CoderInfo.StorageClass = "ExportedGlobal";
      else
         GlobalParam.CoderInfo.StorageClass = "ImportedExtern";
      end

      % generate code and close the model
      slbuild(mdl);
      close_system(mdl,0);
   end

   % add the class export configuration to the "global" project export configuration
   mdlProjCfg = TwinCAT.ModuleGenerator.ProjectExportConfig.Load(codeDirectories(i));
   clsCfg = mdlProjCfg.ClassExportCfg{1};
   projCfg.AddClassExportConfig(clsCfg)
end
```

```
% generate and build the project
TwinCAT.ModuleGenerator.ProjectExporter(projCfg);
```

**TwinCAT XAE 中的配置**

✓ 在 TwinCAT XAE 中创建**一个** DataDefiner 实例（不允许创建更多！）。另一方面，您可以创建任意数量的 DataUser 实例。

1. 对于 DataDefiner，在**参数（初始值）** > **参数 GlobalParam_sharing**（Parameter (Init) > Parameter GlobalParam_sharing）下设置为“Define”。
2. 对于所有 DataUser，在**参数（初始值）** > **参数 GlobalParam_sharing**（Parameter (Init) > Parameter GlobalParam_sharing）下设置为“Inherit”。
3. 创建一个任务。
4. 将该任务分配给所有创建的实例。




5. 激活项目。

⇨ 例如，通过框图更改 DataDefiner 中 GlobalParam 的值并观察对 DataUser 实例的直接影响。

## 4.7.11 创建带 OEM 许可证查询的模块

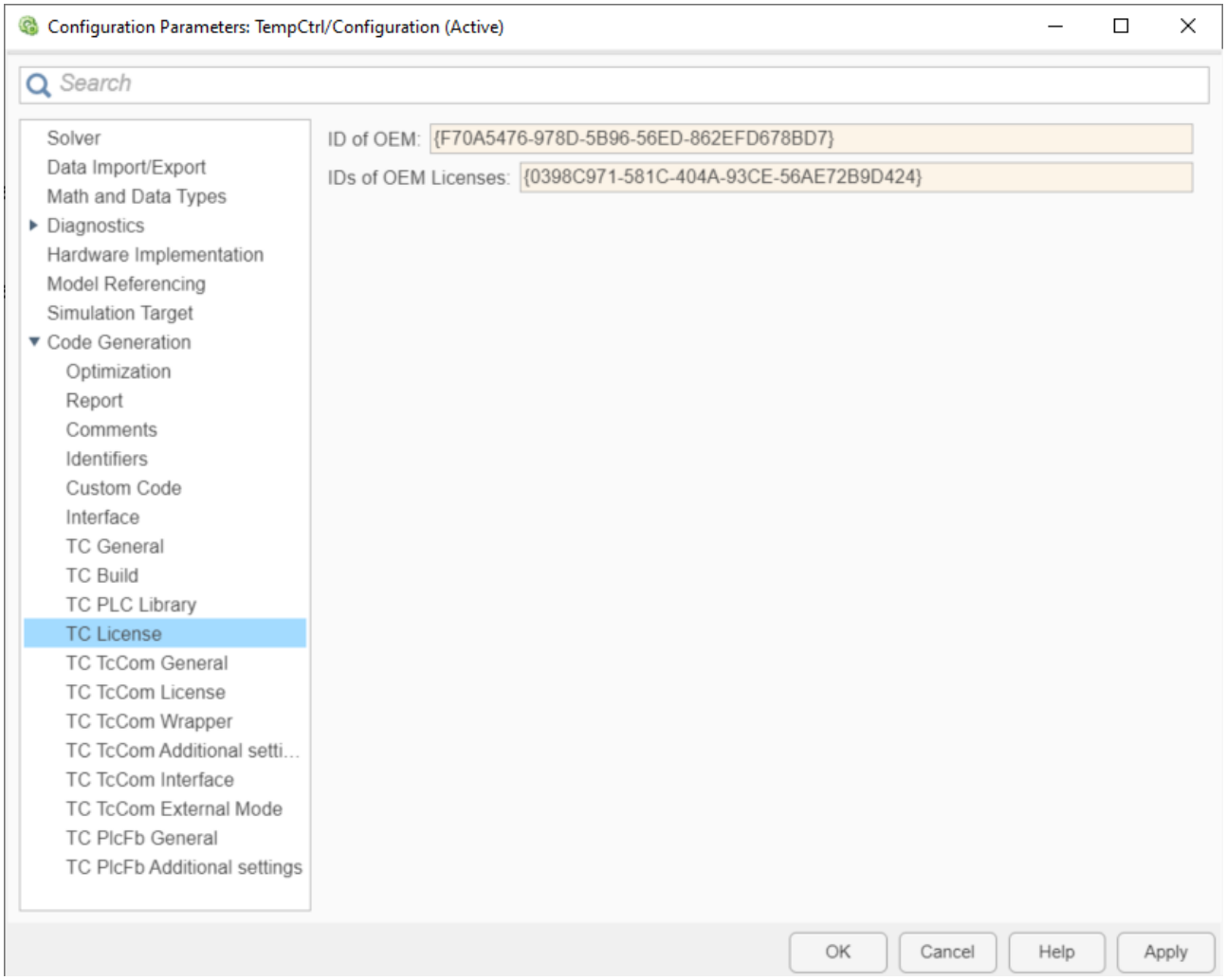
**为什么要将自有许可证与模块相结合?**

如果一个 TwinCAT object 除了 TwinCAT 许可证之外，还绑定了 OEM 许可证，则可以将该 TwinCAT 对象与硬件绑定，从而防止应用程序被克隆。此外，还可以通过这种途径将应用程序的功能授权给终端客户。

如需了解更多信息，请参见软件保护/自有 OEM 许可证。

**Simulink® 中的配置**

- 切换到高级配置级别：
  `TwinCAT.ModuleGenerator.Settings.Change('ConfigurationLevel', 'Advanced')`
- 输入您的 OEM ID 和申请的 OEM 许可证：



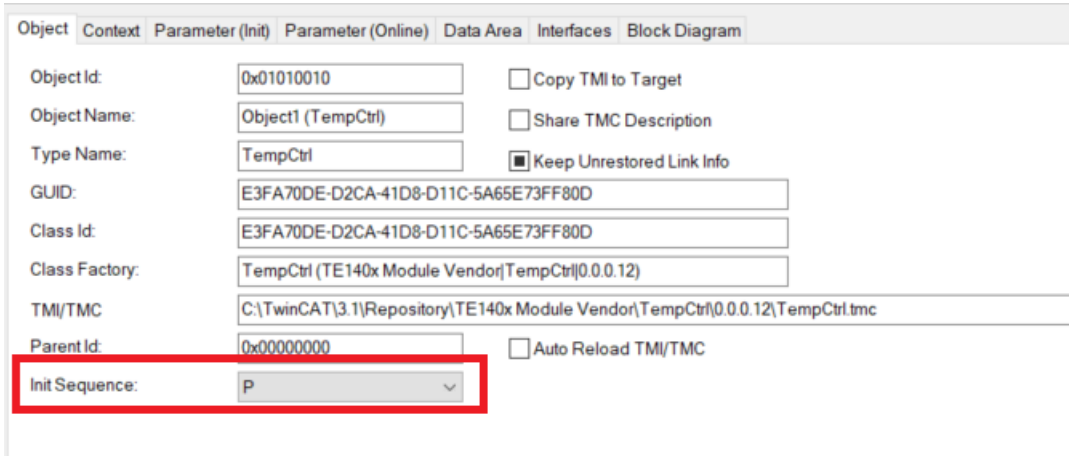

如果使用上述设置创建模块并在 TwinCAT 中实例化，则除了有效的 TwinCAT 许可证（TC1220、TC1320）外，还必须具备有效的 OEM 许可证才能激活 TwinCAT。

### 许可证加密狗注意事项

在加密狗上使用*目标系统*的 OEM 许可证时应注意以下几点：

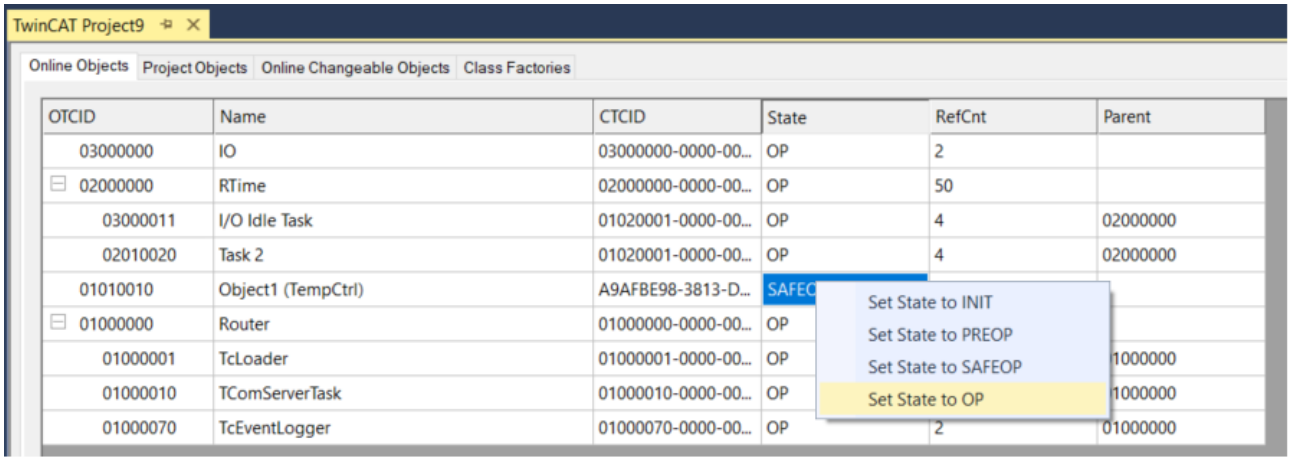
✓ **您使用的是 TcCOM 实例吗?**

1. 将对象实例的初始序列（Init Sequence）设置为 P。




2. 请注意，在这种情况下不能使用自动映射。建议使用 TcCOM wrapper FB 或从 TwinCAT C++ 中调用该模块。
3. 激活配置。

4. 在 TwinCAT 进入运行模式后，将 TcCOM 对象切换到 OP 状态，例如通过 XAE（见下图）、TcCOM wrapper FB 或 ADS。




⇨ 启动进入 OP 状态时，将检查并使用许可证（如果有效）。

✓ **是否使用已创建 PLC 库中的 TcCOM wrapper FB 并引用 TcCOM 静态实例?**

1. 将对象实例的初始序列（Init Sequence）设置为 P（见上文）。
2. 使用 TcCOM wrapper FB 将引用的 TcCOM 切换至 OP 状态。

⇨ 启动进入 OP 状态时，将检查并使用许可证（如果有效）。

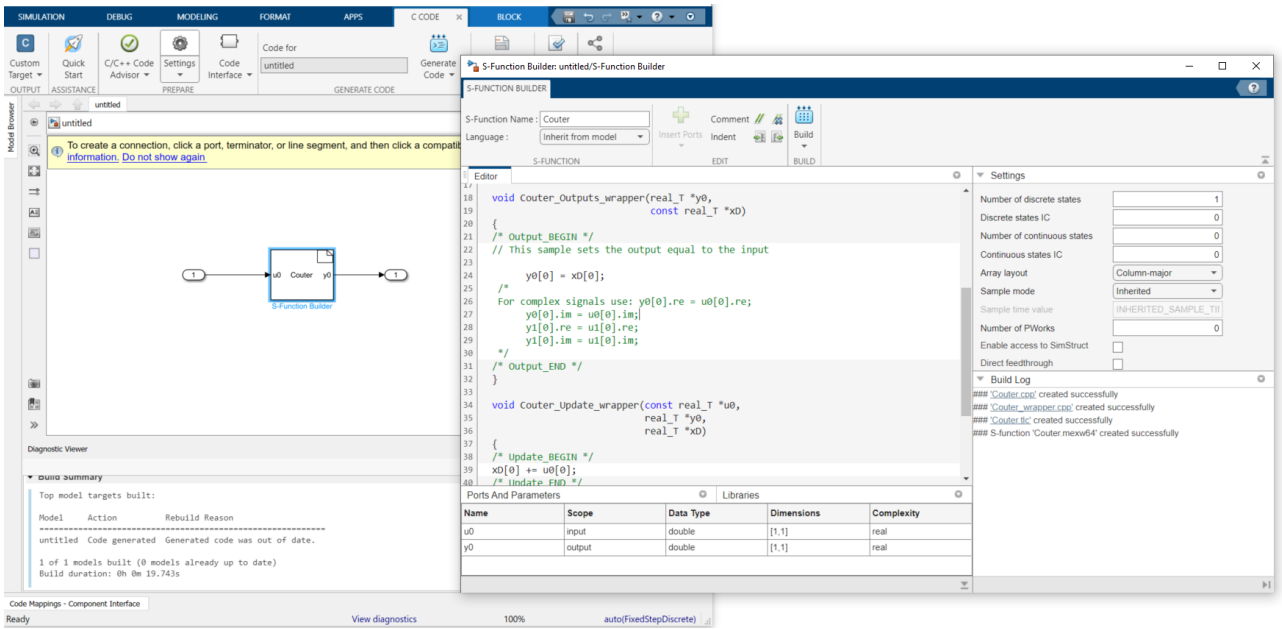
✓ **什么情况下无需关注任何其它事情?**

1. 如果使用了创建的 PLC 库中的 PLC-FB。
2. 如果使用 TcCOM wrapper FB 动态创建 TcCOM。

## 4.7.12 集成自有 C/C++ 代码

Simulink® 的 TwinCAT Target 还支持在 Simulink® 中集成自己的 C/C++ 代码。MathWorks® 为此提供了多种方式，例如通过 S 函数生成器（S Function Builder）。

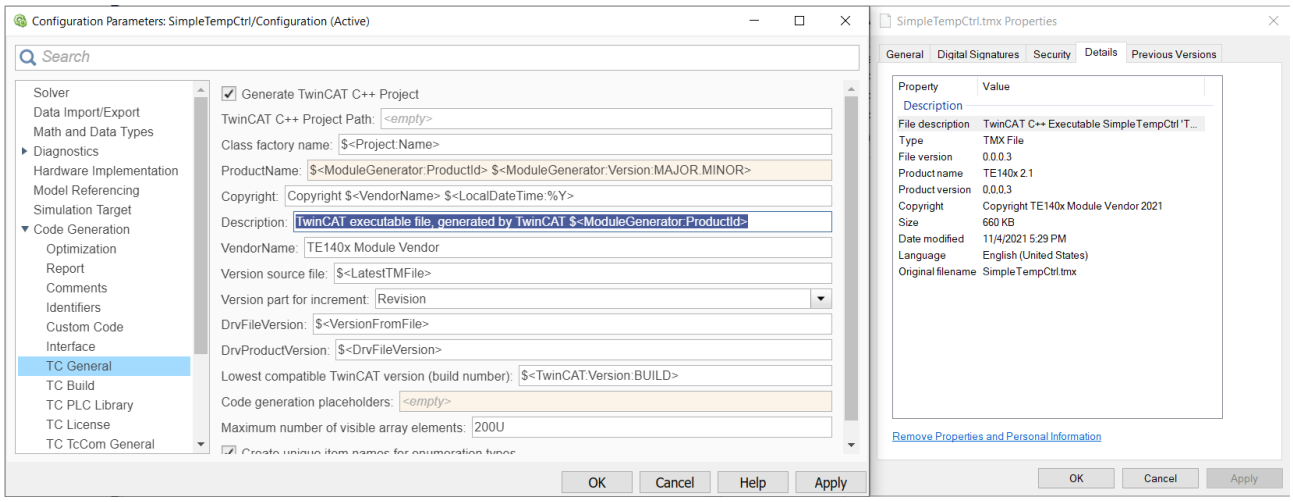
请注意，您将语言设置为“从模型继承”（Inherit from model）。还可以包含库，前提是这些库与平台无关，并以源代码形式提供。例如，不可包含预编译的 DLL。



## 4.7.13 TMX 文件特性的配置

可以通过 Simulink® 对 TMX 文件（TwinCAT Module Executable）中的条目进行参数设置。为此，请切换到高级模式：

```
TwinCAT.ModuleGenerator.Settings.Change('ConfigurationLevel', 'Advanced')
```




**TMX 特性（左）与 Simulink® 中参数（右）的对应关系**

文件描述 -> Description

文件版本 -> DrvFileVersion

产品名称 -> ProductName

产品版本 -> DrvProductVersion

版权 -> Copyright

注：$< > 描述占位符 [▶ 97]。例如，DrvProductVersion 设置为 DrvFileVersion 中的值，而 DrvFileVersion 又从版本源文件中获取值。

## 4.7.14 多任务、并发执行和 OpenMP

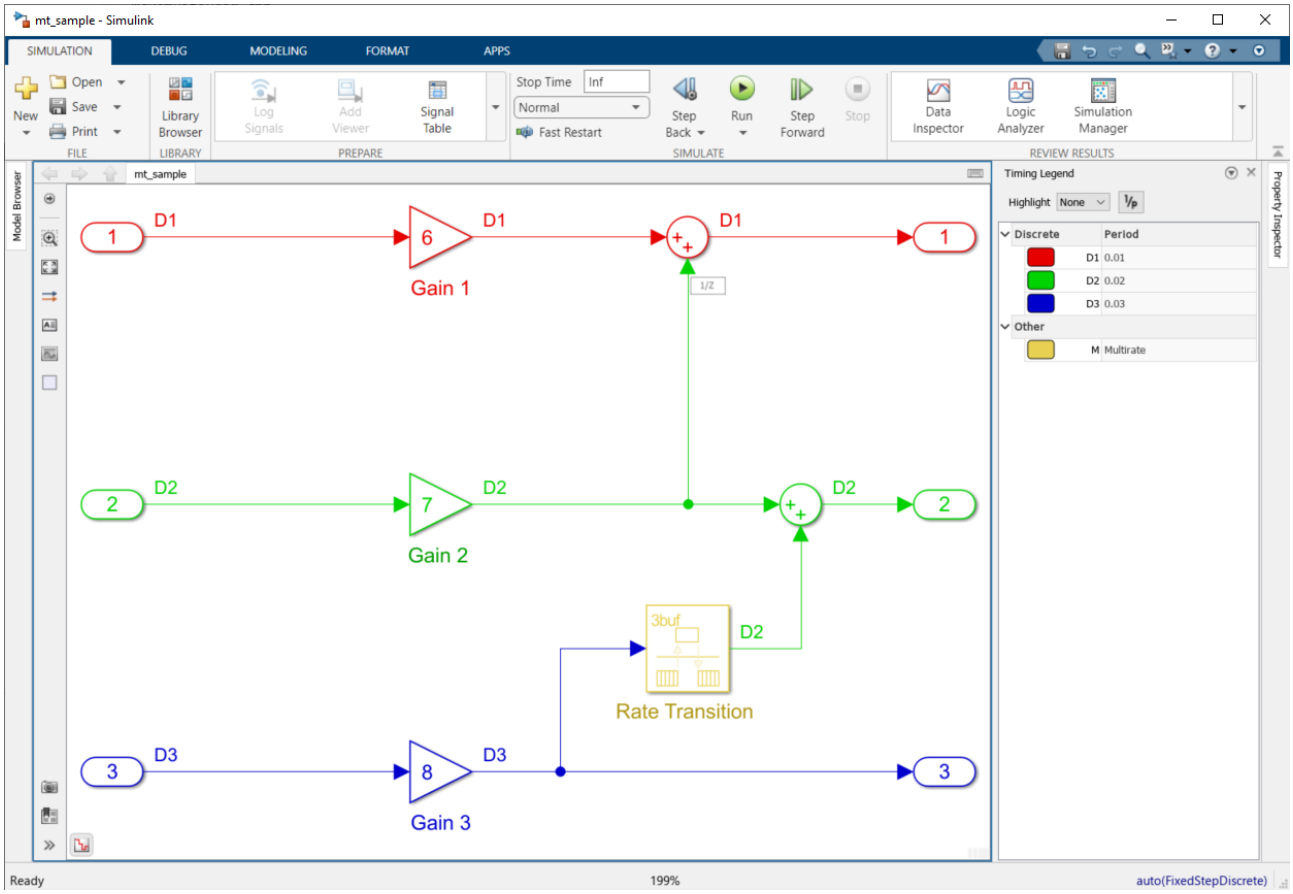
在 Simulink® 中，可以配置模型在多核目标机系统上运行。更多详细信息请参见 MathWorks® 文档。倍福目标机通常提供可与 TwinCAT 3 高效配合使用的多核架构。如下图所示，使用 TwinCAT Target for Simulink® 也可以做到这一点。

本说明对多任务、并发执行和 OpenMP 进行了区分。

- 通过 Multitask [▶ 85]，创建了一个 TcCOM 对象，该对象有多个可用任务。**所有**任务必须在同一核心上运行， 但不会并行处理。
- 通过并发执行 [▶ 87]，还可以创建一个 TcCOM 对象，其中包含多个任务，这些任务可分布在不同的内核上。实际上可以执行并行计算。
- 通过 OpenMP [▶ 88] 创建的 TcCOM 对象带有任务上下文。此外，分布在不同内核上的多个 JobTask 可以并行执行作为 OpenMP 代码生成的代码片段。

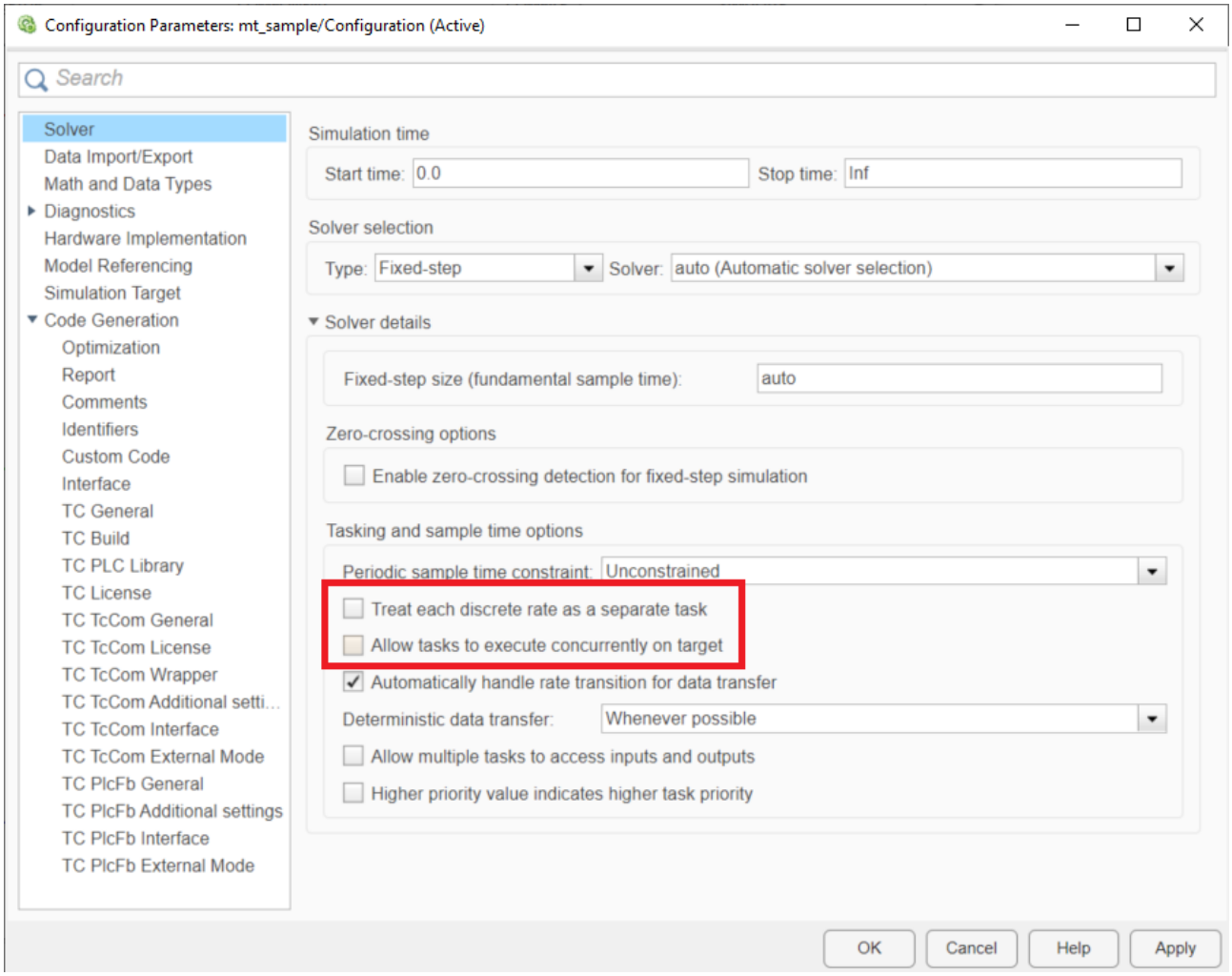
### 多任务和并行执行

以下 Simulink® 中的多级系统可用于描述选项*多任务*（Multitask）和*并发执行*（Concurrent Execution）。该模型有一个显性和一个隐性*速率转换*（rate transition）模块。



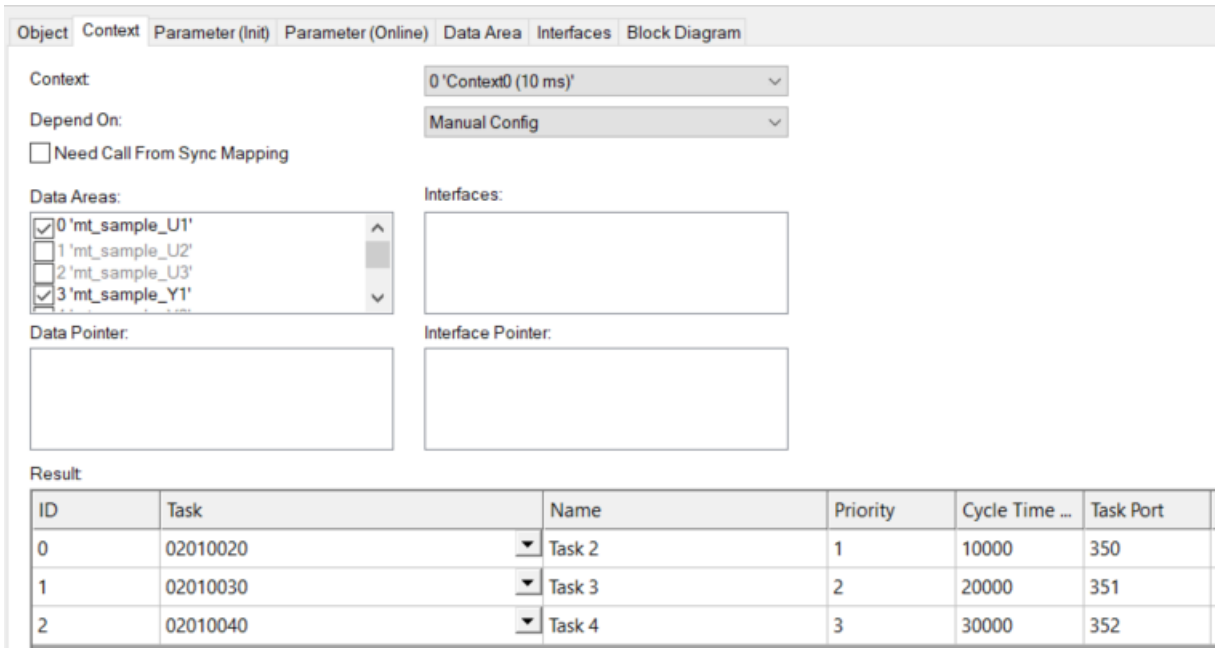

进入**配置参数**（Configuration Parameters），并选择**求解器**（Solver）。可在此选择：

- 将每个离散速率视为单独的任务
- 允许任务在目标上同时执行

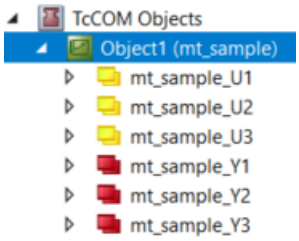



**将每个离散速率视为单独的任务：多任务**

如果在创建 TcCOM 对象时启用选项*将每个离散速率视为单独的任务*（Treat each discrete rate as separate task），将获得一个可为其分配多个任务上下文的对象。本例中有 3 个任务。




输入、输出和所有其它 DataArea 都被划分到不同的上下文中，因此本例中有 3 个输入 DataArea 和 3 个输出 DataArea。




在这种情况下，周期性任务必须全部在同一个内核上。无多任务并行处理。

与只有一个任务接口的 TcCOM 相比，其优势体现在：现在不必在最快的任务周期内完成所有计算（参见调度）。如果上述 Simulink 模型是使用默认设置创建的，而没有启用选项*将每个离散速率视为单独的任务*（Treat each discrete rate as separate task），则只有一个 10 ms 的任务（最快任务）可以链接。这意味着所有计算必须在该时间内完成。将多个任务分配给同一内核，则会禁用此规则，因为多个任务之间会相互挂起（参见优先级）。

| Object | RT-Core | Base Time (ms) | Cycle Time (ms) | Cycle Ticks | Priority |
|---|---|---|---|---|---|
| Task 2 | Default (3) | 1 ms | 10 ms | 10 | 1 |
| Task 3 | Default (3) | 1 ms | 20 ms | 20 | 2 |
| Task 4 | Default (3) | 1 ms | 30 ms | 30 | 3 |
| I/O Idle Task | Core 2 | 1 ms | 1 ms | 1 | 11 |

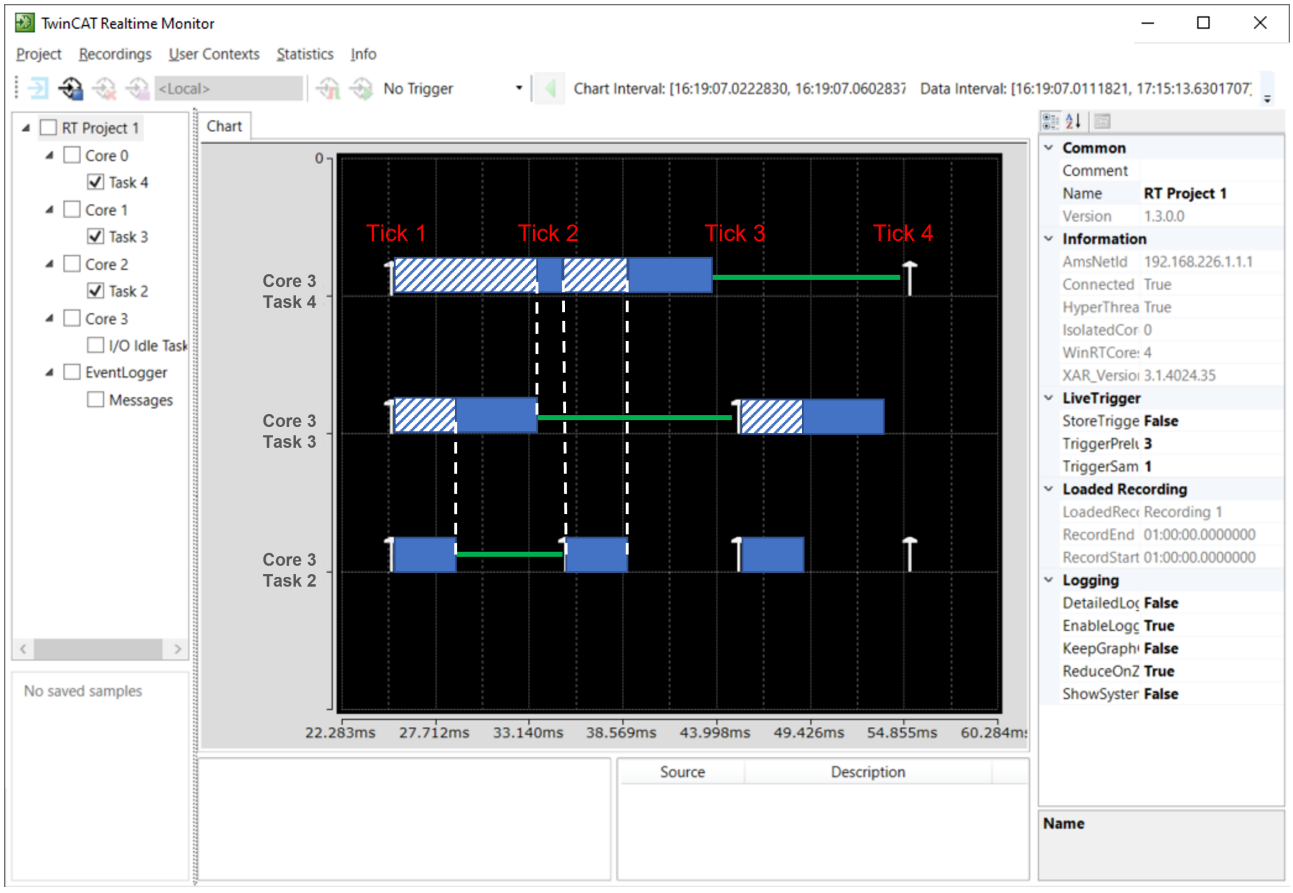
**特性：**

- 不支持PLC-FB功能块。
- 不支持 TwinCAT Usermode Runtime。
- 所有任务分配给同一个内核。
- 必须为最快的任务分配最高优先级（优先级值最小）。第二快的任务采用第二高的优先级，以此类推。

**调度详情：**

下图举例说明了如何分配计算时间。阴影线区域表示，由于存在优先级更高的任务，某项任务在该时间段内可能无法运行。全蓝色区域表示任务正在运行。请注意，这些表面只是随后叠加在实时显示器图像上帮助理解，并非真实图像。

- 在 Tick 1 中，任务 2、任务 3 和任务 4 依次在同一内核上执行。任务 2 和任务 3 的执行不会中断。在向 Tick 2 过渡期间，由于任务 2 的优先级更高，任务 4 的执行被中断。
- 任务 2 在 Tick 2 中首先执行。任务 4 在任务 2 完成后继续执行。
- 任务 2 再次启动，任务 3 在 Tick 3 中进行。




如果出现周期超时，无法遵循调度，则跳过相应任务上下文的执行，直到所有相关上下文都处于适当的状态。在 TcCOM 对象中，可以通过在线参数 `SkippedExecutionCount` 观察到这种行为。

### 允许在目标机上同时执行多个任务：并发执行

如果在创建 TcCOM 对象时启用选项*允许在目标机上同时执行多个任务*（Allow tasks to execute concurrently on target），将获得一个可以分配多个任务上下文的对象。在这种情况下，与前文所述示例一样，有 3 项任务。

同样，DataArea 被分隔到不同的上下文中。与多任务对象的区别在于，现在可以将任务分配给不同的内核，从而真正实现并行处理。

| Object | RT-Core | Base Time (ms) | Cycle Time (ms) | Cycle Ticks | Priority |
|---|---|---|---|---|---|
| Task 2 | Core 2 | 1 ms | 10 ms | 10 | 1 |
| Task 3 | Core 1 | 1 ms | 20 ms | 20 | 2 |
| Task 4 | Core 0 | 1 ms | 30 ms | 30 | 3 |
| I/O Idle Task | Default (3) | 1 ms | 1 ms | 1 | 11 |

**特性：**

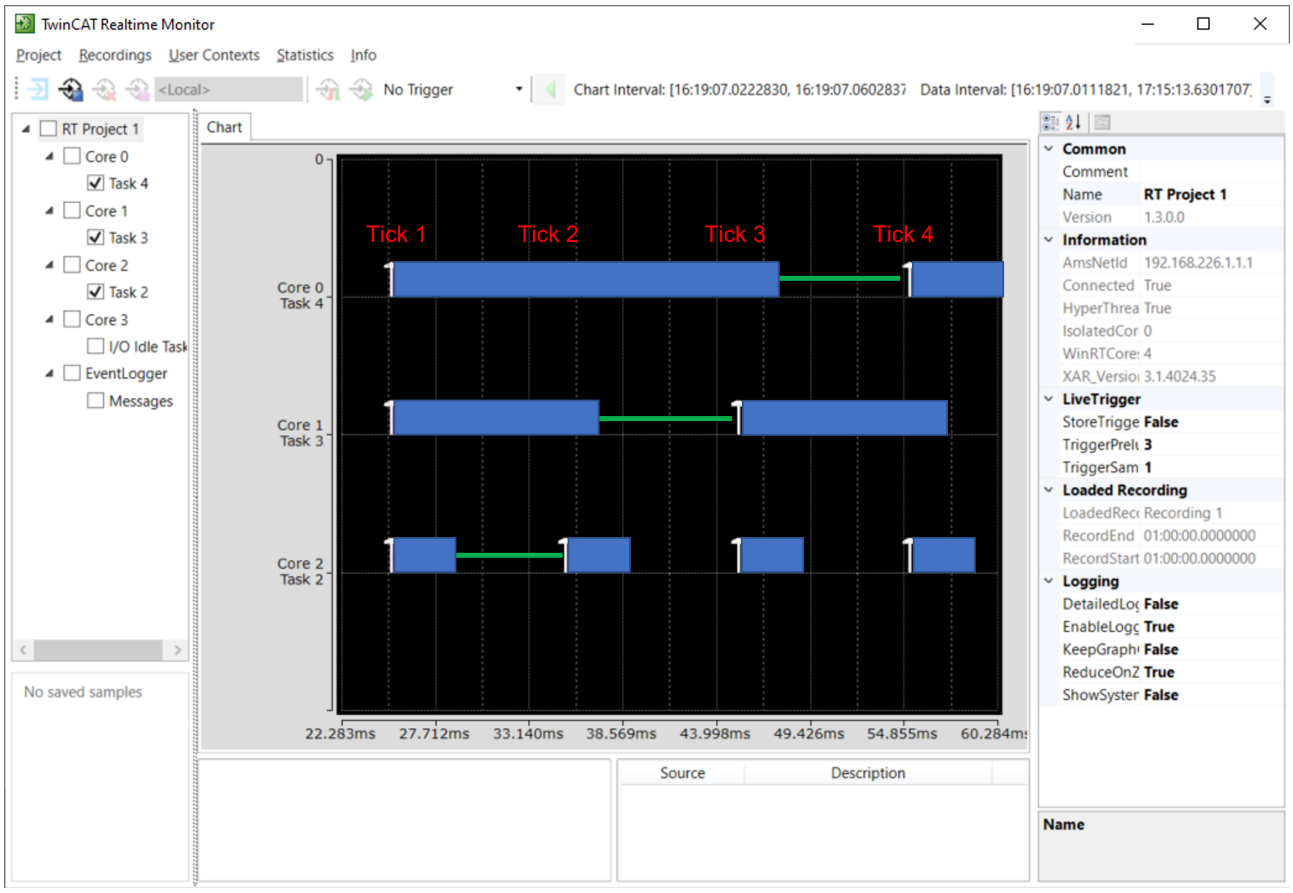
- 不支持PLC-FB功能块。
- 支持 TwinCAT Usermode Runtime。
- 任务可分配给不同的内核。
- 必须为最快的任务分配最高优先级（优先级值最小）。第二快的任务采用第二高的优先级，以此类推。

**调度详情：**

下图举例说明了如何分配计算时间。全蓝色区域表示任务正在运行。请注意，这些表面只是随后叠加在实时显示器图像上帮助理解，并非真实图像。

- 在 Tick 1 中，任务 2、任务 3 和任务 4 在不同内核上并行执行。任务 1 必须在 Tick 2 开始前执行完毕。

- 任务 2 在 tick 2 中再次执行。任务 3 和任务 4 可以继续运行。任务 2 和任务 3 必须在 Tick 3 开始前执行完毕。
- 在 tick 3 中再次执行任务 2 和任务 3。任务 4 可以继续运行。任务 2 和任务 4 必须在 Tick 4 开始前执行完毕。



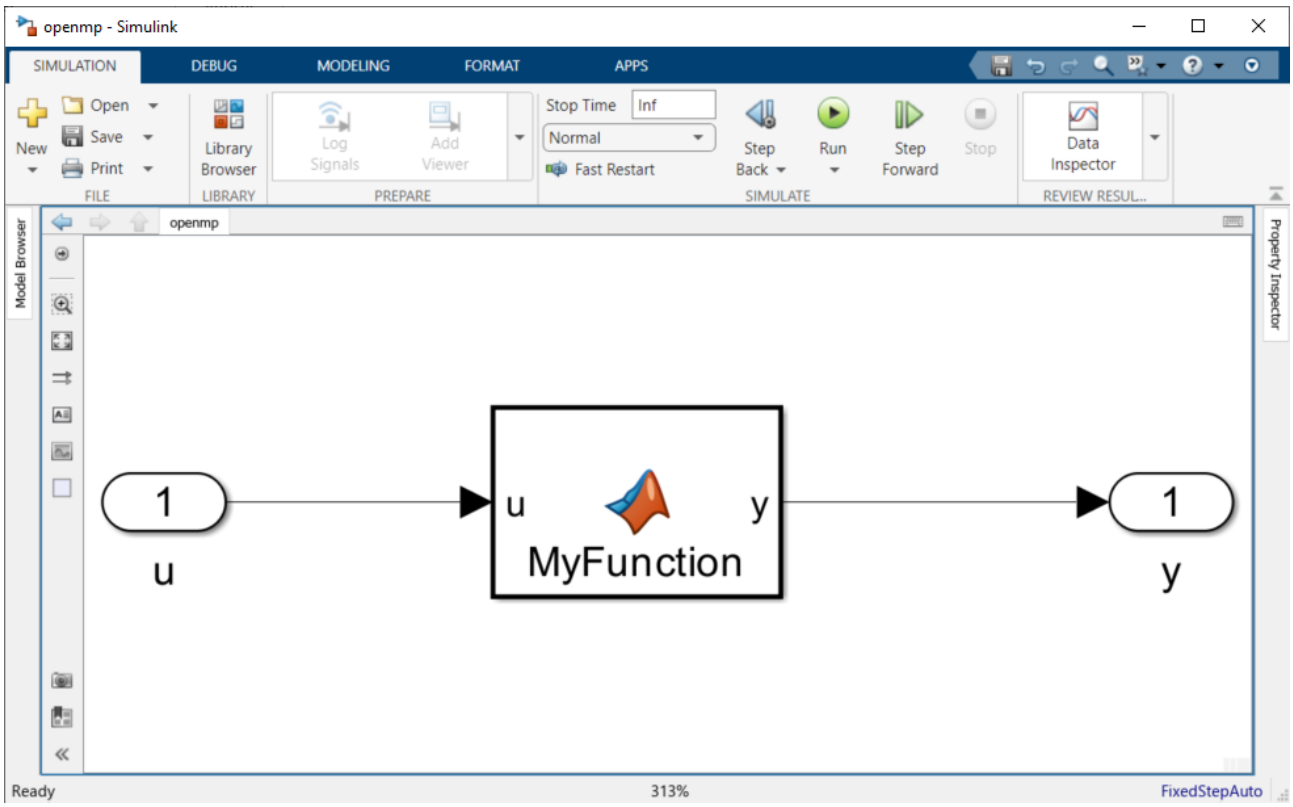

如果出现周期超时，无法遵循调度，则跳过相应任务上下文的执行，直到所有相关上下文都处于适当的状态。在 TcCOM 对象中，可以通过在线参数 `SkippedExecutionCount` 观察到这种行为。

### OpenMP

Simulink Coder™ 或 MATLAB Coder™ 可以生成 openMP 代码。请参见 MathWorks® 文档了解生成此类代码的具体情况。

下面是在 Simulink® 中使用 MATLAB® 函数的示例。MATLAB® 示例可与 TE1401 TwinCAT Target for MATLAB® 结合使用：

```
TwinCAT.ModuleGenerator.Samples.Start('Code parallelization with OpenMP').
```



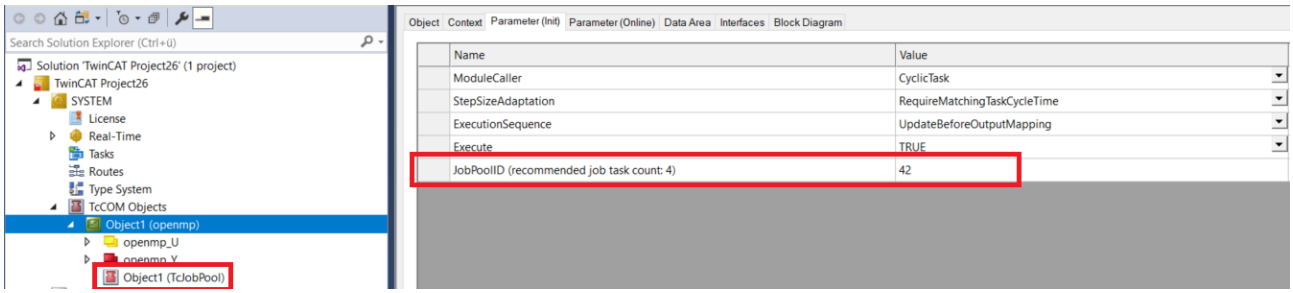

parfor 命令用于并行处理 MATLAB® 函数中的 FOR 循环。在这种情况下，并行工作线程的数量限制为 4。

```
function y = MyFunction(u) %#codegen

A = ones(20,50);
t = 42;

parfor (i = 1:10,4)
    A(i,1) = A(i,1) + t;
end

y = A(1,4) + u;
```

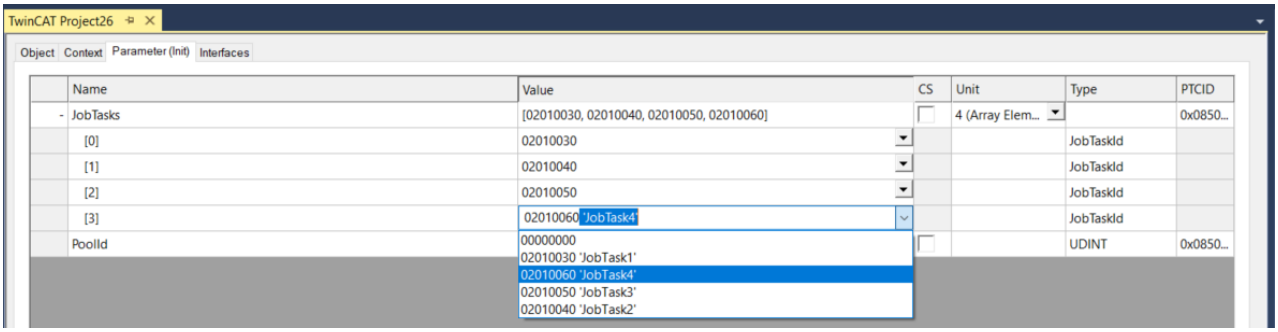
TwinCAT Target 无需对 openMP 进行特殊设置。像往常一样生成 TwinCAT 对象。Simulink® Coder™ 会将此代码编译成 openMP 代码，以便对 C/C++ 代码进行相应的并行化处理。此功能不需要使用 Embedded Coder®。

现在，可在 TwinCAT XAE 中对已创建的 TcCOM 或 PLC-FB 进行实例化，并进行相应配置。与往常一样，对象实例只在“上下文”（Context）选项卡下提供一个周期性任务接口。本例中创建了一个周期时间为 200 ms 的任务 2，并将其分配给对象。

参数（初始化）下有一个参数 JobPoolID。根据 C/C++ 代码，这里还显示了有多少工作线程可以并行工作。JobPool 是 JobTask 的组织单位，JobTask 可在任务节点中创建。



因此，必须通过 Add new item 在 TcCOM 对象（TcCOM Objects）下添加一个 TcJobPool 类型的对象。在 Parameter (Init) 选项卡的 TcJobPool 对象下，应输入 JobPoolId 并引用一组 JobTask。首先定义任务池应合并的 JobTask 数量，然后通过下拉菜单选择 JobTask。




可在 System > Realtime 下将 JobTask 分配给不同的内核。

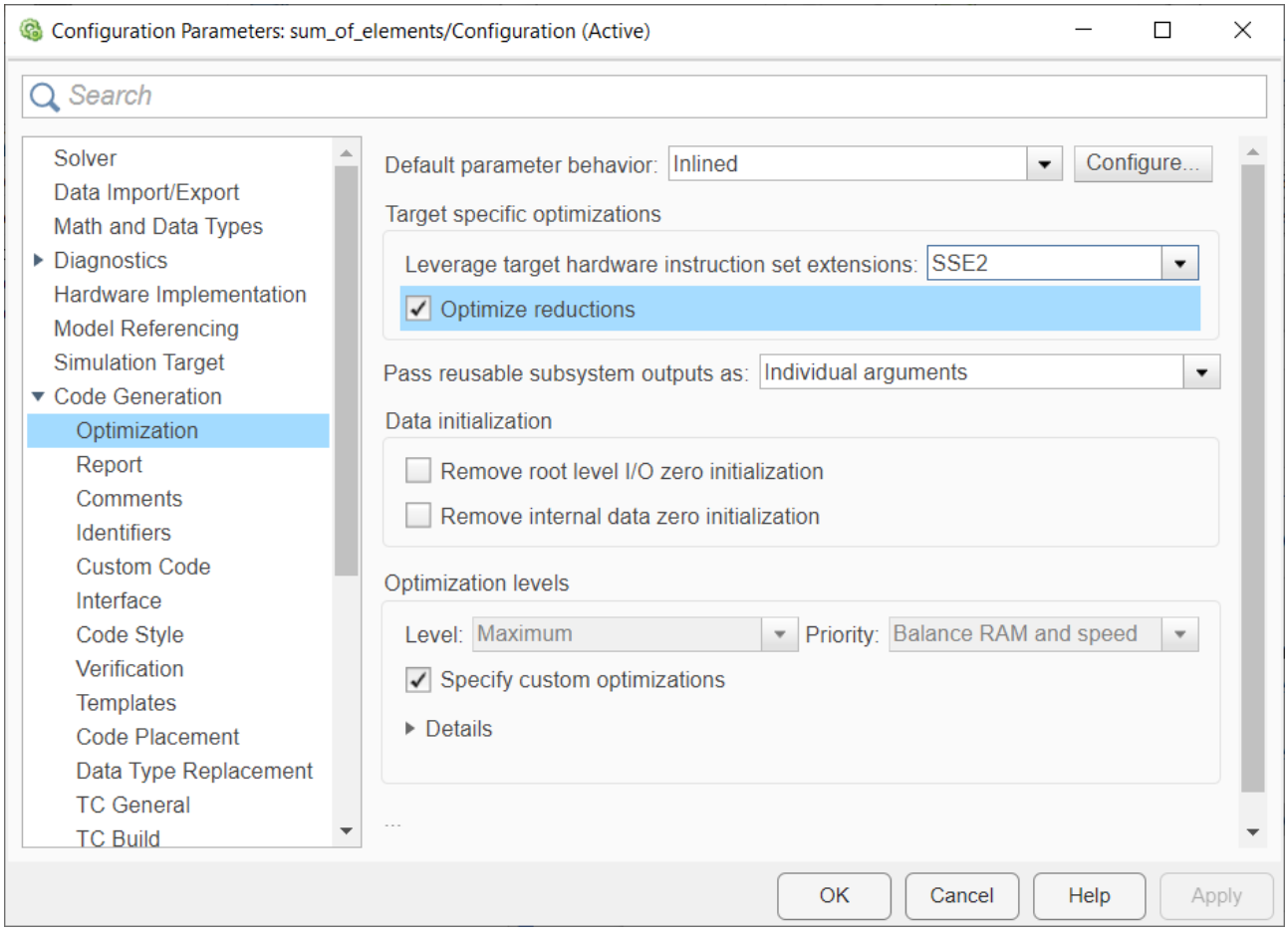
| Object | RT-Core | Base Time (ms) | Cycle Time (ms) | Cycle Ticks | Priority |
|---|---|---|---|---|---|
| Task 2 | Default (4) | 1 ms | 200 ms | 200 | 1 |
| JobTask1 | Core 0 | 1 ms | (none) | 0 | 2 |
| JobTask2 | Core 1 | 1 ms | (none) | 0 | 3 |
| JobTask3 | Core 2 | 1 ms | (none) | 0 | 4 |
| JobTask4 | Core 3 | 1 ms | (none) | 0 | 5 |
| I/O Idle Task | Default (4) | 1 ms | 1 ms | 1 | 11 |

上述配置中的执行过程如下。任务 2 在内核 4 上执行，并周期性地驱动 openMP 对象。然后，生成为 openMP 代码的代码片段可通过 JobPool 将任务外包给配置的 JobTask。JobTask 完成计算后，所有部分结果再次绑定，内核 4 上的任务 2 将代码执行到最后。

## 4.7.15 指令集扩展

指令集扩展可用于实现更好的运行时性能。虽然 Microsoft C++ 编译器的自动矢量器已将代码优化，并在默认情况下自动生成 SSE2 代码，但 *Intrinsics* 明确使用指令集扩展可以提高代码性能。

在使用 TE1400 TwinCAT Target for Simulink® 时，您可以明确要求使用 SSE2 代码。为此，请使用“代码生成 > 优化” (Code Generation > Optimization) 项目下 Simulink Coder™ 的以下属性。




**MATLAB® 中可用的示例**

```
TwinCAT.ModuleGenerator.Samples.Start("Simulink Instruction Set Extensions")
```

**有关代码性能的更多信息**

您可以使用 MathWorks® 的 Embedded Coder® 进一步进行优化。要在 TwinCAT 中使用 Embedded Coder™，请使用 Embedded Coder® 的 TwinCAT Target。例如，您还可以使用 AVX 命令。

同时充分利用求解器设置。一般来说，“离散”求解器的性能会明显优于高阶求解器。

## 4.7.16 符号特性和编译指令的赋予

**特性和赋予分别是什么?**

对于 TcCOM 对象，可将**特性**分配给所有定义，如 DataArea、DataType、SubItem 等。特性被定义为名称与值的对， 可以包含任何附加信息。

**赋予**通常在 PLC 的声明部分使用，例如，它也可以将任何附加信息绑定到变量中。关于 PLC 赋予列表，请参见 PLC 文档。

许多 TwinCAT 功能都使用赋予和特性。例如：

- TwinCAT OPC-UA
  - `{attribute 'OPC.UA.DA' := '1'}`
  - `{attribute 'OPC.UA.DA.Access' := '1'}`
- Analytics Logger
  - `{attribute 'TcAnalytics'}`

- JSON Library Tc3_JsonXml
- …

也可以由用户定义赋予和特性，并用于自己的应用程序。

**如何将符号的特性和赋予结合 TwinCAT 目标机使用?**

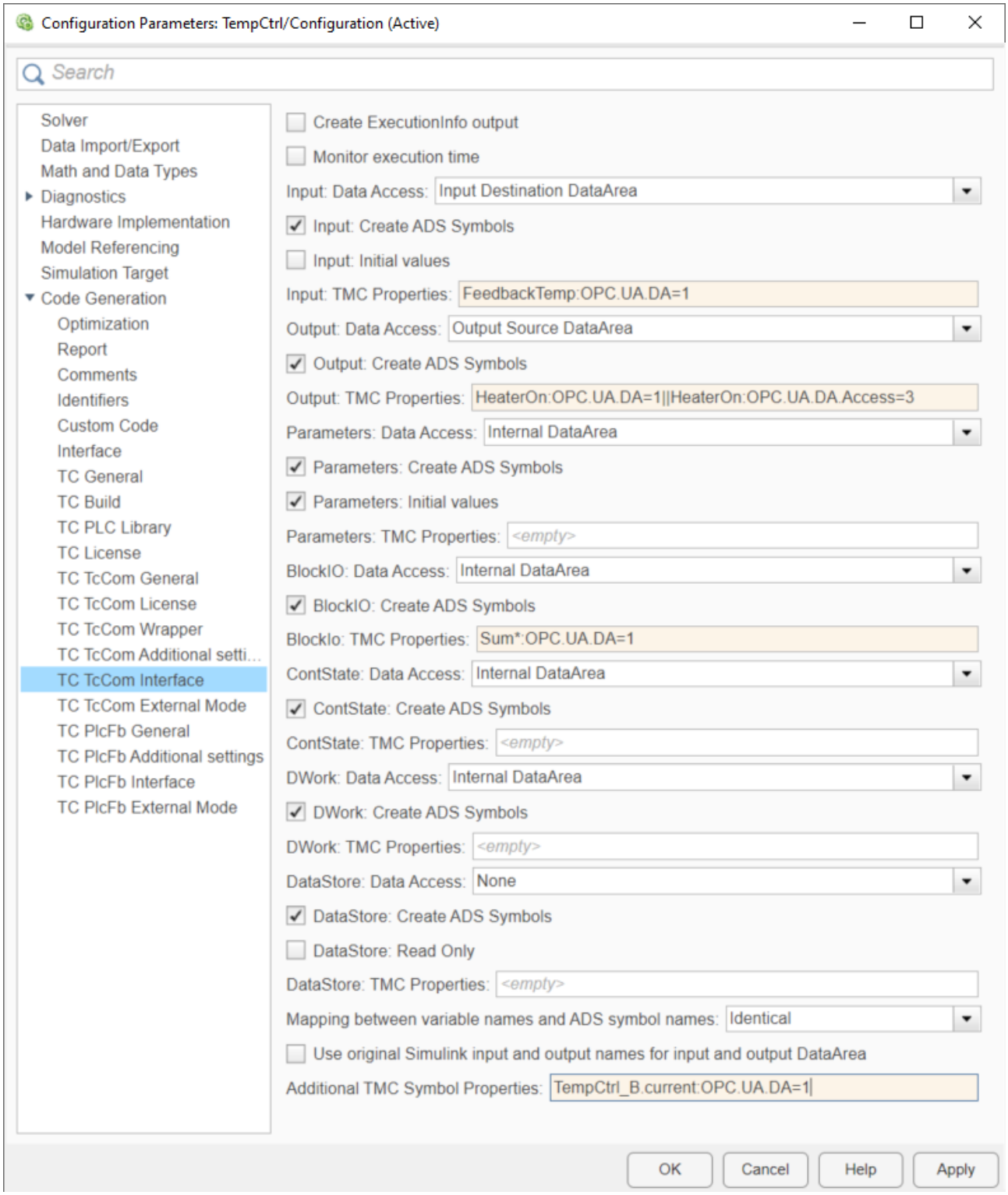
可以通过 TwinCAT 目标机为 ADS symbol 分配特性。ADS symbol 应从变量的角度来理解。

**（TcCOM）TMC 特性**

在 Simulink® 中，可在 TC TcCom 接口（TC TcCom Interface）下的“配置参数”（Configuration Parameters）中以字符串形式自由分配特性。可为 DataArea 定义特性：

- Input：TMC 特性
- Output：TMC 特性
- Parameters：TMC 特性
- BlockIO：TMC 特性
- ContState：TMC 特性
- DWork：TMC 特性
- DataStore：TMC 特性

此外，还可以不受 DataArea 的影响，使用*附加 TMC 符号特性*（Additional TMC Symbol Properties）字段分配特性。




以下符号用于表示 DataArea 特有的 TMC 特性：

```
SymbolName1:PropertyName1=Value1
```

示例：`FeedbackTemp:OPC.UA.DA=1`

如果要分配多个特性，必须用 | 分隔：

```
SymbolName1:PropertyName1=Value1|SymbolName2:PropertyName2=Value2
```

示例：`HeaterOn:OPC.UA.DA=1|HeaterOn:OPC.UA.DA.Access=3`

可以使用 MATLAB R2020b 的**通配符**：

```
*:PropertyName1=Value1
```

* 字符用作通配符，该示例将包含 Value1 的 PropertyName1 分配给 DataArea 中的所有符号。子字符串也可以与通配符结合使用。下面的示例将指定的属性分配给所有以 Sum 开头的符号。

```
Sum*:OPC.UA.DA=1
```

“附加 TMC 符号属性”（Additional TMC Symbol Properties）字段必须使用相同的结构。不过，必须为寻址添加 DataArea。

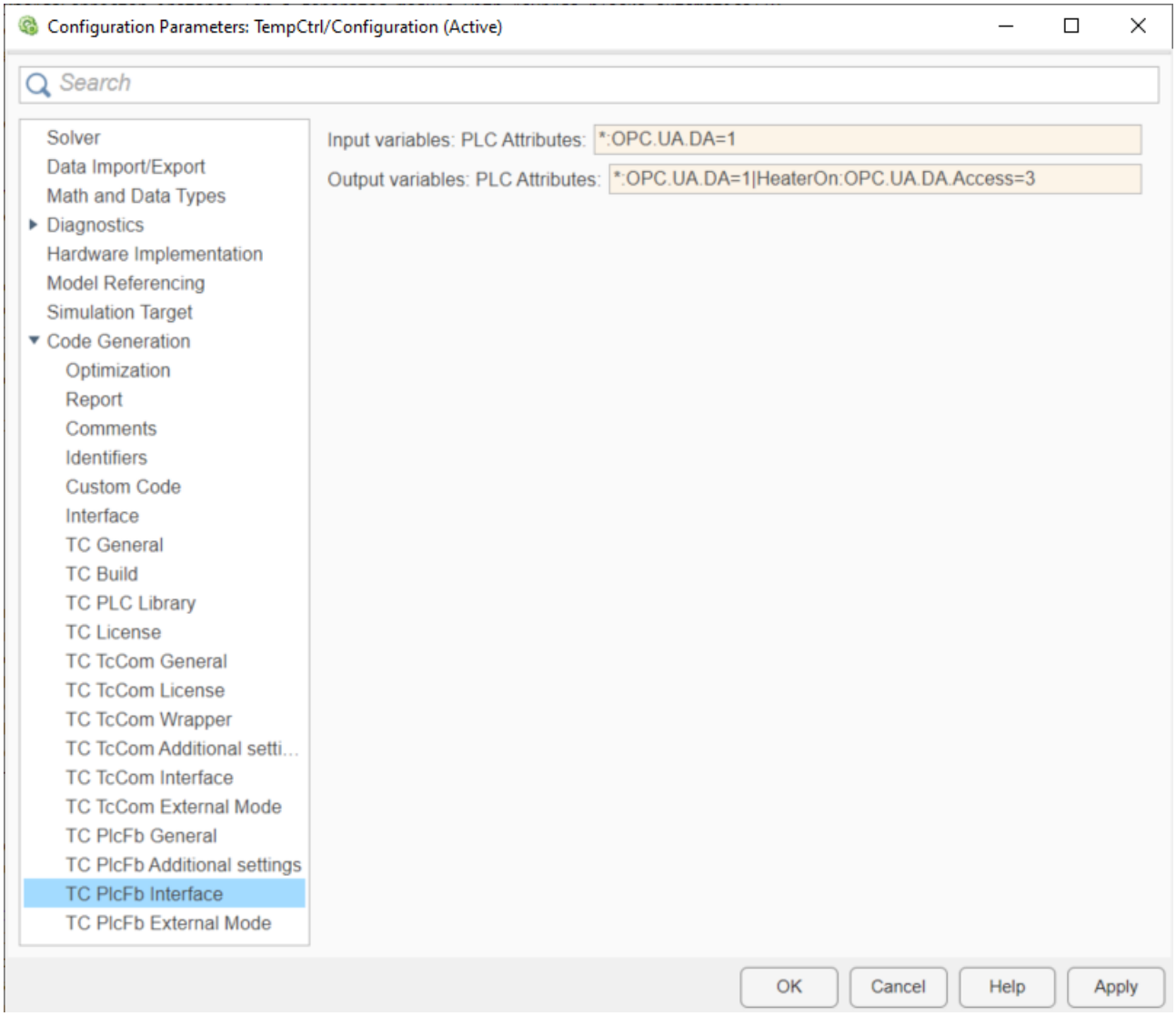
```
DataAreaName1.SymbolName1:PropertyName1=Value1
```

示例：`TempCtrl_B.current:OPC.UA.DA=1`

**PLC 属性**

PLC-FB [▶ 138] 的步骤与 TcCOM 相同，但只适用于输入和输出。使用 TcCOM-Wrapper-FB [▶ 134] 时会引用 TcCOM，以便您根据 TMC 特性进行配置。

PLC-FB 的配置在“TC PlcFb 接口”（TC PlcFb Interface）下进行。



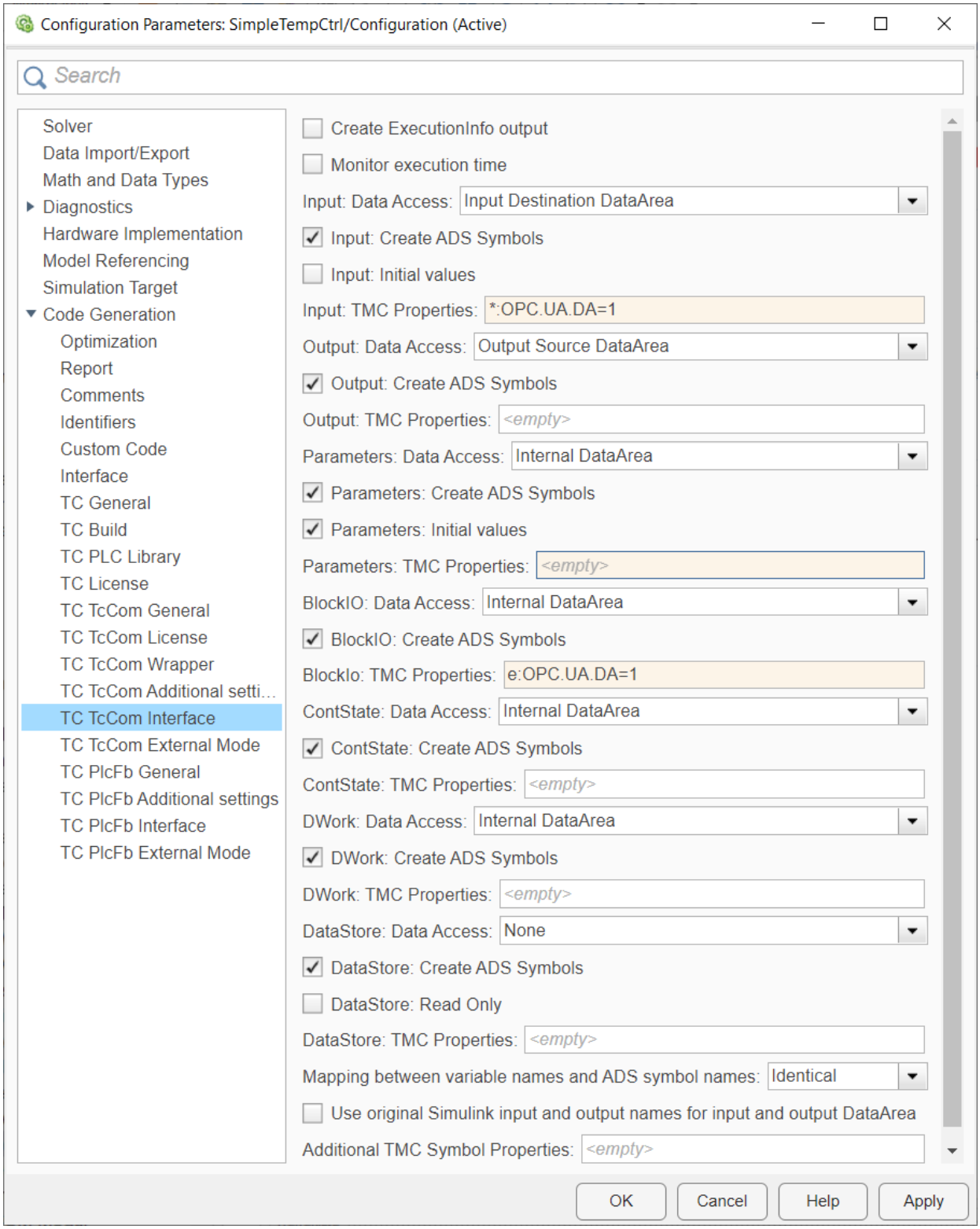

**MATLAB® 中的示例代码**

通过以下命令打开相应的示例：`TwinCAT.ModuleGenerator.Samples.Start(Add Properties to ADS Symbols')`

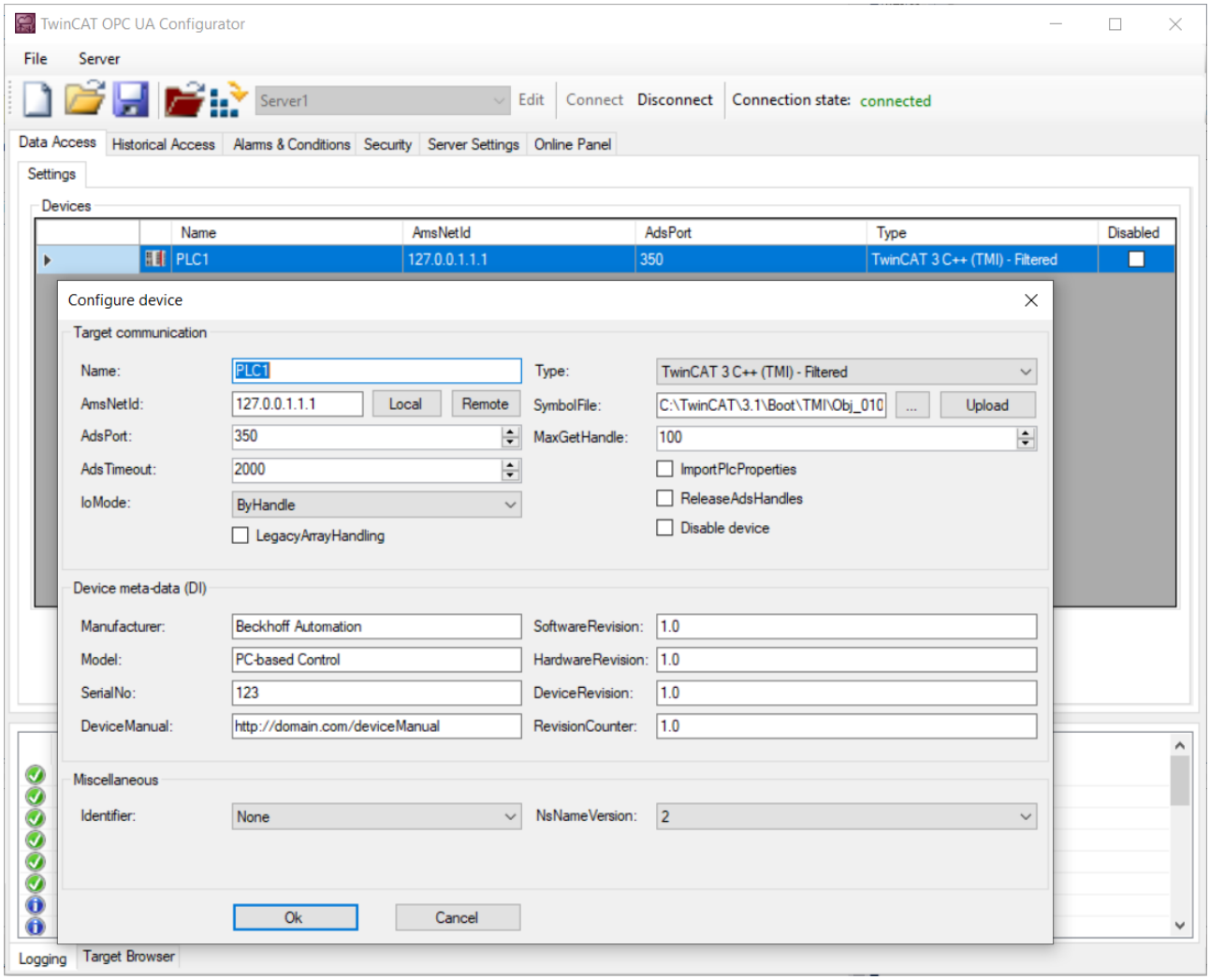
**特性和赋予分别有什么作用?**

例如，在 TcCOM 与**TwinCAT 3 OPC-UA** 结合使用的情况下使用 PLC 赋予或符号特性。请参见 OPC-UA 属性列表。

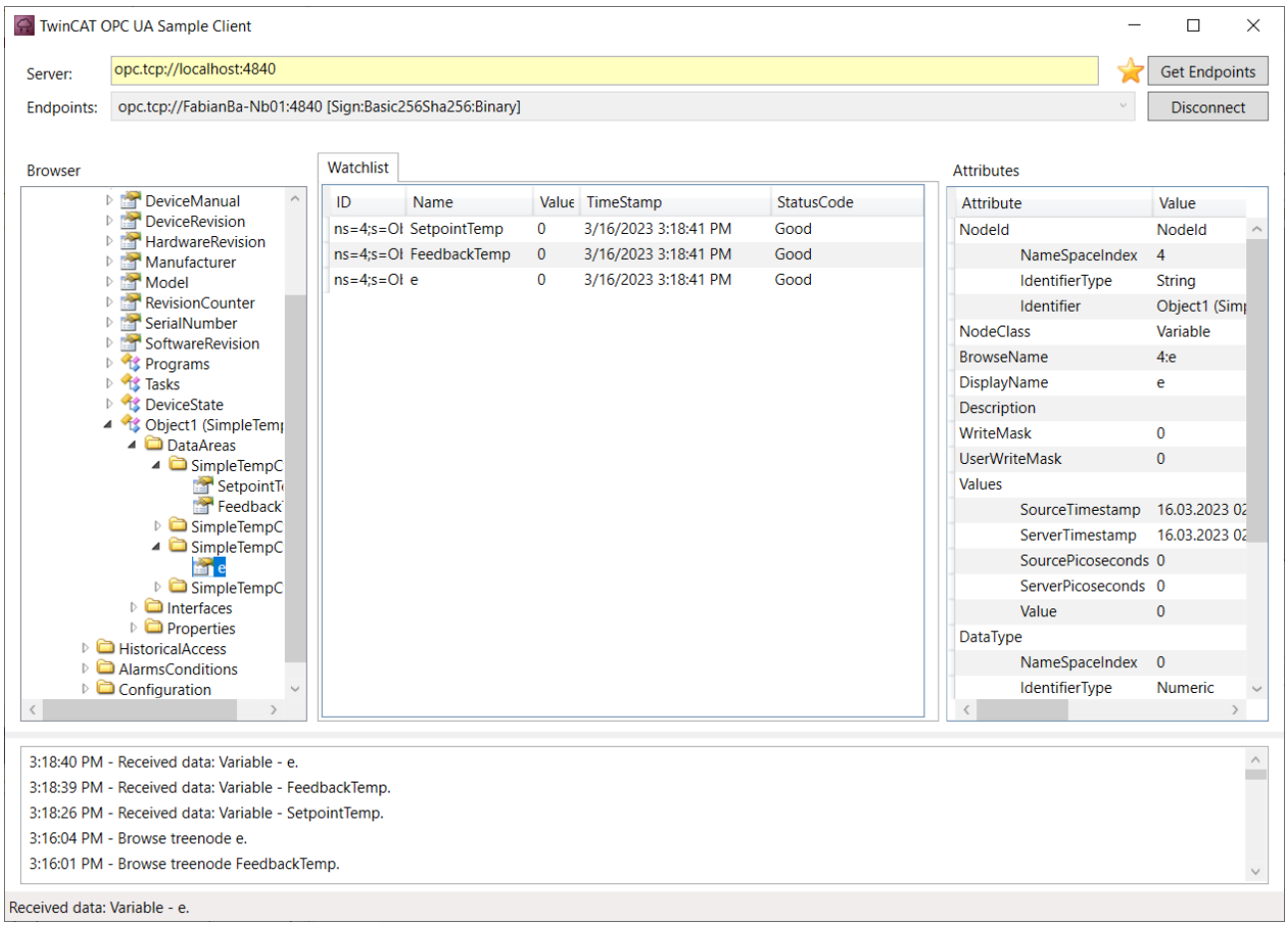
以下示例显示了如何使用 OPC-UA 数据访问属性提供 Input DataArea 的所有输入变量以及仅来自 BlockIO DataArea 的信号 e。通配符 * 用于 Input DataArea。




您必须在 OPC-UA Configurator 右侧的“类型”（Type）下选择“TwinCAT 3 C++ (TMI) - Filtered”，这样服务器中仅显示具有相应属性的符号。如果要查看服务器中的所有符号，只需在此处选择“TwinCAT 3 C++ (TMI) - All”。随后将显示所有符号，无论其特性如何。同时选择 TwinCAT 目标设备上的 TcCOM TMI 文件作为 SymbolFile（Boot/TMI 文件夹）。

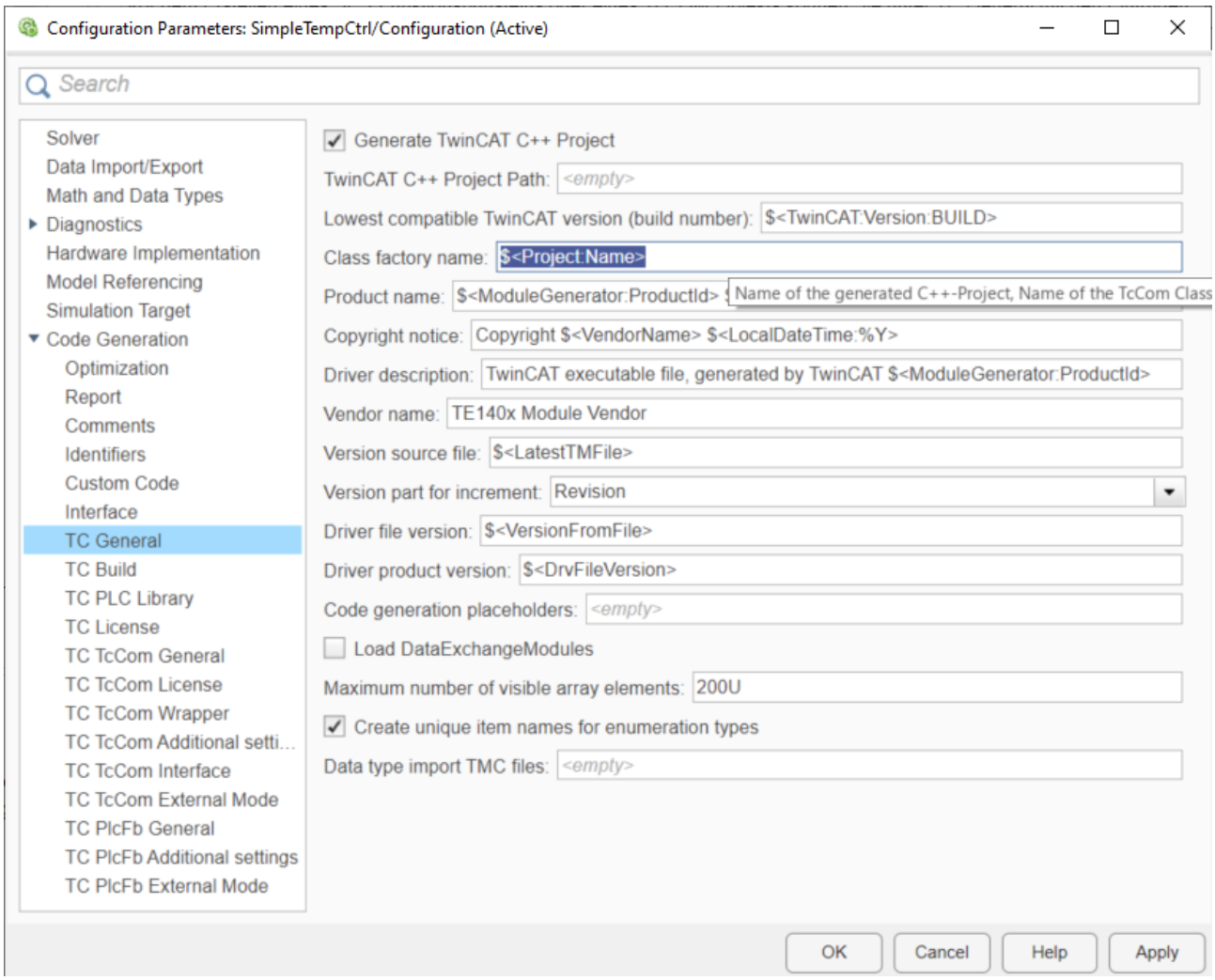



通过客户端连接 OPC-UA 服务器，检查命名空间。这里使用的是 TwinCAT 3 的 OPC-UA 示例客户端。您可以看到，服务器中只显示那些明确分配了 OPC.UA.DA=1 属性的符号。




## 4.7.17 可用占位符

在 Target for Simulink® 配置中使用占位符可以减少配置工作并提高清晰度。配置中使用 *$<PlaceholderName>* 指定占位符。




**什么是占位符?**

占位符可以作为变量抽象地指定配置参数的值。在目标机（模块生成器）、项目和模块（TcCOM 和 PLC FB）层面都有特定的占位符。此外，配置参数本身也是一个占位符，因此可复用。

**示例:**

在上图中，驱动程序产品版本是使用占位符 *$<DrvFileVersion>* 指定的。该占位符指向驱动程序文件版本条目，而该条目又被占位符 *$<VersionFromFile>* 占用。占位符 *$<VersionFromFile>* 代表版本值，版本值的来源由版本源文件参数定义。

**为什么需要占位符?**

使用占位符可以定义参数值，在配置中可在多处复用，或者将它们设置为相互关联的依赖链中（见上例）。

**可用占位符概述**

**配置参数组中的占位符**

| Category | Name | Displayname | Default | Description |
|---|---|---|---|---|
| TC General | Generate | Generate TwinCAT C++ Project | TRUE | Generate a TwinCAT C++ project. If unset, only code artifacts will be generated which can get used to generate C++ projects later [▶ 52]. |

| Category | Name | Displayname | Default | Description |
|---|---|---|---|---|
| | ProjectDir | TwinCAT C++ Project Directory | | Full path to the directory with the VCXPROJ file (e. g. "C:\Temp") |
| | ProjectName | TwinCAT C++ Project Name | | Name to the generated VCXPROJ file (e. g. "MyGeneratedProject.vcxproj") |
| | LowestCompatibleTcBuild | Lowest compatible TwinCAT version (build number) | $<TwinCAT:Version:BUILD> | The lowest TwinCAT build number the generated C++ project and its modules and POUs are to be compatible with. |
| | ClassFactoryName | Class factory name | $<Project:Name> | Name of the generated C++-Project, Name of the TcCOM classfactory and tmx-file name |
| | ProductName | Product name | $<ModuleGenerator:ProductId> $<ModuleGenerator:Version:MAJOR.MINOR> | Product name, used e.g. for the module driver description and the module TMC description [▶ 83]. |
| | Copyright | Copyright notice | Copyright $<VendorName> $<LocalDateTime:%Y> | Copyright notice of the generated module driver file [▶ 83] |
| | Description | Driver description | TwinCAT executable file, generated by TwinCAT $<ModuleGenerator:ProductId> | Driver description [▶ 83] |
| | VendorName | Vendor name | TE140x Module Vendor | Module vendor name, used as the company name of the generated executables in the repository [▶ 40] and the major module group as shown in the TwinCAT XAE module dialog. |
| | VersionSrc | Version source file | $<LatestTMFile> | Path to an existing TMC, TML or XML file containing the previous version value [▶ 61]. |
| | IncrementVersion | Version part for increment | Revision | The part of the version number that is to be incremented [▶ 61]. |
| | DrvFileVersion | Driver file version | $<VersionFromFile> | Executable file version and library version. [▶ 61] |
| | DrvProductVersion | Driver product version | $<DrvFileVersion> | Product version [▶ 61] |
| | CodeGenPlaceholders | Code generation placeholders | | Define custom placeholders |
| | UseDataExchangeModules | Load DataExchangeModules | 0 | Manually set DataExchangeModule [▶ 59] dependency (currently no need to set manually) |

| Category | Name | Displayname | Default | Description |
|---|---|---|---|---|
| | MaxVisibleArrayElements | Maximum number of visible array elements | 200U | Specifies the maximum number of array elements to be displayed in the TwinCAT XAE. In the TwinCAT XAE, larger arrays cannot get expanded and linked to by its individual items |
| | PackOutputPath | Pack output path | | Path to pack the generated TwinCAT C++ project. Optional. Can be a directory or .zip file. See Sample "*Use Continuous Integration Principles with Code Generation*" |
| TC Build | PreferToolArchitectureX64 | Prefer X64 build tools | TRUE | Prefer X64 compiler and linker. Useful for complex source files, where X86 tools may run out of heap space. |
| | CppLanguageStandard | Specify C++ language standard version | Default | Enable supported C++ language features from the specified version of the C++ language standard |
| | Verbosity | Codegeneration and build verbosity | Normal | Verbosity level of code generation and build output messages. Silent and Detailed are other possible values. |
| | Publish | Run the publish step after project generation | TRUE | Start the build procedure after code generation for all selected platforms. The generated module binaries and module descpription files will get copied to the "publish folder". Published modules are automatically located by the XAE and can get instantiated in all TwinCAT 3 projects. If unset, the module generation process will be stopped after code generation. To instantiate in a TwinCAT3 project, the generated C++ project needs to be inserted and built from. |
| | PublishPlatformtoolset | Platform Toolset | Auto | Choose Platform Toolset to build binaries. |
| | PublishConfiguration | Build configuration | Release | Build configuration to build binaries. |
| | PublishTcRTx86 | TwinCAT RT (x86) | TRUE | Publish binaries for platform 'TwinCAT RT (x86).' [▶ 54] |
| | PublishTcRTx64 | TwinCAT RT (x64) | TRUE | Publish binaries for platform 'TwinCAT RT (x64).' [▶ 54] |
| | PublishTcOSx64 | TwinCAT OS (x64) | TRUE | Publish binaries for platform 'TwinCAT OS (x64)' (e.g. TwinCAT/BSD) [▶ 54] |

| Category | Name | Displayname | Default | Description |
|---|---|---|---|---|
| | PublishTcOSARMv8A | TwinCAT OS (ARMV8-A) (TwinCAT XAE >= 3.1.4026) | FALSE | Publish binaries for platform 'TwinCAT OS (ARMV8-A)' (requires TwinCAT XAE >= 3.1.4026) [▶ 54] |
| | ForceRebuildForPublish | Always rebuild all source files on publish | FALSE | Always rebuild all source files on publish |
| | PublishParallel | Build parallel to publish | TRUE | Build parallel on multiple cores to publish |
| | SignTwinCatCertName | Certificate name for TwinCAT signing | | Certificate name for TwinCAT signing with OEM Certificate level 2. [▶ 14] |
| | TmxInstall | Install TMX | TRUE | Install all generated TwinCAT Objects on local XAE (fill local Engineering Repository [▶ 40]). |
| | TmxArchive | TMX Archive | | Name of an optional archive containing all files required to use the generated TwinCAT Objects on another TwinCAT development system. [▶ 57] |
| | MsBuildPublishProperties | MsBuild publish properties | | Set additional MsBuild publish properties. |
| | MsBuildProjProperties | MsBuild project properties | | Set additional MsBuild project properties. |
| | PreCodeGenerationCallbackFcn | Pre code generation callback function | | The defined MATLAB® function is called before code generation [▶ 108]. |
| | PostCodeGenerationCallbackFcn | Post code generation callback function | | The defined MATLAB® function is called after code generation [▶ 108]. |
| | PostPublishCallbackFcn | Post publish callback function | | The defined MATLAB® function is called after publish [▶ 108]. |
| | DeploymentPath | Deployment project | | Full path to a TwinCAT project (.tsproj). Instances of the generated TcCOM Module in the specified project will be upgraded to the newly generated version [▶ 59] |
| | DeployRestart | Activate and restart deployment project | FALSE | If set the specified TwinCAT project will be activated and restarted on the configured target system [▶ 59] |
| | PostDeployCallbackFcn | Post deploy callback function | | The defined MATLAB function is called after deployment [▶ 108] |
| TC PLC library | LibCatPath | PLC library category description file | $<ProjectDir>\$<Name>.libcat.xml | Path to the PLC library category description file |
| | LibraryCategories | PLC library categories | $<VendorName> | Define PLC library category hierarchy. Default only one hierarchy level = vendor. List |

| Category | Name | Displayname | Default | Description |
|---|---|---|---|---|
| | | | | separated with \| possible: <MainCategory>\| <SubCategory1>\|… |
| | GeneratePlcLibrary | Generate a PLC library | FALSE | Generate a PLC library with POUs. [▶ 131] Define containing POUs with parameter TcComWrapperFb and PlcFb>General>Generate. |
| | InstallPlcLibrary | Install the generated PLC library | FALSE | Install the generated PLC library for use in the local TwinCAT XAE/PLC [▶ 131]. |
| | PlcTypePrefixes | Type Prefixes | | Define custom type prefixes |
| | PlcVarPrefixes | Variable Prefixes | `PVOID=p \| BOOL=b \| BOOL32=b \| DATE=d \| TIME_OF_DATE=td \| TIME=t \| LTIME=t \| GUID=n` | Define custom variable prefixes. |
| TC License | OemId | ID of OEM | | ID of OEM. Required for OEM Licence checks [▶ 80] |
| | OemLicenses | IDs of OEM Licenses | | IDs of OEM Licenses. Multiple IDs may be inserted as a comma separated list. "{GUID},{GUID}" [▶ 80] |
| TC TcCom General | Generate | Generate TcCOM Module (TwinCAT Module Class) | TRUE | Generate a TcCOM module class for the model. |
| | OnlineChange | Online change support | FALSE | Allow to switch between different TcCOM module versions without switching TwinCAT runtime to config mode. |
| | ModuleProperties | TMC Properties | | Additional properties added to the module description in the TMC file: Name1=Value1\| Name2=Value2\|... |
| | GroupName | GroupName | TE140x\| Simulink Modules | Minor module group name in the TwinCAT XAE module dialog |
| | GroupDisplayName | GroupDisplayName | $<GroupName> | Minor module group description in the TwinCAT XAE module dialog |
| | GroupIcon | GroupIcon | $<TE140x:Icon> | Optional module group icon in the TwinCAT XAE module dialog |
| | ModuleIcon | ModuleIcon | $<TE140x:Icon> | Optional module icon in the TwinCAT XAE module dialog |
| | InitExceptionHandling | Floating point exception handling during initialization | CallerExceptions | Configures how to throw, suppress or handle floating point exceptions during initialization [▶ 145]. |

| Category | Name | Displayname | Default | Description |
|---|---|---|---|---|
| | UpdateExceptionHandling | Floating point exception handling during update | CallerExceptions | Configures how to throw, suppress or handle floating point exceptions during cyclic execution [▶ 145]. |
| | AdditionalIncludeFiles | Additional include files | | Additional files required to be included after rtwtypes.h |
| TC TcCom License | OemLicenses | IDs of OEM License | $<Project:OemLicenses> | IDs of OEM Licenses. Multiple IDs may be inserted as a comma separated list. "{GUID},{GUID}" [▶ 80] |
| TC TcCom Wrapper | TcComWrapperFb | TcCom Wrapper FB | FALSE | Generate a PLC Functionblock simplifying the interaction between a PLC and an instance of the generated TcCOM module [▶ 134] |
| | TcComWrapperFbProperties | TcCom Wrapper FB properties | FALSE | Generate properties for accessible data in the referenced TcCOM object [▶ 134] |
| | TcComWrapperFbPropertyMonitoring | TcCom Wrapper FB property monitoring | NoMonitoring | NoMonitoring: Online values of properties are not monitored in the PLC online view, CyclicUpdate: Update property values in the PLC online view cyclically, ExecutionUpdate: Update property values in the PLC online view when the property getter or setter is called [▶ 134] |
| TC TcCom Additional settings | ModuleCaller | Default module caller | CyclicTask | CyclicTask: Call module via TwinCAT Task. Module: Cal module from another TwinCAT module (see e.g. TcCOM-Wrapper-FB). |
| | CallerVerification | Verify caller | Default | Verify the caller context to prevent concurrent execution of the model code and corresponding DataArea mappings. Skip verification to reduce the execution time. |
| | StepSizeAdaptation | Default StepSize adaptation mode | RequireMatchingTaskCycleTime | Configure how to handle differences between the default model step size(s) and the cycle time of the assigned task(s). |
| | ExecutionSequence | Default execution sequence | UpdateBeforeOutputMapping | Configure the execution order of input mapping, model code execution and output mapping. |
| | ExecuteModelCode | Execute model code after startup | TRUE | Start cyclic execution of the model code after startup by default. If FALSE, Module Parameter Execute needs to be set to TRUE to start execution of code. |

| Category | Name | Displayname | Default | Description |
|---|---|---|---|---|
| | BlockDiagramExport | Export BlockDiagram | TRUE | Export graphical block diagram information for monitoring and optional debugging on the generated TwinCAT module in TwinCAT XAE [▶ 118] |
| | ResolveMaskedSubsystems | Resolve Masked Subsystems | FALSE | Resolve masked subsystems in the block diagram |
| | ExtendSignalResolution | Extended resolution of signals in block diagram | FALSE | Intensified search for assignments of variables and block diagram signals (blue signals). This option increases the build time. [▶ 159] |
| | BlockDiagramVariableAccess | Access to VariableGroup not referenced by any block | AssignToParent | Variables from a block within an unresolved subsystem are either assigned to the next higher visible block or hidden in the block diagram. |
| | BlockDiagramDebugInfoExport | Export BlockDiagram debug info | TRUE | Export additional information required to debug the module using the block diagram [▶ 122]. |
| TC TcCom Interfaces | ExecutionInfoOutput | Create ExecutionInfo output | FALSE | Create additional output DataAreas containing execution and exception information [▶ 145]. |
| | MonitorExecutionTime | Monitor execution time | FALSE | Calculate and expose the execution time of the module as an ADS variable for monitoring purposes. |
| | InputDataAccess | Input: Data Access | Input Destination DataArea | Defines how the input variables are exposed in TwinCAT [▶ 67]. |
| | InputCreateSymbols | Input: Create ADS Symbols | TRUE | Create ADS symbol information for the input variables [▶ 67] |
| | InputInitValues | Input: Initial values | FALSE | Create module parameters for the input variables to allow definition of initial values [▶ 67] |
| | InputProperties | Input: TMC Properties | | Additional properties added to the Input symbol description in the TMC file. [▶ 91] |
| | OutputDataAccess | Output: Data Access | Output Source DataArea | Defines how the output variables are exposed in TwinCAT [▶ 67]. |
| | OutputCreateSymbols | Output: Create ADS Symbols | TRUE | Create ADS symbol information for the output variables [▶ 67]. |
| | OutputProperties | Output: TMC Properties | | Additional properties added to the Output symbol description in the TMC file. [▶ 91] |
| | ParametersDataAccess | Parameters: Data Access | Internal DataArea | Defines how the model parameter variables are exposed in TwinCAT [▶ 67] |
| | ParametersCreateSymbols | Parameters: Create ADS Symbols | TRUE | Create ADS symbol information for the model parameter variables [▶ 67]. |

| Category | Name | Displayname | Default | Description |
|---|---|---|---|---|
| | ParametersInitValues | Parameters: Initial values | TRUE | Create module parameters for the model parameter variables to allow definition of initial values [▶ 67]. |
| | ParametersProperties | Parameters: TMC Properties | | Additional properties added to the Parameters symbol description in the TMC file. [▶ 91] |
| | BlockIoDataAccess | BlockIO: Data Access | Internal DataArea | Defines how the BlockIO variables are exposed in TwinCAT [▶ 67] |
| | BlockIoCreateSymbols | BlockIO: Create ADS Symbols | TRUE | Create ADS symbol information for the BlockIO variables [▶ 67]. |
| | BlockIoProperties | BlockIO: TMC Properties | | Additional properties added to the BlockIO symbol description in the TMC file. [▶ 91] |
| | ContStateDataAccess | ContState: Data Access | Internal DataArea | Defines how the continuous state variables are in TwinCAT [▶ 67] |
| | ContStateCreateSymbols | ContState: Create ADS Symbols | TRUE | Create ADS symbol information for the continuous state variables [▶ 67]. |
| | ContStateProperties | ContState: TMC Properties | | Additional properties added to the ContState symbol description in the TMC file. [▶ 91] |
| | DWorkDataAccess | DWork: Data Access | Internal DataArea | Defines how the DWork variables are exposed in TwinCAT [▶ 67] |
| | DWorkCreateSymbols | DWork: Create ADS Symbols | TRUE | Create ADS symbol information for the DWork variables [▶ 67]. |
| | DWorkProperties | DWork: TMC Properties | | Additional properties added to the DWork symbol description in the TMC file. [▶ 91] |
| | DataStoreDataAccess | DataStore: Data Access | None | Defines how the DataStore variables are exposed in TwinCAT [▶ 67] |
| | DataStoreCreateSymbols | DataStore: Create ADS Symbols | TRUE | Create ADS symbol information for the DataStore variables [▶ 67]. |
| | DataStoreReadOnly | DataStore: Read Only | FALSE | Restrict ADS access to be read only for the DataStore variables [▶ 67]. |
| | DataStoreProperties | DataStore: TMC Properties | | Additional properties added to the DataStore symbol description in the TMC file. [▶ 91] |
| | SymbolProperties | Additional TMC Symbol Properties | | Additional properties added to specific symbol descriptions in the TMC file. [▶ 91] |
| | VariableSymbolMapping | Mapping between variable names and ADS symbol names | Identical | Defines the TwinCAT symbol names for the generated C/C++ variables. 'Identical': Symbol name equals variable name, |

| Category | Name | Displayname | Default | Description |
|---|---|---|---|---|
| | | | | 'Classic': Use symbol names known from TE1400 Release 1.2.x.x [▶ 67] |
| TC TcCom External Mode | ExtModeRtAllowExecutionCommands | Allow RealTime execution commands via External Mode | FALSE | Allow to start and stop model code execution via External Mode [▶ 142]. |
| | ExtModeRtWaitForStart | Wait for RealTime execution start command via External Mode | FALSE | Wait for External Mode [▶ 142] connection before starting model code execution. |
| | ExtModeRtAllowForParameterChange | Allow to change parameters via External Mode | FALSE | Allow to change parameter online values via External Mode [▶ 142]. |
| TC PlcFb General | Generate | Generate TwinCAT PLC Function Block | TRUE | Generate a PLC-FB for the model [▶ 138]. |
| | InitExceptionHandling | Floating point exception handling during initialization | CallerExceptions | Configures how to throw, suppress, or handle floating point exceptions during initialization [▶ 145]. |
| | UpdateExceptionHandling | Floating point exception handling during update | CallerExceptions | Configures how to throw, suppress, or handle floating point exceptions during cyclic execution [▶ 145]. |
| TC PlcFb License | OemLicenses | IDs of OEM License | $<Project:OemLicenses> | IDs of OEM Licenses. Multiple IDs may be inserted as a comma seperated list. "{GUID},{GUID}" [▶ 80] |
| TC PlcFb Additional settings | MonitorExecutionTime | Monitor ExecutionTime | FALSE | Calculate and expose the execution times of TwinCAT modules as an ADS variable for monitoring purposes. |
| PlcFb->Interface | InputAttributes | Input variables: PLC Attributes | | Additional attributes added to the PLC FB Input variables. |
| | OutputAttributes | Output variables: PLC Attributes | | Additional attributes added to the PLC FB Input variables. |
| TC PlcFb External Mode | ExtModeRtAllowExecutionCommands | Allow RealTime execution commands via External Mode | FALSE | Allow to start and stop model code execution via External Mode [▶ 142]. |
| | ExtModeRtWaitForStart | Wait for RealTime execution start command via External Mode | FALSE | Wait for External Mode connection before starting model code execution [▶ 142]. |
| | ExtModeRtAllowForParameterChange | Allow to change parameters via External Mode | FALSE | Allow to change parameter online values via External Mode [▶ 142]. |

**目标机层（模块生成器）占位符**

该组中的占位符可用于目标机/项目和模块层级。

| Placeholder name | Description |
|---|---|
| ModuleGenerator:ProductName | Product name of the module generator |

| Placeholder name | Description |
|---|---|
| ModuleGenerator:Version | Version of the module generator |
| TwinCAT:Version | Version of local TwinCAT installation |
| UsablePlatformToolsets | Available and supported platform toolsets |
| LocalDateTime:Format | Actual local time as string where Format must be defined like the format string for std::put_time (e.g. '%Y-%m-%d') |
| UtcDateTime:Format | Actual UTC time as string where Format must be defined like the format string for std::put_time (e.g. '%Y-%m-%d') |
| EnvironmentVarName | Any environment variable defined for the system, the current user or the current process (MATLAB®). |

**项目层占位符**

该组中的占位符可用于项目和模块层级。

| Placeholder name | Description |
|---|---|
| Project:Name | Name of the project file (without directory and extension) |
| Project:Dir | Project file directory |
| Project:Ext | Project file extension |
| Project:Path | Full project file path |
| Project:Guid | Project GUID |
| Project:LibraryID | LibraryID of the generated repository driver |
| Project:VendorName | Company name part of the LibraryID |
| Project:DriverName | Driver name part of the LibraryID |
| Project:DrvFileVersion | Version part of the LibraryID |
| Project:LatestTMFile | Path to an existing corresponding TML or TMC file with the highest library version (searching project directory and repository) |
| Project:LatestTMFile:Repository | Path to an existing corresponding TML or TMC file with the highest library version (searching only repository) |
| Project:LatestTMFile:ProjectDir | Path to an existing corresponding TML or TMC file with the highest library version (searching only project directory) |
| Project:VersionFromFile | Version read from file defined by "VersionSrc" |

**模块层占位符（TcCOM 和 PLC FB）**

该组中的占位符只能在模块层级使用。

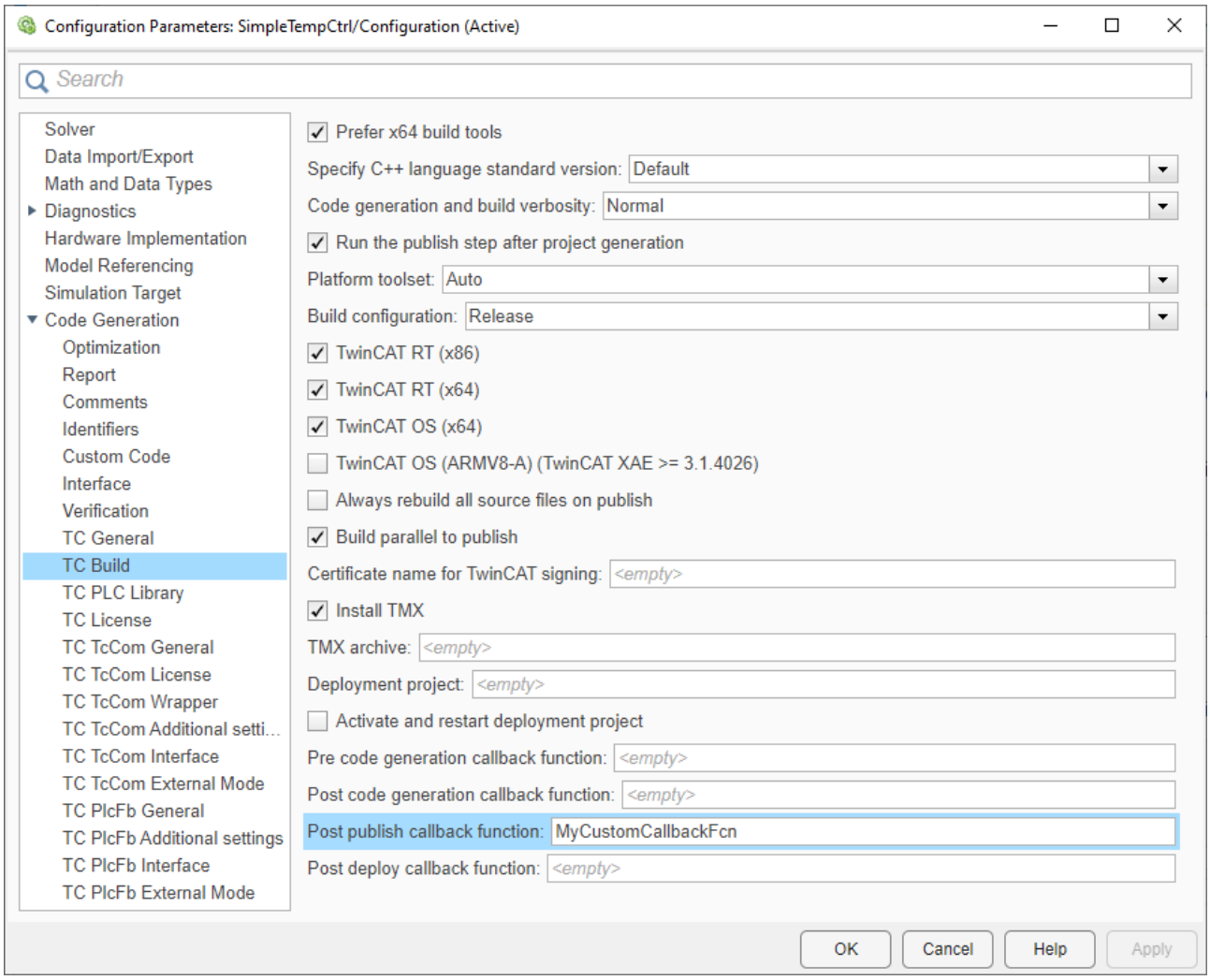
| Placeholder name | Description |
|---|---|
| Module:Name | Name of the TcCom module or PLC FB |
| Module:ClsId | Class ID of the TcCom module (TcCom only) |
| Module:ContextCount | Number of task contexts |
| Module:ClassName | Name of the TcCom module |
| Module:CppClassFileName | Name of the corresponding .h and .cpp files |
| Module:ModelName | Name of the corresponding Simulink Model (TE1400 only) |
| Module:MFileName | Name of the corresponding M-File (TE1401 only) |
| Module:FileFilterName | Visual Studio project filter name |

## 4.7.18 使用回调函数

有三种不同的回调函数：

- **代码生成前回调函数：**模型转换为 C++ 代码前的回调函数。
- **代码生成后的回调函数：**模型转换为 C++ 代码后的回调函数。
- **发布后回调函数：**为配置的平台创建 C++ 项目后的回调函数。
- **部署后回调函数：**回调将在“部署项目”中指定的项目更新后执行。

在此处输入所创建的 MATLAB® 函数的名称即可调用。




MATLAB® 函数将 ProjectExporter 对象作为传输参数传递：

```
function MyCallback(obj)
…
return
```

该对象的特性中包含当前编译版本的配置。

```
ProjectExporter with properties:
ProjectGenerator: [1×1 TwinCAT.ModuleGenerator.ProjectGenerator]
Configuration: [1×1 TwinCAT.ModuleGenerator.ProjectExportConfig]
Project: [1×1 TwinCAT.ModuleGenerator.Project]
State: [1×1 struct]
ClassExporters: {[1×1 TwinCAT.ModuleGenerator.Simulink.ModelExporter]}
AdditionalExports: [1×1 containers.Map]
```

**MATLAB® 中的示例代码**

通过以下命令打开相应的示例：`TwinCAT.ModuleGenerator.Samples.Start('Callbacks')`

# 4.8 模块在 TwinCAT 中的应用

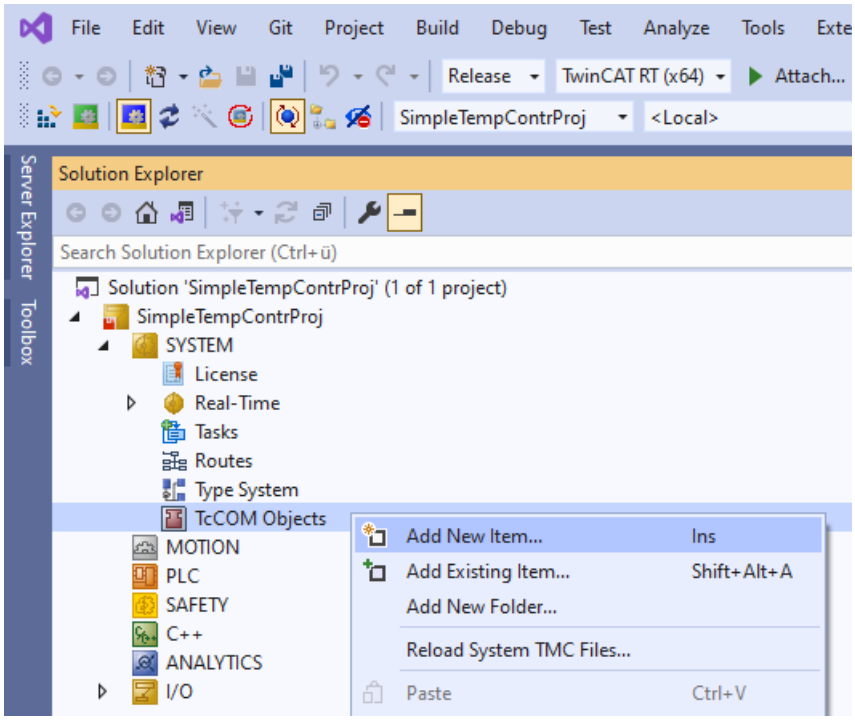
使用 Target for Simulink® 创建的 TcCOM 和功能块可在 TwinCAT XAE 中无缝使用。在任何 TwinCAT XAE 系统上使用的唯一要求是需使用 TwinCAT XAE 3.1.4024.7 及更高版本。**无**需安装 MATLAB®、完整版 Visual Studio 等，因为您可以使用已经为 TwinCAT 编译好的对象和描述文件。

只需将 Engineering Repository 文件夹复制到所选的工程系统中即可。另请参见 TwinCAT 对象 [▶ 40]和 共享已创建的 TwinCAT objects [▶ 57]。手动复制时始终保持文件夹结构：
*%TwinCATInstallDir% \3.1\Repository\<TE140x Module Vendor>\<ModelName>\<Version>\.*

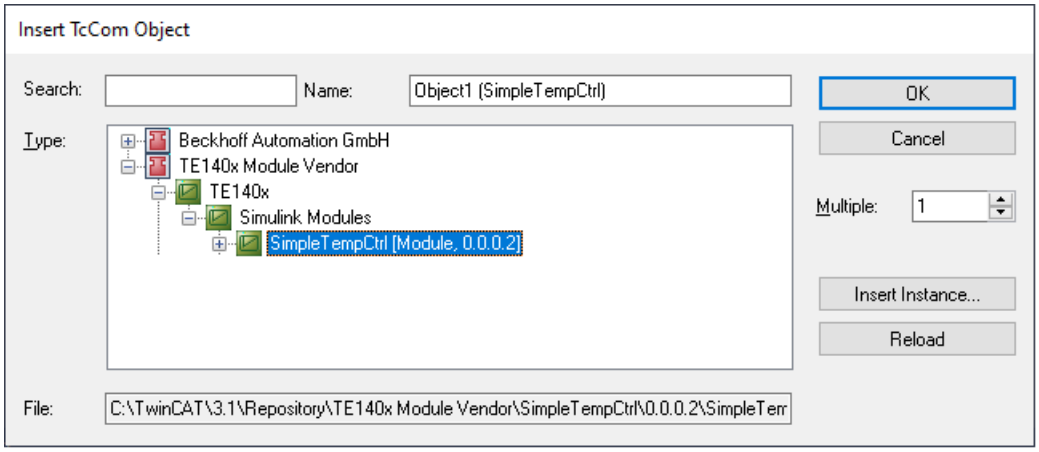
## 4.8.1 使用 TcCOM 模块

**在 TwinCAT 中添加 TcCOM**

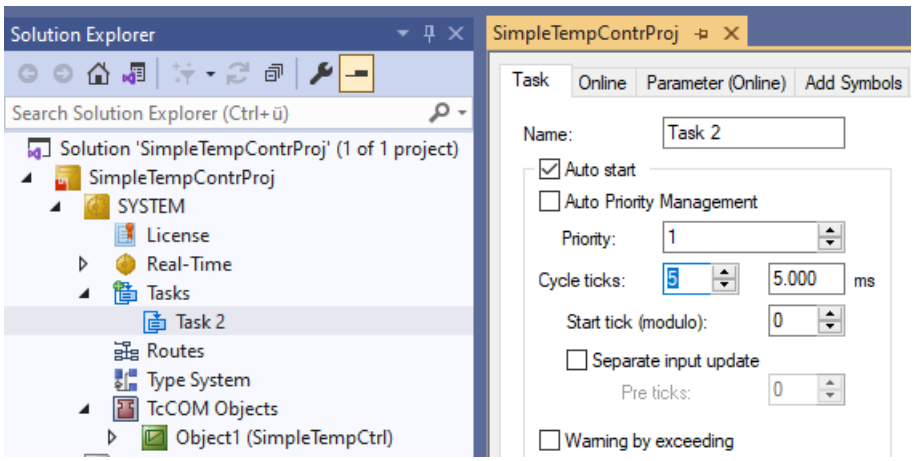
1. 打开 TwinCAT（TwinCAT XAE 或 Visual Studio 环境中的 TwinCAT）。
2. 实例化一个新的 TcCOM 对象。




3. 选择所需的对象。

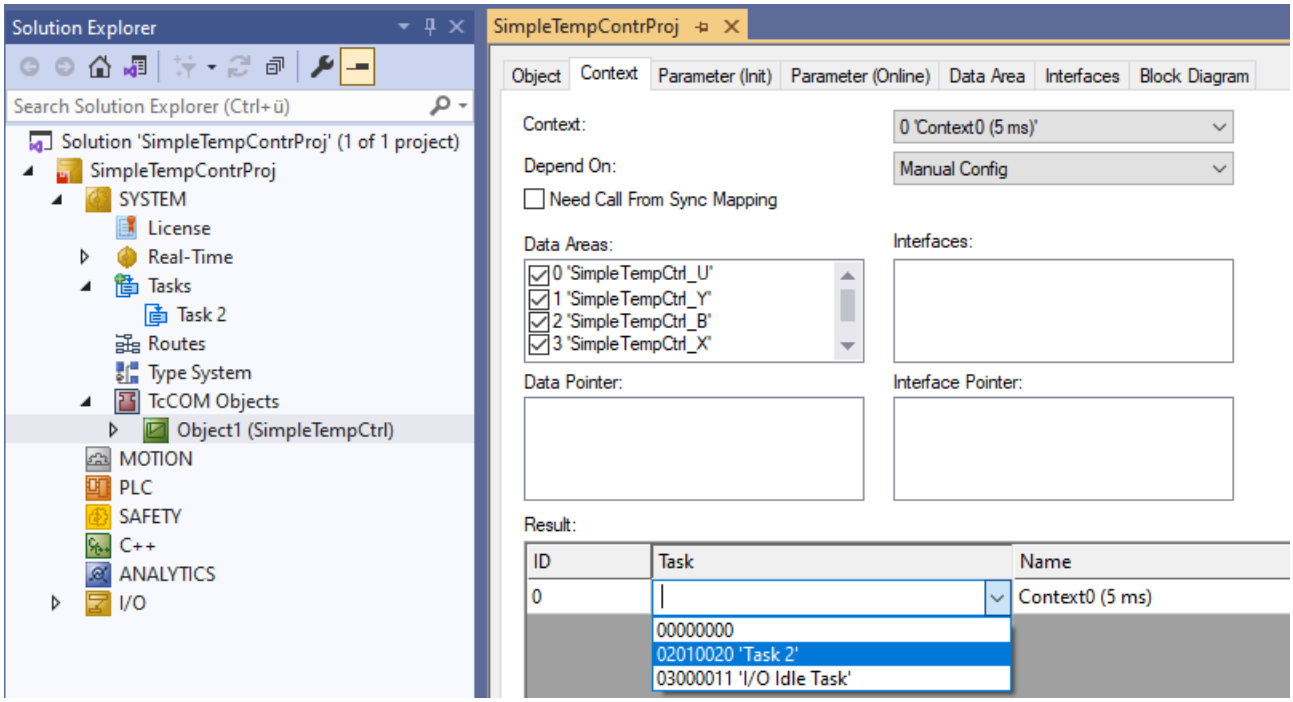



4. 创建一个周期性任务。

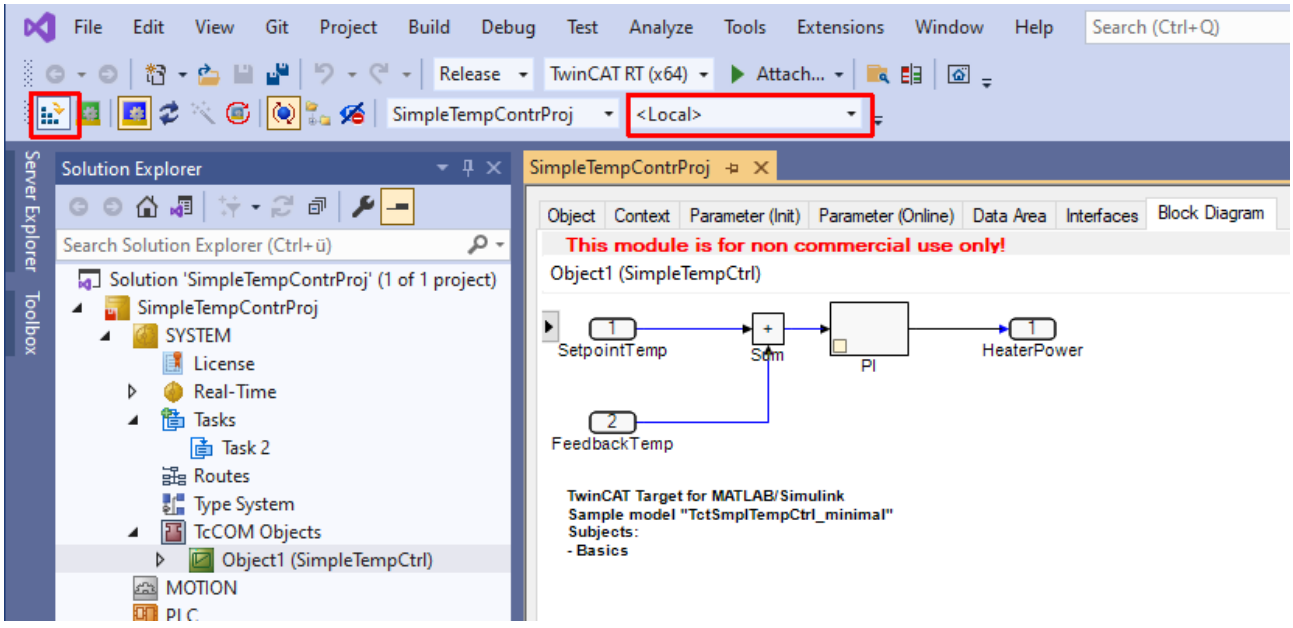



5. 将创建的任务分配给 TcCOM 实例。

请注意，任务的周期时间和 Simulink® 中的 SampleTime（此处为 5 ms）相符。

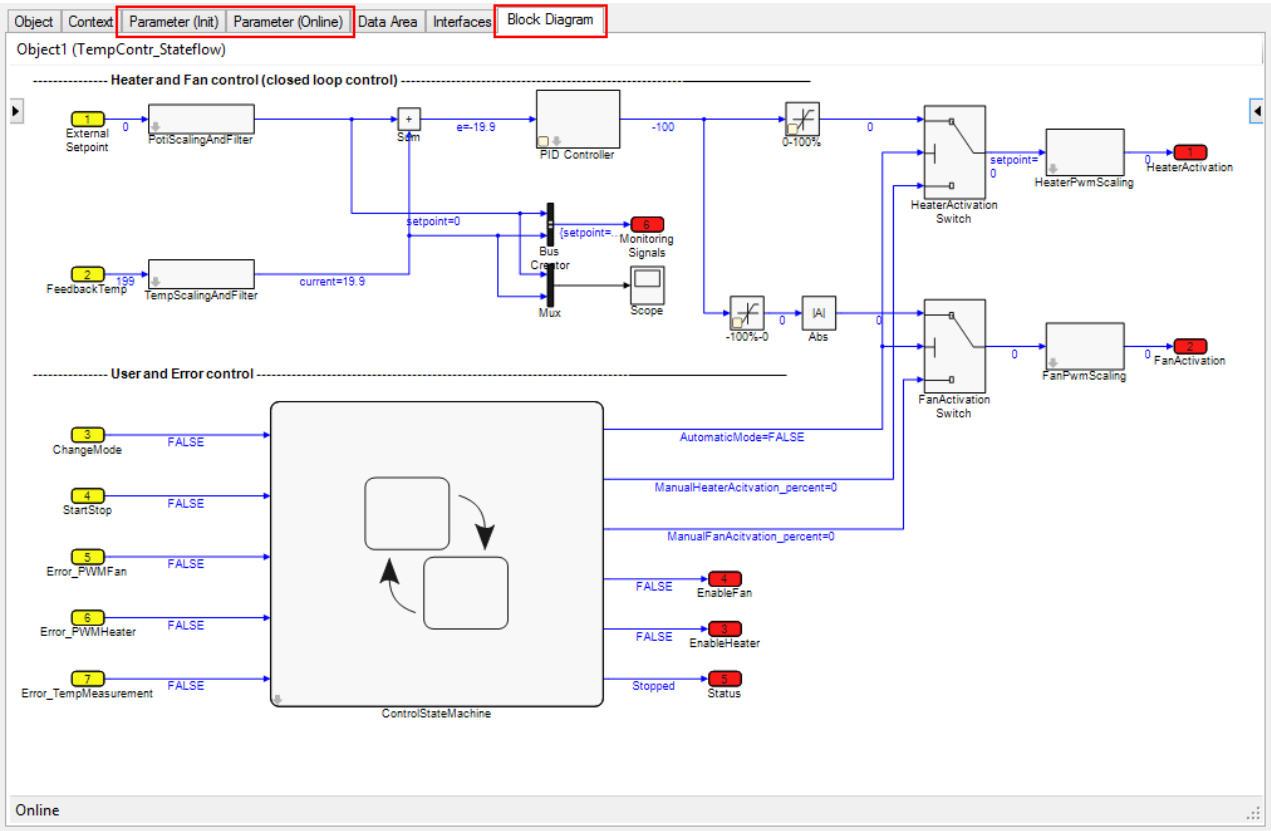



6. 激活配置。



### 4.8.1.1 模块实例的参数设置

**如果是一个 TcCOM 实例，在哪些选项卡中可以找到参数?**




可以在“参数（初始值）”（Parameter (Init)）、“参数（在线）”（Parameter (Online)）和“框图”（Block Diagram）中查看和更改参数。在 Data Area 中，可以看到已创建的 DataArea 及其内容（参数名称和数据类型），但不能通过 XAE 手动更改这里的任何值。

另请参见最佳实践：访问 TcCOM 数据 [▶ 71]了解 TcCOM 上的数据访问方法。

#### 默认值、启动值、在线值和预备值

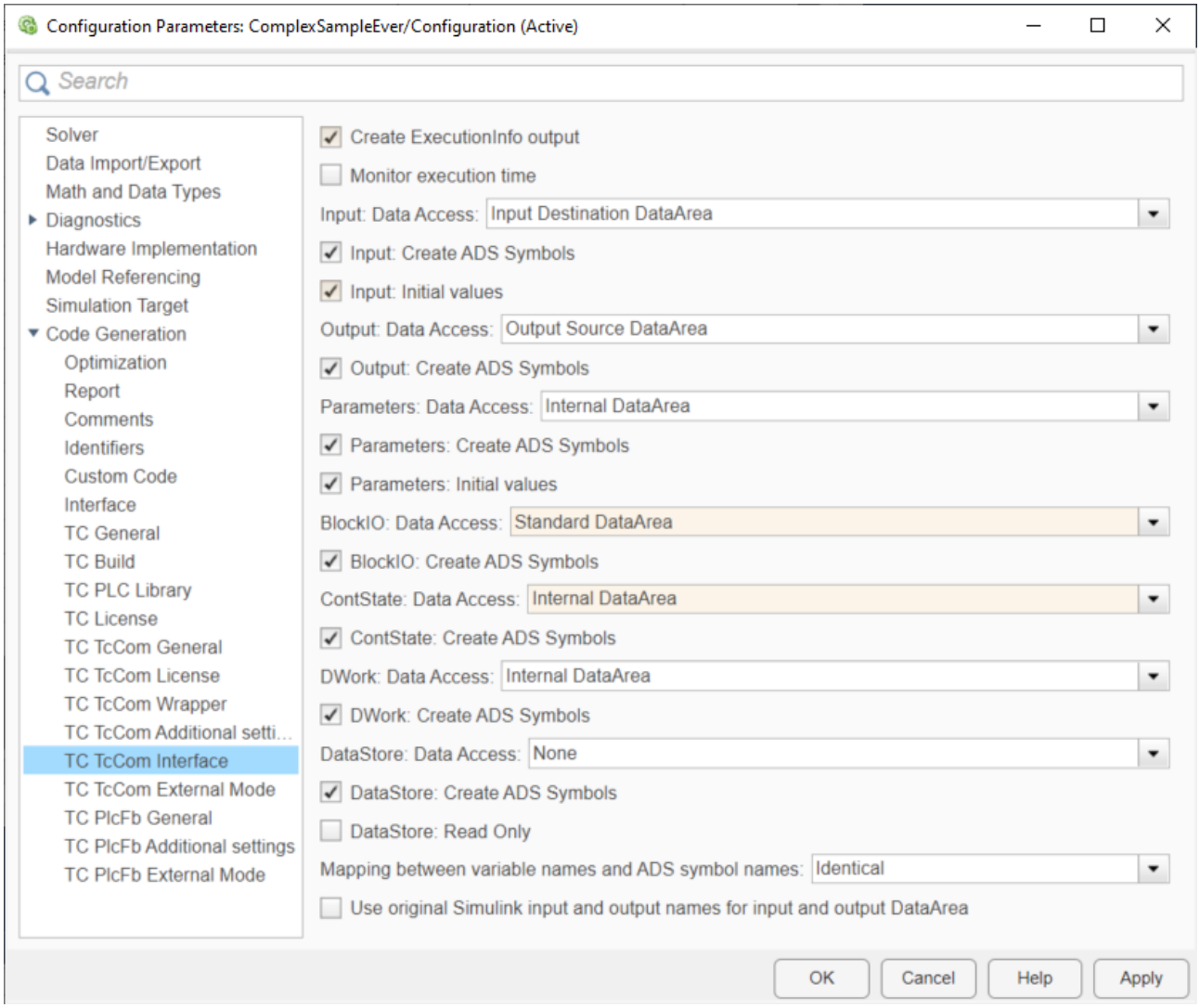
参数可以有不同的状态/特性。定义如下：

- **默认值**是代码生成过程中的参数值（在 Simulink® 中设置）。它们不可更改地存储在模块描述文件（*.tmc）中，即模块类描述。
- **启动值**存储在 TwinCAT 项目文件中，并写入模块实例。启动值在目标系统上有相应的位置，并定义模块实例启动时的值。
- 只有在目标系统上启动了 TcCOM 实例，才能使用**在线值**。它们显示运行模块中的当前参数值。还可以在运行时期间更改该值。
- 只要可以使用在线值，就可以指定**预备值**。因此可以同时更改多个参数值并将其写入模块。

#### 模块实例参数化

#### TC TcCom 接口设置示例

为了解释模块实例的参数化，假定模块具有以下属性。




**基于这些设置的 TcCOM 属性：**

- 模型参数 <ModelName>_P 的结构被创建为模块参数，因为“参数：初始值”（Parameters: Initial Values）已启用。如果代码接口封装未设置为*可复用函数*（Reusable function），则可以访问作为 DataArea 的模型参数。

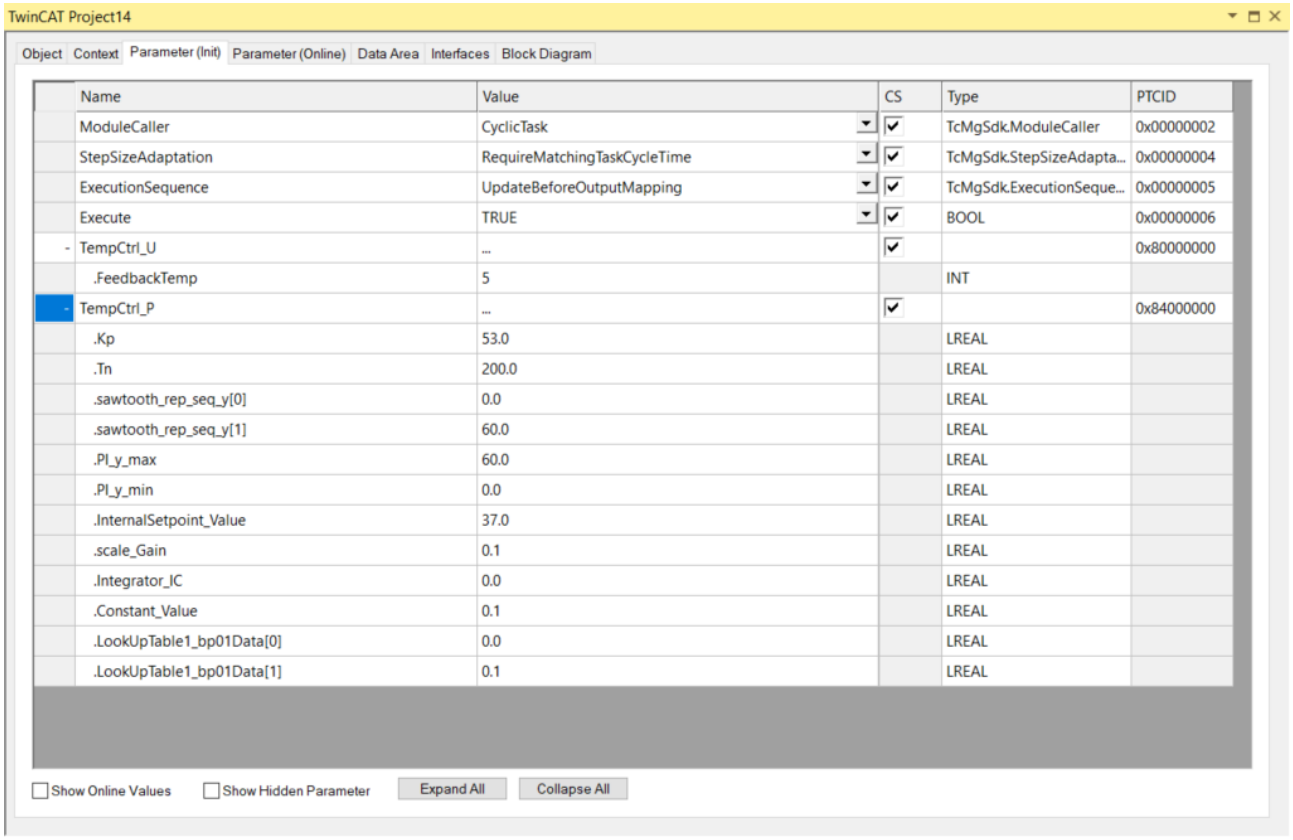
> **将模型参数输入为可调参数**
>
> Simulink Coder™ 模型参数默认设置为“内联”。这意味着使用的对象内存更小，生成的代码在运行时也得到了优化，但通常只有少数几个参数可以在运行时通过此设置进行更改。
> 使用*优化 > 默认参数行为*（Optimization > Default parameter behavior）下的*配置参数*（Configuration Parameter）“可调”（Tunable），，，这样就可以在运行时对模型进行参数化。

- BlockIO 作为标准 DataArea 创建，因此在 TcCOM 的流程映像中可见，并可通过 DataPointer 链接。
- 启用“创建 ExecutionInfo 输出”（Create ExecutionInfo output），因此有另一个输出源类型 DataArea，其中包含实例的执行信息。
- 在所有 DataArea 启用“创建 ADS 符号”（Create ADS Symbols），因此可以通过 ADS 符号名称访问 DataArea 中的所有参数。
- 由于启用了“输入：初始值”（Input: Initial Values），因此为模型输入创建模块参数。

有关设置选项的说明，请参见配置对 TcCOM 对象数据的访问权限 [▶ 67]。

**参数（初始值）中可能的参数设置**

模块参数可在“参数（初始值）”（Parameter (Init)）中找到，这些参数不会发生周期性变化，但会发生非周期性变化。在 Config 模式下或未启动 TcCOM 时，不显示在线值。



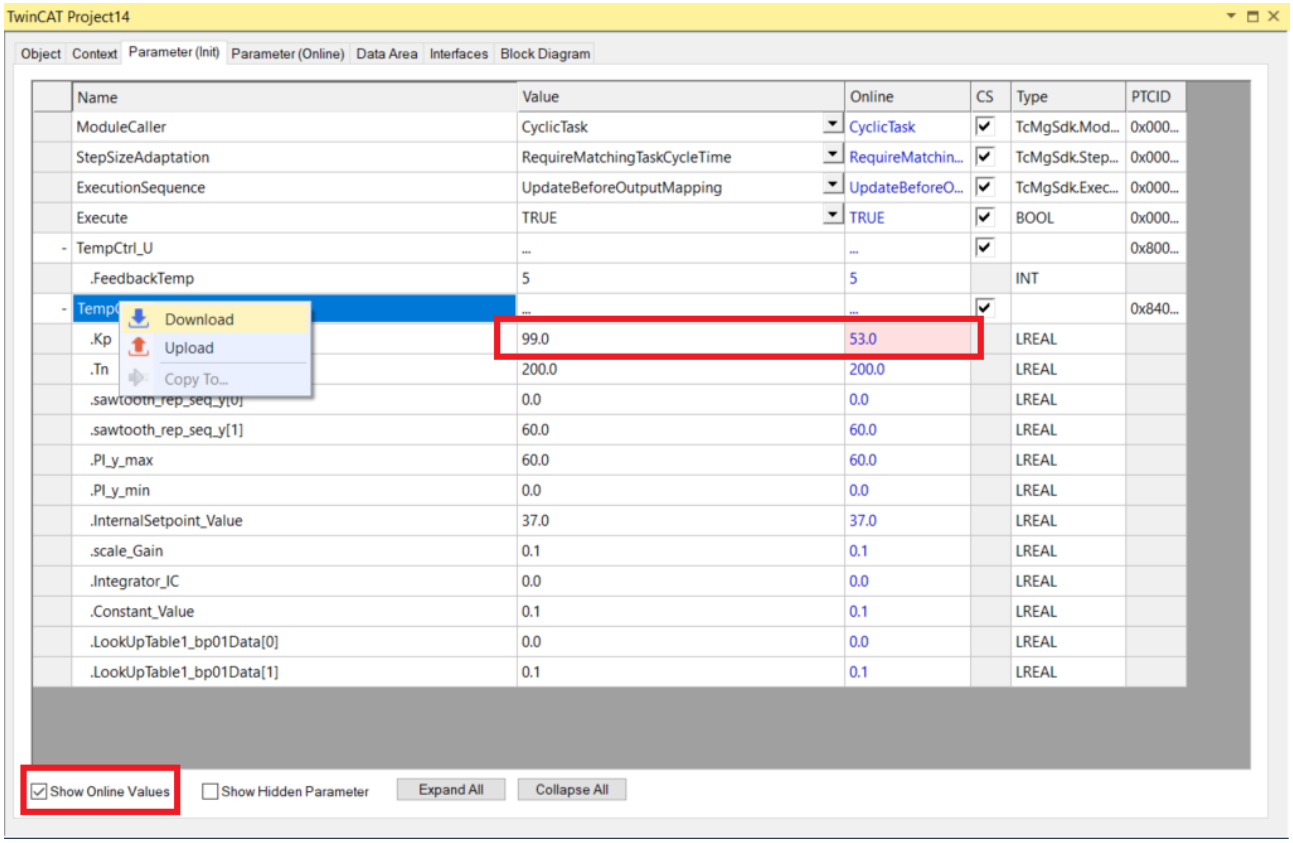

### 定义启动值

启动值可在“参数（初始值）”（**Parameter (Init)**）或“框图”（**Block Diagram**）选项卡中设置（参见以下章节）。为此，请在模块实例的“值”（**Value**）列中选择要调整的条目并输入数值。以这种方式设置的值目前只能在 XAE 的项目文件中使用。“激活配置”（Activate Configuration）可将设定值（连同整个项目）加载到目标系统。

*ModuleCaller*、*StepSizeAdaption* 或 *ExecutionSequence* 等设置不能在模块运行时更改，只能定义为启动值。Execute 等其它模块参数或模型参数也可在线更改。

### 在线更改数值

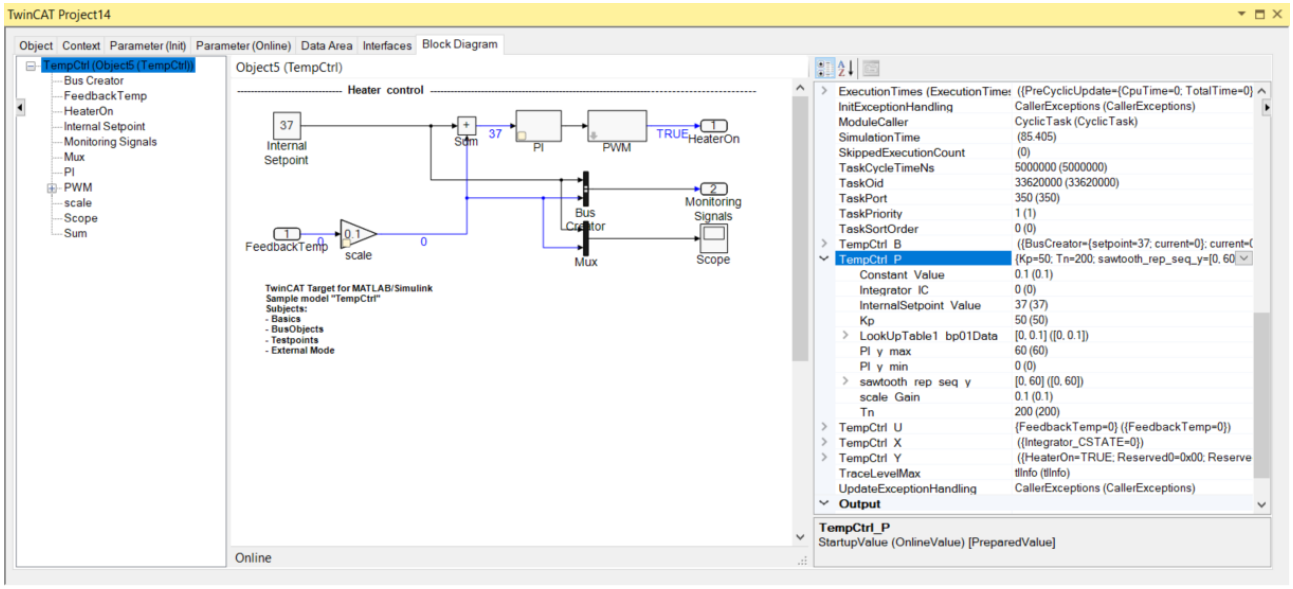
如果 TcCOM 模块已启用，则可以使用**显示在线值**（Show Online Values）来显示实例的当前值。可在**数值**（Value）列中输入一个新值，使其成为预备值。如果目标系统尚未加载预备值，“数值”（Value）与“在线”（Online）之间有偏差的字段会显示红色标记。完成所有更改后，可以右键单击结构层，使用“下载”（Download）功能在目标系统中设置预备值。请注意，这不会更改目标系统上实例的启动值。为此，必须首先激活目标系统上的当前配置。同样，也可以使用“上传”（Upload）将当前在线值设置为项目中的启动值。同样，在编译和下载项目之前，运行时系统中的更改不会生效，请参见最佳实践：访问 TcCOM 数据 [▶ 71]。



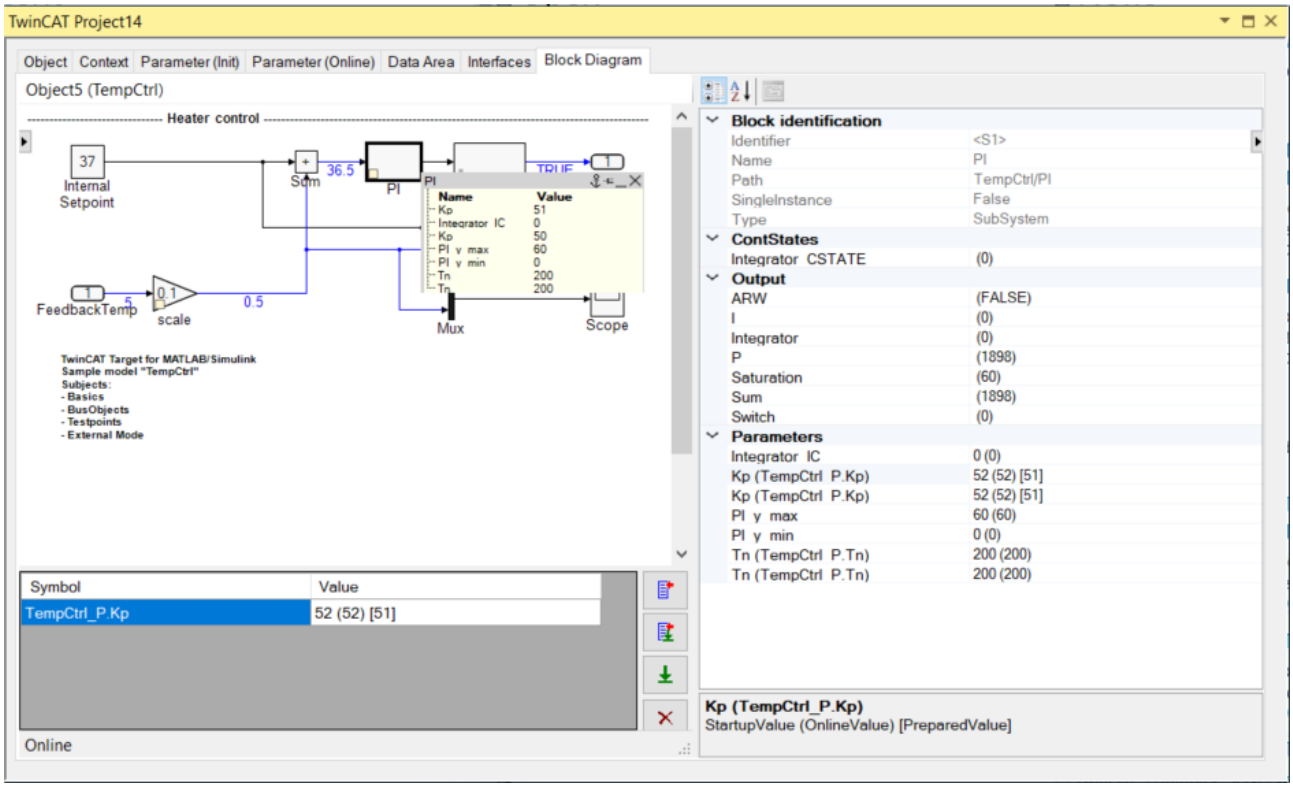

**可在框图中进行的参数设置**

仅当处于框图顶层（“<root>”）时，TwinCAT 3 框图中才会显示参数范围（窗口右侧）内的所有参数。如果是在子系统中或已经选择一个功能块，则只显示活动功能块的参数。

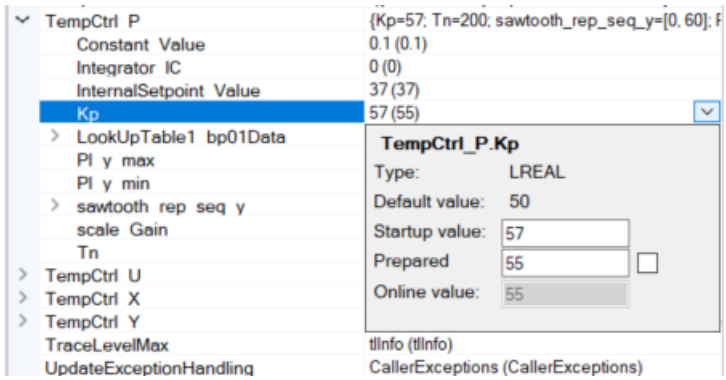
在框图中，可以读取和写入在线值以及更改启动值。



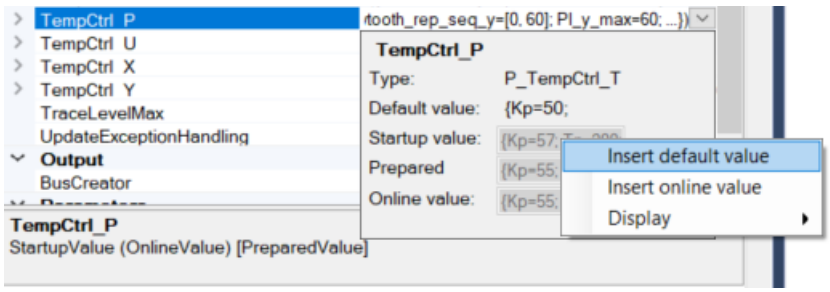

选择左下角有黄点的图块。直接单击黄点会打开一个上下文菜单，可在其中看到该图块的在线值，也可以对其进行更改。如果更改了某个值，该值将被输入到预备列表中。完成所有更改后，可将预备值写入目标系统。可以在预备列表中选择要将预备值设置为在线值还是启动值。




也可以进入框图右侧区域的参数列表，并在那里更改数值。找到要更改的参数，并单击列表右侧的向下箭头。在这里，可对可编辑字段进行更改。如果将鼠标指针移至参数名称（下图中为 TempCtrl_P.Kp）上，则会显示 ADS 地址信息。右键单击参数名称，可将参数的 ADS 地址信息复制到剪贴板。



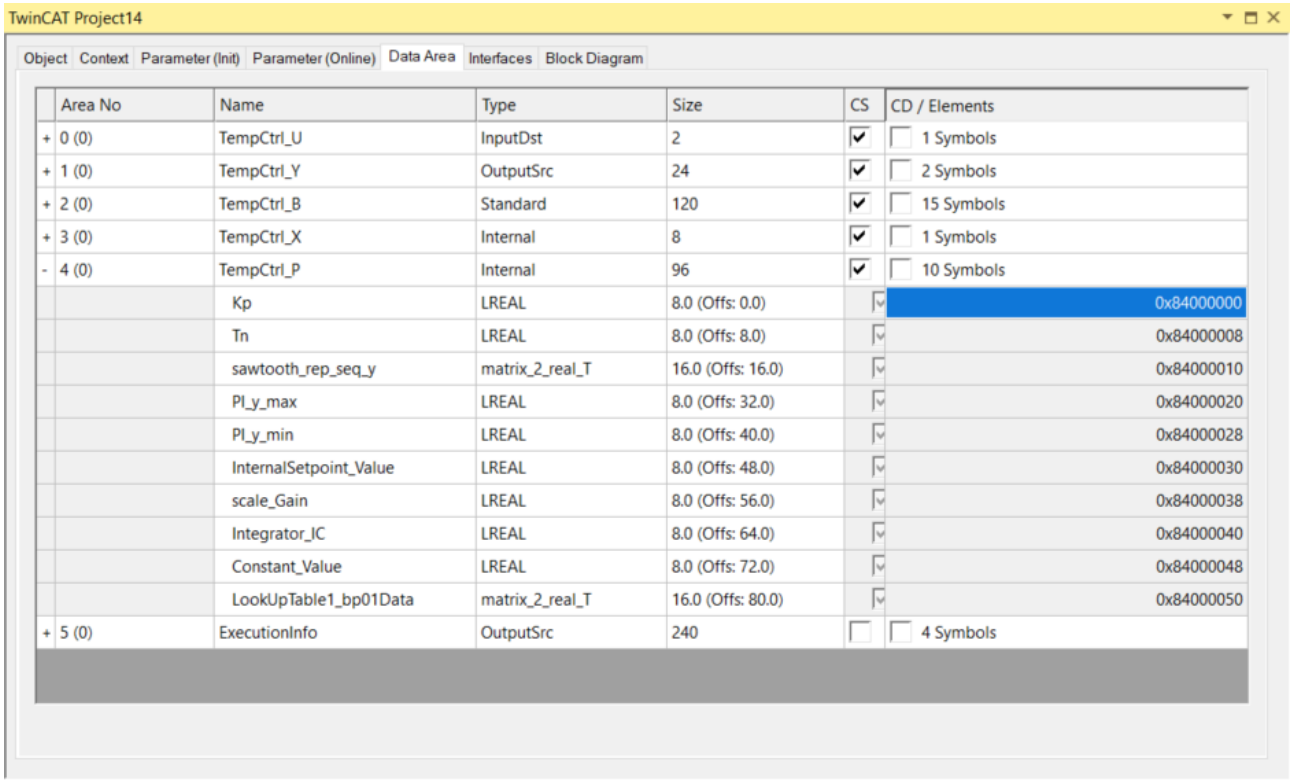

还可以更改整个结构。例如，选择 <ModelName>_P 结构，即包含所有模型参数的结构，并在此处选择向下箭头。例如，右键单击**启动值**（Startup Values）会出现一个上下文菜单。可在此将结构的所有当前在线值设置为启动值，或者也可以将启动列表重置为默认值。




### 与 DataArea 交互

无法通过 XAE 更改 DataArea。只能看到 DataArea 类型（请参见类型列），并推断出 DataArea 可以使用哪些交互选项，请参见配置对 TcCOM 对象数据的访问权限 [▶ 67]。

通过 CS 列可以查看是否为该 DataArea 生成 ADS 符号名称。用户可以编辑 CS。如果启用 CS，则可以使用目标浏览器，例如用于搜索 ADS 符号并将其包含在作用域配置中。如果禁用 CS，则只能通过索引组（Index Group）和索引偏移（Index Offset）来访问 DataArea 的数据。索引组是实例的对象 ID（请参见 "对象 "选项卡），索引偏移显示在 "CD/元素 "列中。




| | Area No | Name | Type | Size | CS | CD / Elements |
|---|---|---|---|---|---|---|
| + | 0 (0) | TempCtrl_U | InputDst | 2 | ☑ | ☐ 1 Symbols |
| + | 1 (0) | TempCtrl_Y | OutputSrc | 24 | ☑ | ☐ 2 Symbols |
| + | 2 (0) | TempCtrl_B | Standard | 120 | ☑ | ☐ 15 Symbols |
| + | 3 (0) | TempCtrl_X | Internal | 8 | ☑ | ☐ 1 Symbols |
| - | 4 (0) | TempCtrl_P | Internal | 96 | ☑ | ☐ 10 Symbols |
| | | Kp | LREAL | 8.0 (Offs: 0.0) | | 0x84000000 |
| | | Tn | LREAL | 8.0 (Offs: 8.0) | | 0x84000008 |
| | | sawtooth_rep_seq_y | matrix_2_real_T | 16.0 (Offs: 16.0) | | 0x84000010 |
| | | PI_y_max | LREAL | 8.0 (Offs: 32.0) | | 0x84000020 |
| | | PI_y_min | LREAL | 8.0 (Offs: 40.0) | | 0x84000028 |
| | | InternalSetpoint_Value | LREAL | 8.0 (Offs: 48.0) | | 0x84000030 |
| | | scale_Gain | LREAL | 8.0 (Offs: 56.0) | | 0x84000038 |
| | | Integrator_IC | LREAL | 8.0 (Offs: 64.0) | | 0x84000040 |
| | | Constant_Value | LREAL | 8.0 (Offs: 72.0) | | 0x84000048 |
| | | LookUpTable1_bp01Data | matrix_2_real_T | 16.0 (Offs: 80.0) | | 0x84000050 |
| + | 5 (0) | ExecutionInfo | OutputSrc | 240 | ☐ | ☐ 4 Symbols |

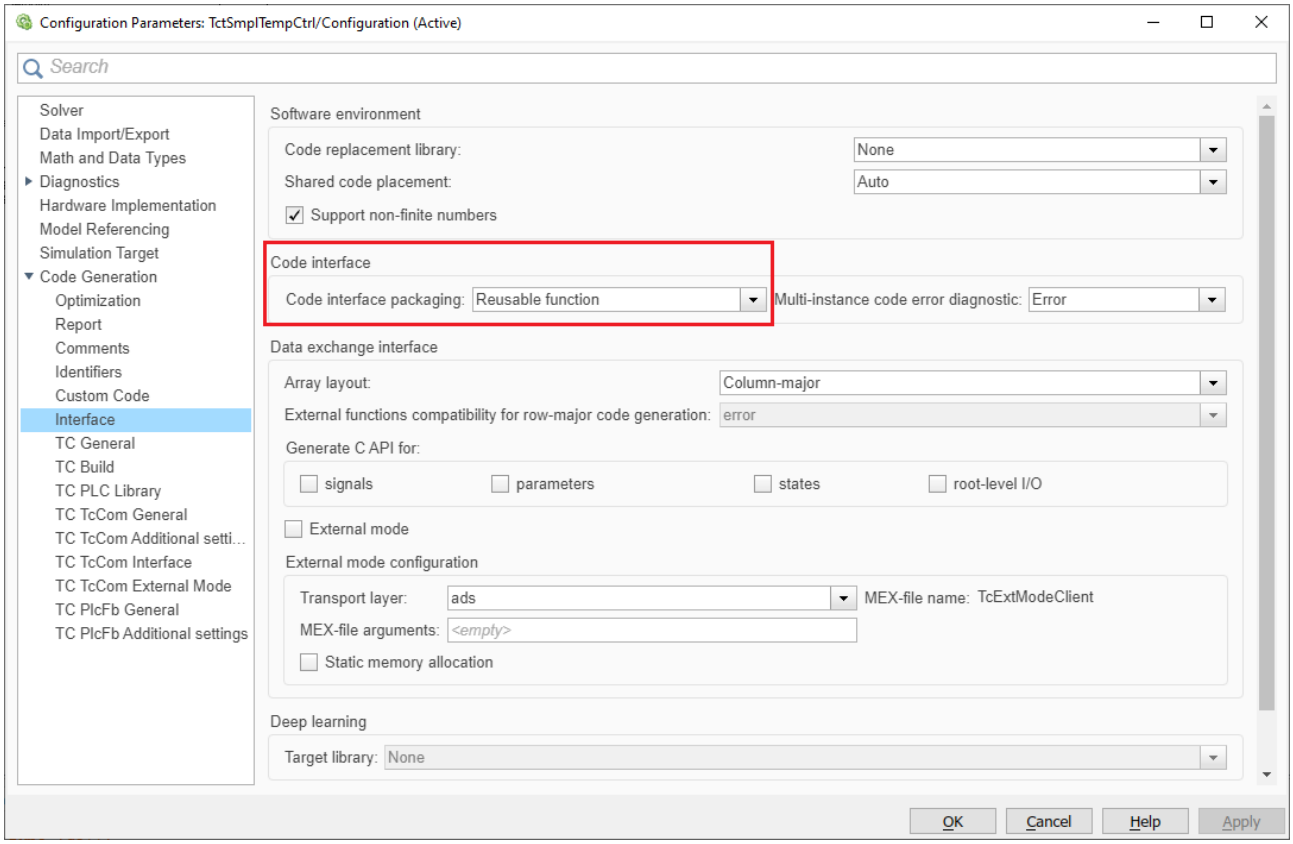
有关与 DataAreas 交互的更多信息，请参见配置对 TcCOM 对象数据的访问权限 [▶ 67]和最佳实践：访问 TcCOM 数据 [▶ 71]。

### 4.8.1.2 多个模块实例的参数设置

在上一节中已经介绍过，TcCOM 的实例可以在 TwinCAT 中进行参数化，甚至可以偏离 Simulink® 中的参数。

如果在 TwinCAT 解决方案中使用了多个 TcCOM 实例，则在实例的单独参数化方面存在不同的选项。如要实现以下三个选项，必须在 Simulink® 中使用**代码接口封装**（Code interface packaging）设置。

确保将*默认参数行为*（Default parameter behavior）设置为*优化*（Optimization）下的*可调*（Tunable）。




✓ **所有实例应具有相同的参数。**

1. 将参数设置为“可复用函数”（Reusable function）。这是选择目标 *TwinCatGrt.tlc* 时的默认值。
2. 在 TwinCAT 中创建多个 TcCOM 实例。
3. 在参数（初始化）下，将 <ModelName>_P_Sharing 参数配置为*定义*（define）或*继承*（inherit）。定义指定了用*继承*（inherit）配置的所有从属实例的参数化。

⇨ 只能配置一个带有*定义*（define）的实例。

✓ **应该可以单独为每个实例设置参数。**

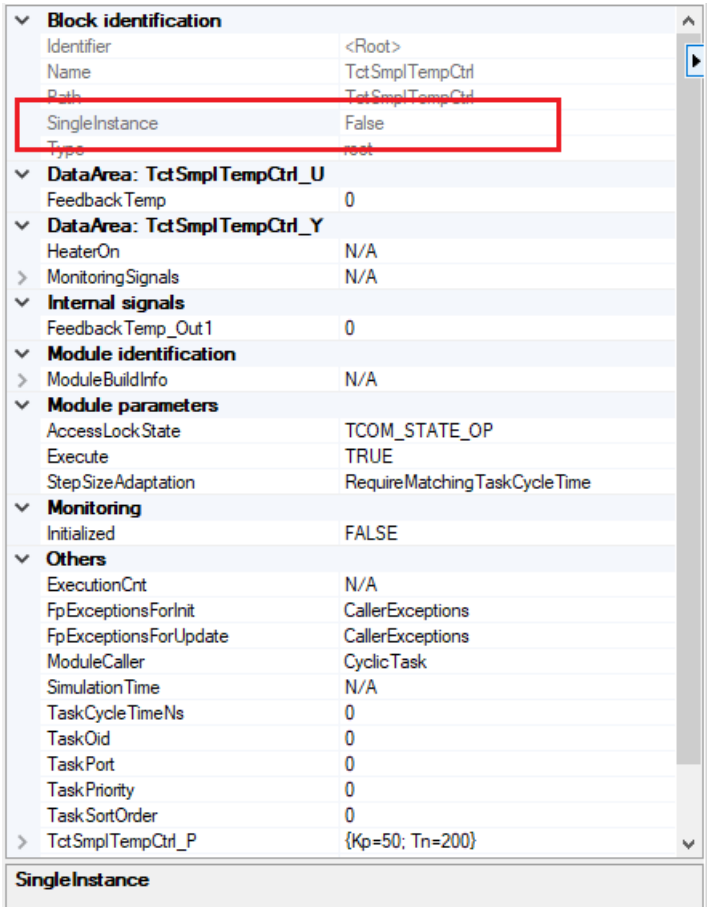
1. 将参数设置为“C++ 类”。
2. 在 TwinCAT 中创建多个 TcCOM 实例。

⇨ 未生成 <ModelName>_P_Sharing 参数。每个实例都可以单独设置参数。

✓ **项目中只能有一个实例。**

1. 将参数设置为“不可复用函数”（Nonreusable function）。
   ⇨ 如果您在 TwinCAT 中创建了多个 TcCOM 实例，则在激活解决方案时会收到一条错误信息。
   ⇨ 一个 TcCOM 实例是否可以多次实例化，可以从 TC3 BlockDiagram 中看出。
2. 为此，请转到右侧的参数范围。在“功能块标识”（Block Identification）下，可以看到参数“SingleInstance”。

⇨ “False”值表示可多次实例化。因此，“True”表示一个实例。




**设置也适用于 PLC 功能块的使用**

*代码接口封装（Code Interface Packaging）*的设置对于使用 TcCOM 和使用 PLC 功能块具有相同的含义。

**打开参数设置示例**

在 MATLAB® 中，打开多实例示例：`TwinCAT.ModuleGenerator.Samples.Start("Create Multiple Instances of the Same TcCOM Module")`

### 4.8.1.3 在 TwinCAT 中使用框图

#### 4.8.1.3.1 Simulink®-TcCOM

如果使用 TwinCAT Target for Simulink® 创建了 TwinCAT object，并在过程中导出了框图，则 Simulink® 模型的框图可以作为控件显示在 TwinCAT XAE 中。

4.8.使用框图

1.3.

1.1

在从 MATLAB® 或 Simulink® 生成 TcCOM 模块时，可对框图导出进行配置。如果已启用导出功能，则可在 TwinCAT 开发环境中模块实例的“框图”（Block Diagram）选项卡下找到框图。

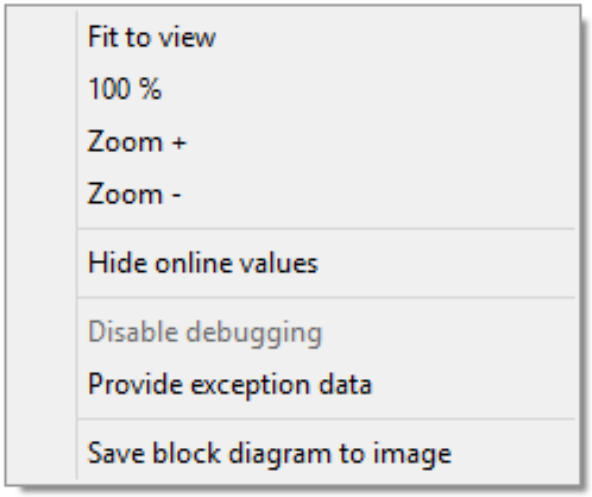
通过快捷键、拖放功能和上下文菜单，可以浏览 TcCOM 模块的层次结构，查看参数值，显示信号值并获取可选的附加调试信息。

**快捷键功能：**

| 快捷键 | 功能 |
|---|---|
| Space | 缩放至框图选项卡的当前大小 |

| 快捷键 | 功能 |
|---|---|
| Backspace | 切换到下一个更高的层级 |
| ESC | 切换到下一个更高的层级 |
| CTRL + “+” | 放大 |
| CTRL + “-” | 缩小 |
| F5 | 连接调试器<br>（必须激活 *系统 - > 实时 - > C++ 调试器 - > 启用 C++ 调试器*（System- > Real-Time -> C++ Debugger -> Enable C++ Debugger）） |

**上下文菜单功能：**

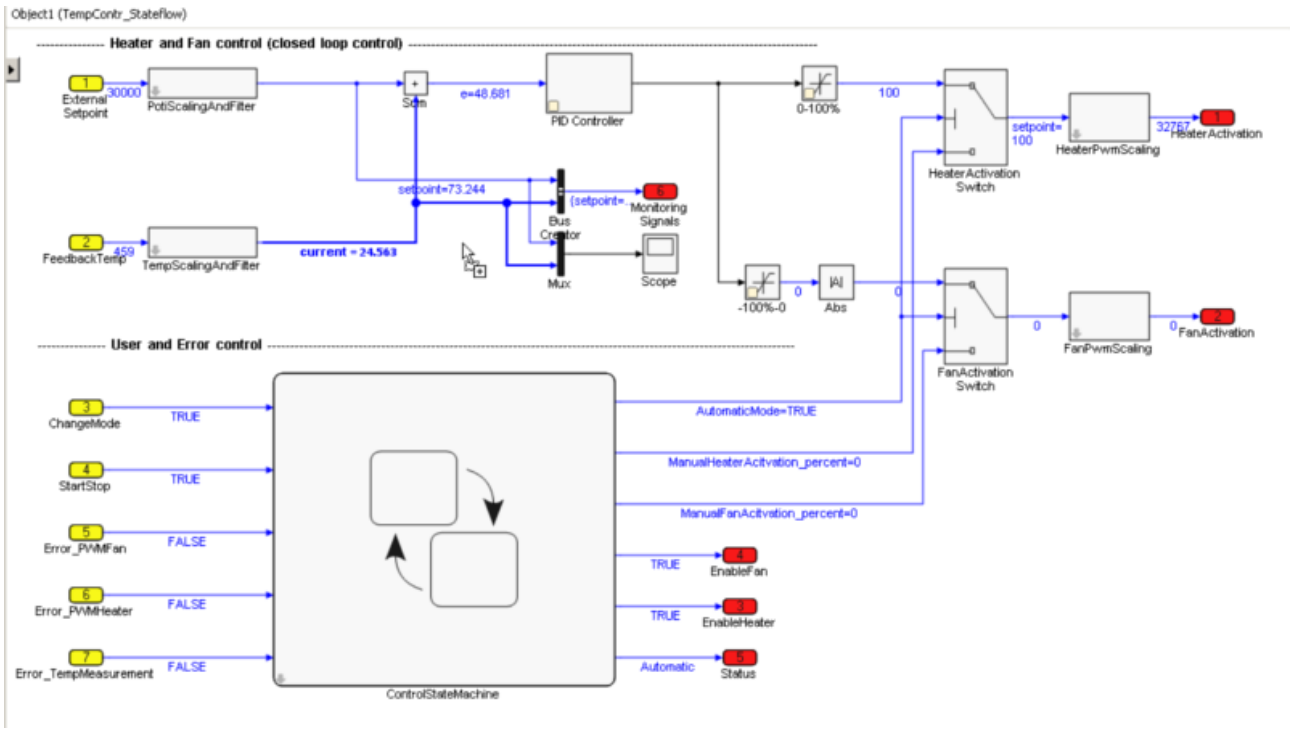



4.8.显示信号曲线
1.3.
1.2

显示信号曲线通常有助于验证和故障排除。框图提供以下选项：

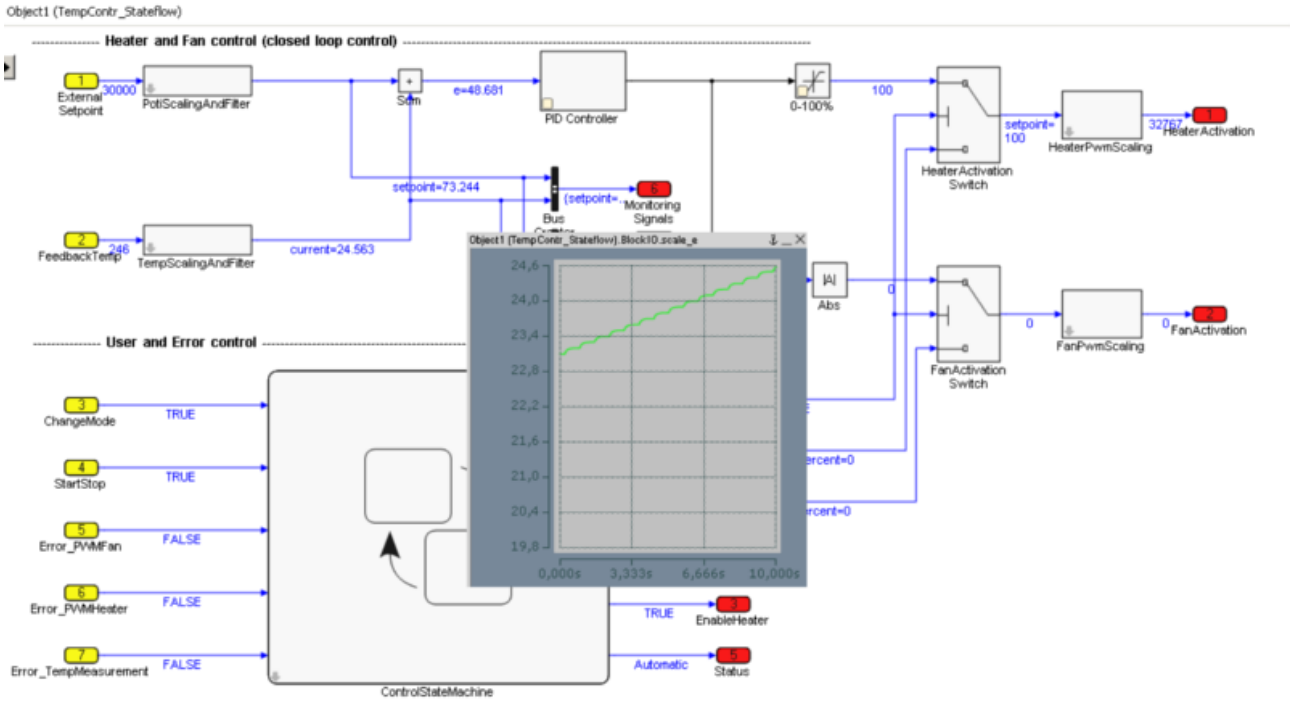
**在框图中显示信号曲线**

框图中提供在窗口中显示信号曲线的选项。为此，可将信号或图块拖放到框图的空闲区域。




在框图中创建作用域

下拉后，框图中将打开一个作用域窗口。




在框图中显示作用域

作用域窗口的标题栏提供以下选项：

| | |
|---|---|
| ✖ | 关闭窗口 |
| ⚓ | 将窗口保持在所有框图层次结构的最前面 |
| — | 窗口最小化到标题栏 |

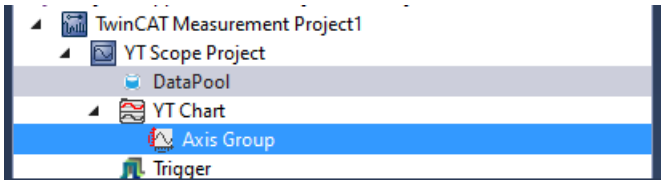
在框图控件 [▶ 165]中显示作用域需要Scope View Professional（TE1300）许可证。TwinCAT XAE 中不需要 Scope View Professional 许可证。

在 Simulink® 总线框图中创建作用域窗口时，总线的所有信号都会直接显示在作用域窗口中。

框图中的作用域窗口可用于快速概览。如要进行更详细的分析，建议在 TwinCAT Measurement 项目中分析信号。

### 在 TwinCAT 3 Scope 中显示信号曲线

如果不是在框图控件中，而是在 TwinCAT Measurement 项目的轴组中下拉，则信号会添加到该轴组中。




在 TwinCAT 3 Scope 中添加信号

4.8.框图中的模块参数设置

1.3.

1.3

如要对 TcCOM 实例进行参数设置，可直接在框图中使用**参数窗口**。此外，还可以使用“属性表”，该表可在框图右侧边展开或折叠。不同的参数值之间存在基本区别：

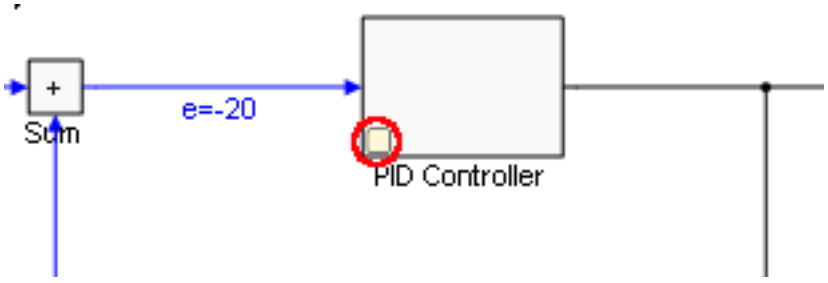
### “默认”、“启动”、“在线”和“预备”

在框图属性表的下拉菜单中可以找到以下类型的值：

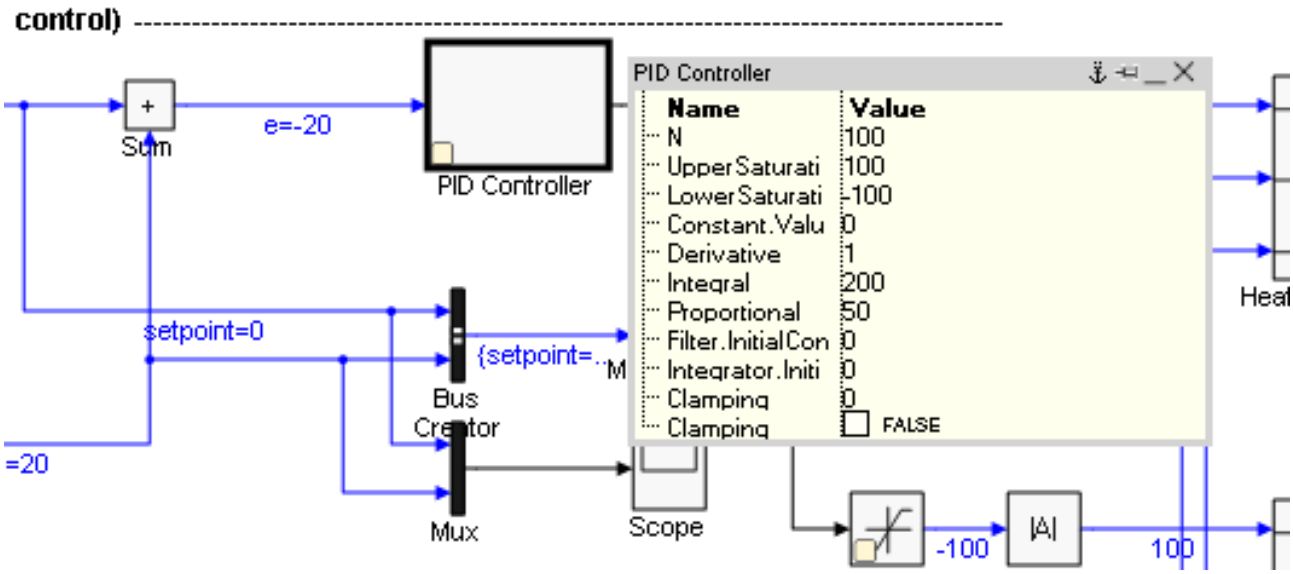
- **默认值**是代码生成过程中的参数值。它们总是存储在模块描述文件中，能够在参数更改后恢复生产设置。
- TwinCAT 启动模块实例后，**启动值**储存在 TwinCAT 项目文件中并下载到模块实例中。
  输入过程映像的启动值也可以在 Simulink® 模块中指定。这样，模块就可以在输入值不为零的情况下启动，无需将输入值链接至其他过程映像。内部信号和输出信号没有起始值，因为它们始终会在第一个周期内被覆盖。
- 只有当模块在目标系统上启动时，才能使用**在线值**。它们显示运行模块中的当前参数值。还可以在运行时期间更改该值。不过在这种情况下，必须通过上下文菜单启用相应的输入字段，以防止意外输入。
- 只要可以使用在线值，就可以指定**预备值**。它们可用于保存各种值，从而将其统一写入模块。如果已指定预备值，这些值将显示在框图下方的表格中。使用列表右侧的按钮可将预备值下载为在线值并/或保存为起始值，或将其删除。

### 框图中的参数设置

可参数化的图块在框图中用黄色方框标出。




双击图块或单击黄色方框，会弹出一个窗口，其中显示可更改的参数。




如果更改了某个值，可以通过以下键盘命令来应用：

| | |
|---|---|
| CTRL + Enter | 直接设置在线值 |
| SHIFT + Enter | 设置启动值 |
| Enter | 设置预备值 |

标题栏中的图标具有以下功能：

| | |
|---|---|
|  | 关闭窗口 |
|  | 将窗口保持在所有框图层级的最前面 |
|  | 在当前框图层级保持窗口打开状态 |
|  | 窗口最小化到标题栏 |

4.8.调试
1.3.
1.4

有多种方法可用于查找使用 MATLAB®/Simulink® 创建的 TcCOM 模块中的错误，或分析 TwinCAT 项目整体架构中的模块行为。

**框图中的调试**

如果在生成 TcCOM 模块时导出了框图，则框图可以显示在 TwinCAT 开发环境中，并用于在相应模块实例中进行调试。为此，框图使用 Microsoft Visual Studio 调试器，该调试器可通过 TwinCAT 调试器端口与 TwinCAT Runtime 连接。按照“调试” () 下的 C++ 部分所述连接调试器。

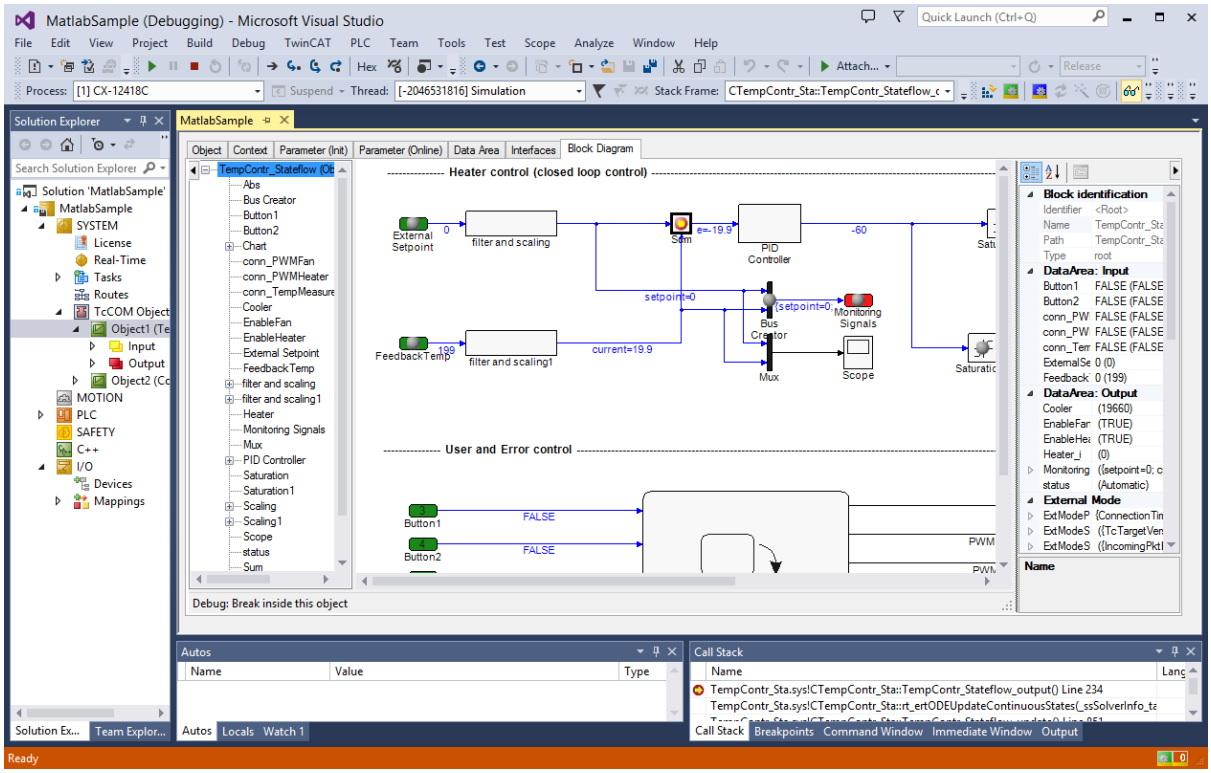
在框图内进行调试的前提条件：

- 工程系统中必须有 TcCOM 模块的 C/C++ 源代码，且 Visual Studio 调试器必须能够找到它。理想情况下，应在生成代码的系统上进行调试。如果模块是在其它系统上创建的，通常可以通过将 Visual Studio 项目集成到 TwinCAT C++ 部分来获取相关的 C/C++ 源代码。文件 <ModelName>.vcxproj 位于构建目录中，请参见
- 该模块必须是使用**调试**（Debug）配置创建的。在代码生成后直接发布时，请在**发布配置**（publish configuration）下的区域选择**调试**（Debug）设置。从 TwinCAT 的 C++ 部分发布模块时，必须启用解决方案 C++ 节点中的调试器；参见 C/C++ 文档中的“调试”章节。

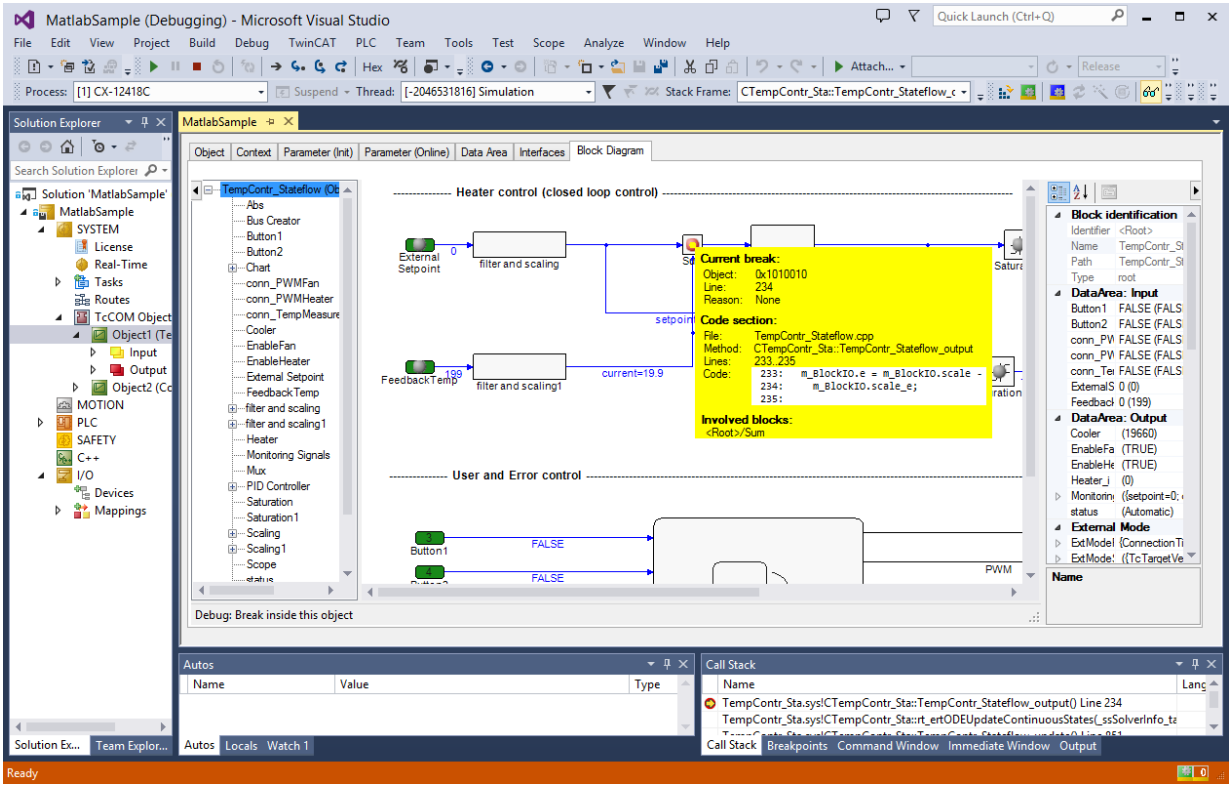
- 在代码生成过程中，必须在 **Tc 高级**（Tc Advanced）下的编码器设置中启用**导出框图**（Export block diagram）和**导出框图调试信息**（Export block diagram debug information）选项。
- 如 中所述，在 TwinCAT 项目中，必须启用调试器端口。

### 在框图中设置断点

1. 将调试器连接到 TwinCAT Runtime 后，可能出现的断点将分配给框图中的图块，并以点的形式表示。单击所需的断点可将其激活，以便在下一次执行相关功能块时使模块实例停止执行。点的颜色提供有关断点当前状态的信息：
   - 灰色：断点未启用
   - 红色：断点已启用。下一次执行该功能块时，程序将停止运行
   - 中间黄点：断点触发。程序在此时停止执行
   - 中间蓝点：断点触发（与黄色相同），但是在模块的不同实例中。



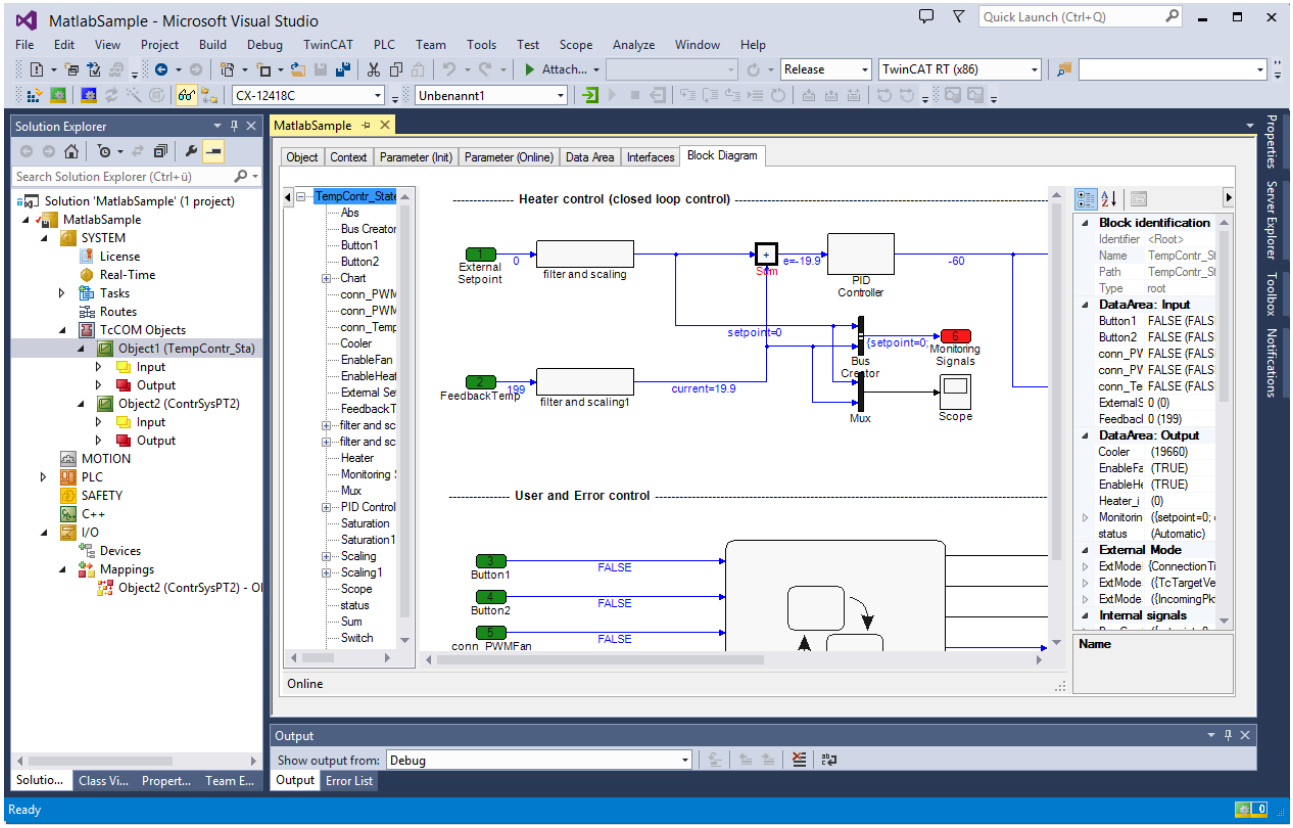
2. 其它信息（例如，相应的 C++ 代码段）可在断点的工具提示中找到：




断点并非总是分配给单个功能块。在许多情况下，多个功能块的功能会被合并在一个代码段，甚至是底层 C++ 代码的一行中。这意味着多个功能块可以共享同一个断点。因此，激活框图中的断点也可能导致其它框图中的点显示发生变化。

**评估例外情况**

如果在 TcCOM 模块的处理过程中出现异常，例如除以零，则可在框图中显示发生异常的点。为此，TcCOM 模块必须满足上述要求，并且必须在 TwinCAT 项目中启用 C++ 调试器（）。连接调试器后（可能在异常发生前，也可能在异常发生后），只要能将导致异常的代码行分配给一个功能块，框图中就会突出显示导致异常的功能块。功能块名称以红色显示，功能块本身以粗体标出。



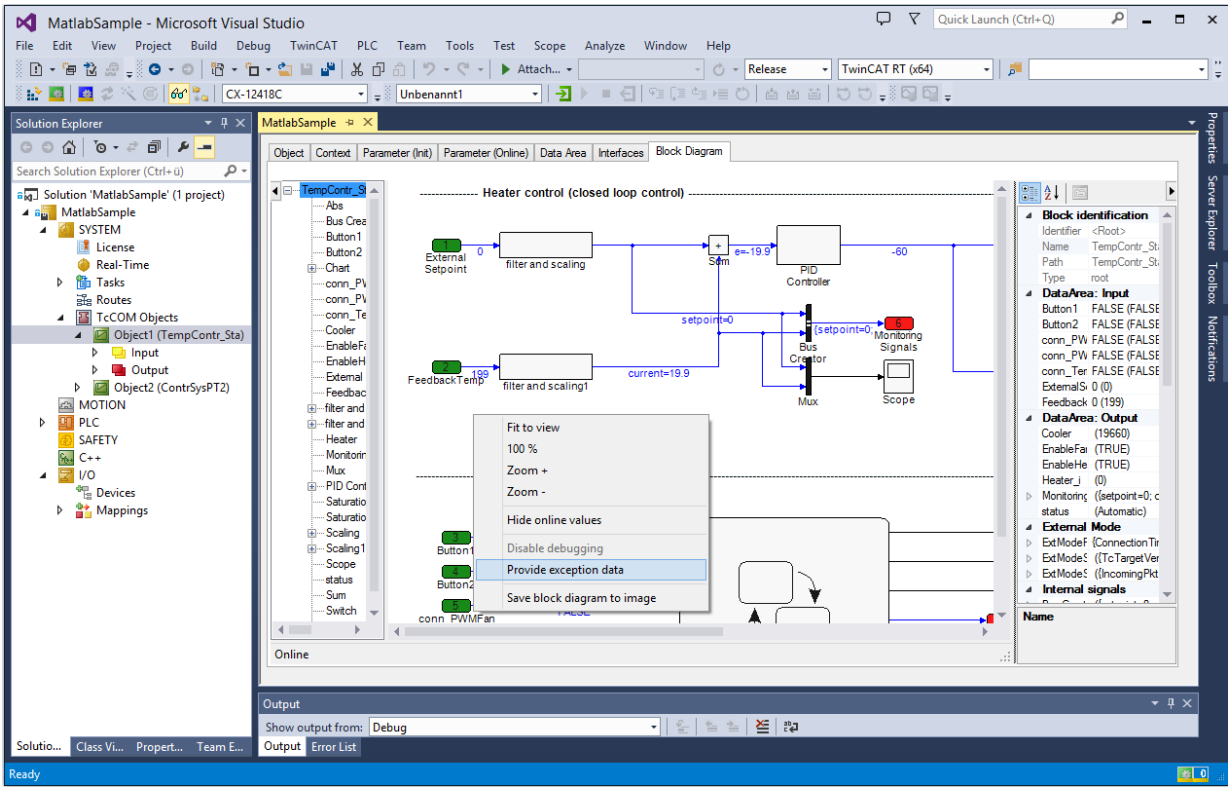

**无需源代码即可手动评估异常**

即使工程系统中没有模块源代码或 C++ 调试器未激活，一旦出现异常，还是可以在框图中突出显示出错位置。

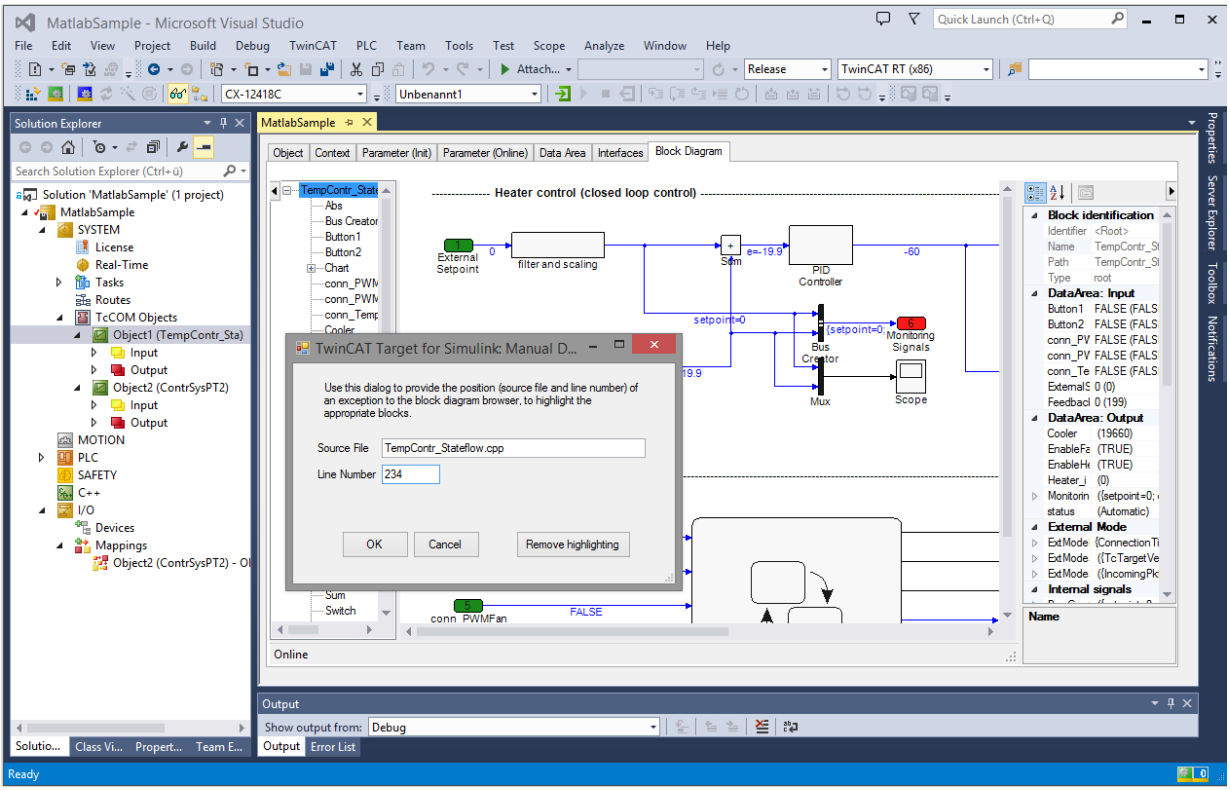
通常，发生错误时总会生成一条错误信息，指示源文件和源代码中的哪一行。在许多情况下，该信息可用于将异常分配给框图中的某个功能块。具体操作如下：

✓ 高亮显示框图中出错位置的前提条件是已生成调试信息（**Tc 高级**（Tc Advanced）下编码器设置中的选项**导出框图调试信息**（Export block diagram debug information））。

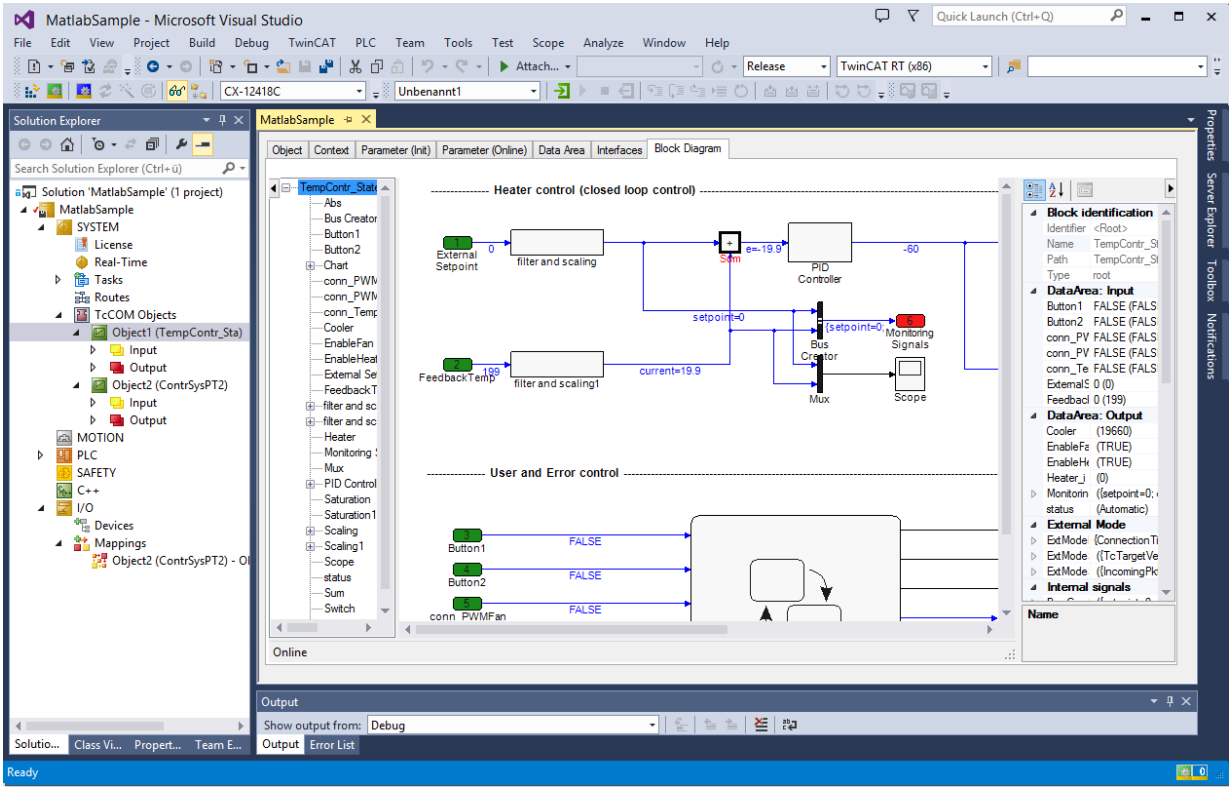
3. 从框图的上下文菜单中选择条目**提供异常数据**（Provide exception data）：




4. 在打开的对话框中，输入错误信息中提供的源代码文件和行编号：



5. 与行编号相关的功能块名称以红色显示，功能块本身以粗体标出：




### 4.8.1.4 在运行时在线更改 TcCOM

通过“在线更改”功能，可在运行时 PC 上在运行时交换 TcCOM 对象，即无需停止 TwinCAT。

关于 TcCOM 在线更改的说明，请参见：在线更改 TcCOM [▶ 65]。

**在 MATLAB® 中打开示例**

使用以下参数，在 MATLAB® 中打开一个示例：

```
TwinCAT.ModuleGenerator.Samples.Start("Online Change for TcCOM Modules")
```

### 4.8.1.5 ToFile 功能块和 MAT 文件记录

您可以对 Simulink® 模型进行配置，以便在 TwinCAT runtime 中以 TcCOM 对象的形式在运行时 PC 的文件系统中生成 MAT 文件。

**观察运行时 PC 上的写入权限**

注意要写入路径的写入权限。

#### MAT 文件日志

来自 Simulink® 的配置：

- 在**代码生成 > 接口 > MAT 文件日志**（Code Generation > Interface > MAT-file logging）下启用 MAT 文件日志，请参见 MathWorks® 文档。
- 启用**代码生成 > TC 常规 > 加载数据交换模块**（Code Generation > TC General > Load DataExchangeModules）。

如果使用这些设置创建了 TcCOM 对象，并在运行时系统上的 TwinCAT 配置中激活了该对象，那么模型信号将根据 MathWorks® 指定的属性保存在 MAT 文件中。

MAT 文件创建在运行时 PC 的文件系统中，位于 TwinCAT 启动目录下。

**To File 功能块**

Simulink® 中的配置：

- 通过路径**代码生成 > 接口 > MAT 文件日志**（Code Generation > Interface > MAT-file logging）启用 MAT 文件日志。
- 启用**代码生成 > TC 常规 > 加载数据交换模块**（Code Generation > TC General > Load DataExchangeModules）。
- 在**ToFile 功能块**（ToFile Block）中指定完整路径，例如 *C:|Logs|MyLog.mat*。如果只指定文件名，则将在 TwinCAT 启动目录下创建 MAT 文件。
- 在 **ToFile 功能块**（ToFile Block）中选择**功能块参数**（Block Parameters）：*保存格式：数组*（Save format: Array）。

如果使用这些设置创建了一个 TcCOM 对象，并在运行时系统的 TwinCAT 配置中激活了该对象，则会在配置的位置创建一个 MAT 文件。

- MAT 文件在运行时填充新数据，其所需内存大小也会随时间推移而相应增加。
- 完成 MAT 文件的终止方法在 Transition Preop Init. 中执行。为此，可以将 TcCOM 对象移动到初始状态，或者将 TwinCAT Runtime 设置为配置模式。

***注意***

**足够的存储空间**

请注意，必须在目标系统上保留足够的存储空间，以避免运行时 PC 出现不可预测的行为。

**TwinCAT 文件写入器**

您可以使用 TwinCAT 文件写入器在运行时精确控制数据记录。TwinCAT 文件写入器可以终止规定大小的文件包，并在运行时系统上生成一组指定的文件。因此，不会有达到目标系统内存极限的危险。

- .mat 文件的大小上限可调整
- .mat 文件的最大数量可调整
- 可选择通过 TcCOM 模块参数暂停写入
- 不支持所有数据类型

该区块的文件可在这里找到。

**还请参阅有关此**

TwinCAT File Writer [▶ 35]

### 4.8.1.6 从 PLC 调用 TcCOM

请参见应用 TcCOM Wrapper FB [▶ 134]章节。

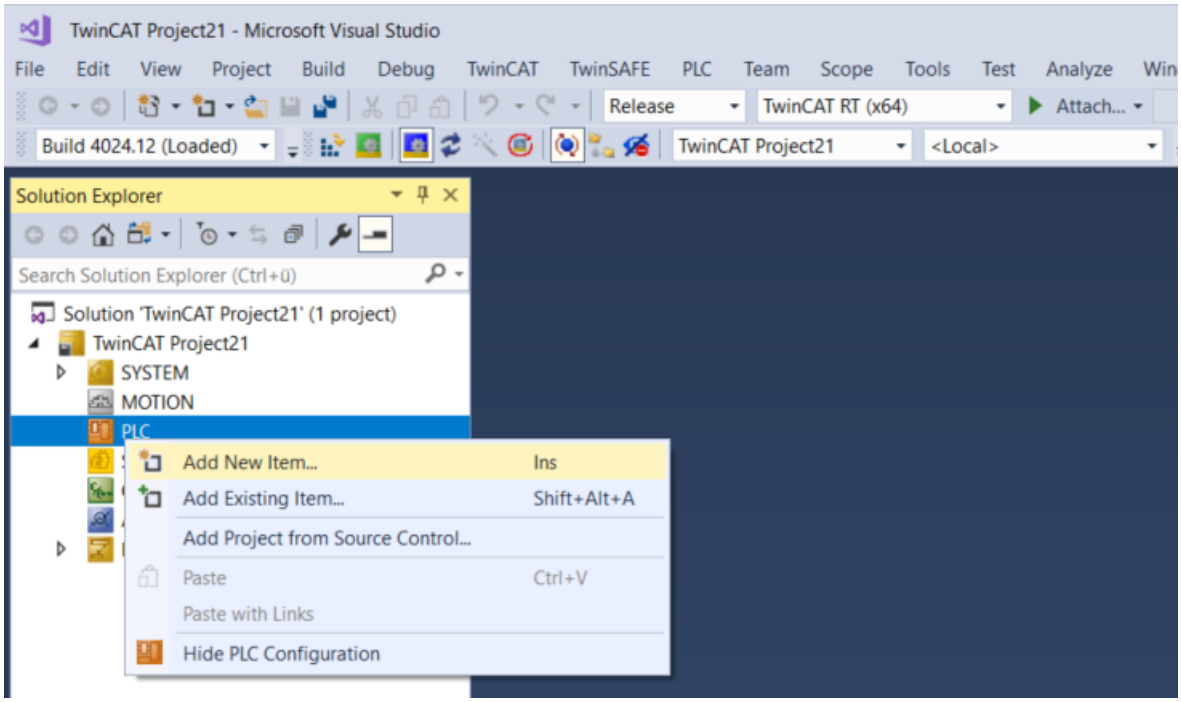
## 4.8.2 使用 PLC 库

除 TcCOM 对象外，还可以创建 PLC 库。该 PLC 库包含**两个**不同**类型**的功能块：

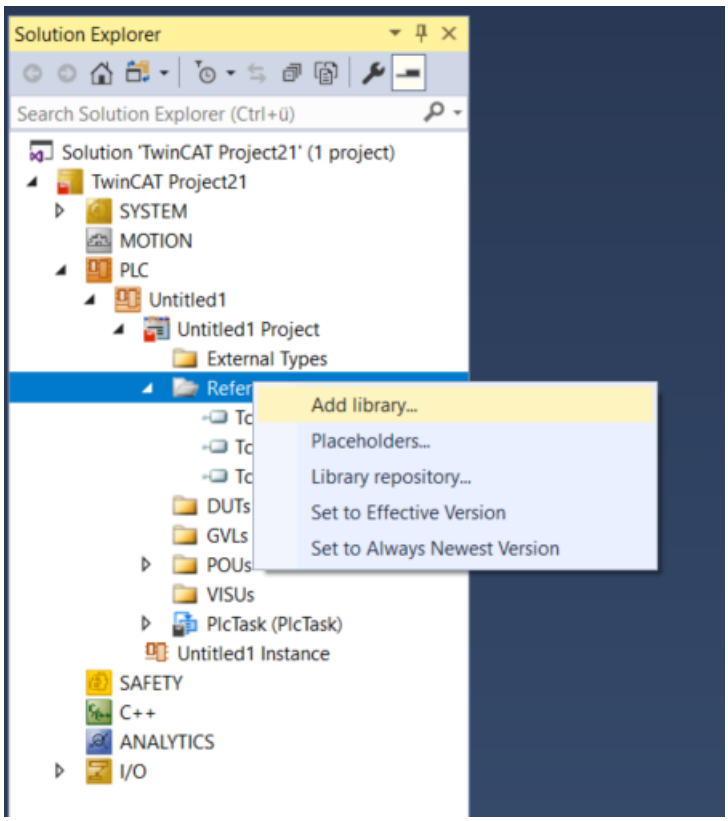
- *POU* 文件夹中的功能块 *FB_<modelname>* 包含 Simulink® 模型的完整功能，即源代码直接存在于 FB 中。该 FB 在下文中叫做 **PLC-FB**。
- 子文件夹 *POU/TcCOM Wrapper* 包含作为 TcCOM 对象封装器的 FB，即功能不是直接存在于 FB 中，而是外包给 TcCOM 实例。这些封装器在下文中叫做 **TcCOM-Wrapper-FB**。

**简要概述**

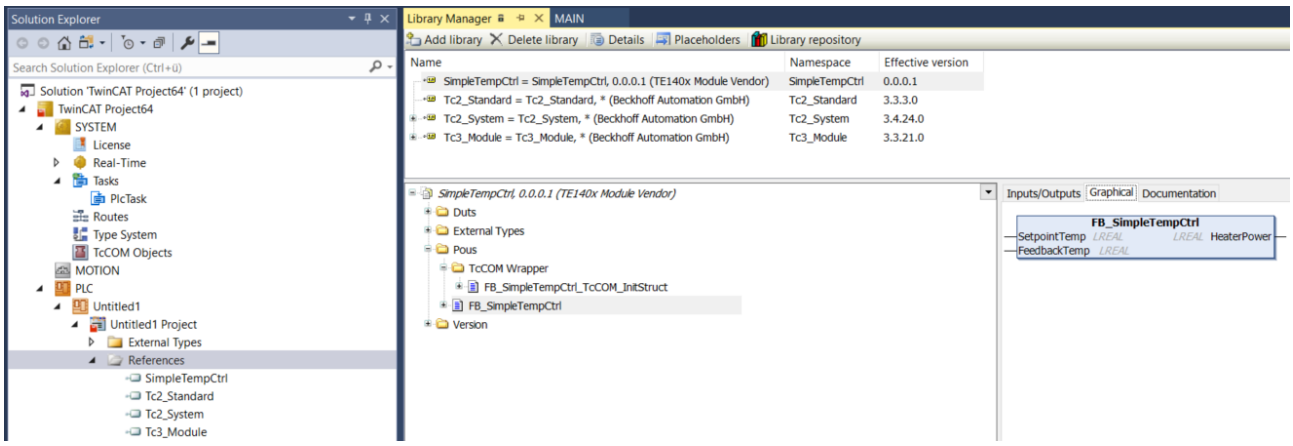
- 创建 PLC 项目：



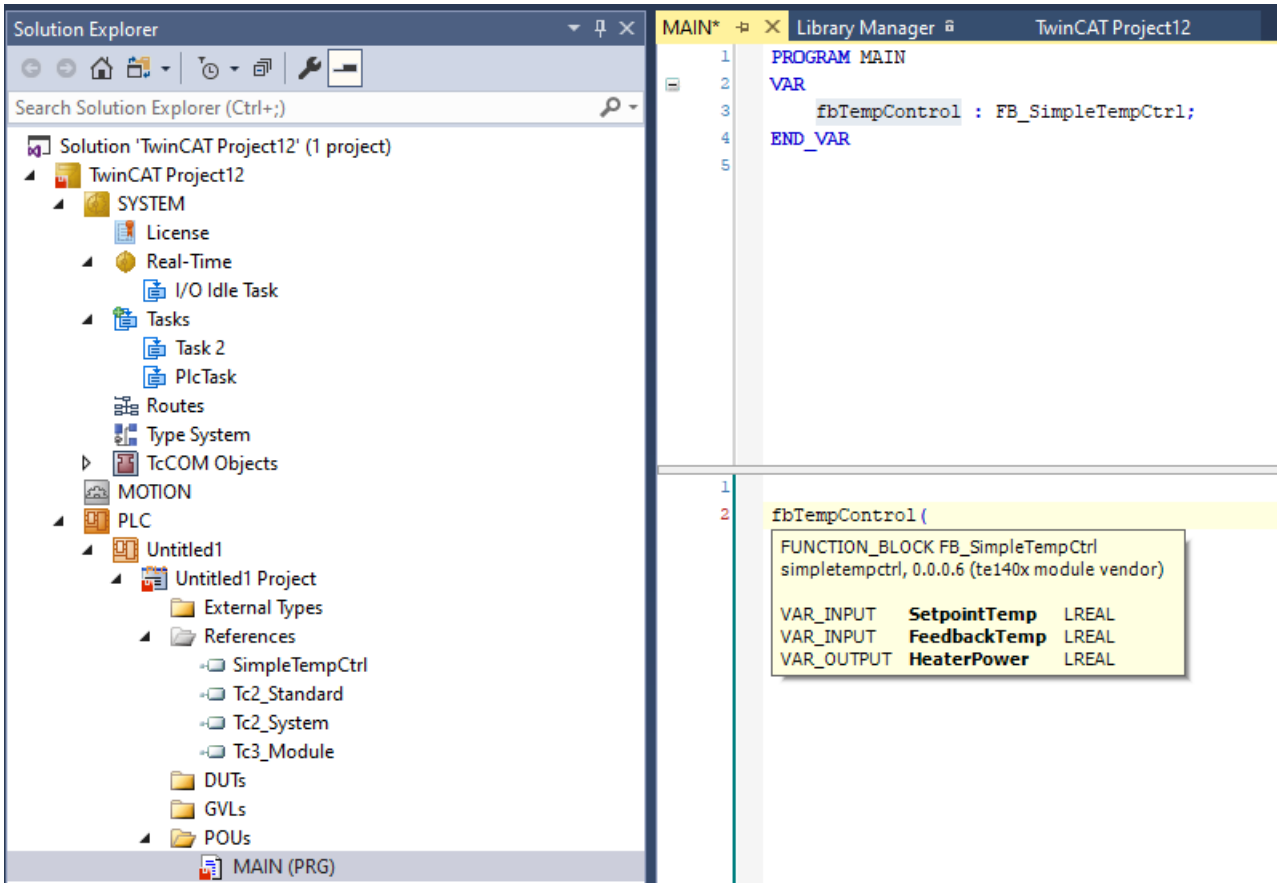

- 加载 PLC 库：



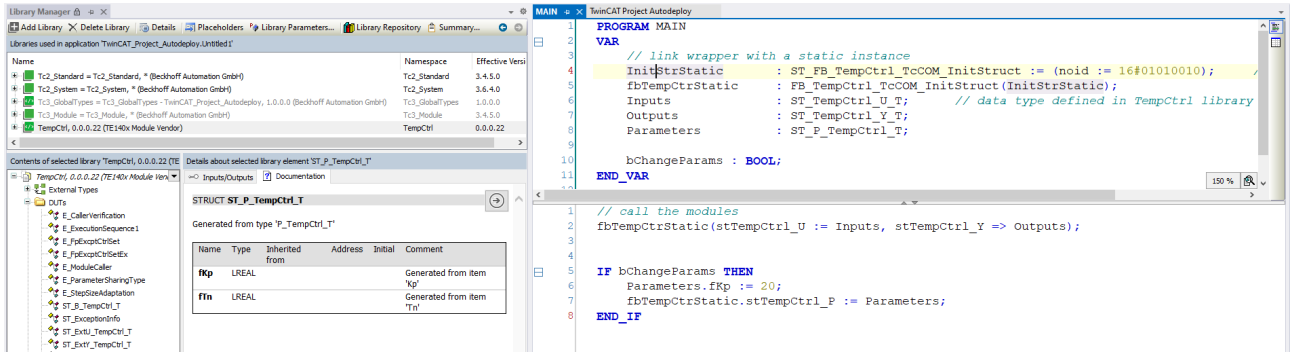

- 查看 PLC 库的内容：




- 实例化功能块 *FB_<modelname>*（PLC-FB [▶ 138]）并在 PLC 代码中使用：




- 或者：使用 TcCOM Wrapper FB [▶ 134] 并在 PLC 代码中使用：



### 4.8.2.1 创建并安装 PLC 库

#### 配置 PLC 库的内容

如使用 PLC 库 [▶ 128]所述，PLC 库可包含不同的功能块。

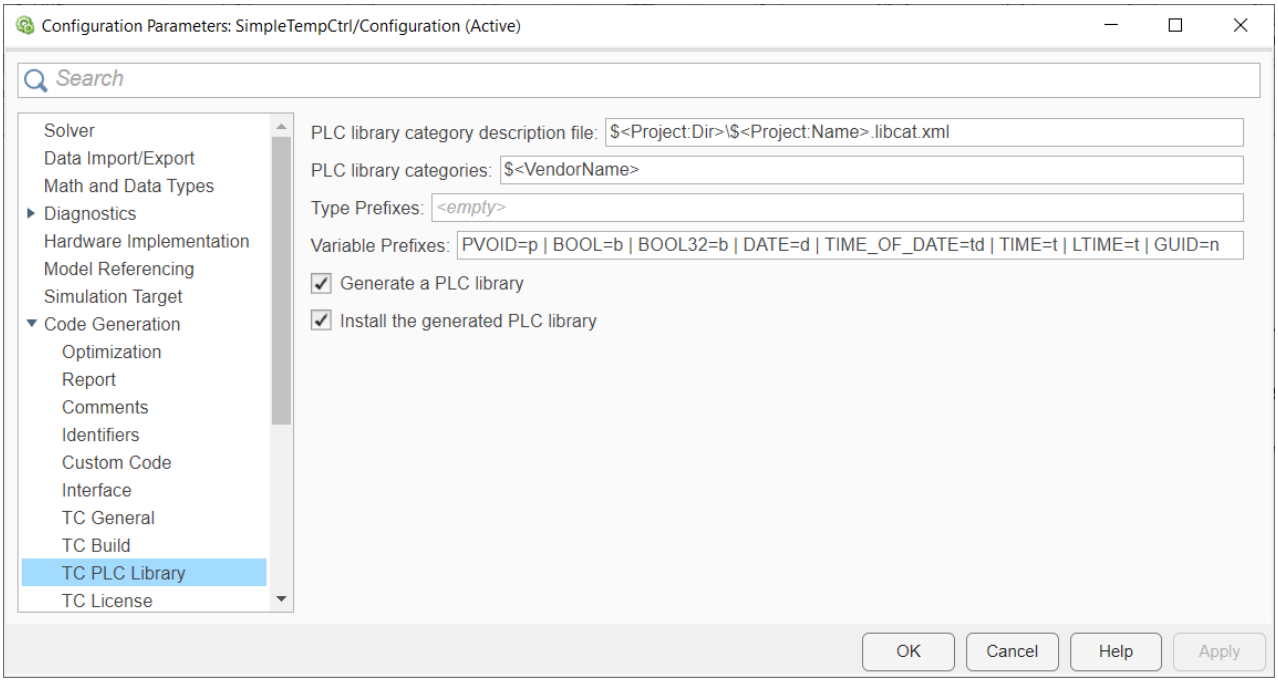
- *POU* 文件夹中的功能块 *FB_<modelname>* 包含 Simulink® 模型的完整功能，即源代码直接存在于 FB 中。该 FB 在下文中叫做 **PLC-FB**。
- 子文件夹 *POU/TcCOM Wrapper* 包含作为 TcCOM 对象封装器的 FB，即功能不是直接存在于 FB 中，而是外包给 TcCOM 实例。这些封装器在下文中叫做 **TcCOM-Wrapper-FB**。

下面，您将学习如何配置两个功能块以及如何安装创建的 PLC 库。

#### 4.8.2.1.1 配置和安装 PLC 库

#### 配置类型和变量前缀

您可以在**代码生成 > TC PLC**（Code Generation > TC PLC）库中配置在创建的 PLC 库中使用的单个类型和变量前缀。默认情况下，变量前缀是根据这一约定生成的（参见变量和实例的标识符）。




以*管道分隔列表*的形式输入所需的前缀。

#### 安装 PLC 库

如上所述，您可以配置哪些功能块（PLC-FB [▶ 132] 和/或 TcCOM-Wrapper-FB [▶ 132]）应包含在 PLC 库中。为了在 PLC 中使用创建的功能块，必须在 TwinCAT 开发系统中安装相应的 PLC 库。

**情况 1：**

✓ 您可以在要对 PLC 编程的同一台 PC 上使用 Target for Simulink®。

1. 创建 PLC 库，并直接从 Simulink® 安装到本地开发系统中。
2. 为此，请在**TC PLC 库**（TC PLC Library）中选择相应的复选框：

⇨ 成功构建后，新版本的 PLC 库可直接在本地 TwinCAT XAE 中使用。

**情况 2：**

✓ 您希望在任一 TwinCAT XAE 系统上使用 PLC 库。

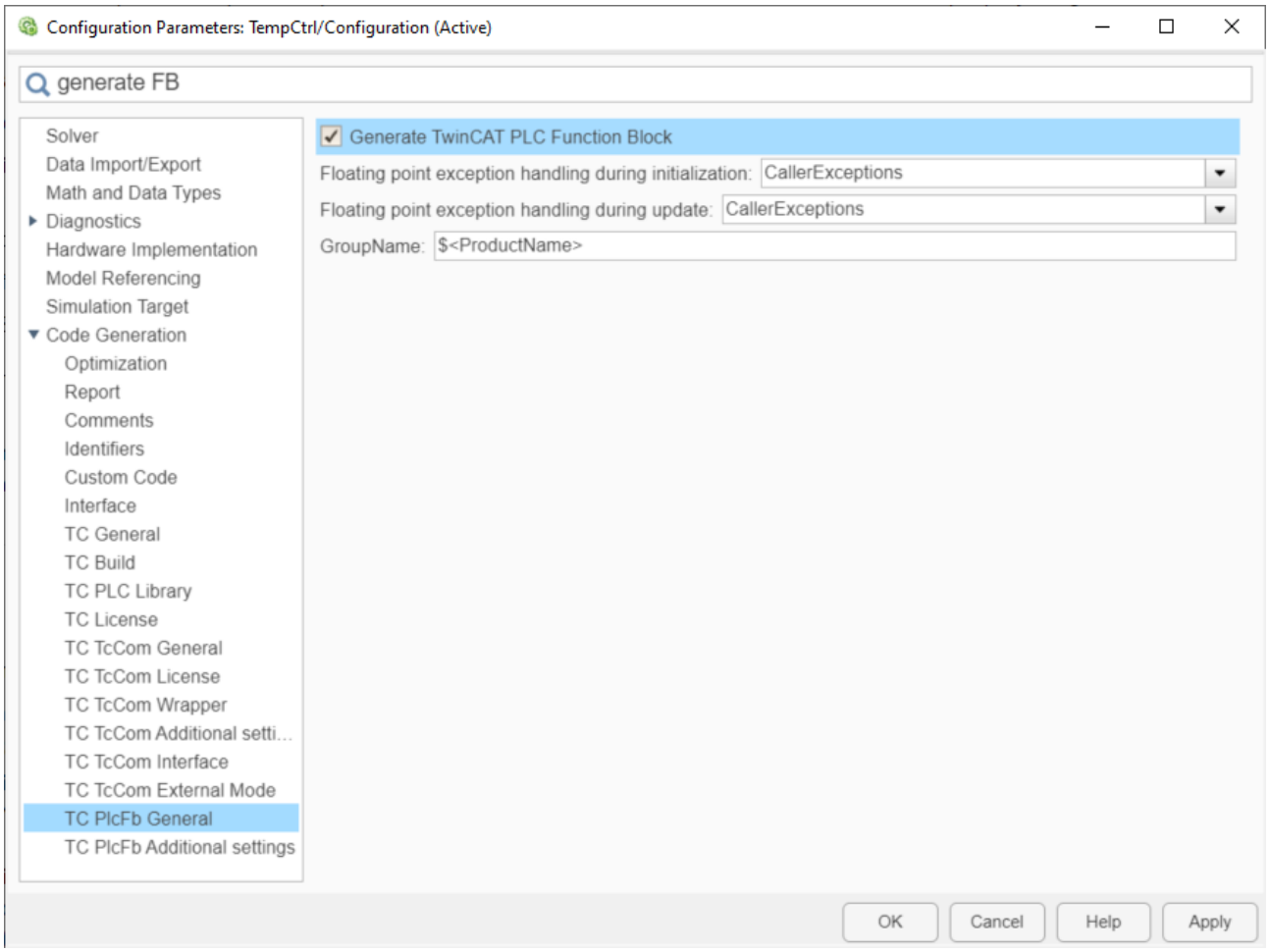
1. 创建 TMX 存档 [▶ 57]。
2. 将 TMX 存档 [▶ 57]复制到您选择的 TwinCAT 开发系统。
3. 在解压缩TMX 存档 [▶ 57]时安装 PLC 库。

⇨ TwinCAT XAE 中可使用存档中包含的库。

#### 4.8.2.1.2 创建并配置 PLC-FB

导航至**配置能数 > 代码生成 > TC PlcFb 常规**（Configuration Parameters > Code Generation > TC PlcFb General）。

在此处选中复选框*生成 TwinCAT PLC 功能块*（Generate TwinCAT PLC Function Block）。在标准配置中，复选框被选中。

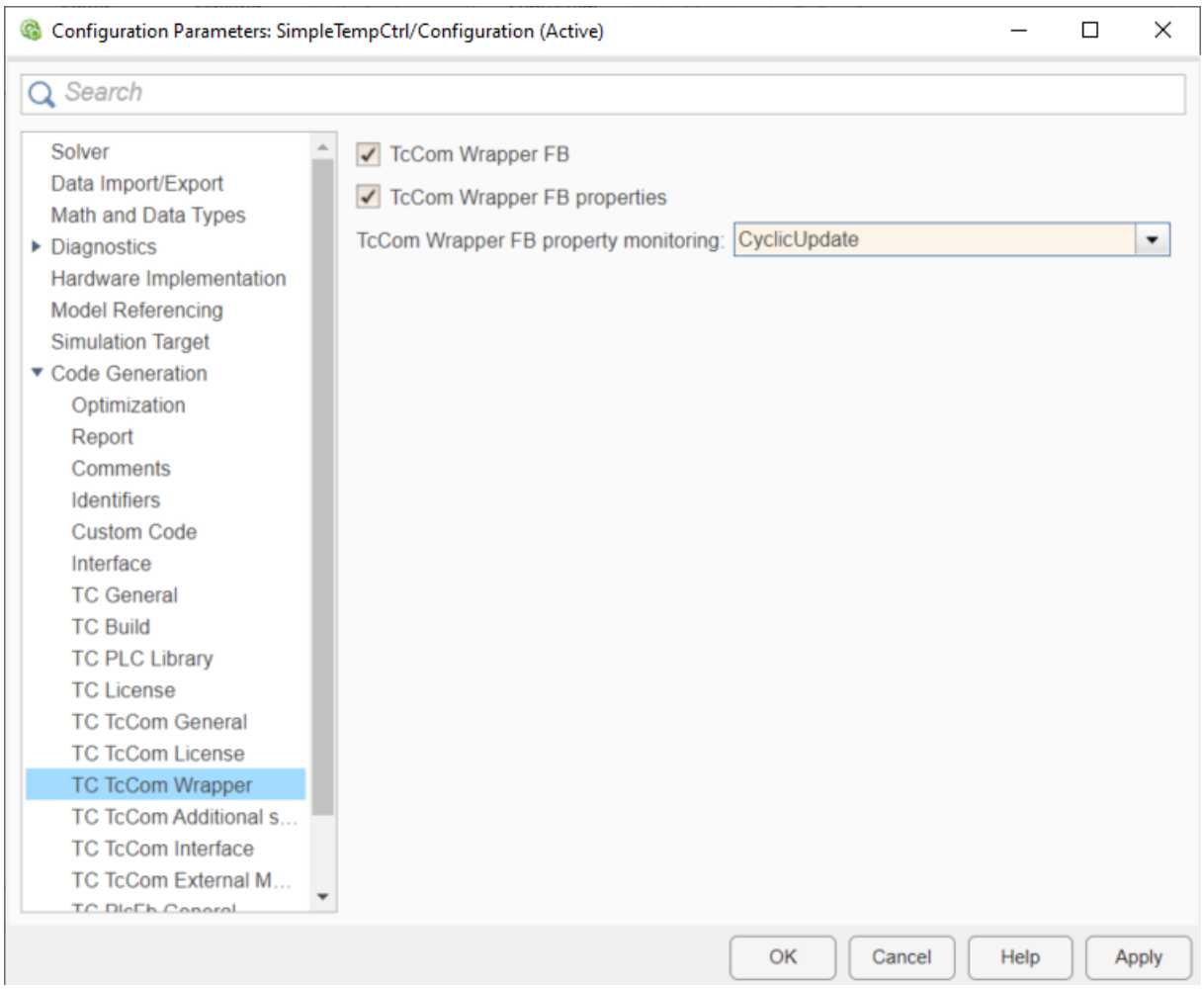



其他设置涉及 PLC-FB 的浮点异常处理。这些设置必须与 TcCOM 对象的设置分开进行，请参见异常处理 [▶ 145]。

#### 4.8.2.1.3 创建并配置 TcCOM-Wrapper-FB

导航至**配置参数 > 代码生成 > TC TcCom Wrapper**（Configuration Parameters > Code Generation > TC TcCom Wrapper）。

在此处选中 *TcCom Wrapper FB* 复选框。在标准配置中，该复选框未被选中。




您可以使用复选框 *TcCom Wrapper FB properties* 配置是否将 FB 上的模块参数创建为属性。参见配置对 TcCOM 对象数据的访问权限 [▶ 67]，尤其是*参数：初始值（Parameter: Initial values）*。

您可以在此处了解如何应用生成的 Wrapper：应用 TcCOM Wrapper FB [▶ 134]。

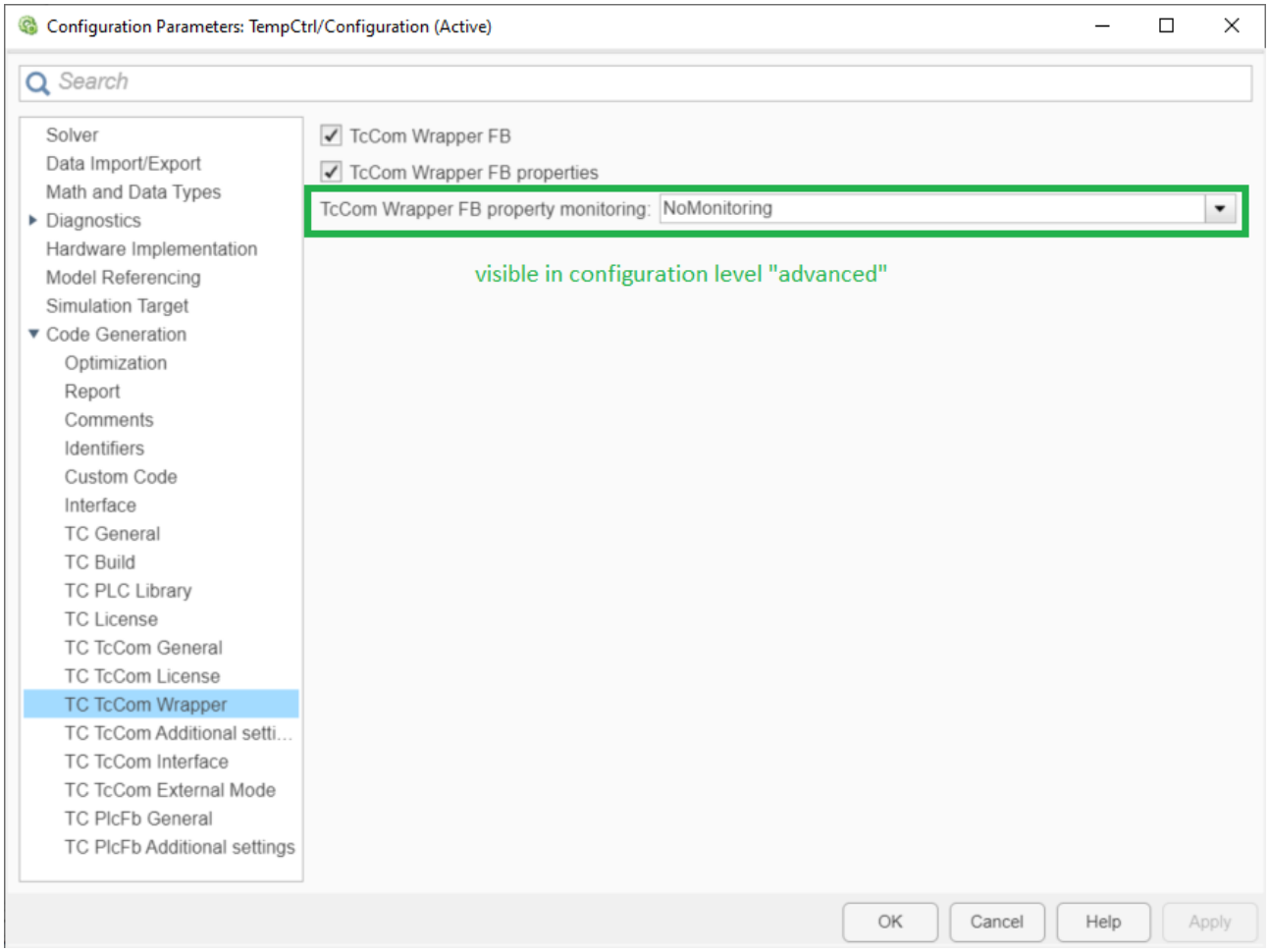
**示例**

通过设置 Tc TcCom Interfaces 下的*参数：初始值*（Parameter: Initial Values），模型参数将作为模块参数创建（默认打开）。现在使用 TcCom Wrapper FB properties 选项创建 “TcCom Wrapper FB”。将监控特性设置为 CyclicUpdate，可直接在联机视图中查看特性值的变化。

然后就可以访问模块参数，请参见如下示例：

```
PROGRAM MAIN
VAR

   // dynamic instance: create TcCOM from PLC
   InitStrDyn : ST_FB_TempCtrl_TcCOM_InitStruct_InitStruct := (
                    nTaskOid:= 16#02010030,                    // take TaskOID of PlcTask
                    eModuleCaller:= E_ModuleCaller.Module );   // set module caller to "call by mod
ule"
   fbTempCtrDyn : FB_TempCtrl_TcCOM_InitStruct(InitStrDyn);

   Outputs    : ST_TempCtrl_Y_T;   // output
   Inputs     : ST_TempCtrl_U_T;   // input
   Parameters : ST_P_TempCtrl_T;      // parameter

   bChange: BOOL;
END_VAR
```

```
fbTempCtrDyn(stTempCtrl_U := Inputs, stTempCtrl_Y => OutputsDyn);

IF bChange THEN
   Parameters.Kp := 10;
   fbTempCtrDyn.stTempCtrl_P := Parameters;
END_IF
```




在“高级”配置级别，还可以指定特性的监控属性（monitoring attribute）。默认情况下，设置为"无监控"（No Monitoring），即不设置任何属性。

| Simulink® 中的设置 | 特性的属性 |
|---|---|
| ExecutionUpdate | {attribute 'monitoring' := 'variable'} |
| CyclicUpdate | {attribute 'monitoring' := 'call'} |

监控属性会影响在线视图中属性值的可见性，即当您登录 PLC 并希望监控 FB 上特性的当前值时。

- 无监控：在线视图中不显示数值。
- 循环更新：循环更新并显示特性值。
  **在 PLC 的登录状态下执行附加代码。**
- 执行更新：只有在执行代码中调用特性的 getter/setter 方法时，才会在联机视图中更新特性值。**这将迅速导致刺激，只有在极少数情况下才适用。**

**TcCOM Wrapper 无在线更改功能**

如果 TcCOM 具有在线更改功能，则不会生成 TcCOM-Wrapper-FB，因为 PLC 库的版本和 TcCOM 对象的版本必须始终匹配，而在线更改 TcCOM 时无法保证这一点。

### 4.8.2.2 应用 TcCOM Wrapper FB

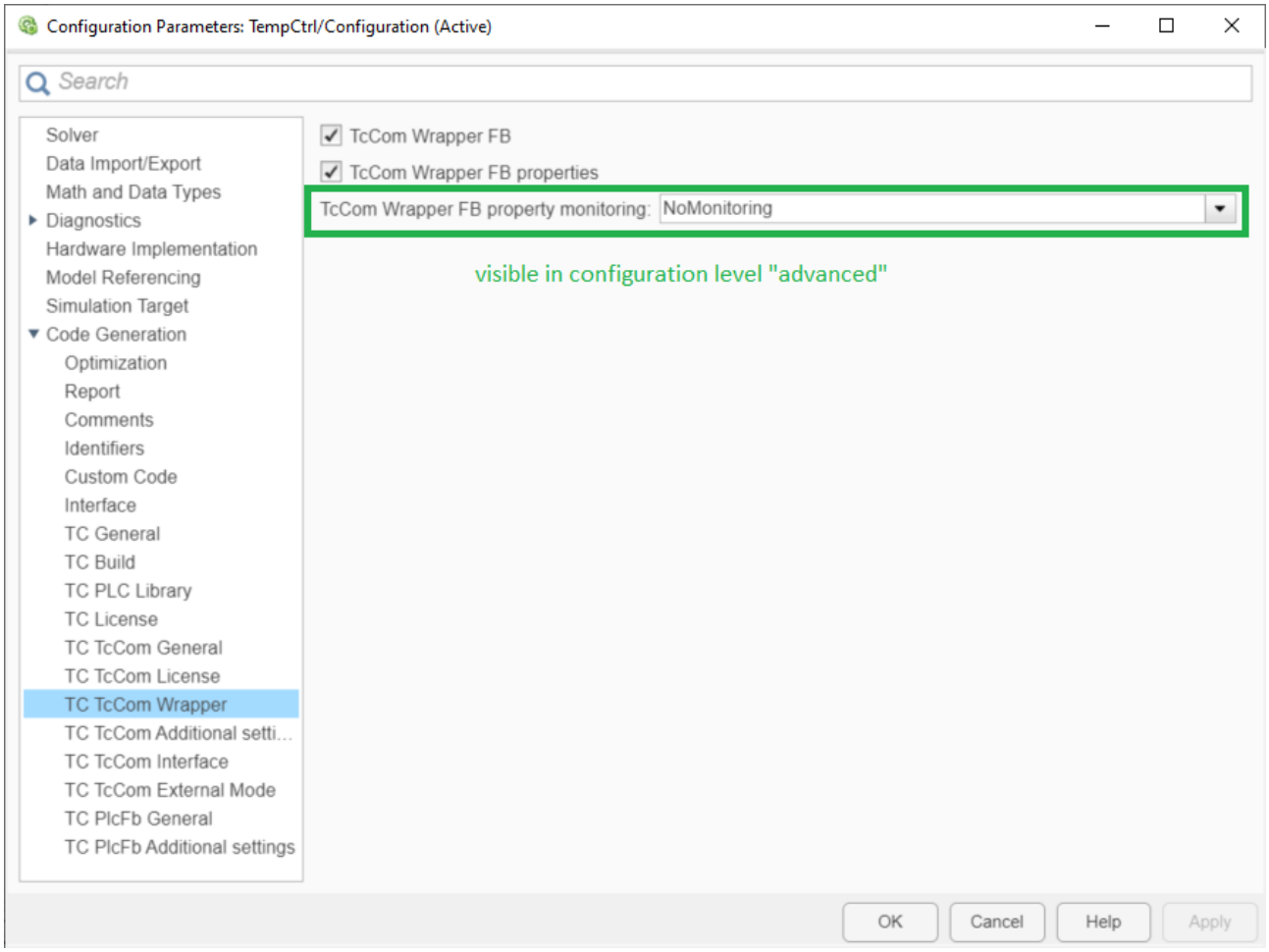
从 PLC 调用 TcCOM 对象有两种方法：

1. 引用静态对象实例
2. 动态实例化来自 PLC 的对象

⇨ 这两种方式都使用生成的 PLC 库中的 TcCOM Wrapper 功能块。

### 在 Simulink® 中配置 TcCOM Wrapper 功能块

浏览至**配置参数 > 代码生成 > TC TcCom 封装器**（Configuration Parameters > Code Generation > TC TcCom Wrapper）。在此处选中 **TcCom Wrapper FB** 复选框。在标准配置中，该复选框未被选中。可以使用 **TcCom Wrapper FB 特性**（TcCom Wrapper FB properties）复选框配置是否将功能块上的模型参数创建为特性。




在“高级”配置级别，还可以指定特性的监控属性（monitoring attribute）。默认情况下，设置为“不监控”（No Monitoring），即不设置任何属性。

| Simulink® 中的设置 | 特性的属性 |
|---|---|
| ExecutionUpdate | {attribute 'monitoring' := 'variable'} |
| CyclicUpdate循环更新 | {attribute 'monitoring' := 'call'} |

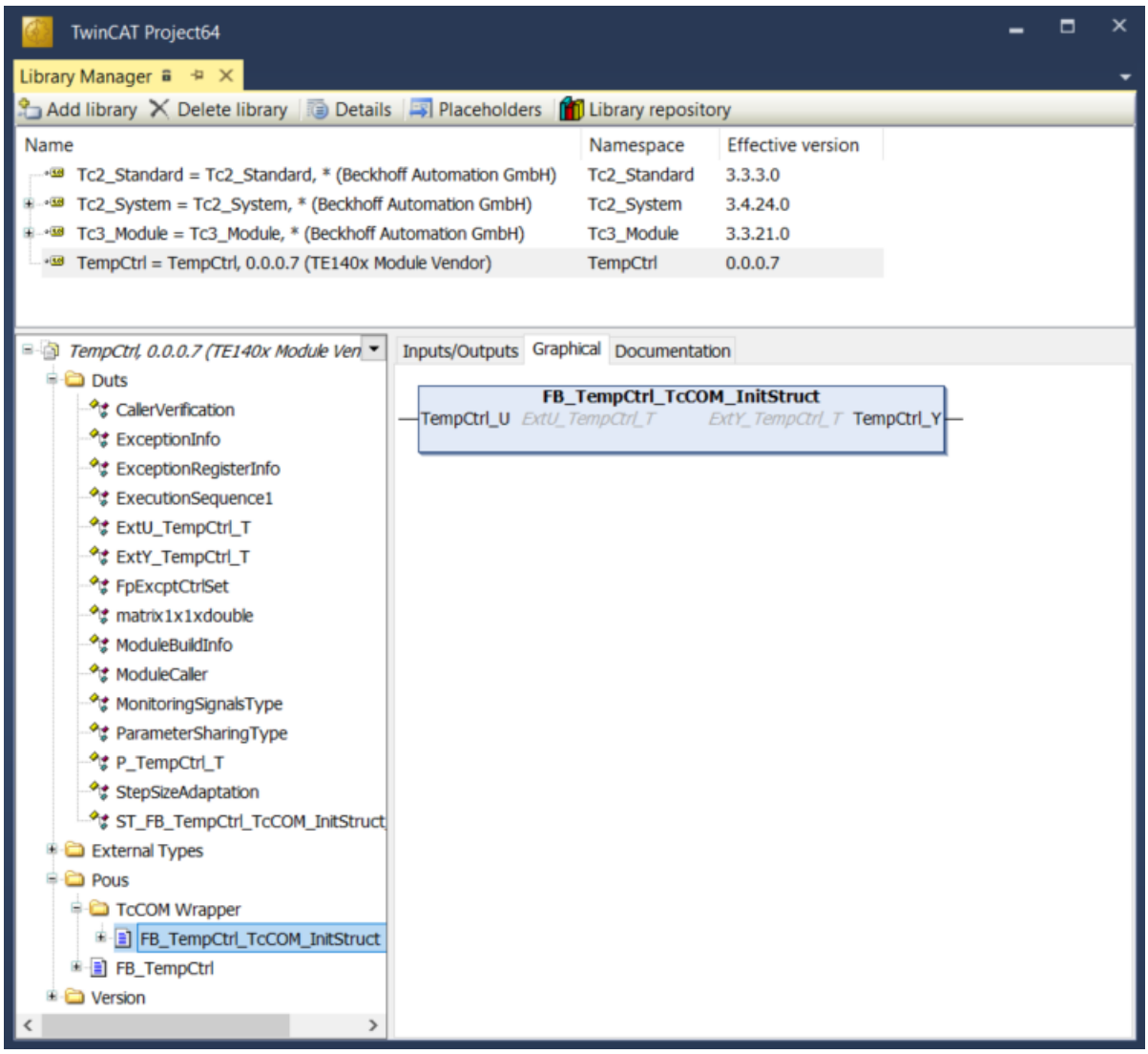
监控属性会影响属性值在线视图中的显示，也就是您登录 PLC 并希望监控 FB 上的当前特性值时。

- 无监控（No Monitoring）：在线视图中不显示数值。
- 循环更新（Cyclic Update）：循环更新并显示特性值。
  **在 PLC 的登录状态下执行附加代码。**
- 执行更新（Execution Update）：只有在执行代码中调用特性的 getter/setter 方法时，在线视图中才会更新特性值。**这将迅速导致刺激，只有在极少数情况下才适用。**

详情另请参见创建和安装 PLC 库 [▶ 131]或创建 TcCOM-Wrapper-FB。

### 创建 TcCOM Wrapper 功能块实例

1. 创建 PLC 项目。
2. 在**引用**（References）下添加所需的库。




⇨ 在 **Pous/TcCOM Wrapper** 下，可以得到一个可在 PLC 中实例化的功能块。此外，还在 *Duts* 文件夹中创建必要的数据类型。

### 版本 1：引用静态模块实例

该功能块可用于访问之前在 XAE 中创建的模块实例，如在**系统 > TcCOM 对象**（System > TcCOM Objects）下。如果是静态实例，必须在声明功能块实例时转移相应模块实例的对象 ID。

- TcCOM 对象实例和调用的 PLC 必须在同一个任务中运行。
- 在 TcCOM 对象实例中，确保 Parameter (Init) 下的 *ModuleCaller* 条目设置为*模块*（Module），而不是*周期性任务*（CyclicTask）。
- 在这种情况下，TcCOM 所需的内存从系统的*非分页池*中获取。

声明

```
// link wrapper with a static instance
InitStrStatic       : ST_FB_TempCtrl_TcCOM_InitStruct := (noid := 16#01010010);   // OID from object1
 in System > TcCOM Objects
fbTempCtrStatic     : FB_TempCtrl_TcCOM_InitStruct(InitStrStatic);
Inputs              : ST_TempCtrl_U_T;      // data type defined in TempCtrl library
Outputs             : ST_TempCtrl_Y_T;
```

执行代码

```
fbTempCtrStatic(stTempCtrl_U := Inputs, stTempCtrl_Y => Outputs);
```

### 版本 2：从 PLC 进行动态实例化和引用

该功能块还可用于从 PLC 生成 TcCOM 对象并链接到封装器。

- 必须使用 PLC 任务的 TaskOid 来指定调用封装器的实时任务。
- 还必须在此处（通过 Init 结构）将 ModuleCaller 设置为*模块*（Module）。
- 在这种情况下，TcCOM 所需的内存来自路由器内存。

声明

```
// dynamic instance: create TcCOM from PLC
InitStrDyn          : ST_FB_TempCtrl_TcCOM_InitStruct := (
                        nTaskOid      := 16#02010030,             // take TaskOID of PlcTask
                        eModuleCaller   := E_ModuleCaller.Module );      // set module caller to "ca
ll by module"
fbTempCtrDyn        : FB_TempCtrl_TcCOM_InitStruct(InitStrDyn);
OutputsDyn          : ST_TempCtrl_Y_T;
```

执行代码

```
fbTempCtrDyn(stTempCtrl_U := Inputs, stTempCtrl_Y => OutputsDyn);
```

**上图所示图形的源代码可在 MATLAB® 中通过命令窗口获取：**

```
TwinCAT.ModuleGenerator.Samples.Start("TcCOM Wrapper Function Blocks")
```

### 使用 TcCOM Wrapper FB 的特性

通过 FB 上的特性，可以轻松地与 TcCOM 模块参数进行交互，另请参见最佳实践：访问 TcCOM 数据 [▶ 71]。

### 示例

通过设置 Tc TcCom Interfaces 下的*参数：初始值*（Parameter: Initial Values），模型参数将作为模块参数创建（默认打开）。现在使用 TcCom Wrapper FB properties 选项创建 “TcCom Wrapper FB”。将监控特性设置为 CyclicUpdate，可直接在联机视图中查看特性值的变化。

然后就可以访问模块参数，请参见如下示例：

```
PROGRAM MAIN
VAR

   // dynamic instance: create TcCOM from PLC
   InitStrDyn : ST_FB_TempCtrl_TcCOM_InitStruct_InitStruct := (
                    nTaskOid:= 16#02010030,                    // take TaskOID of PlcTask
                    eModuleCaller:= E_ModuleCaller.Module );   // set module caller to "call by mod
ule"
   fbTempCtrDyn : FB_TempCtrl_TcCOM_InitStruct(InitStrDyn);

   Outputs    : ST_TempCtrl_Y_T;   // output
   Inputs     : ST_TempCtrl_U_T;   // input
   Parameters : ST_P_TempCtrl_T;      // parameter

   bChange: BOOL;
END_VAR
```

```
fbTempCtrDyn(stTempCtrl_U := Inputs, stTempCtrl_Y => OutputsDyn);

IF bChange THEN
   Parameters.Kp := 10;
   fbTempCtrDyn.stTempCtrl_P := Parameters;
END_IF
```

### 使用 ADI 接口

**⚠ 警告**

**读写权限不受限制**

通过 ITc_ADI 接口可以获得指向 DataArea 内存区域的指针。因此，在该区域的读写不受任何限制。

以下示例说明了如何以只读模式访问 BlockIO DataArea：

```
stInitTemp : ST_Funktionsblock_SimpleTempCtrl_TcCOM_InitStruct := (nOid := 16#01010010);
FunktionsblockTempCtr  : Funktionsblock_SimpleTempCtrl_TcCOM_InitStruct(stInitTemp);
stTempCtr_BlockIO  : ST_B_SimpleTempCtrl_T;
pStTempCtr_BlockIO : POINTER TO ST_B_SimpleTempCtrl_T;
hr : HRESULT;
```

```
(* read data are via ADI Interface *)
// get a pointer to Data Area
hr := FunktionsblockTempCtr.ipADI.GetImagePtr(size := SIZEOF(stTempCtr_BlockIO), offs := 0, adi_x :=
 2, ppData := ADR(pStTempCtr_BlockIO));
```

```
IF hr = 0 THEN
   // copy data to a local variable
   MEMCPY(ADR(stTempCtr_BlockIO), pStTempCtr_BlockIO, SIZEOF(stTempCtr_BlockIO));
   // always release the pointer!
   FunktionsblockTempCtr.ipADI.ReleaseImagePtr(pData := pStTempCtr_BlockIO);
END_IF;
```

Object | Context | Parameter (Init) | Parameter (Online) | Data Area | Interfaces | Block Diagram

| | Area No | Name | Type | Size | CS | CD / Elements |
|---|---|---|---|---|---|---|
| + | 0 (0) | SimpleTempCtrl_U | InputDst | 16 | ☑ | ☐ 2 Symbols |
| + | 1 (0) | SimpleTempCtrl_Y | OutputSrc | 8 | ☑ | ☐ 1 Symbols |
| - | 2 (0) | SimpleTempCtrl_B | Internal | 64 | ☑ | ☐ 8 Symbols |
| | | e | LREAL | 8.0 (Offs: 0.0) | | 0x82000000 |
| | | P | LREAL | 8.0 (Offs: 8.0) | | 0x82000008 |
| | | Integrator | LREAL | 8.0 (Offs: 16.0) | | 0x82000010 |
| | | Sum | LREAL | 8.0 (Offs: 24.0) | | 0x82000018 |
| | | Saturation | LREAL | 8.0 (Offs: 32.0) | | 0x82000020 |
| | | Switch | LREAL | 8.0 (Offs: 40.0) | | 0x82000028 |
| | | I | LREAL | 8.0 (Offs: 48.0) | | 0x82000030 |
| | | ARW | BOOL | 1.0 (Offs: 56.0) | | 0x82000038 |
| + | 3 (0) | SimpleTempCtrl_X | Internal | 8 | ☑ | ☐ 1 Symbols |
| + | 4 (0) | ExecutionInfo | OutputSrc | 240 | ☐ | ☐ 4 Symbols |

其中，`adi_x` 和 `offs` 被传递给 GetImagePtr 方法。这些参数决定了 DataArea 本身（本例中为 2 号 DataArea（SimpleTempCtrl_B））和待读/写区域中的数据区，本例中没有偏移量和 DataArea 的总大小（即整个 DataArea）。

写入时，必须在上述示例的 `MEMCPY` 上相应地交换源文件和目标文件。

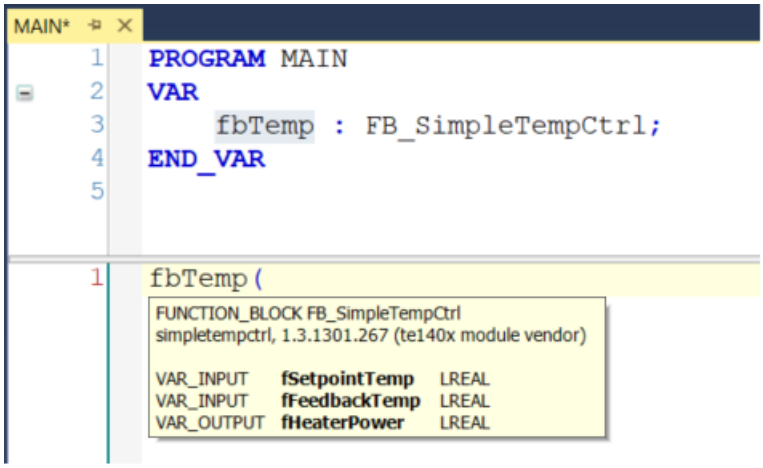
关于如何与 TcCOM 交互的更多说明，请参见最佳实践：访问 TcCOM 数据 [▶ 71]。

### 4.8.2.3 使用 PLC 功能块（PLC-FB）

PLC 功能块（PLC-FB, FB_<modelname>）在 PLC 库中主要用于简单应用。与 TcCOM（或 PLC Wrapper FB）相比，也有一些限制。

**应用**

从已创建的 PLC 库中创建一个或多个 PLC-FB 实例。编写源代码时，可以使用 IntelliSense 方便地查看预期的输入和输出。




如果总线对象在 Simulink® 中被用作输入或输出，这些数据类型会自动定义为 PLC 库中的结构。

**限制条件**

PLC-FB 不允许通过功能块的特性访问模型参数。不过，有两种方法可以更改模型参数：

- 关于External 模式 [▶ 142]
- 使用可复用代码 [▶ 116]时，TcCOM 实例可以定义参数。PLC-FB 的所有实例都会自动设置为“inherit”，并采用默认 TcCOM 实例的参数。

此外，在 TwinCAT XAE 中，PLC-FB 不包含框图 [▶ 118]。功能块的调试可通过 TwinCAT C++ 调试器（如此处所述）或 External 模式进行。

**还请参阅有关此**

调试 [▶ 139]

#### 4.8.2.3.1 在线更改 PLC 库

TwinCAT 处于运行模式时，可以在 TwinCAT XAE 中切换 PLC 库版本，并通过在线更改将其加载到正在运行的应用程序中。这意味着无需重启 TwinCAT，即可更新 PLC 库中的所有功能块。

**逐步流程：**

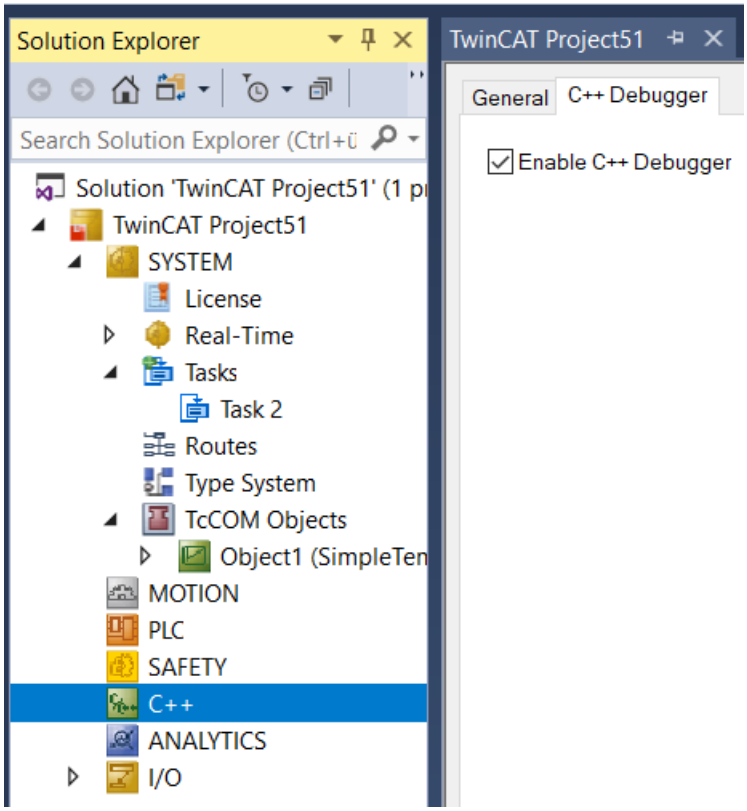
1. 使用 TwinCAT Target for Simulink® 创建第一个 PLC 库版本。
2. 在 PLC 项目中包含此 PLC 库版本。
3. 使用第一个 PLC 库版本（例如，版本 0.0.0.1）激活 TwinCAT 配置。
4. 调整 Simulink® 模型，并从中创建 PLC 库版本（0.0.0.2）。
5. 在**引用**（References）的 PLC 中选择新创建的 PLC 库版本（可能需要在 XAE 系统上安装新库）。
6. 选择**构建 > 构建解决方案**（Build > Build Solution）重建项目。
7. 选择**登录 > 在线更改登录**（Login > Login with online change）（更多信息请参见 PLC 文档）。

### 4.8.3 调试

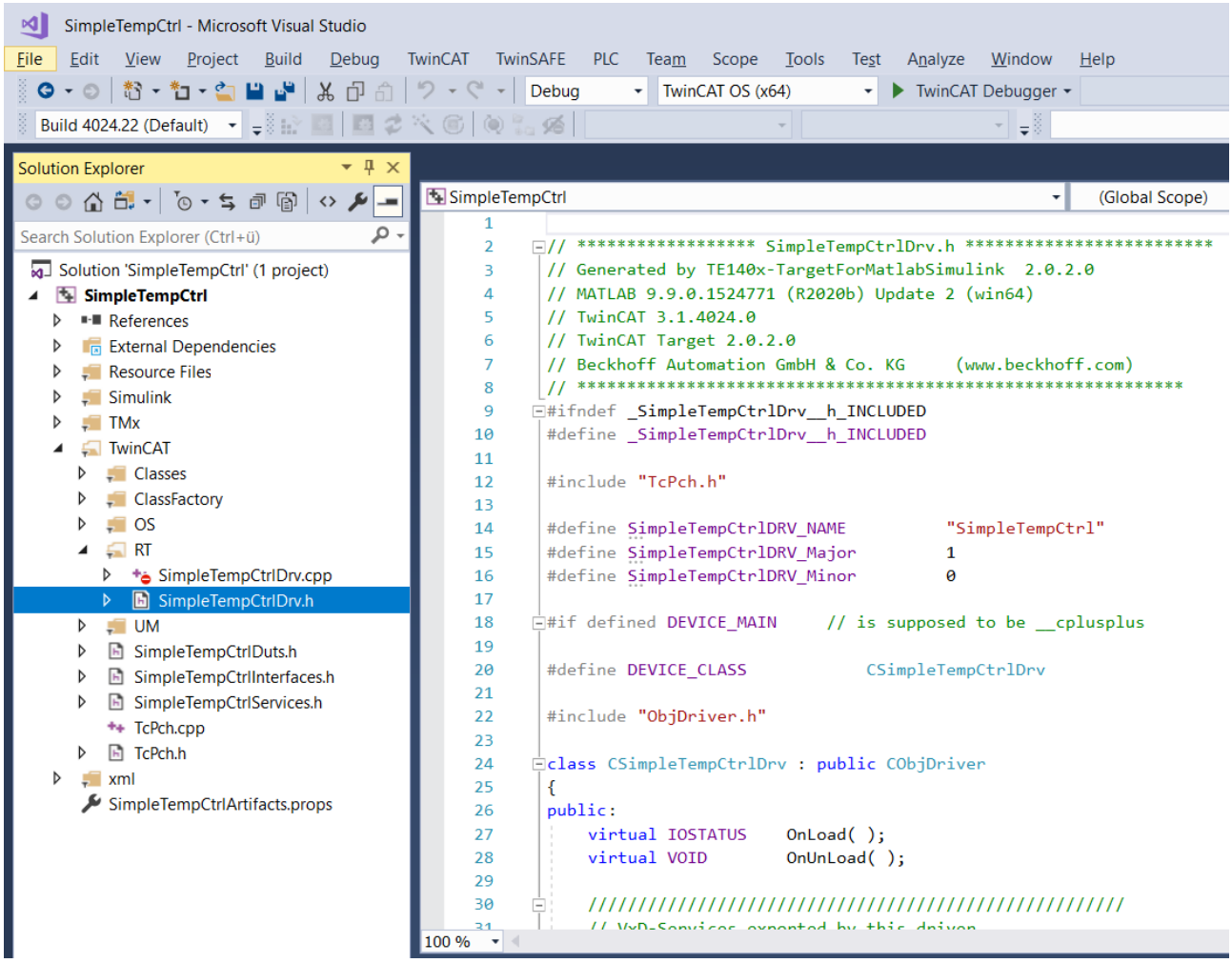
除了通过 External 模式 [▶ 142]和TwinCAT XAE 中的框图 [▶ 122]进行调试外，还可以使用创建的 C++ 项目以传统方式进行调试。

**逐步流程：**

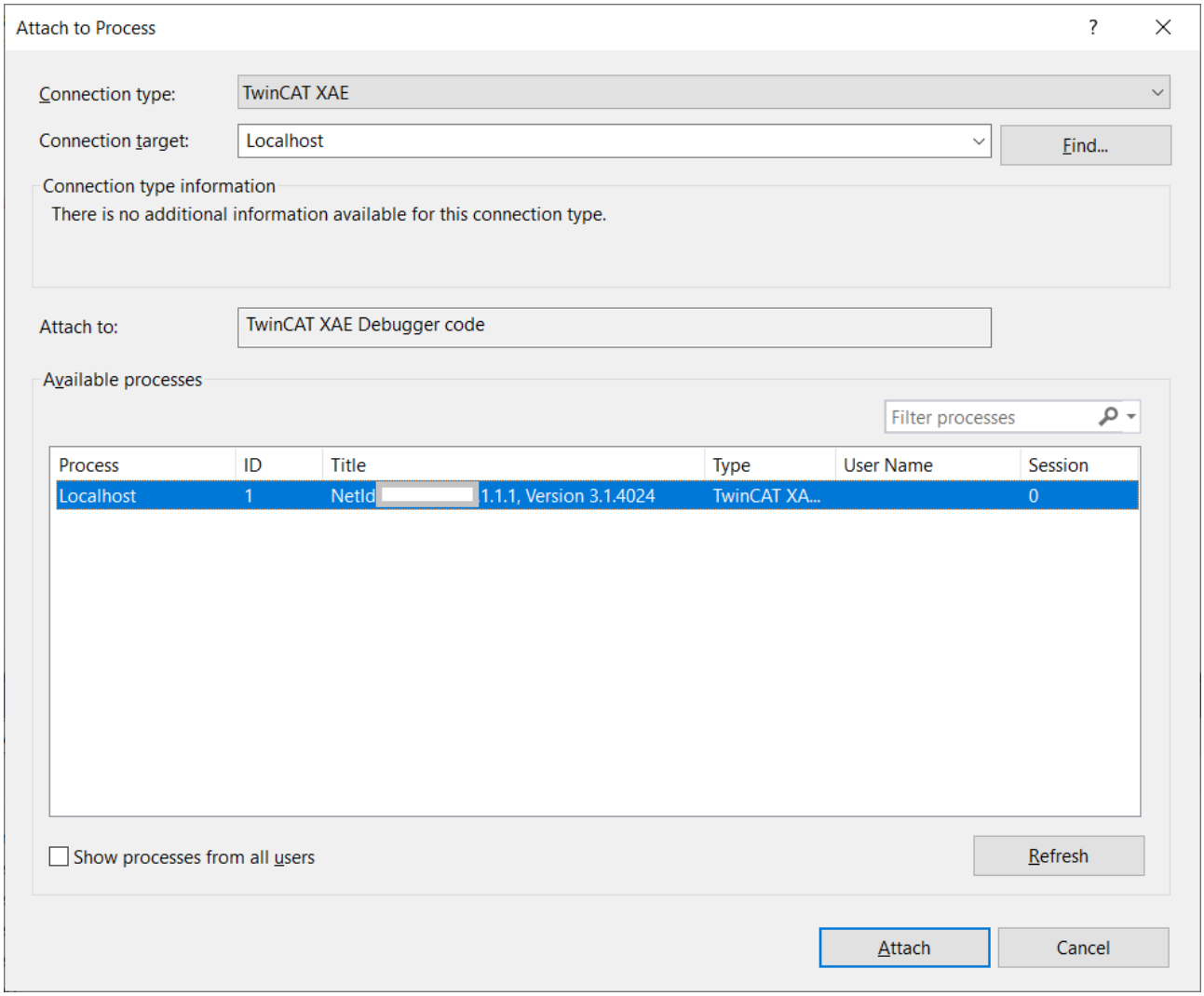
1. 确保已激活 TwinCAT 应用程序并启用 C++ 调试器。



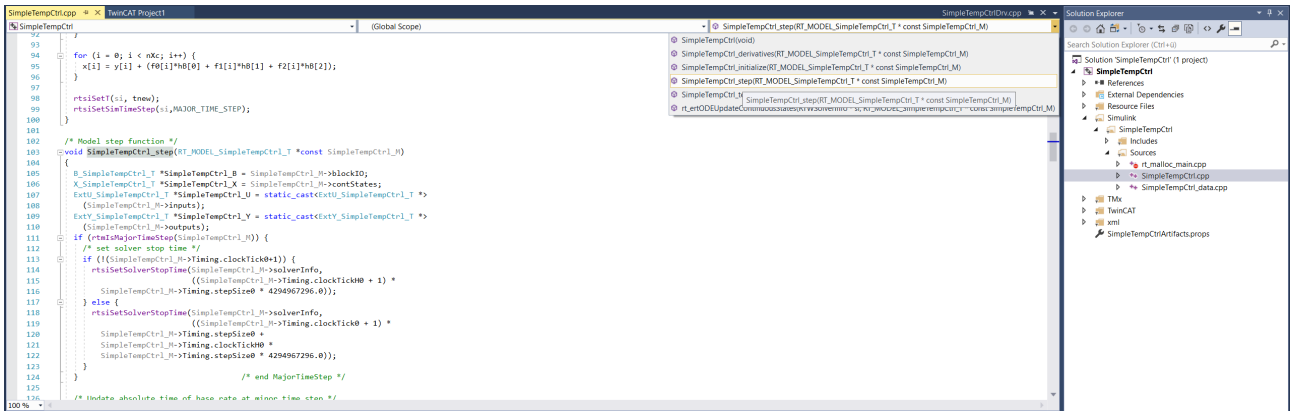

2. 打开代码生成过程中创建的 C++ 项目，该项目属于要调试的模块。
   该项目位于 *<SimulinkModelName>_tcgrt* 文件夹中，该文件夹在启动代码生成过程时在当前 MATLAB® 路径下创建。
3. 在该文件夹中搜索文件 *<SimulinkModelName>.vcxproj*。
   可以单独在 Visual Studio 中打开 *<SimulinkModelName>.vcxproj*，也可以使用“添加现有项”（Add existing Item）将 vcxproj 文件添加到 C++ 下的 TwinCAT 解决方案中。




4. 在菜单栏中选择**调试 > 附加到过程**（Debug > Attach to Process），“连接类型”（Connection Type）选择“TwinCAT XAE”，在连接目标中选择所需的目标系统。然后选择**附加**（Attach）。




5. 在 C++ 代码中设置断点，然后像往常一样逐步浏览代码。提示：执行代码时会使用阶跃函数，该函数可在 **Simulink > Sources > <SimulinkModelName>.cpp** 文件夹中找到。



## 4.8.4 连接 External 模式

通过 External 模式可以从 Simulink® 环境连接到运行中的 TcCOM 对象或 TwinCAT XAR 中的 PLC 功能块实例。

**对代码接口封装的限制**

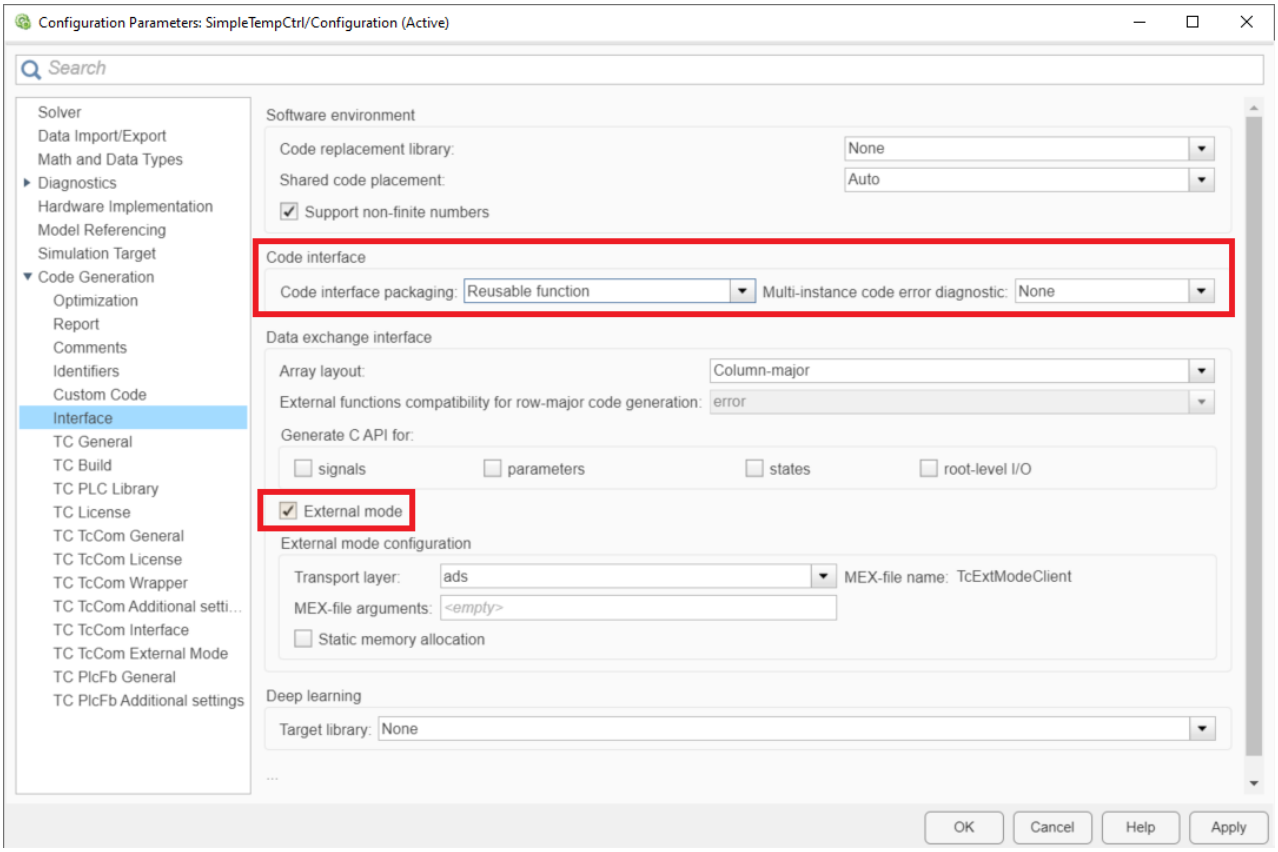
代码接口封装对 TwinCAT 中已创建类的多个实例的行为 [▶ 116]进行定义。如果要使用 External 模式，则不得设置 C++ 类。

### Simulink® 中的仿真时间设置为“inf”

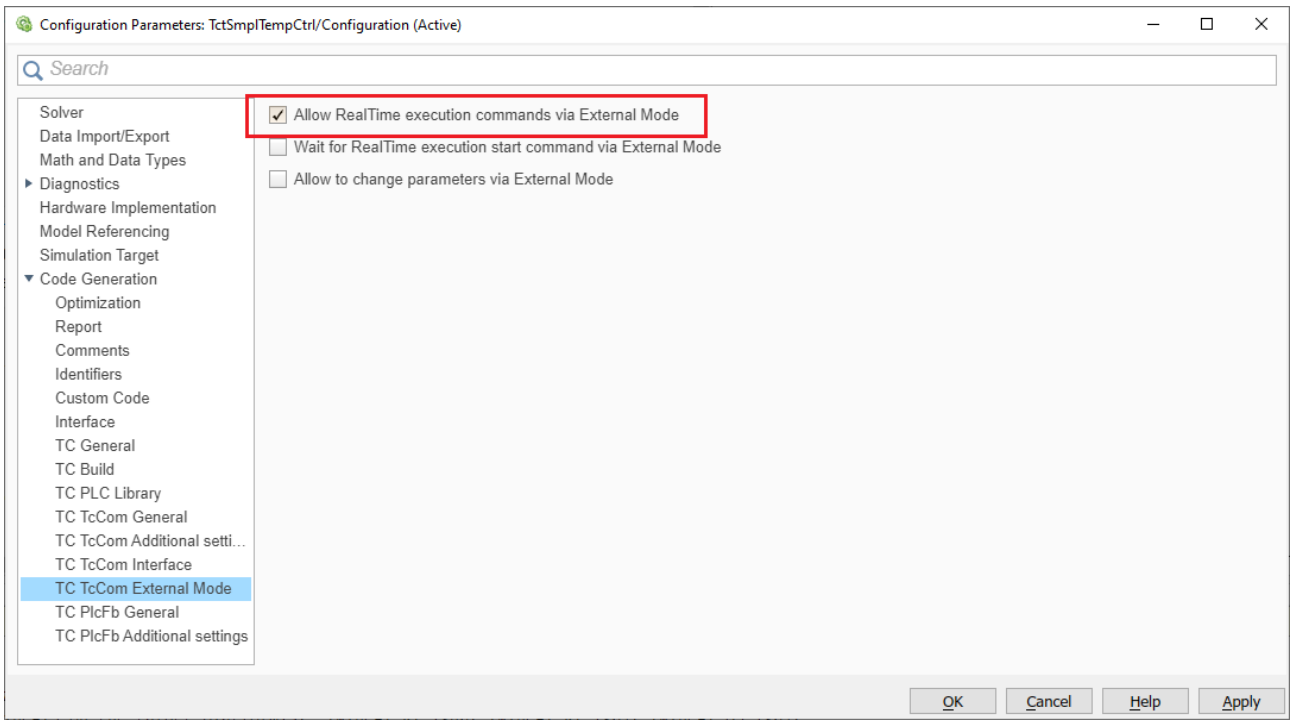
将 Simulink® 仿真时间设为“inf”。在 TwinCAT 中运行时，在规定时间后停止执行模块是无意义的。

✓ Simulink® 中的代码生成设置

1. 在**代码生成 > 接口**（Code Generation > Interface）下，设置 **External 模式**（External mode）参数。
   - 请注意**代码接口封装**（Code interface packaging）的以下几项设置：
     - 不可复用功能（Nonreusable function）：允许
     - 可复用功能（Reusable function）：允许，将“多实例代码错误诊断”（multi-instance code error diagnostic）设置为**无**（None）
     - C++ 类（C++ Class）：不允许

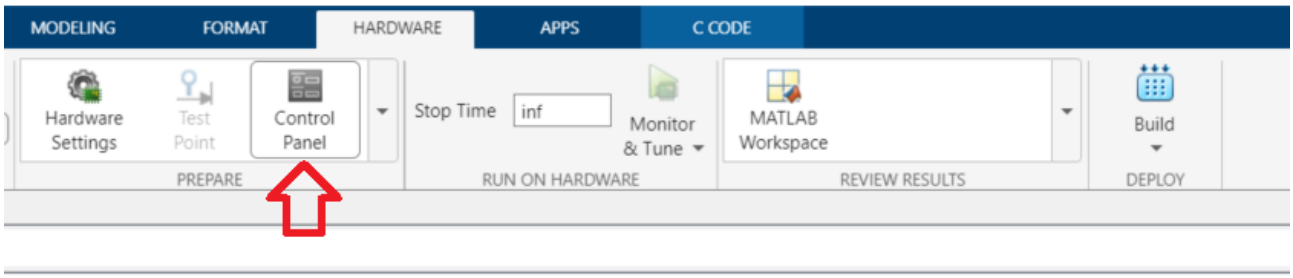


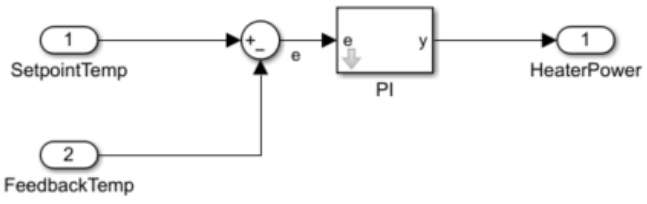

2. 定义 External 模式的权限（TcCOM 和 PlcFb 单独调整）。
   ⇨ 在以下截屏中显示为 **TcCOM** 的示例。



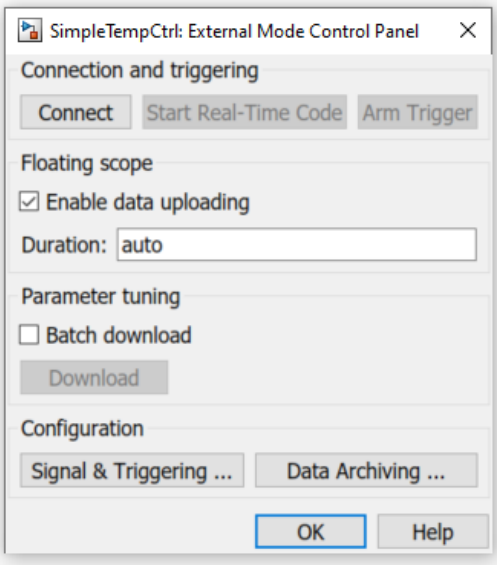

✓ 使用 External 模式连接运行时对象

3. 打开 External 模式**控制面板**。
4. 选择**连接**（Connect）。

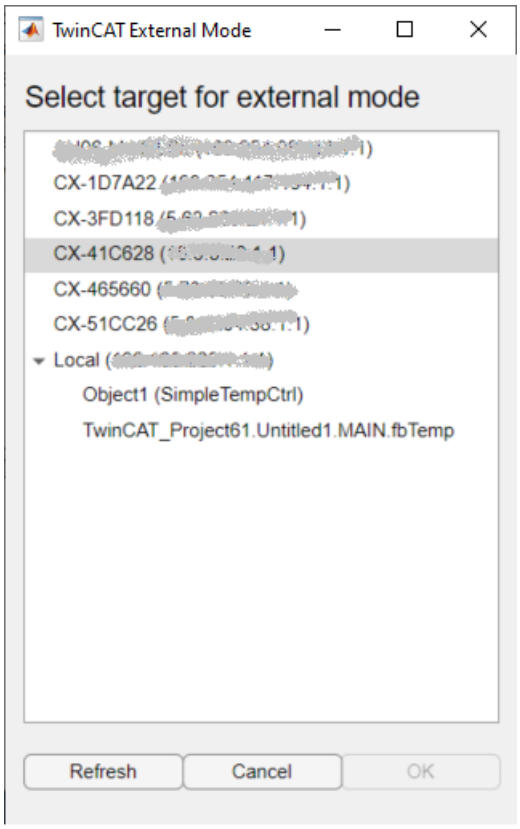



**TwinCAT Target for MATLAB/Simulink**
**Sample model "SimpleTempCtrl"**
**Subjects:**
**- Basics**



5. 选择连接的目标和对象实例。




⇨ 选择**确定**（OK）后，将连接到对象。External 模式控制面板上的**连接**（Connect）按钮已更改为**断开**（Disconnect），可以在 Simulink® 中看到从目标传输的仿真时间。

如上图所示，在目标中选择对象时，External 模式既适用于 TcCOM 实例，也适用于 PLC-FB 实例。

**需要双向 ADS 路由**

External 模式需要双向 ADS 路由。单向路由会导致通信超时。

## 4.8.5 异常处理

在 TwinCAT 中处理从 MATLAB® 或 Simulink® 自动生成的 C++ 代码时，会在运行时发生浮点异常，例如在编程过程中将一个意外值传入函数。下文将介绍如何处理此类异常。

### 什么是浮点异常?

当中央处理器的浮点运算单元指示执行算术上不完全可执行的运算时，就会发生浮点异常。IEEE 754 对这些情况进行了定义：*不精确*、*下溢*、*溢出*、*除以零*、*无效运算*。如果出现上述情况之一，则会设置一个状态标志，指示算术运算无法准确执行。此外，还规定每个算术运算必须返回一个结果 – 在大多数情况下，这个结果可能导致忽略异常。

例如，除以零的结果是 +inf 或 -inf。如果在进一步的代码中将数值除以 inf，则结果为零，因此不会出现任何相应的问题。但是，如果对 inf 进行乘法运算或其他算术运算，这些都是*无效的运算*，其结果表示为非数值（NaN）。

### TwinCAT 运行时如何处理异常情况?

**TwinCAT C++ 调试器未激活**

以下说明仅适用于 C++ 调试器在 TwinCAT 运行时系统中**未**激活的情况。启用 C++ 调试器后，异常情况会被调试器捕获并处理，请参见调试 [▶ 139]。

### 默认行为

TwinCAT 默认设置为，程序在发生除以零（divide-by-zero）和无效运算（invalid-operation）时停止执行，TwinCAT 发出错误信息。

**任务设置：浮点异常**

该默认设置可在每个 TwinCAT 任务的级别上进行更改。如果未选中“浮点异常”（Floating Point Exception）复选框，则异常**不会**导致 TwinCAT 停止，也**不会**发出错误消息。然后，该设置对该任务调用的所有对象都有效。因此，在应用中必须注意在程序代码中相应地处理 NaN 和 inf 值。

**检查 NaN 和 Inf**

例如，如果通过映射将 NaN 传递给一个已激活浮点异常的 TwinCAT 对象，则对 NaN 进行算术运算自然会导致该对象出现异常，并随后导致 TwinCAT 停止。因此，必须在映射后直接检查 NaN 或 inf。在 PLC 中，Tc2_Utilities 库中有相应的函数，如 LrealIsNaN。

**Try-Catch 语句**

处理异常的另一种方法是将异常嵌入 try-catch 语句中。在 PLC 中，指令 __TRY、__CATCH、__FINALLY、__ENDTRY 可用于此目的。如果在调用任务中启用了浮点异常，且异常发生在 Try-Catch 中，则会在 Catch 分支中捕获异常并进行处理。因此，在这种方法中，没有变量被设置为 inf 或 NaN。不过，还需要注意的是，Try 分支中的代码只运行到出现异常为止，然后跳转到 Catch 分支。在应用程序代码中，需要注意的是，Try 分支的内部状态可能并不一致。

**转储文件**

从 TwinCAT 3.1.4024.22（XAR）开始，如果 TcCOM 对象出现异常，可在运行时创建转储文件。

**对象层出现异常时的行为规范**

除了影响任务层发生异常时的行为外，还可以在 TwinCAT object 层级指定行为，即生成的 TcCOM 或生成的 PLC 功能块（PLC-FB [▶ 138]）。

在对象层，可通过 TwinCAT Target for Simulink® 实现各种操作。不过，下面介绍的所有选项基本上都是以上述原理为基础的。

**定义出现异常时的对象行为**

共有 9 种不同的设置可供选择。

- **CallerExceptions**（默认）：根据调用任务的配置触发异常。
- **ThrowExceptions**：无论任务如何配置，TwinCAT object 中的异常在任何情况下都会触发。
  - 异常会导致 TwinCAT 错误信息和 TwinCAT 停止。
- **SuppressExceptions**：无论任务如何配置，都不会触发异常。
  - 异常不会导致 TwinCAT 停止。
  - 输出或内部状态可以是 NaN 或 inf。
- **LogExceptions**：触发异常，但不会导致 TwinCAT 停止。
  - 异常不会导致 TwinCAT 停止。
  - 输出或内部状态可以是 NaN 或 inf。
  - ExecutionInfo 输出中包含关于当前周期中发生异常的信息。如果一个周期内出现多个异常，输出只显示第一个异常。再次调用 TwinCAT object 时，信息将被重置。
- **LogAndHold**：触发异常。停止执行 TwinCAT object。
  - 异常不会导致 TwinCAT 停止。
  - 输出或内部状态可以是 NaN 或 inf。
  - ExecutionInfo 输出中包含关于当前周期中发生异常的信息。如果一个周期内出现多个异常，输出只显示第一个异常。再次调用 TwinCAT object 时，信息将被重置。
  - 发生异常后，TwinCAT object 停止执行。TwinCAT 本身仍处于运行模式。重新启动执行：ReleaseObjectStop [▶ 150]。

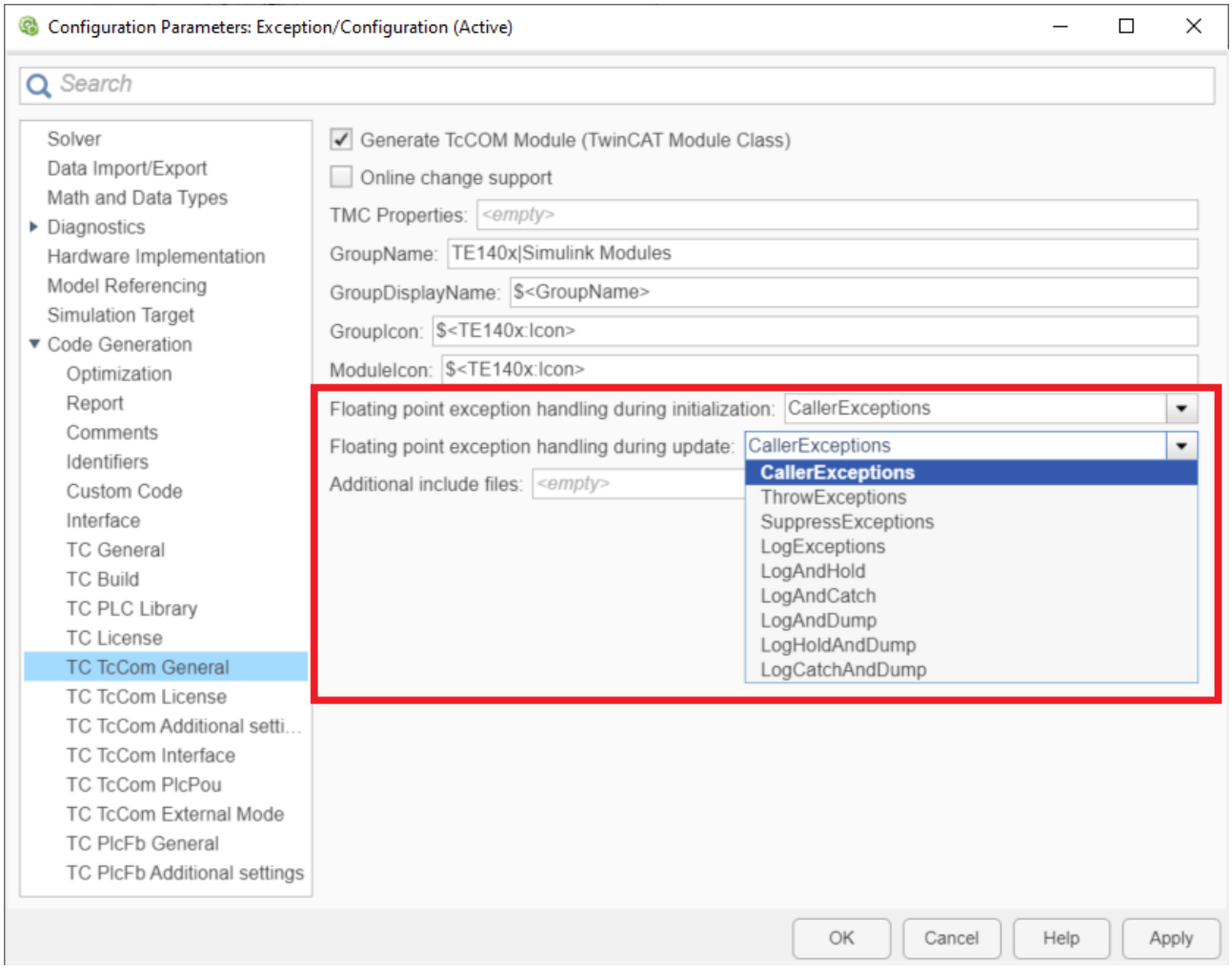
- **LogAndCatch**：在 TwinCAT 对象中使用 try-catch 捕捉异常。停止执行 TwinCAT object。
  - 异常不会导致 TwinCAT 停止。
  - 输出或内部状态**不能**包含 NaN 和 inf。
  - ExecutionInfo 输出中包含关于当前周期中发生异常的信息。
  - 代码在出现异常时结束执行。程序在此时跳转到 catch 结点，即内部状态可能不一致。
  - 发生异常后，TwinCAT object 停止执行。TwinCAT 本身仍处于运行模式。重新启动执行：ReleaseObjectStop [▶ 150]。
- **LogAndDump、LogHoldAndDump 和 LogCatchAndDump**
  - 行为与 LogExceptions 类似
  - 此外，运行时系统中的转储文件存储在 TwinCAT 文件夹 *Boot* 中。关于转储文件的更多信息，请参见这里。

**针对 64 位目标系统的版本建议**

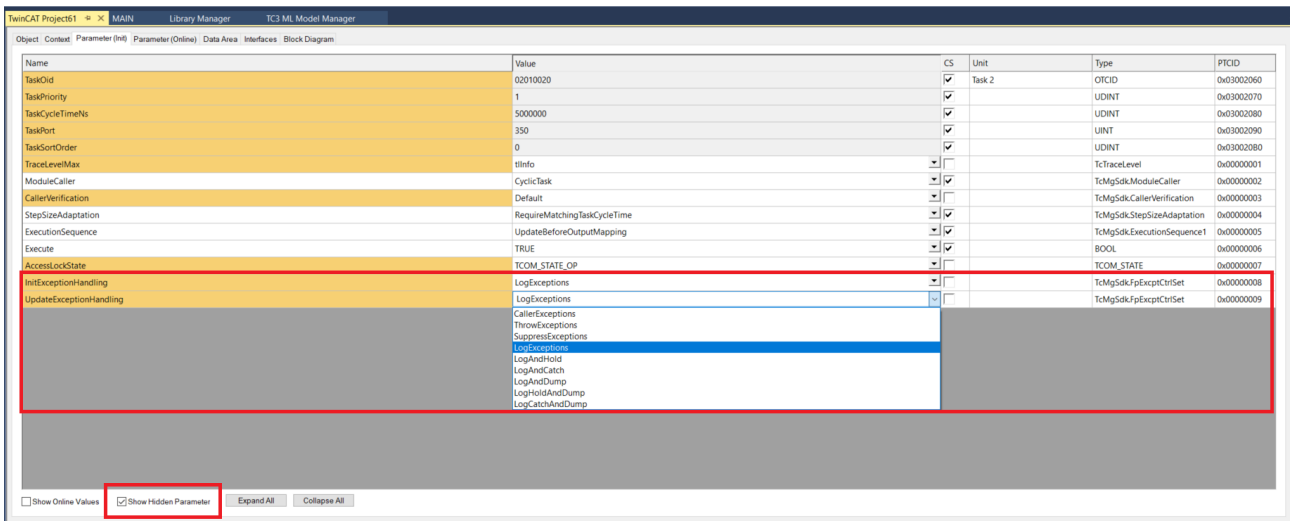
使用“LogExceptions”、“LogAndHold”、“LogAndDump”、“LogHoldAndDump”设置时，建议至少使用 XAR 版本 3.1.4024.35 和 TE1400 版本 2.4.2.0。

**<< TcCOM 的设置 >>**

可以在 Simulink®“代码生成”（Code Generation）设置中的“TC TcCOM 常规”（TC TcCOM General）下定义发生异常时 TcCOM 对象的行为。必须分别为 TcCOM 的初始化阶段和运行阶段（更新阶段）定义行为。



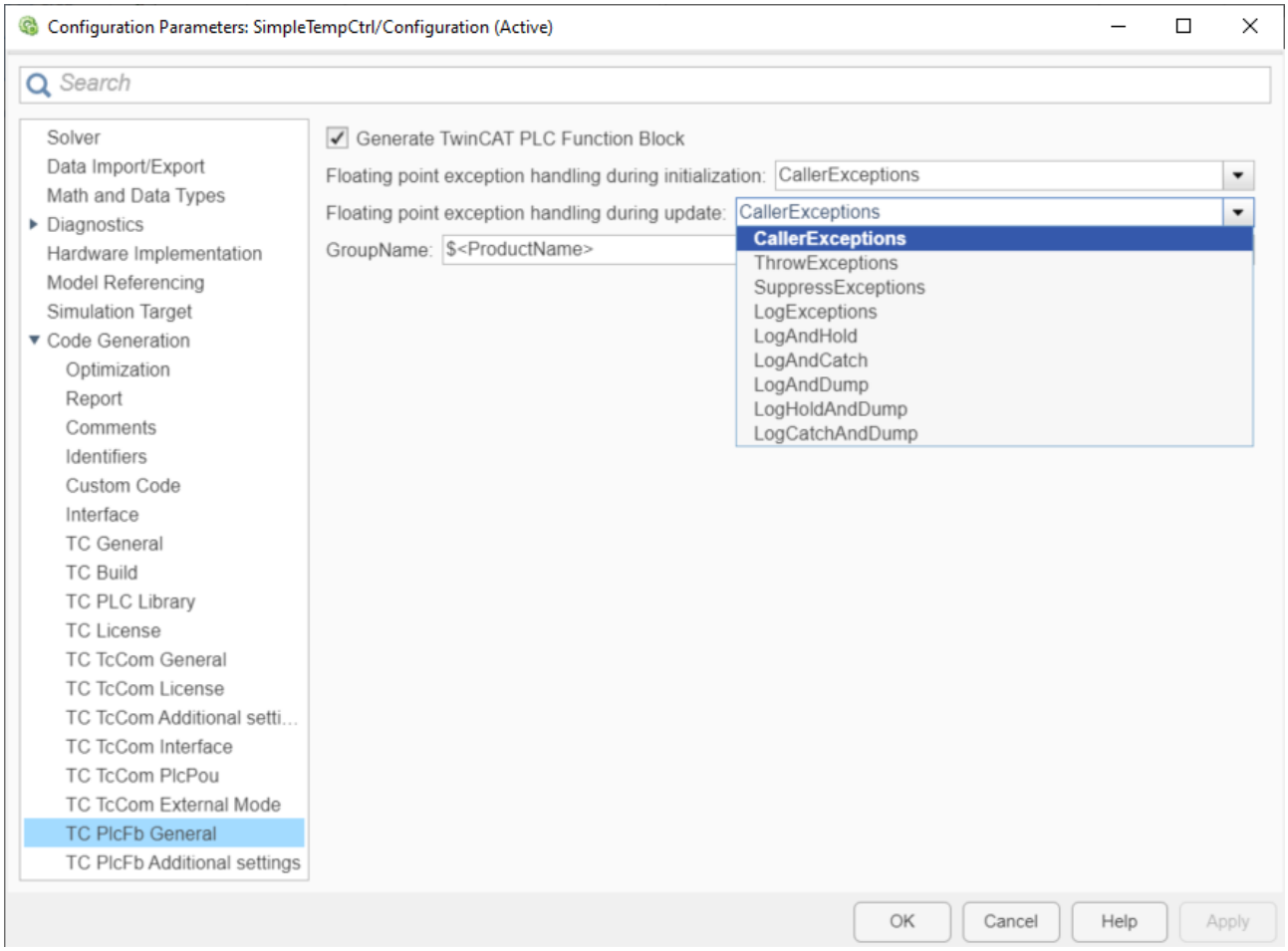

如果在 TwinCAT 中使用已编译好的 TcCOM，也可以在之后更改对象实例的设置。为此，请使用“参数（初始值）”（Parameter (Init)）选项卡并选择**显示隐藏参数**（Show hidden Parameters）。




**<< PLC-FB 的设置 >>**

必须在 **TC PlcFb 常规**（TC PlcFb General）选项卡下对 PLC-FB（PLC 库中的 PLC 功能块 FB_<ModelName>）进行独立于 TcCOM 对象的设置。在 TwinCAT 中使用功能块时，不能随后调整异常选项。

请注意，另一个 PLC 功能块 FB_<ModelName>_TcCOM 是 TcCOM 对象的封装器，因此在使用该功能块时，TcCOM 区域的异常设置生效。

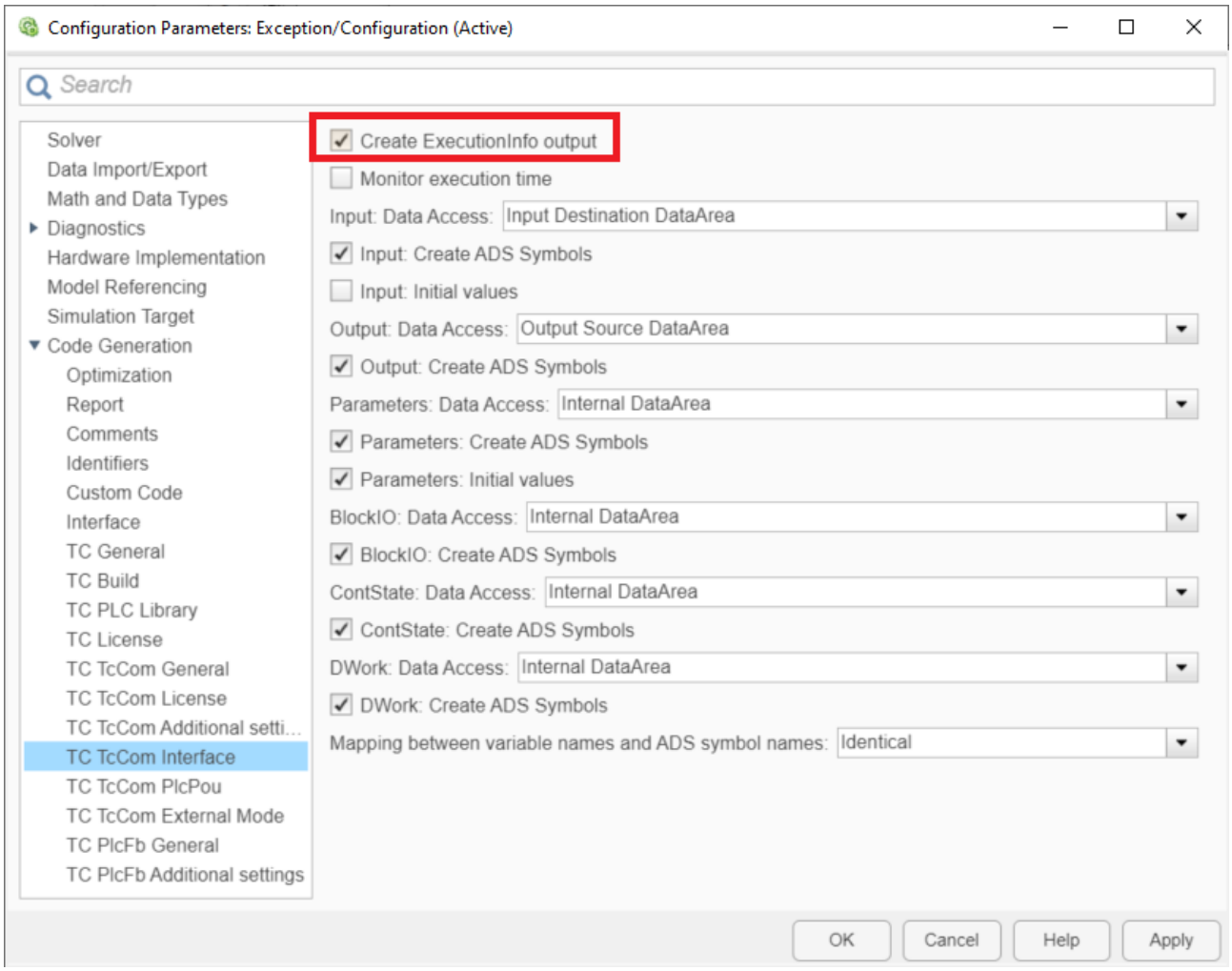



**可选的 ExecutionInfo 输出**

如果在对象层处理异常，那么在对象输出端提供与发生异常相关的相应信息也是合理的。该输出可用于查询是否发生异常、异常类型、是否已写入转储文件等。

**<< TcCOM 的设置 >>**

可以通过“TC TcCom 接口”（TC TcCom Interface）条目为 TcCOM 对象激活额外的“ExecutionInfo”输出。




ExecutionInfo 输出的结构包含以下条目：

**ExecutionInfo 结构**

| 条目 | 数据类型 | 含义 |
|---|---|---|
| CycleCount | ULINT | 当前周期计数（与异常无关） |
| ExceptionCount | ULINT | 目前发生的异常数量 |
| ActException | TcMgSdk.ExceptionInfo | 对当前异常的更详细解释（仅为当前周期中的第一个异常） |

**TcMgSdk.ExceptionInfo**

| 条目 | 数据类型 | 含义 |
|---|---|---|
| ExceptionCode | DINT | 异常代码 |
| TmxName | STRING(127) | 抛出异常的 tmx 驱动程序的名称。 |
| TmxVersion | ARRAY[0..3] OF UDINT | 抛出异常的 tmx 驱动程序的版本。 |
| InstructionAddr | UDINT | 内存中的相对地址；发生异常的位置。 |
| ReturnAddr | ARRAY[0..3] OF UDINT | 返回地址 |
| DumpCreated | BOOLEAN | 如果为异常创建了转储文件，则为 TRUE。 |

通过 *InstructionAddr* 可以判断带有给定 *ExceptionCode* 的异常是否总是出现在源代码的同一位置。如果重复异常的 *InstructionAddr* 相同，则表示总是出现在代码的同一位置。通过 *ReturnAddr* 可以看到导致异常位置的调用来自何处。这样就可以判断导致异常的调用是否总是采用相同的调用路径。如果代码是从 Tmx 驱动程序外部调用的，则 *ReturnAddr* 中的值为 0。

| 异常代码 | 含义 |
|---|---|
| 0xC000008E | 除零 |
| 0xC000008F | 结果不明确 |
| 0xC0000090 | 无效动作 |

如果使用 TcCOM Wrapper FB [▶ 134]，则可在功能块中使用 ExecutionInfo 结构。请注意，根据 TwinCAT 编程约定，条目带有与数据类型相对应的前缀。

**<< PLC-FB 的设置 >>**

根据上述定义，PLC-FB 始终包含 `nExceptionCount` 和 `stActiveException` 作为特性。也就是说，无需单独设置复选框即可获得这些特性。与 TcCOM 相比，唯一不可用的参数是周期计数，因为如果需要，PLC 本身就可以轻松实现。

### 处理 TwinCAT 对象的执行停止

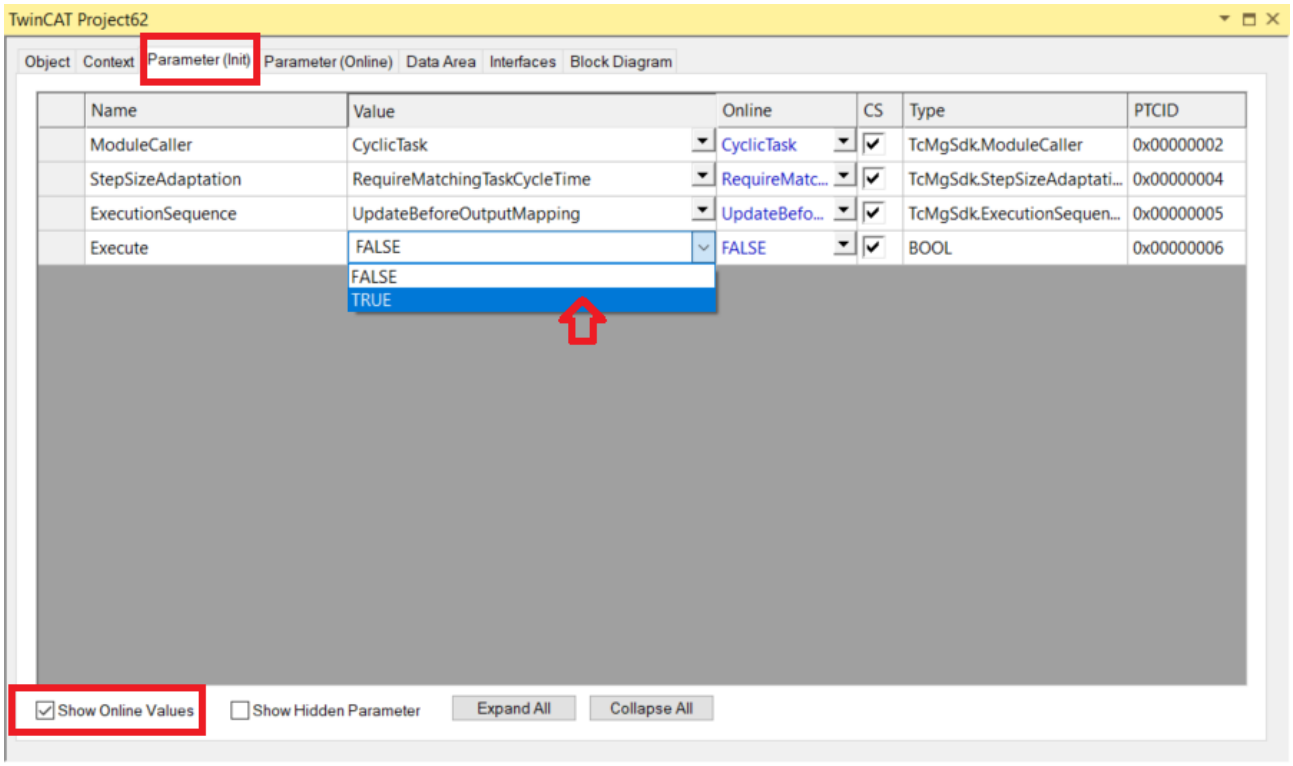
### LogAndHold 和 LogHoldAndDump

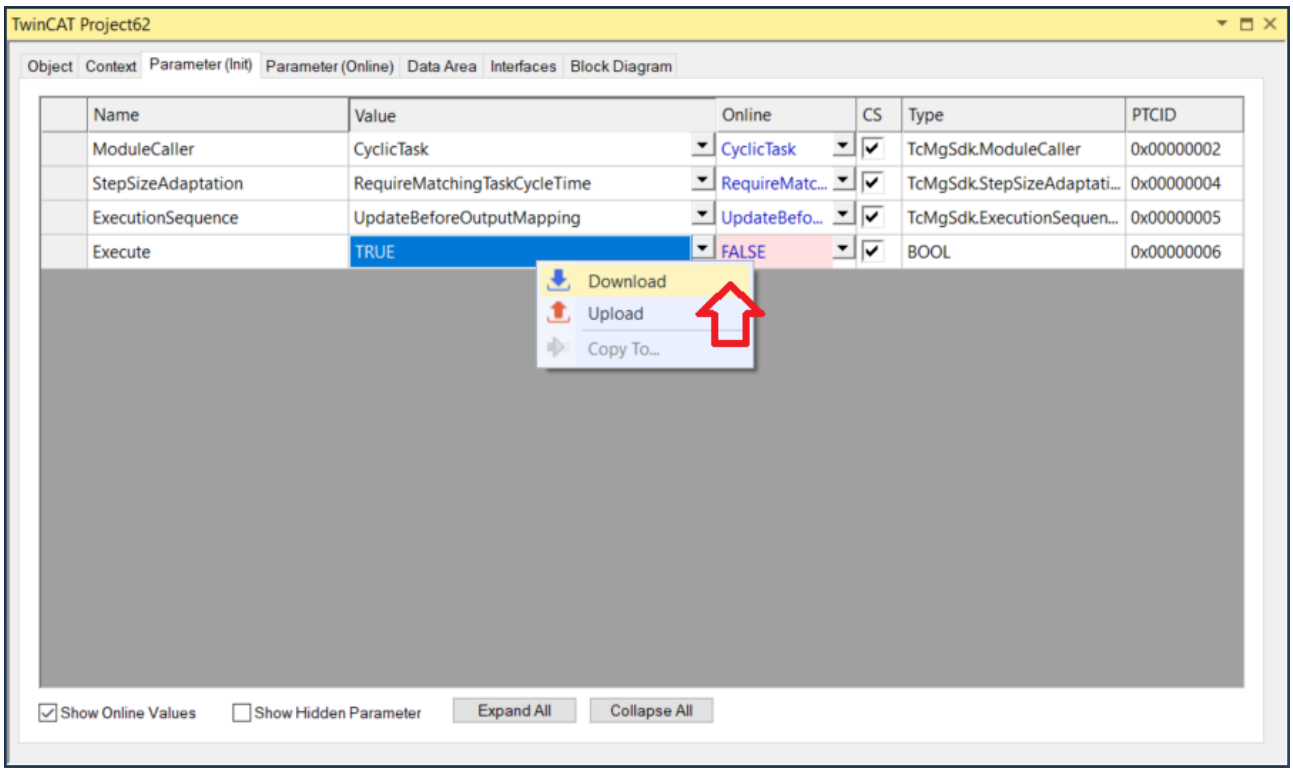
出现异常时，通过将 Execute 参数设置为“FALSE”，停止执行 TcCOM 对象或 PLC 功能块（PLC-FB）中的相关代码。

**<< TcCOM 的设置 >>**

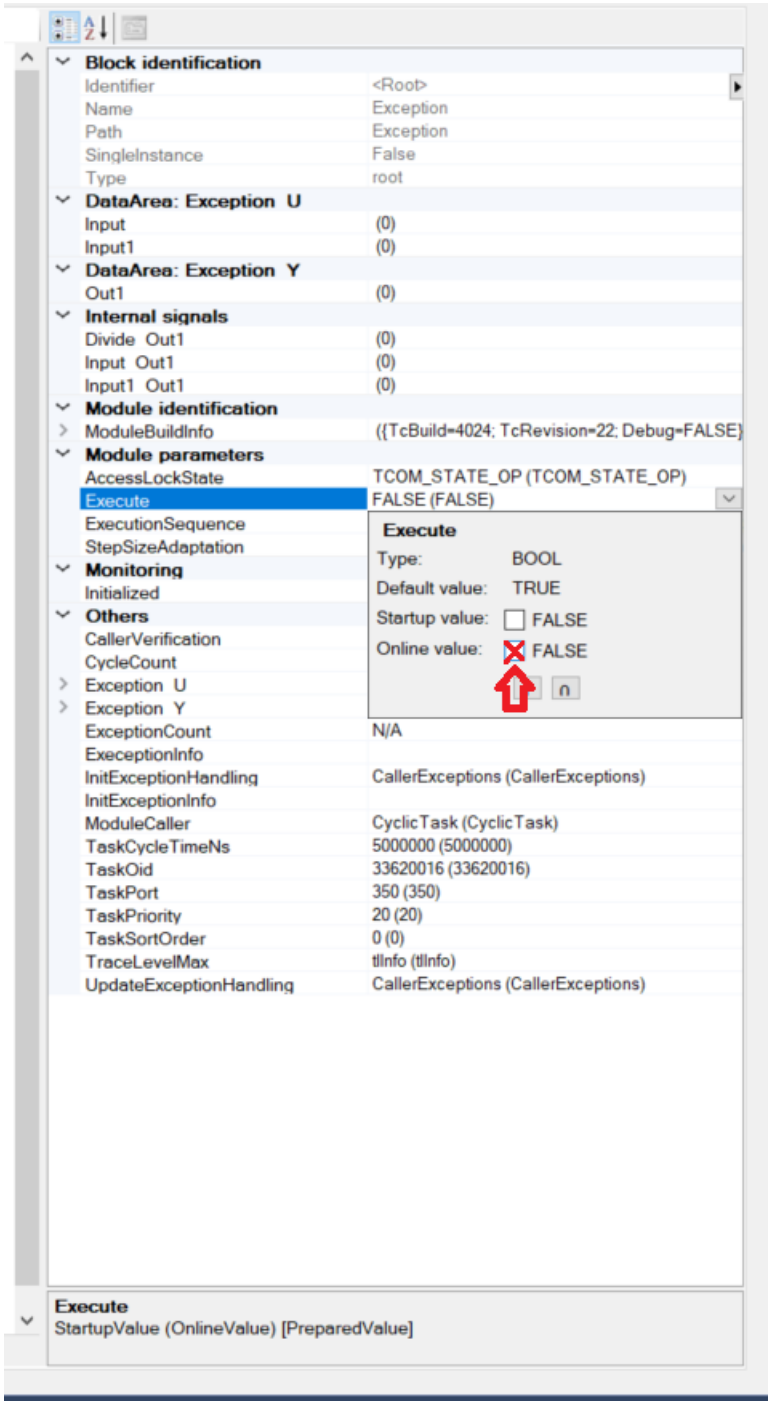
参数 Execute 可通过 XAE 和 ADS 读取或写入。

在 XAE 中，可以在参数（Init）下显示和更改 TcCOM 对象的在线值。

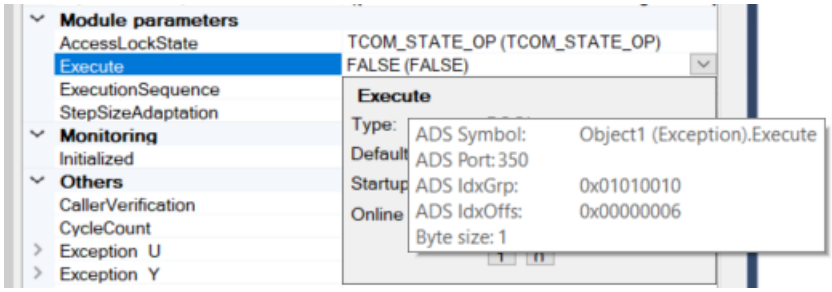




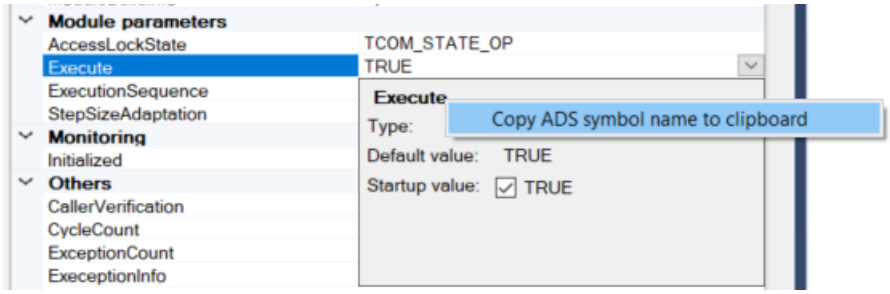

在框图中，该参数在*模块参数*（Module parameters）下提供。




如果将鼠标移至更改对话框中的执行名称上，与所有其它参数一样，将会显示该参数的 ADS 地址。这样也可以通过 ADS 设置参数。




右键单击名称 **Execute**，还可以将 ADS symbol 信息保存到剪贴板。这也适用于所有其它参数。




如果使用 TcCOM Wrapper FB，可以通过写入 bExecute 特性来更改 Execute 参数。

**<< PLC-FB 的设置 >>**

Execute 参数可在 FB 上作为具有读写权限的特性 `bExecute` 使用。使用该特性可以重新启动 FB 的执行。

**LogAndCatch 和 LogCatchAndDump**

除了参数 *Execute* 之外，在线参数 *Initialized* 也会在 LogAndCatch 和 LogCatchAndDump 的情况下变为 FALSE。模块必须经过重新初始化之后才能再次执行计算。这一点很有必要，因为在异常发生时，由于代码执行终止，因此内部状态可能会不一致。

**<< TcCOM 的设置 >>**

只能通过将 TcCOM 对象返回到“Init”状态并再次切换到 OP 状态来重新执行初始化。在运行时，只能关闭没有映射的 TcCOM 对象，否则活动映射会阻止关闭。如果 TcCOM 上有活动映射，只有重新启动整个 TwinCAT Runtime 才能进行新的初始化。因此，建议使用 TcCOM Wrapper FB [▶ 134]。它可用于从 PLC 调用 TcCOM，无需任何映射即可访问其输入和输出。因此，TcCOM 对象也可以在运行时重新初始化。

**TcCOM Wrapper FB 的特性设置**

以下读取的是示例代码特性，例如 bExecute。使用 TcCom Wrapper FB 特性组和特性的“CyclicUdate”选项创建 TcCom Wrapper FB，以便下述代码与封装器匹配。

```
PROGRAM MAIN
VAR
   stInitTemp : ST_FB_SimpleTempCtrl_TcCOM_InitStruct := (nOid := 16#01010010);
   fbTempCtr  : FB_SimpleTempCtrl_TcCOM_InitStruct(stInitTemp);
   Inputs     : ST_ExtU_SimpleTempCtrl_T;
   Outputs    : ST_ExtY_SimpleTempCtrl_T;
   ExecutionOut : ST_ExecutionInfo2;
END_VAR
```

```
// check if TcCOM is in OP mode and all set
IF fbTempCtr.bExecute = TRUE AND fbTempCtr.bInitialized = TRUE AND fbTempCtr.nObjectState = TCOM_STA
TE.TCOM_STATE_OP THEN

    // call the module
    fbTempCtr(stSimpleTempCtrl_U := Inputs, stSimpleTempCtrl_Y => Outputs, stExecutionInfo => Execut
ionOut);

    // handle exceptions
    IF ExecutionOut.ActException.ExceptionCode <> 0 THEN
       // collect exception information
       (* ..... *)

       // reinit TcCOM
       fbTempCtr.Reinit(stReInit := stInitTemp);
    END_IF

 END_IF
```

请注意，ReInit 方法是同步执行的，也就是说，根据周期时间和重新初始化所需的时间，可能会出现周期超时。

**<< PLC-FB 的设置 >>**

可以使用 FB 上的特性 `bInitialized` 来检查存储的模块是否未初始化（不再初始化）。在此处只有读取权限。目前无法通过 FB 上的方法重新进行初始化。必须重新启动 PLC Runtime，或者整个 TwinCAT Runtime。

**转储文件**

写入转储文件可能需要几个周期。最好为相关的 TcCOM 对象或 PLC-FB 单独使用一个不会阻止任何重要任务的任务。

仅当 TwinCAT XAR 版本不低于 3.1.4024.22 时才能写入转储文件，否则会收到相应的警告。

采用设置 *LogAndDump* 时，发生异常后代码会继续循环执行，但异常也相应地会循环发生，从而导致反复循环超时。因此，在转储文件写入后，参数 *UpdateExceptionHandling* 的在线值将设为 *LogExceptions*，即停用转储文件的写入，但随后可以再次启用，例如通过 ADS 或“参数（初始值）”（Parameter (Init)）下的 XAE 进行干预。

创建的转储文件存储在运行时 PC 的启动文件夹中，可从该文件夹复制到另一台 PC 进行分析。如果使用的 TwinCAT 版本低于 3.1.4024.x，可以使用 WinDbg 打开转储文件并开始分析。

### 4.8.6 使用实时监控器时间戳

诸如 tic 和 toc 等 MATLAB® 命令是分析 MATLAB® 中代码段性能的常用方法。在 TwinCAT 运行时期间，这些命令不能以这种形式使用。

为此，TwinCAT 提供了 TwinCAT 实时监控器，该监控器可以评估源代码中的时间戳，并将其显示出来以供分析。MATLAB® 代码支持设置实时监控器时间戳，即在 MATLAB® 中设置时间戳，并在 TwinCAT 中生成和实例化代码后由实时监控器进行评估。在 MATLAB® 中运行时间戳，结果会输出到 MATLAB® 控制台。

**类：TwinCAT.ModuleGenerator.Realtime.LogMark**

方法：开始、停止和标记

MATLAB® 文档：`doc("TwinCAT.ModuleGenerator.Realtime.LogMark")`

**MATLAB® 中的示例**

```
TwinCAT.ModuleGenerator.Samples.Start("BaseStatisticsLogMark")
```

时间戳仅限对 MATLAB® 代码使用，并需要适当嵌入 MATLAB® 功能块，以便在 Simulink® 中使用。

## 4.9 常见问题

### 4.9.1 运行时更改模型参数

能否在 TwinCAT 运行期间更改模型参数?

是，请注意以下设置：

- 优化 > 默认参数行为：Tunable（Optimization > Default parameter behavior: Tunable）
  如果以这种方式设置参数，则可以在运行时设置模型参数。另请参见 模块实例的参数设置 [▶ 111]。
- 接口 > 代码接口封装（Interface > Code interface packaging）
  这里有“不可重用函数”（Non-reusable function）、“可重用函数”（Reusable function）和“C++ 类”（C++ Class）选项。这些设置会影响您是否可以在 TwinCAT 中实例化多个 TcCOM 实例，以及是否可以单独或相互关联地设置其模型参数。另请参见多个模块实例的参数设置 [▶ 116]。

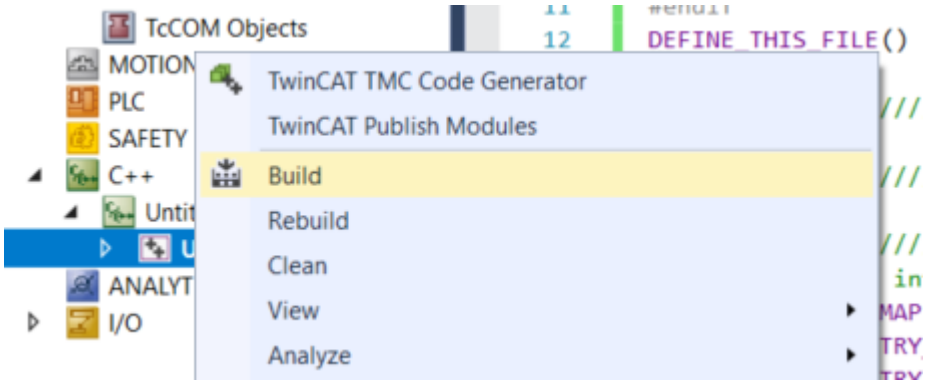
### 4.9.2 构建示例失败

提供的所有示例（在 MATLAB® 命令窗口中通过 `TwinCAT.ModuleGenerator.Samples.List` 列出）均已通过倍福自动化有限公司的测试检查。如果构建的示例仍不能成功运行，很可能是在开发用 PC 上进行设置时需要进行某些调整。

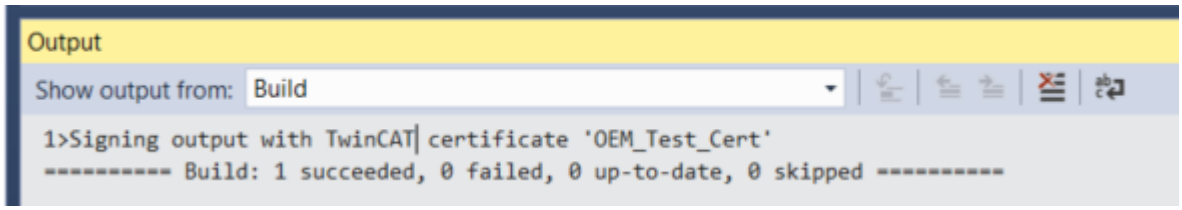
✓ 如要在不受 MATLAB® 影响的情况下测试平台工具集，请在 TwinCAT 中创建一个 TwinCAT Versioned C++ 项目（在 Visual Studio 中打开 TwinCAT）。

1. 右键单击 C++ 树项上的**添加新项**（Add New Item）。

2. 然后选择**使用 Cyclic Caller 的 TwinCAT Module 类**（TwinCAT Module Class with Cyclic Caller）。
   ⇨ 一个 C++ 项目会出现在 TwinCAT 树型结构中的 C++ 下。
3. 构建 C++ 项目并在 TwinCAT 中查看输出窗口。




⇨ 输出窗口应返回“1 succeeded”（1 已成功），表示已完成构建过程。如果不是这种情况，请检查是否在 Visual Studio 中安装了 **Desktop development with C++** 选项。




### 4.9.3 TwinCAT XAE 中的框图表示问题

使用 TwinCAT Target for Simulink® 2.x.xxxx.x 及更高版本创建的 TcCOM 模块需要 TC3 BlockDiagram 1.4.1419.0 及更高版本才能在 TwinCAT XAE 中正确显示。

在哪里可以找到 TC3 BlockDiagram 的版本?

- 在**控制面板 > 倍福 TwinCAT 3 BlockDiagram**（Control Panel > Beckhoff TwinCAT 3 BlockDiagram）的“程序和功能”（Programs and Features）下。
- 在 TwinCAT XAE 的框图中 > 右键单击窗口 >**关于 TC3 BlockDiagram**（About TC3 BlockDiagram）。

如果安装的 TwinCAT XAE 包含较早版本的 TC3 BlockDiagram：

- 是否可以安装 *TwinCAT Tools for MATLAB and Simulink* 安装程序？这其中包括一个新的 TC3 BlockDiagram 版本。
- 可以联系倍福支持部门，申请单独的 TC3 BlockDiagram 安装程序。

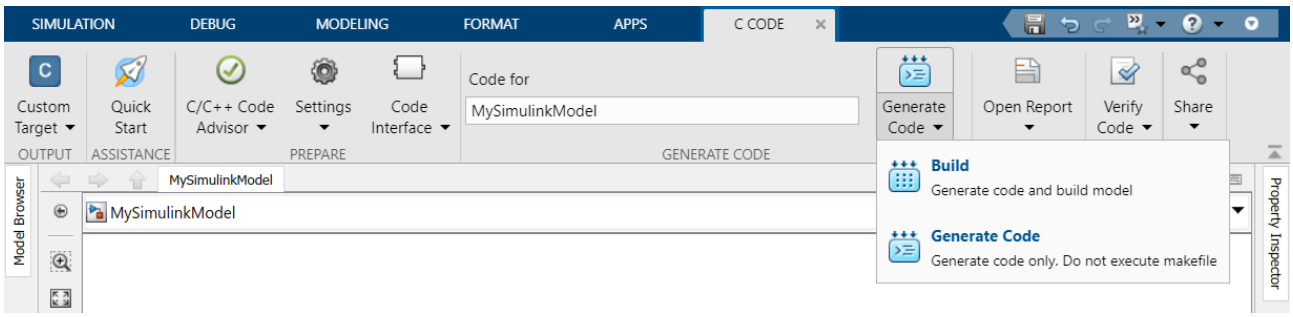
### 4.9.4 是否可以同时使用 TE1400 的 1.2.x 版本和 2.x 版本吗?

是的，可以。只需在系统上安装这两种产品，并选择适当的目标来区分两个版本即可。

*TwinCatGrt.tlc* 用于 2.x 版本，*TwinCAT.tlc* 用于 1.2.x 版本

### 4.9.5 “构建”和“生成代码”有什么区别?

在 Simulink Coder™ 应用程序中，您可以选择“构建”（Build）或“生成代码”（Generate Code）：




如果将 *TwinCatGrt.tlc* 设置为目标，则两个选项的功能相同，因为 *TwinCatGrt.tlc* 不会通过 MathWorks® 的 makefile 运行。

如果不想运行构建过程，而只想生成 C++ 代码，请取消选中 TC Build 下的**在项目生成后运行发布步骤**（Run the publish step after project generation）复选框。

**“构建”和“发布”有什么区别?**

“发布”是指在特定的 TwinCAT 平台上连续执行构建过程。通过 Simulink®，可为 TC Build 中激活的平台逐一创建相应的二进制文件，以便随后决定在哪个目标平台上使用编译后的函数。

## 4.9.6 无法在 TwinCAT 中更改模块参数

在 *TwinCarGrt.tlc* 中，将“内联”（Inlined）设置为参数 **Default parameter behavior** 的默认值。将其更改为“Tunable”或通过**配置**（configure）*按钮配置哪些参数应标记为“Tunable”*。

## 4.9.7 重新加载 TMI/TMC 时映射丢失

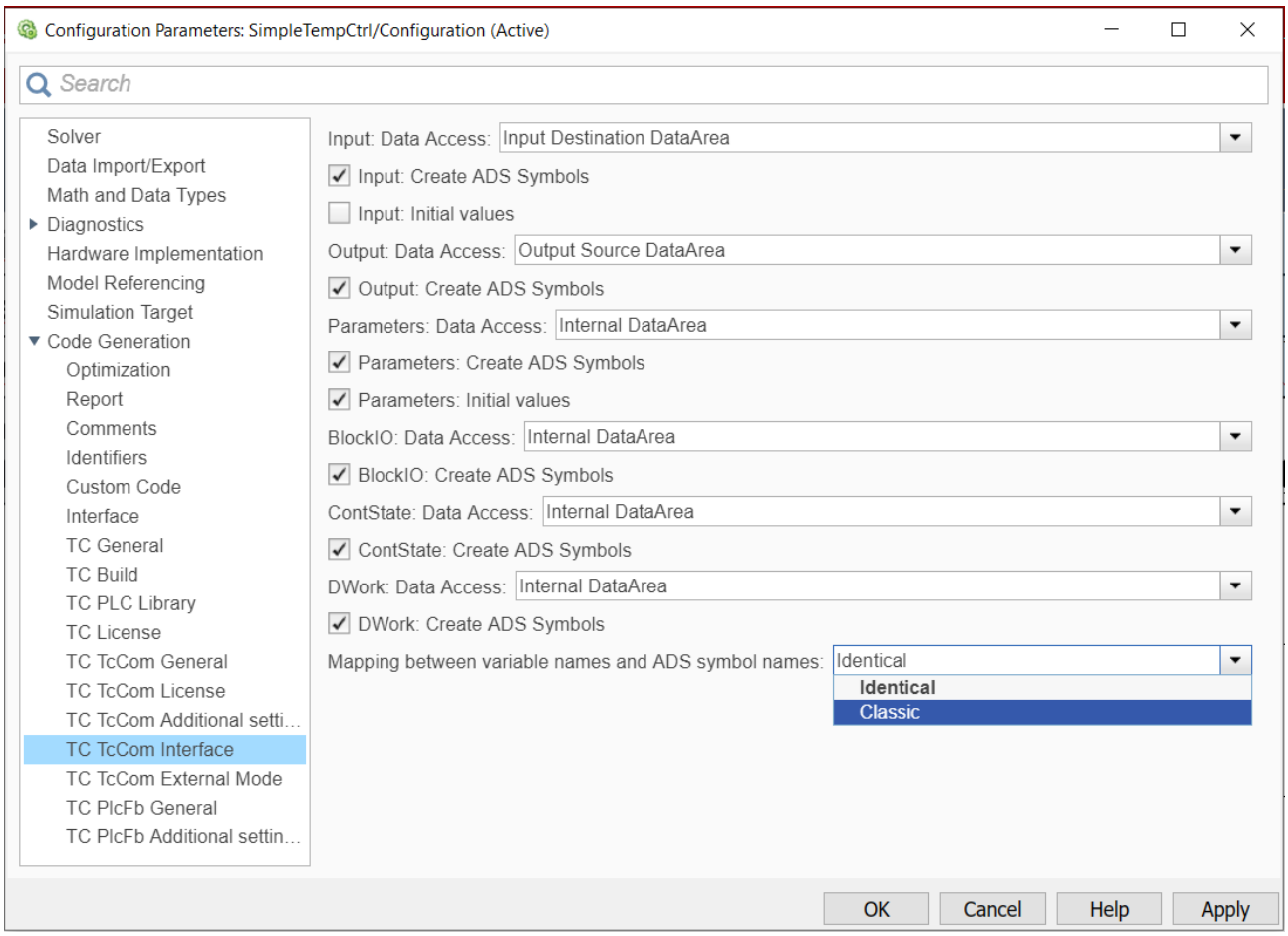
**挑战:**

您的 TwinCAT 解决方案中已经有了 TcCOM 对象，这些对象是使用 Target for Simulink® 版本 1.2.xxxx.x 创建的。现在，您需要使用 Target for Simulink® 版本 2.x.x.x 创建一个新的 TcCOM 对象，并使用“重新加载 TMI/TMC 文件...”（Reload TMI/TMC File...）替换现有 TwinCAT 解决方案中新创建的 TcCOM。

在默认设置中，这样做会丢失映射信息。

**解决方案:**

在“TC TcCOM 接口”（TC TcCOM Interface）下，将“变量名与 ADS symbol 名之间的映射”（mapping between variable names and ADS symbol name）设置为“经典”（Classic），并以此创建新的 TcCOM 对象。




这意味着，如果现在使用“重新加载 TMI/TMC 文件...”（Reload TMI/TMC File...）替换 TwinCAT 中旧的 TcCOM 对象，映射仍会保留。

## 4.9.8 在 .NET 中集成框图控件

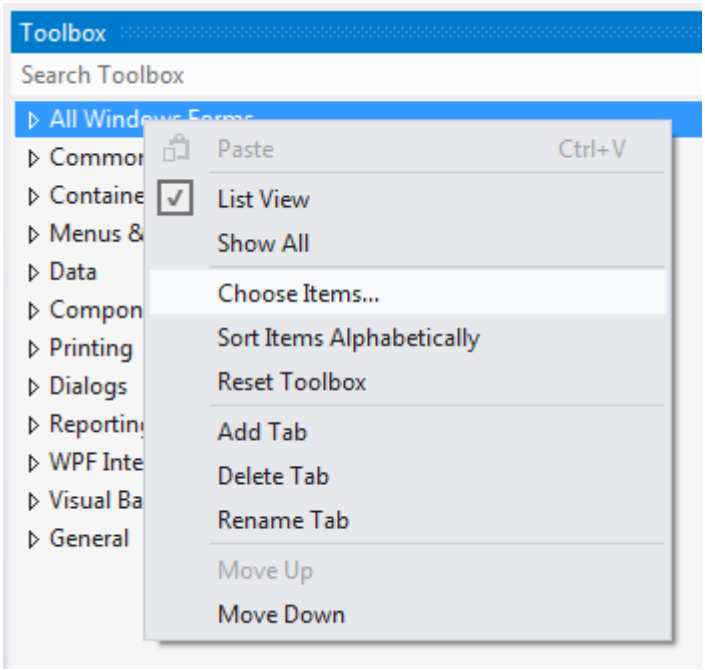
在 TwinCAT XAE 环境中显示框图的选项也可以集成到单独的可视化中。

可在此处下载示例程序：https://infosys.beckhoff.com/content/1033/te1400_tc3_target_Matlab/Resources/11697311755.zip

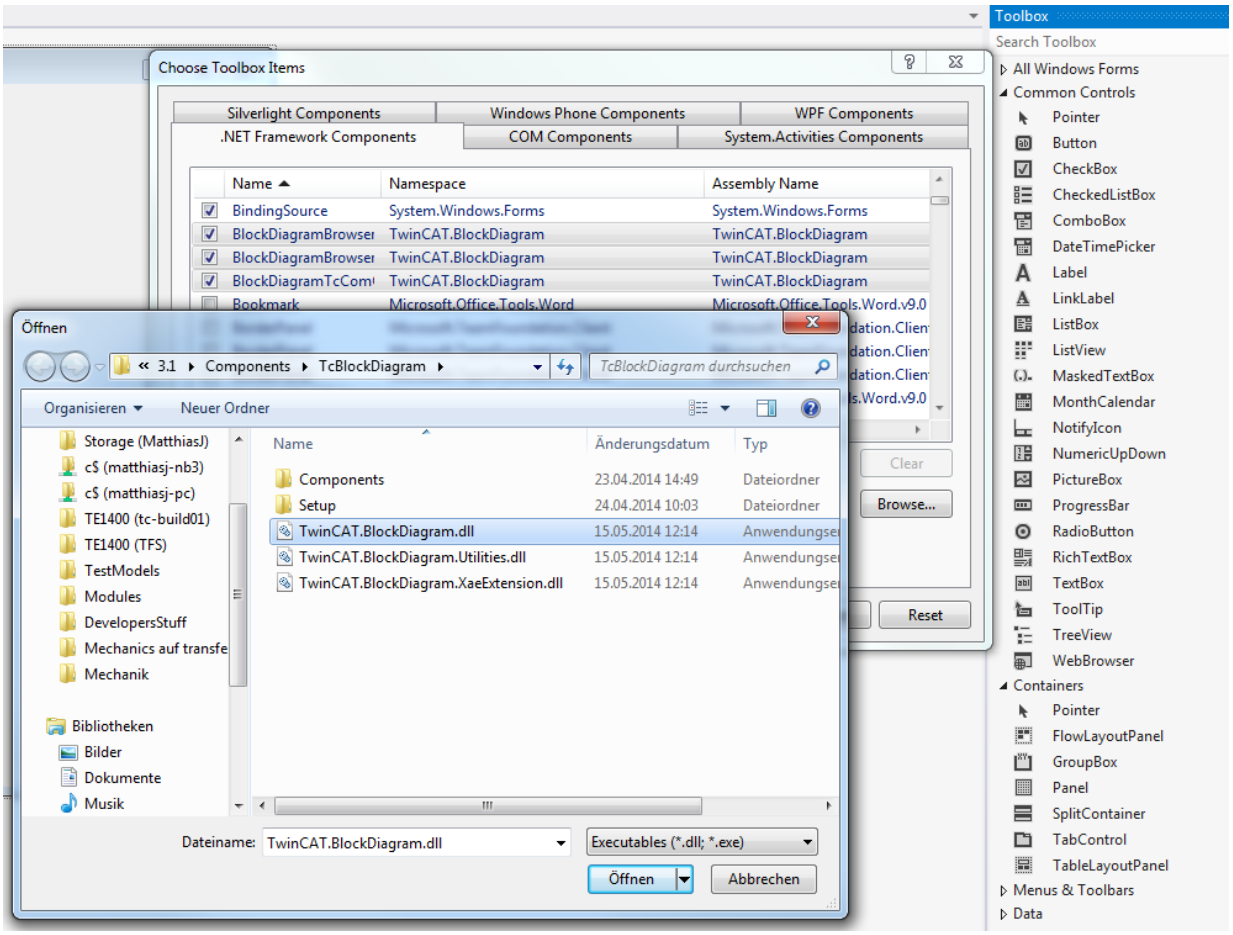
✓ 需要执行以下步骤：

1. 创建一个新的 Windows Forms 应用程序。
2. 将 TwinCAT.BlockDiagram.dll 添加到工具箱中。

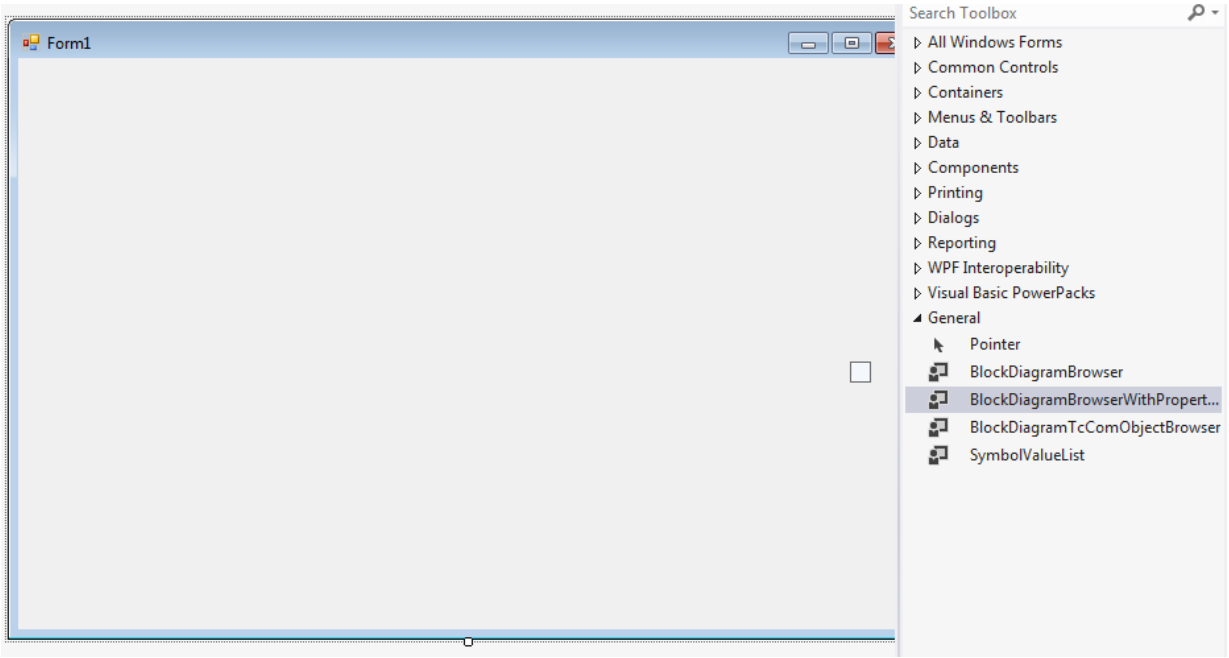
3. 为此，请在上下文菜单中选择**选择项...**（Choose Items...）条目。




4. 按照 *<TwinCAT installation path>|3.1|Components|TcBlockDiagram* 路径浏览至 TwinCAT.Blockdiagram.dll。



5. 使用拖放功能将 TcBlockdiagram 控件实例添加到 Windows Forms 对象中（BlockDiagramBrowser 或 BlockDiagramTcComObjectbrowser）。



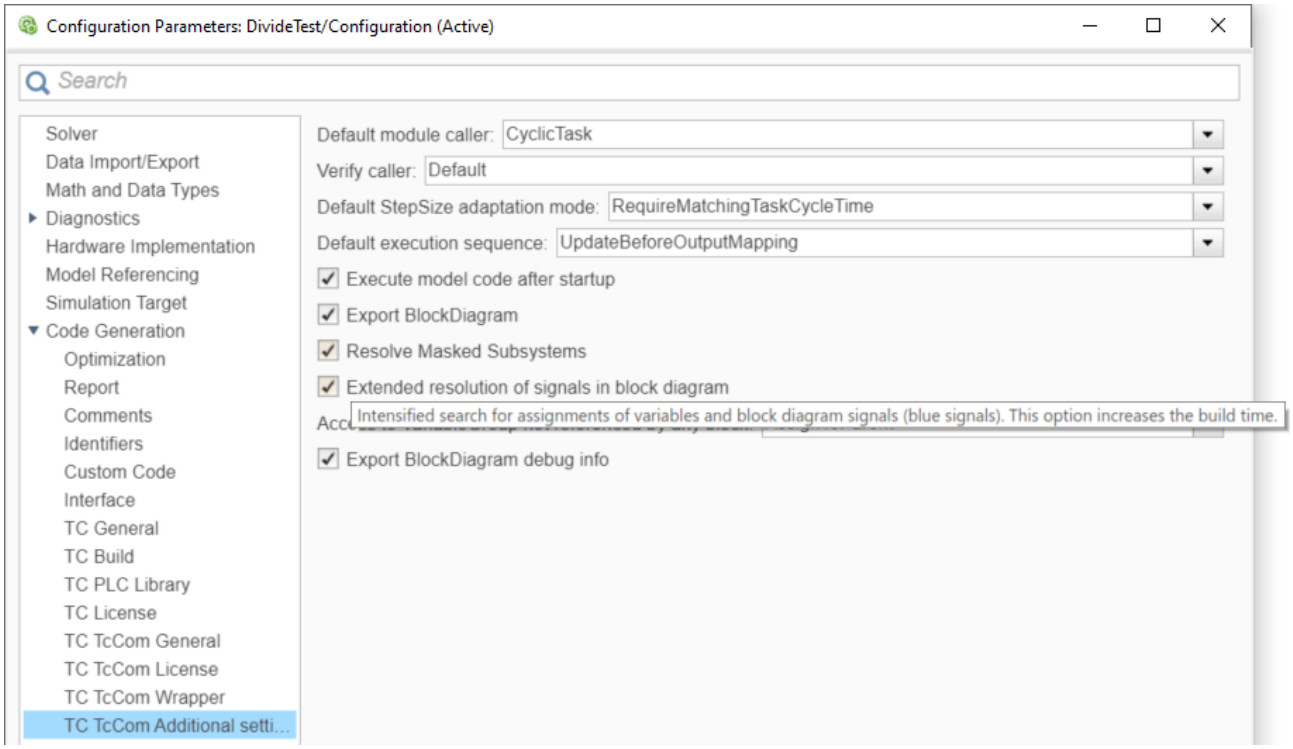

## 4.9.9 TwinCAT 框图中的可观测信号

哪些设置会影响 TwinCAT 框图中“蓝色信号”的数量?

基本上，TwinCAT Target for Simulink® 需要在 C/C++ 代码中设置一个变量，该变量可以分配给框图中的信号。如果框图中的某个信号没有以蓝色标记，则表示不存在变量（因为在生成代码时，由于代码优化而放弃了该变量），或者变量无法赋值。

如要通过代码优化抑制变量遗漏，可以使用**测试点**（在 Simulink® 中）：请参见*将信号配置为测试点*（在 Simulink® 文档中）。

在 **TC TcCom 附加设置**（TC TcCom Additional settings）中，还可以设置条目**框图中的扩展信号分辨率**（Extended resolution of signals in block diagram）。如果激活此条目，则会加强对变量和信号赋值的搜索。请注意，这种搜索需要时间，因此导出框图的时间会更长。




## 4.9.10 使用 Simulink® 字符串

Simulink® 字符串是明确允许的，可与 TwinCAT Target for Simulink® 一起使用。

### 限制

根据 MATLAB® 发布版本、代码接口封装设置和设定的 C/C++ 标准，Simulink® Coder™ 会将 Simulink® 字符串编译成 `std::string` 数据类型。

如果将此 Simulink® 字符串用作模型*输入*或模型*输出*，应注意这些条目不能通过映射与 TwinCAT 中的其他对象连接。TwinCAT 中的映射假定数据类型大小是静态的，而 `std::string` 并非如此。

**MATLAB® 中的详细示例**

本示例详细介绍了 TwinCAT Build 4024 和 4026 的主题：

```
TwinCAT.ModuleGenerator.Samples.Start('Using Simulink Strings')
```

### 使用 TwinCAT 3.1 Build 4026 进行处理

用于 TwinCAT Build 4026 及以上版本的 TwinCAT SDK 允许使用 `std::string` 实施。这意味着字符串也可以在映射中实施。

该选项有两个限制条件：

- 在 Simulink® Coder™ 生成的函数中，作为局部变量使用的 std::string 元素会增加所需的堆栈大小，因为 std::string 数据位于堆栈上。
- 编译时的最大字符串大小限值由用户选择。例如，如果生成的代码连接的两个字符串超过规定大小，就会出现异常。请相应地设计您的模型，并对其进行验证，例如通过代码检查，以确保这种情况不会发生。

使用以下设置可成功构建 Simulink® 模型：

```
set_param(modelName,'TcProject_CppLanguageStandard','stdcpp20');
set_param(modelName,'TcProject_UseStaticStlString','on');
```

根据 Simulink® 参数 *DynamicStringBufferSize* 的配置，底层字符数组的大小默认设置为 256。

如果需要，可以通过项目参数 *StaticStlStringCapacity* 独立于 Simulink® 参数 *DynamicStringBufferSize* 设置底层字符数组的大小：

```
set_param(modelName,'TcProject_StaticStlStringCapacity','255');
```

对于 TcCOM Data Area，每个 std::string 对应的 TwinCAT 类型是一个结构类型，其中有一个数据成员和一个大小成员。此外，创建的 TcCOM Data Area 必须相互兼容。如果设置字符串值，则必须同时设置数据成员和大小成员。大小是会被设置为数据值的字符数组的长度，介于 0 和 StaticStlStringCapacity 的值之间。选择大于 StaticStlStringCapacity 的值会导致未定义的行为。

PLC-FB 的可用性与 TcCOM 模块类似，std::string 是一个具有大小成员和数据成员的结构。写入字符串值需要同时设置大小成员和数据成员，而读取字符串值可能需要用户将从数据元素读取的字符串截断到合适的大小。

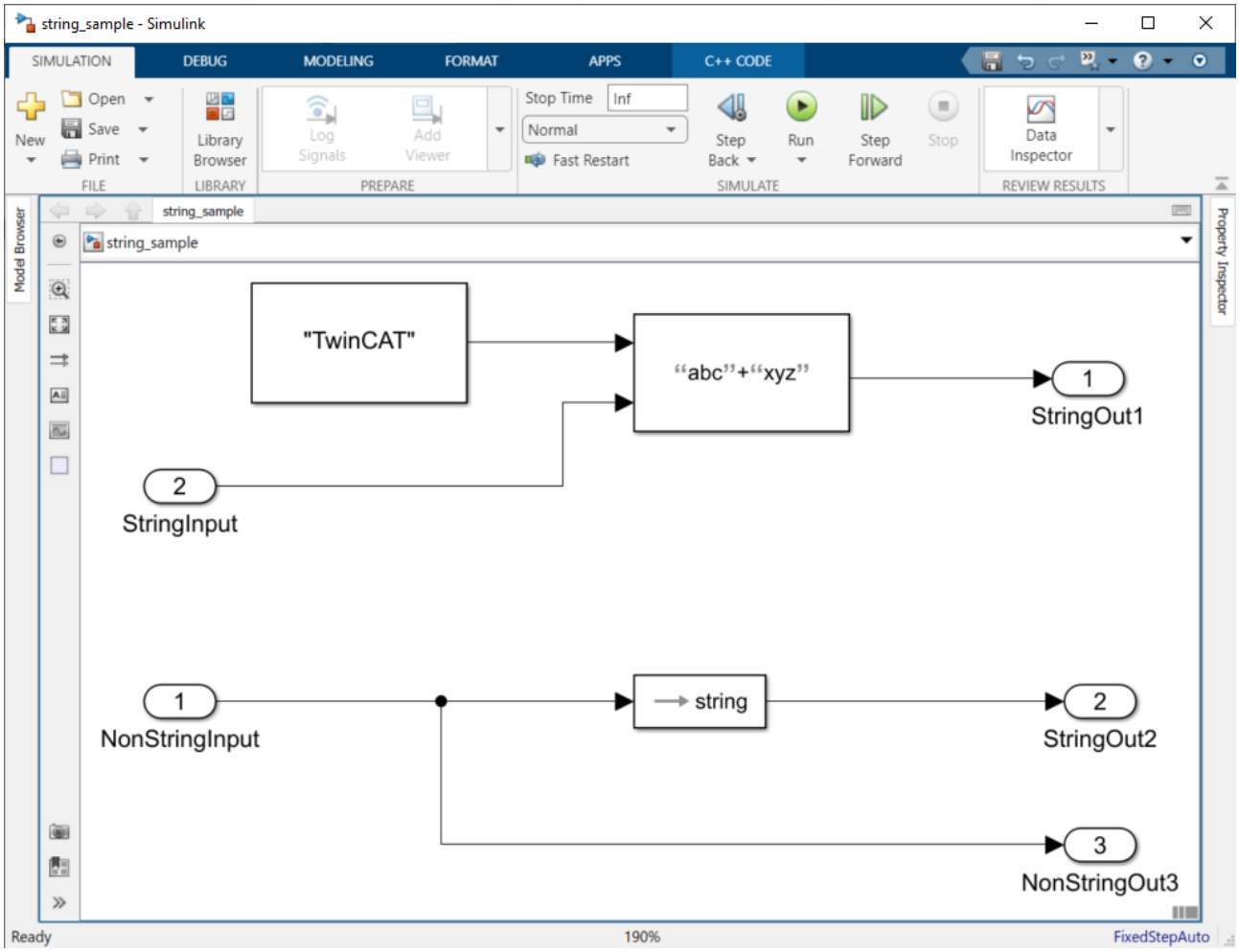
### 使用 TwinCAT 3.1 Build 4024 进行处理

为了仍能使用输入和输出，建议使用 PLC-FB（无需映射）或 TcCOM Wrapper FB。如果是 FB 的标准输入和输出，Simulink® 字符串条目在两种情况下都不会显示。需要通过 FB 上的 getter 和 setter 方法分别设置。

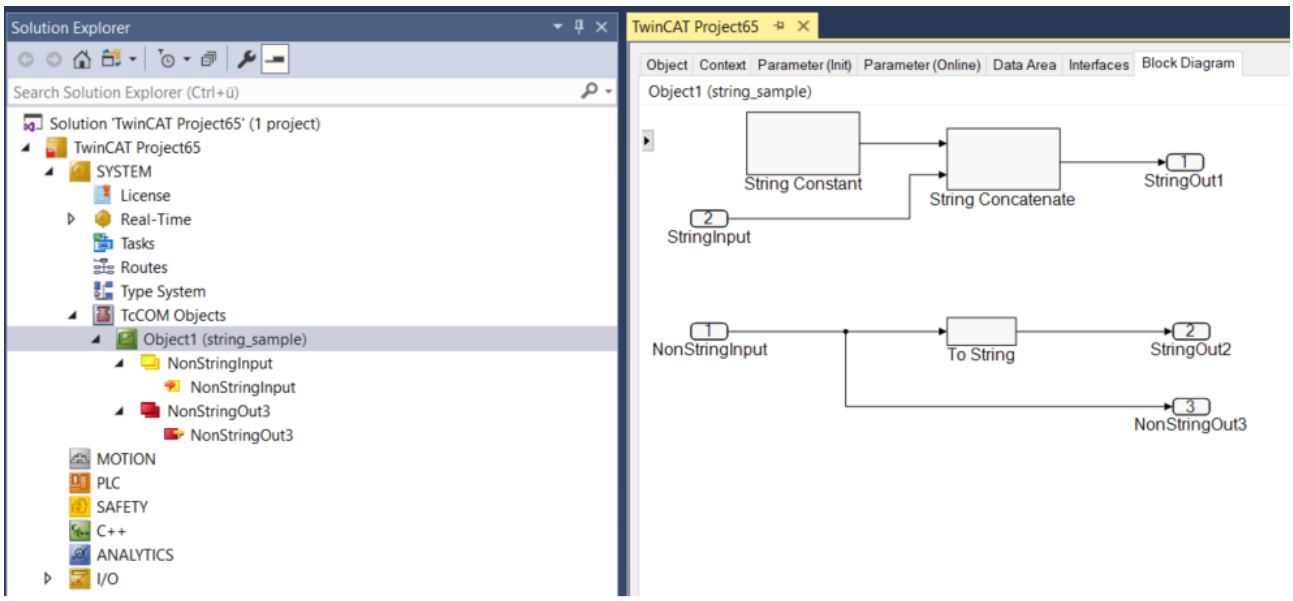
> **Simulink® 总线与 Simulink® 字符串：使用受限**
>
> 目前不支持以下情况：Simulink® 字符串如果被 Simulink® Coder™ 映射为 `std::string`，则不能在作为模型输入或输出的 Simulink® 总线中使用。

### 示例

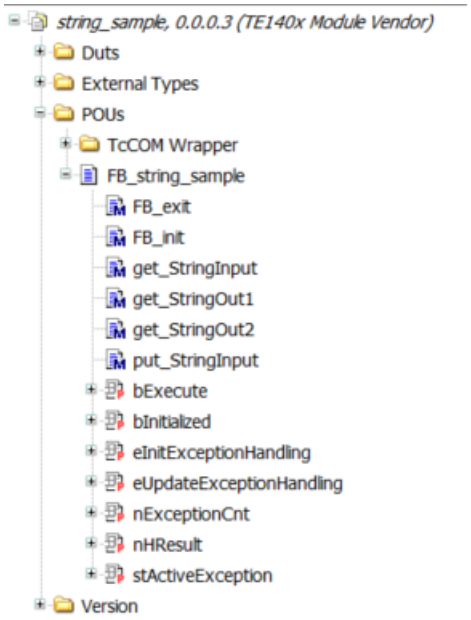
下面是字符串和非字符串混合输入和输出的 Simulink® 模型。

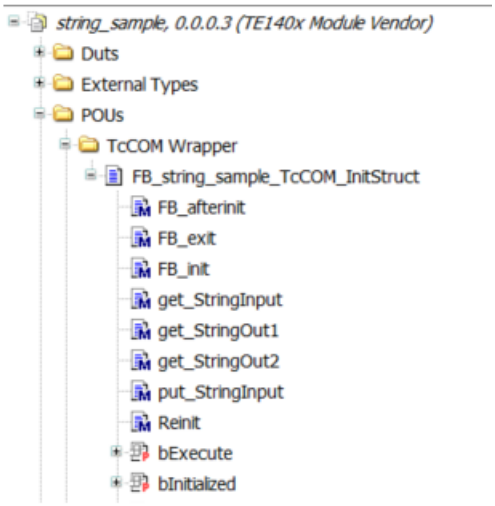



使用 MATLAB® R2022a 和“C++ 类”代码接口封装编译该模型时，数据类型 `std::string` 由 Simulink® Coder™ 生成。从下图中可以看出，TwinCAT 的过程映像中**没有**字符串输入和输出。
过程映像中只有非字符串输入和输出。




如要写入和读取字符串，必须使用 TcCOM Wrapper FB 或 PLC FB。功能块中会自动创建相应的 getter 和 setter 方法。




使用 PLC-FB 的示例代码：

```
VAR
   fbStringSample : FB_string_sample;

   myStringIn   : T_MaxString;
   myStringOut1 : T_MaxString;
   myStringOut2 : T_MaxString;
   nSize : ULINT;
   nSize2: ULINT;
   fIn   : LREAL;
   fOut  : LREAL;
END_VAR
```

```
// put sting input
fbStringSample.put_StringInput(c_str := ADR(myStringIn));

// call function
fbStringSample(fNonStringInput := fIn, fNonStringOut3 => fOut);

// get string outputs
nSize := SIZEOF(myStringOut1);
fbStringSample.get_StringOut1(c_str := ADR(myStringOut1), size := nSize);  // size in VAR IN OUT!

nSize2 := SIZEOF(myStringOut2);
fbStringSample.get_StringOut2(c_str := ADR(myStringOut2), size := nSize2);
```

## 4.9.11 实时执行模块是否有局限性?

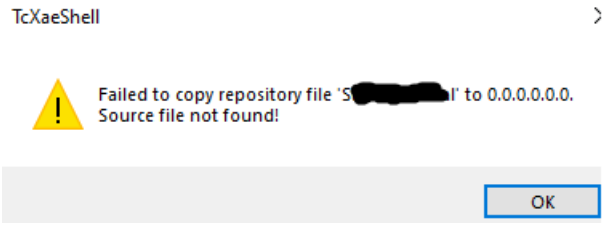
并非所有可以在 Simulink® 非实时条件下进行的访问操作都能在 TwinCAT 实时环境中进行。下文介绍了已知的局限性:

- **直接文件访问**:无法从 TwinCAT Runtime 直接访问 IPC 文件系统。如要写入 .mat 文件,请使用 TwinCAT File Writer 功能块,而不是 Simulink® 的 ToFile 功能块。
- **直接硬件访问**:直接访问设备/接口需要相应的驱动程序,例如 RS232、USB、网卡...
  无法从实时上下文访问操作系统的设备驱动程序。因此,目前还无法通过 Instrument Controller Toolbox™ 为非实时操作轻松建立 RS232 通信,然后将其直接用于 TwinCAT Runtime。然而,TwinCAT 为连接外部设备提供了多种通信选项,请参见 TwinCAT 3 Connectivity TF6xxx。
- **访问操作系统 API**:不能直接从 TwinCAT Runtime 使用操作系统的 API。例如,在 C/C++ 代码中集成 *windows.h*。例如,如果使用 DSP Systems Toolbox™ 中 FFT 功能块的 FFTW 方法(但不使用 Radix 2 方法),Simulink Coder™ 将集成这一方法。
- **预编译库**:在 Simulink Coder™ 生成代码期间,可能没有生成独立于平台的 C/C++ 代码,但包含了预编译库。在这种情况下,无法在 TwinCAT 中实时执行。

## 4.9.12 信息:复制存储库失败

激活 TwinCAT 配置后,TwinCAT XAE 中会出现以下错误信息:

复制存储库文件失败...(Failed to copy repository file...)




原因通常是工程存储库中没有文件或没有找到所有文件,尤其是 TMX 驱动程序。TwinCAT XAE 试图将驱动程序复制到 XAR 系统,但没有找到正确的文件(检查工程存储库中是否有用于所选 XAR 系统目标平台的驱动程序)。

XAE "错误"(Error)对话框中产生的错误示例:

*'TCOM Server' (10): Error loading repository driver 'C:\TwinCAT\3.1\Boot\Repository\<model vendor>\<model name>\<version>\<tmx-name>' - hr = 0xc0000225*

由于 TMX 文件无法复制到 XAR 系统,因此无法加载。产生的另一个错误是链接错误:"无法链接外部功能"(Could not link external function)。

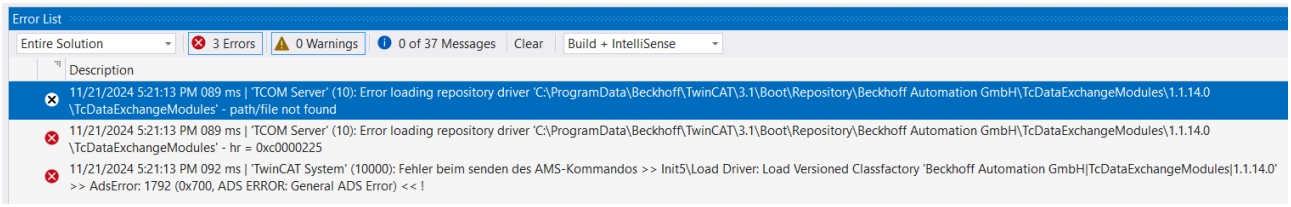
✓ **有关此错误模式的信息和补救措施:**

1. 什么是工程存储库?在哪里可以找到它?
   ⇨ 请参见自动创建的文件 [▶ 40]。
2. 如何确保将已编译的模型复制到其它 XAE 系统时,始终传输所有必要的文件,并且文件夹结构保持正确?
   ⇨ 参见 TMX 压缩包 [▶ 57]。

## 4.9.13 无法加载 DataExchange 模块

使用 Simulink® 生成的对象激活配置时出现**错误图像**:

加载存储库驱动程序时出错 ..... TcDataExchangeModules .....




**解决方案：**

在某些情况下，需要在开发用 PC 上安装 **DataExchange modules**，以便使用在另一个系统上创建的 TwinCAT objects。

这些情况包括：

- 创建模块时使用了“External Mode”选项。
- 创建的模块使用 TwinCAT File Writer [▶ 35] 或 MAT 文件记录。
- 创建的模块包含来自 TE1410 TwinCAT Interface for MATLAB®/Simulink® 的功能块。

在其他情况下，创建的 TwinCAT 对象与 DataExchange 模块没有依赖关系。

**安装 DataExchange 模块**

TwinCAT 3.1 Build 4026

```
tcpkg install TwinCAT.XAE.TMX.DataExchange
```

TwinCAT 3.1. Build 4024

DataExchange-Modules 安装程序随着 MATLAB® 和 Simulink® 安装程序专用 TwinCAT 工具复制到以下文件夹，这样安装了 TE14xx-ToolsForMatlabAndSimulink 安装程序的员工便可将 DataExchange 模块安装程序分发给相关同事。

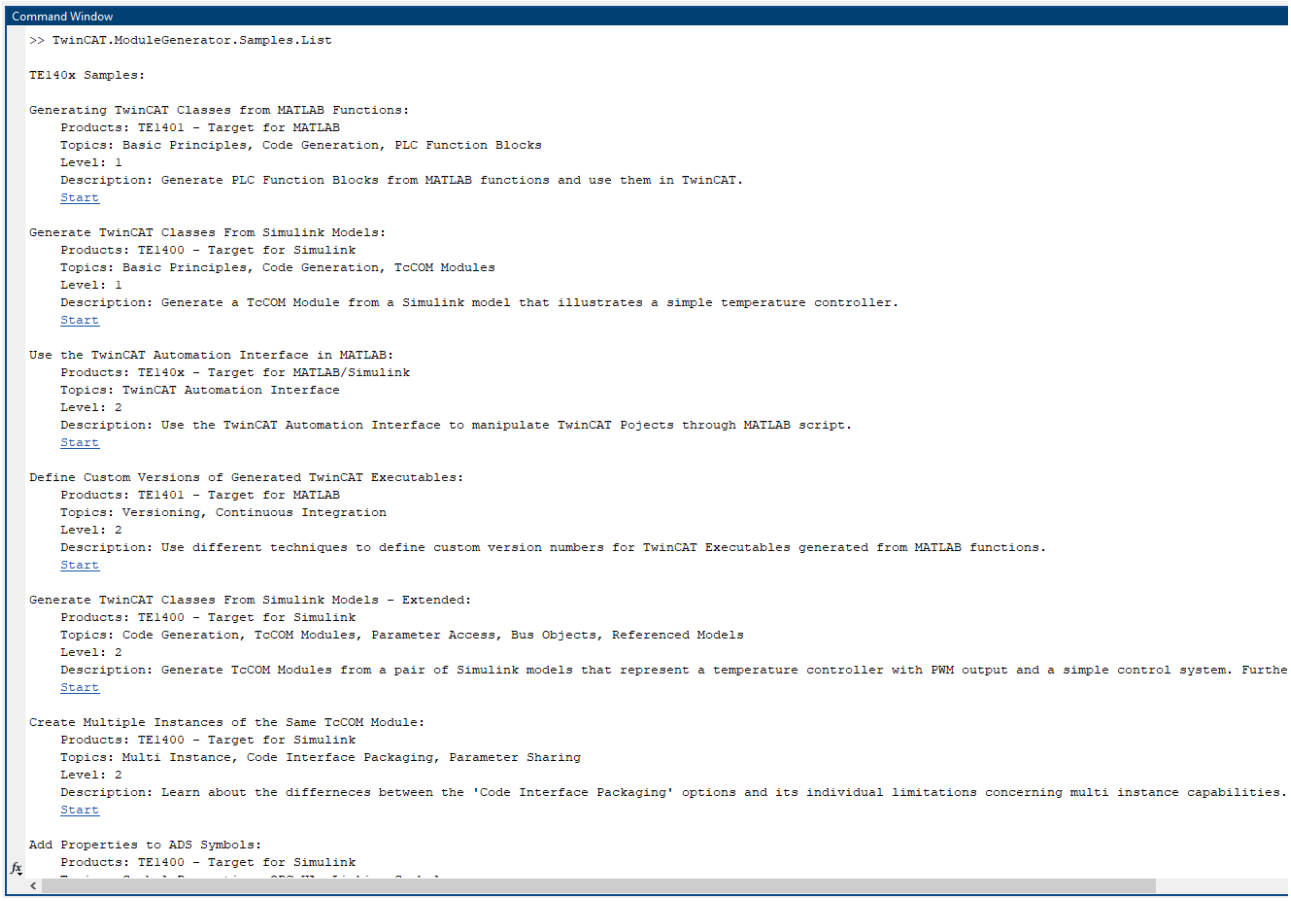
```
<TwinCATInstallDir>\TwinCAT\Functions\TE14xx-
ToolsForMatlabAndSimulink\TE140x\SDK
```

## 4.10 示例

Beckhoff Automation 提供的示例随设置安装在您的系统上。

可以使用以下命令显示所有可用示例：

```
TwinCAT.ModuleGenerator.Samples.List
```




单击蓝色的开始链接即可获取示例。为此，会将示例代码复制到用户目录，这样就不会更改原始示例。可以相应地使用示例副本并进行尝试，

还可用于显示和启动单个示例：

```
TwinCAT.ModuleGenerator.Samples.Show(SampleName)
TwinCAT.ModuleGenerator.Samples.Start(SampleName)
```

参数 SampleName 以字符串形式传递，例如：

```
TwinCAT.ModuleGenerator.Samples.Start("Generate TwinCAT Classes From Simulink Models")
```

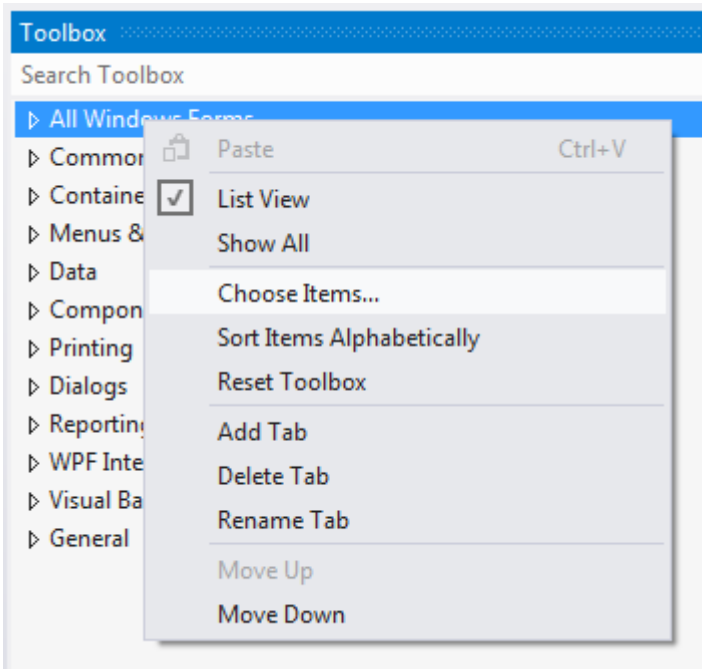
## 4.10.1 集成框图控件

在 TwinCAT XAE 环境中显示框图的控件也可以作为控件集成在自己的可视化系统中。

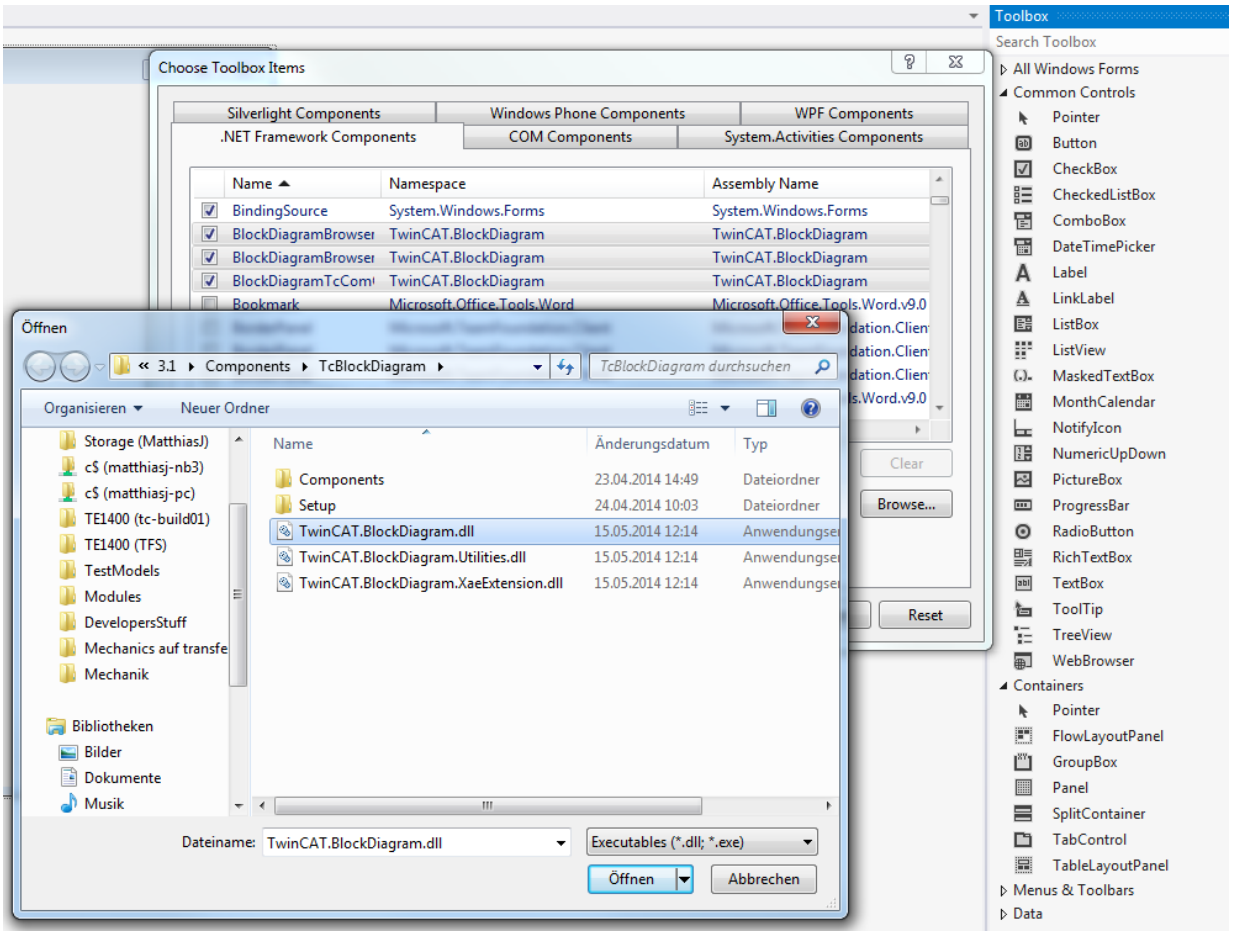
需要执行以下步骤：

1. 创建一个新的 Windows Forms 应用程序。
2. 将 TwinCAT.BlockDiagram.dll 添加到工具箱中：

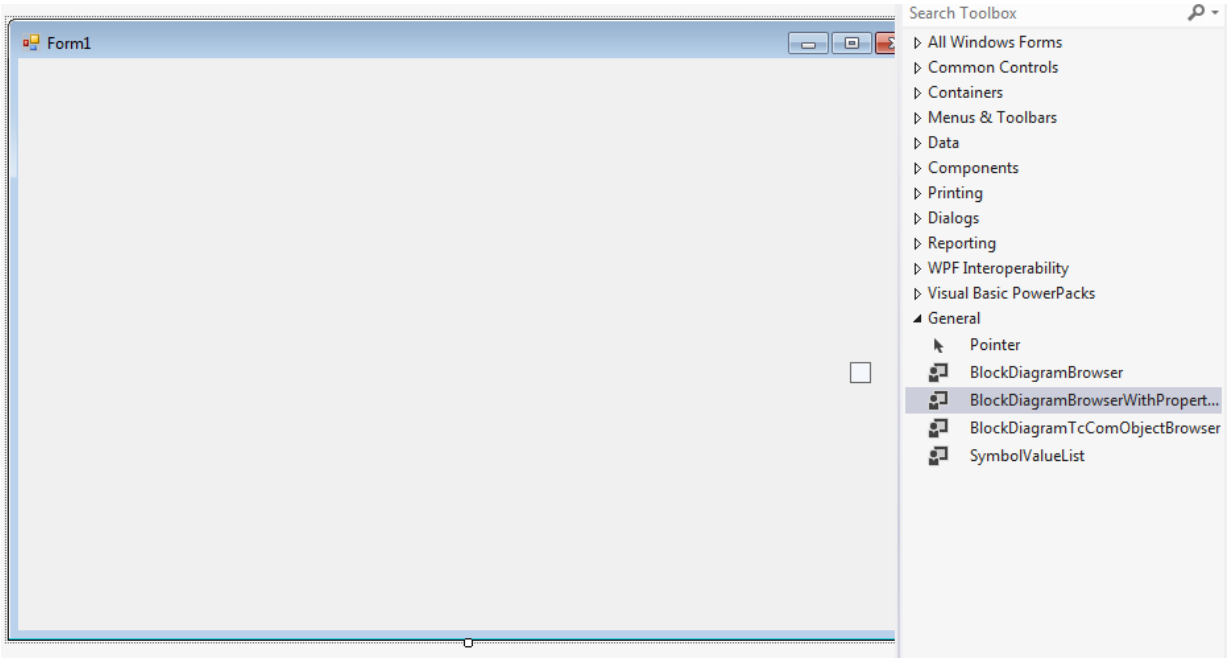
3. 为此，请在上下文菜单中选择“选择项...”（Choose Items...）条目。




4. 按照 *<TwinCAT installation path>|3.1|Components|TcBlockDiagram* 路径浏览至 TwinCAT.Blockdiagram.dll。



5. 使用拖放功能将 TcBlockdiagram 控件实例添加到 Windows Forms 对象中。

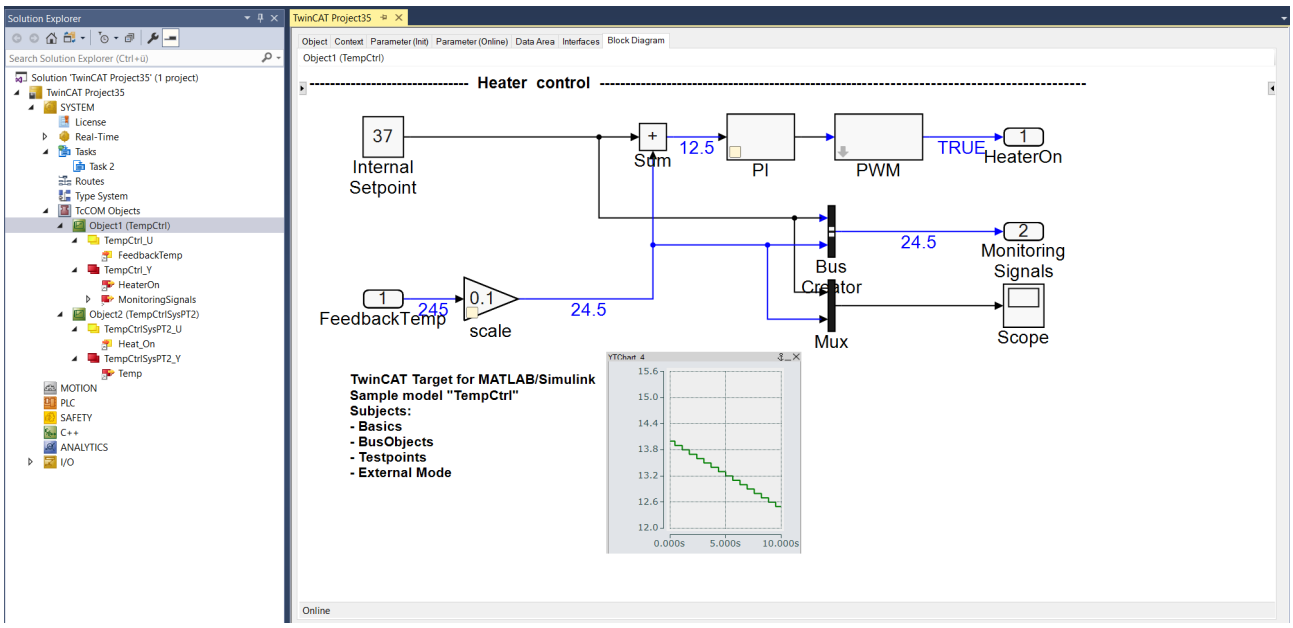



## 4.10.2 亲自试用创建的 TwinCAT 对象

✓ 您没有 MATLAB®，或者您只是想使用针对 Simulink 的 TwinCAT Target® 试用编译后的 TwinCAT 对象？然后按照下面的说明进行操作。

1. https://infosys.beckhoff.com/content/1033/te1400_tc3_target_Matlab/Resources/14632780043.zip
2. 解压缩 ZIP 并执行 tmx 压缩包。
   ⇨ 压缩包中包含两个 tmx-archive [▶ 57] 作为可执行文件。
3. 执行这些操作。
   ⇨ 这将把 TwinCAT 对象解压缩到正确的 TwinCAT 路径（开发环境存储库）中。
4. 打开 XAE 并创建实例。
   ⇨ 现在您可以在 TwinCAT 3 中创建新的 TcCOM 对象，可以在“TE140x 模块供应商” – TE140x – Simulink 模块下找到这两个对象。
5. 创建实例并分配 TwinCAT 任务。
6. 激活配置。
7. 将用于签名的证书列入目标的信任列表。
   ⇨ 目标系统必须加载所用的 tmx 文件。它们带有签名，由 OEM 证书创建。所使用的 OEM 证书很可能还不在白名单中的目标系统上。
8. 据此将证书加入白名单。
9. 激活配置。

⇨ 激活配置后，您可以在框图 [▶ 118]中观察对象的行为并更改参数。




在工程系统上进行必要的安装：

TwinCAT 3.1 Build 4026 下载项

- TwinCAT 标准版
- TwinCAT 经典框图

TwinCAT 3.1 Build 4024

- TwinCAT 3 XAE 设置

# 5 技术支持和服务

倍福公司及其合作伙伴在世界各地提供全面的技术支持和服务，对与倍福产品和系统解决方案相关的所有问题提供快速有效的帮助。

### 下载搜索器

我们的下载搜索器包含我们供您下载的所有文件。您可以通过它搜索我们的应用案例、技术文档、技术图纸、配置文件等等。

可供下载的文件格式多种多样。

### 倍福分公司和代表处

若需要倍福产品的本地支持和服务，请联系倍福分公司或代表处！

倍福遍布世界各地的分公司和代表处地址可在倍福官网上找到：http://www.beckhoff.com.cn

该网页还提供更多倍福产品组件的文档。

### 倍福技术支持

技术支持部门为您提供全面的技术援助，不仅帮助您应用各种倍福产品，还提供其他广泛的服务：

- 技术支持
- 复杂自动化系统的设计、编程和调试
- 以及倍福系统组件的各种培训课程

| | |
|---|---|
| 热线电话： | +49 5246 963-157 |
| 电子邮箱： | support@beckhoff.com |

### 倍福售后服务

倍福服务中心提供所有售后服务：

- 现场服务
- 维修服务
- 备件服务
- 热线服务

| | |
|---|---|
| 热线电话： | +49 5246 963-460 |
| 电子邮箱： | service@beckhoff.com |

### 倍福公司总部

Beckhoff Automation GmbH & Co. KG

Huelshorstweg 20
33415 Verl
Germany

| | |
|---|---|
| 电话： | +49 5246 963-0 |
| 电子邮箱： | info@beckhoff.com |
| 网址： | www.beckhoff.com |

## Trademark statements

Beckhoff®, TwinCAT®, TwinCAT/BSD®, TC/BSD®, EtherCAT®, EtherCAT G®, EtherCAT G10®, EtherCAT P®, Safety over EtherCAT®, TwinSAFE®, XFC®, XTS® and XPlanar® are registered trademarks of and licensed by Beckhoff Automation GmbH.

## Third-party trademark statements

AMD is a trademark of Advanced Micro Devices, Inc.

Arm, Arm9 and Cortex are trademarks or registered trademarks of Arm Limited (or its subsidiaries or affiliates) in the US and/or elsewhere.

DSP System Toolbox, Embedded Coder, MATLAB, MATLAB Coder, MATLAB Compiler, MathWorks, Predictive Maintenance Toolbox, Simscape, Simscape™ Multibody™, Simulink, Simulink Coder, Stateflow and ThingSpeak are registered trademarks of The MathWorks, Inc.

Intel, the Intel logo, Intel Core, Xeon, Intel Atom, Celeron and Pentium are trademarks of Intel Corporation or its subsidiaries.

The registered trademark Linux® is used pursuant to a sublicense from the Linux Foundation, the exclusive licensee of Linus Torvalds, owner of the mark on a worldwide basis.

Microsoft, Microsoft Azure, Microsoft Edge, PowerShell, Visual Studio, Windows and Xbox are trademarks of the Microsoft group of companies.

更多信息:
**www.beckhoff.com/te1400**

Beckhoff Automation GmbH & Co. KG
Hülshorstweg 20
33415 Verl
Germany
电话号码: +49 5246 9630
info@beckhoff.com
www.beckhoff.com