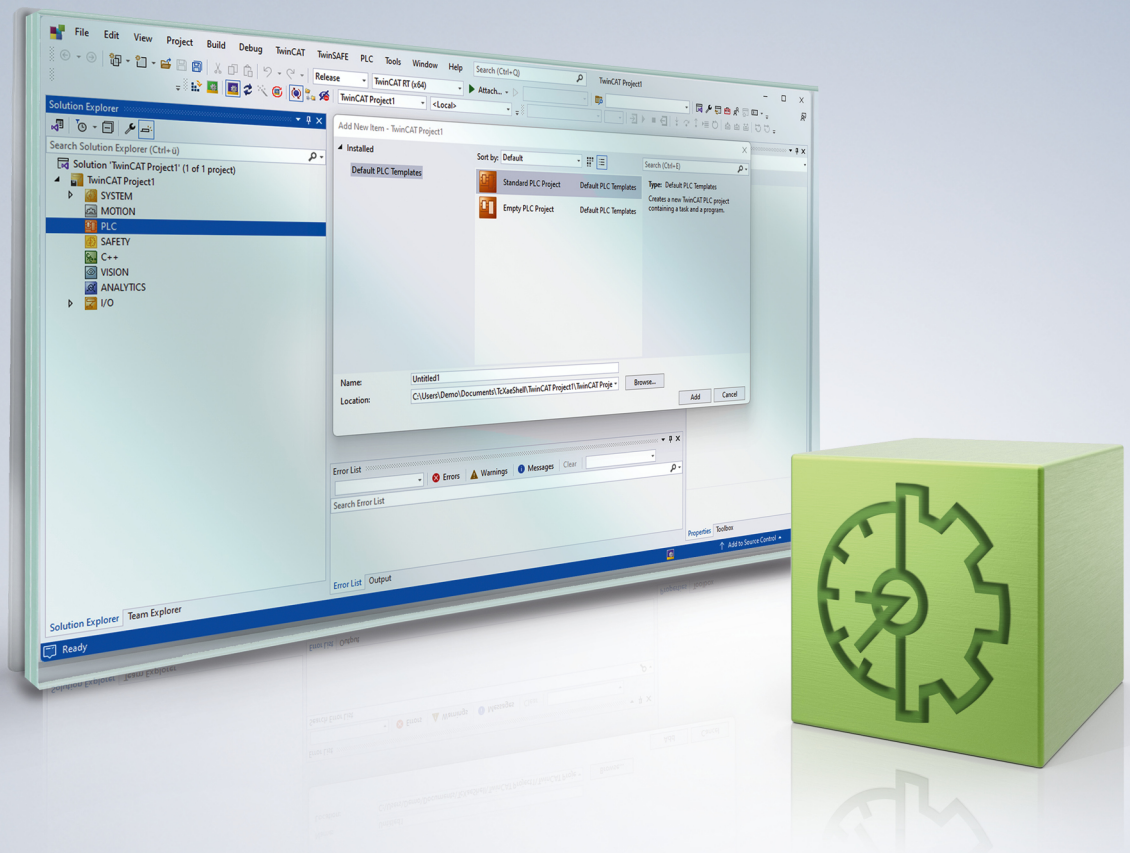
BECKHOFF New Automation Technology

手册 | ZH

TwinCAT 3

基础知识



2025-12-01 | 版本： 1.3.1

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®、ATRO®、EtherCAT®、EtherCAT G®、EtherCAT G10®、EtherCAT P®、MX-System®、Safety over EtherCAT®、TC/BSD®、 TwinCAT®、TwinCAT/BSD®、TwinSAFE®、XFC®、XPlanar® 和 XTS®是 Beckhoff Automation GmbH
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 Beckhoff Automation GmbH 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。

2 实时

根据 DIN 44300 标准，实时，或者说实时运行的定义如下：
"实时运行是计算系统的一种运行模式，在这种模式下，用于处理数据的程序以在规定时间内提供处理结果的方式持续运行"。
换句话说，可在规定的可以保证的时间内提供应用程序的输出值（根据内部状态和输入值计算）。此规定时间也称为周期。

应用程序本身可以由多个程序块组成，这些程序块同理也会调用其他程序或功能块等。（另请参见 IEC 61131-3 标准）。程序块可以指定到实时任务，由调度程序按周期和优先级调用。

TwinCAT 3 Real-Time 是一种扩展实时系统，当前的 TwinCAT 3.1 版本可运行于微软 Windows 7 以上操作系统以及 TwinCAT/BSD 和倍福 RT Linux® 下。为了满足工业过程控制的上述要求，TwinCAT 3 Real-Time 支持以下功能：

- Real-time（实时系统）可以调度
- 多进程并列执行
- 支持多核处理器
- 直接硬件访问

TwinCAT 3 多核支持功能允许将某些可用的核专门用于 TwinCAT，或者与所在操作系统共享。因此在后续章节中这些核被称为“isolated（隔离核）”或“shared（共享核）”。

实时调度能力

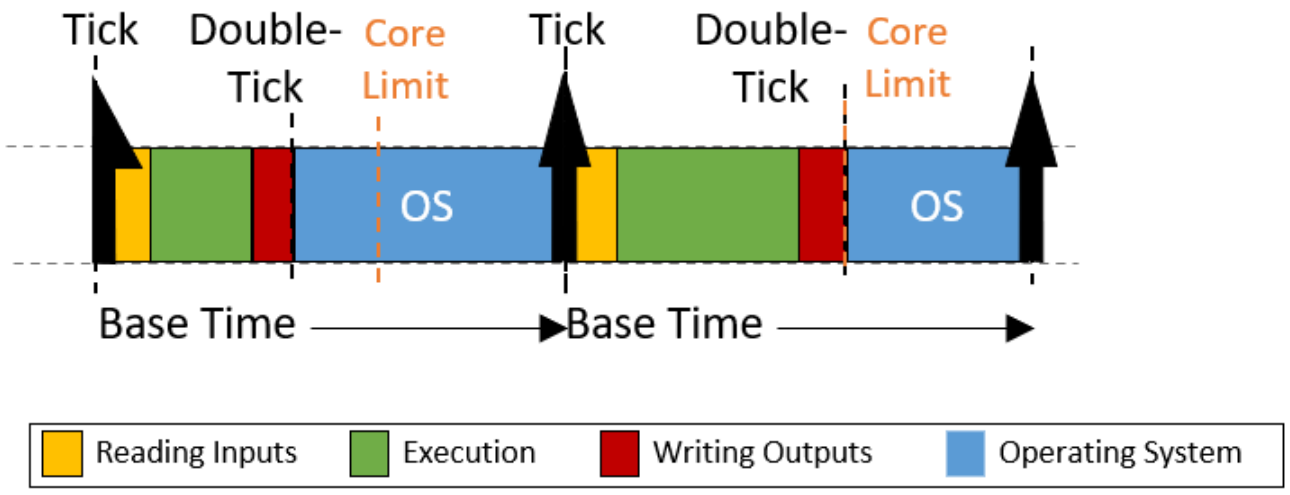
TwinCAT 3 Real-Time 采用双时标法运行。这意味着无论是切换到实时模式还是从实时模式切换回来，都是由一个中断信号触发的。切换到实时模式时触发的中断同时也会启动调度。经过一个可调整时间段后（至少在设置循环时间的 90% 之后），TwinCAT 会以非实时模式下切换回“共享”核，以便客户操作系统有足够的计算时间来满足硬件功能等所需的响应时间。隔离核是例外。

调度指的是根据规定的循环时间和优先级确定各项任务的处理顺序和处理时间的（系统）进程。严格遵守处理时间可确保符合上文所述的实时性。

由所有实时内核上的同步基本时标触发，每个实时内核的调度均在 TwinCAT 3 Real-Time 中独立进行计算。这样可保证在不同核上运行的实时任务不会相互干扰。除非在用户程序中使用联锁对其进行明确编程。

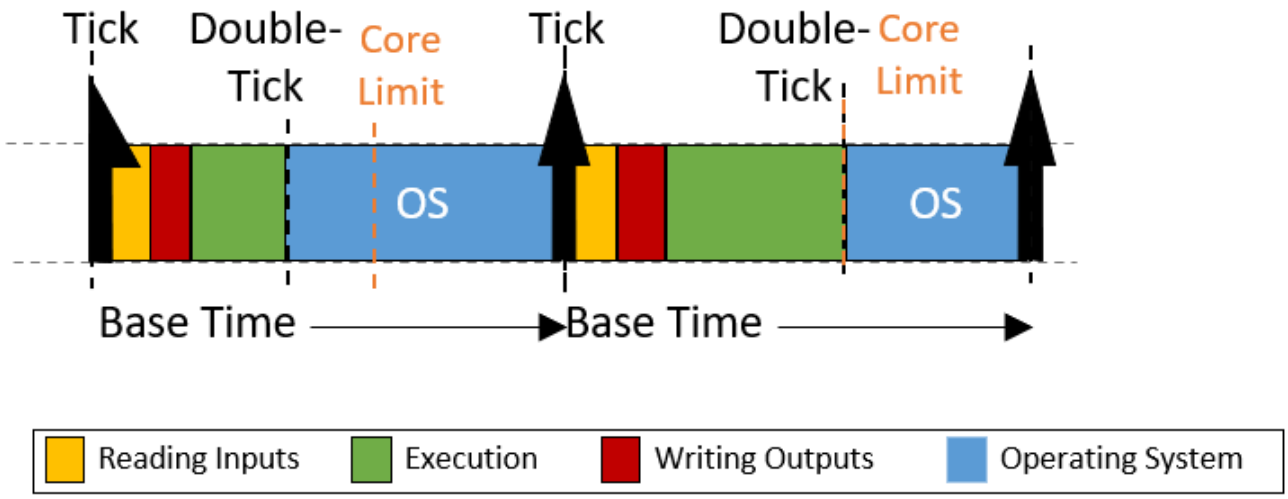
根据循环时间派生任务优先级的调度方式也称为速率单调调度。TwinCAT 3 Real-Time 会自动激活"自动优先级管理"选项。由于该选项并不总是每个应用的最佳解决方案，因此可以手动调整优先级。

PLC 任务调用的表示示例



图中显示了 PLC 任务的调用。在发生实时时标后，调度程序会调用 PLC 任务。这样，PLC 应用程序即可获得当前输入值（输入更新），然后该应用程序会处理该值（循环更新）。最后会将结果写入输出（输出更新）。此操作完成后，设备将切换到非实时模式（双时标）。如图所示，用户程序的执行时间可能会有所不同，具体取决于根据程序的内部状态执行的代码。因此，写入输出的时间也会有所不同。根据总线系统驱动任务的不同，这可能会导致总线报文的发送出现相同程度的变化。

使用"在任务开始时 I/O"调用任务的样例



通过使用"在任务开始时 I/O"选项，可更改任务内的处理顺序，以便在读取输入后直接写入（前一个循环的）输出，然后再执行应用程序。虽然要到下一个循环才会写入输出，但这种设置的优点是，每个循环中输出写入进程/总线的时间均完全相同。

抢占式多任务处理

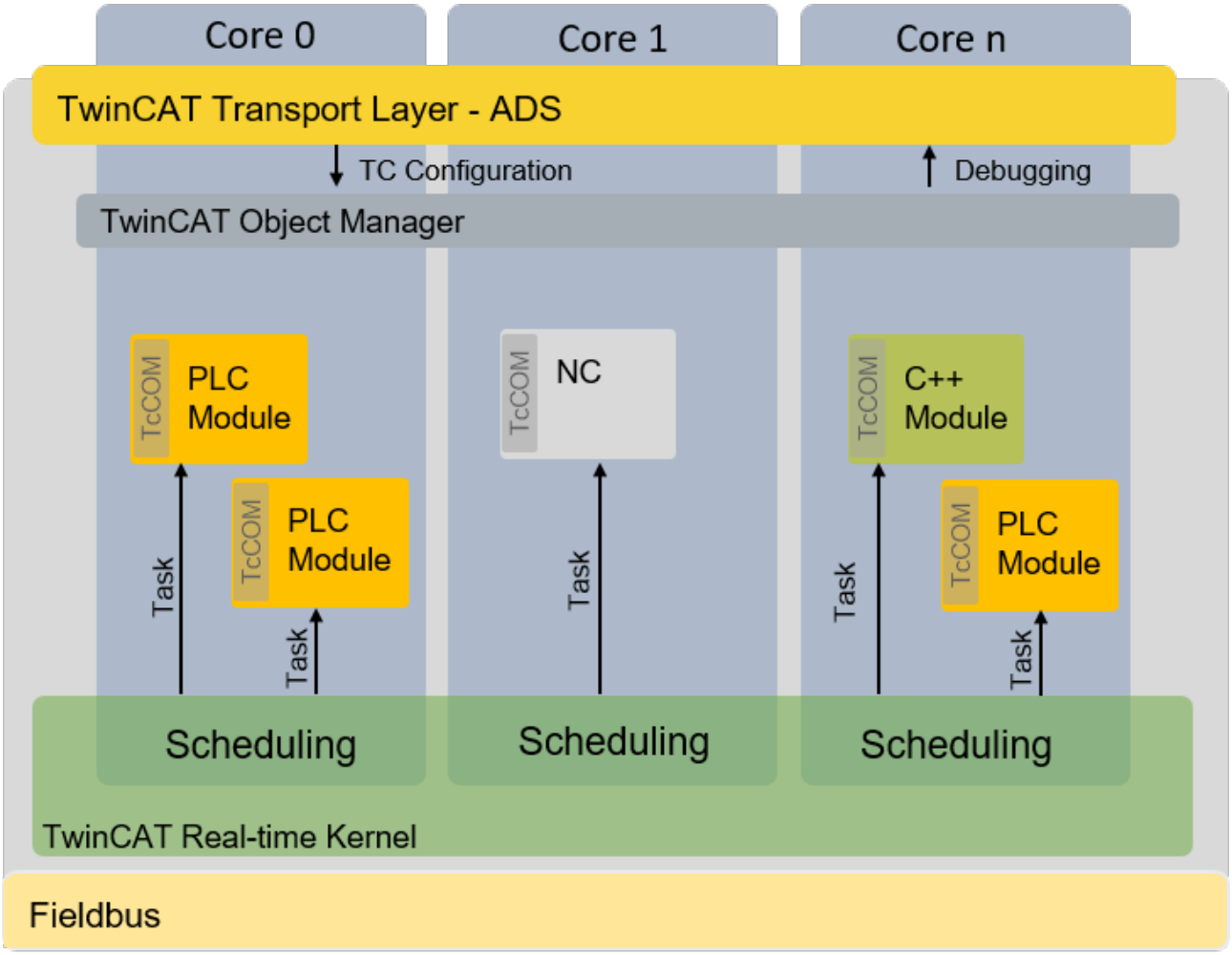
抢占式多任务处理是指在发生中断（如被优先级更高的进程中断）时，系统会保存进程（CPU 和浮点寄存器）的当前状态，并暂停当前的进程。如果出现这种情况，调度程序会根据任务的优先级确定要执行的（新）进程。 被中断的进程完成后，进程上下文就会恢复，并继续执行“旧”流程。

直接硬件访问

为了实现确定性（可复现）实时行为，TwinCAT 3 Real-Time 需要直接访问硬件。为此，TwinCAT 3 Real-Time（实时系统）必须在 Windows 或 TwinCAT/BSD 的内核模式下执行 。除其他事项外，这种方式使得 TwinCAT Real-Time 可以直接访问网络端口，以及发送和接收实时以太网报文（如 EtherCAT）。在倍福 RT Linux® 下，该实时系统通过用户模式下的实时扩展组件运行。直接硬件访问通过特殊的网络驱动程序实现。

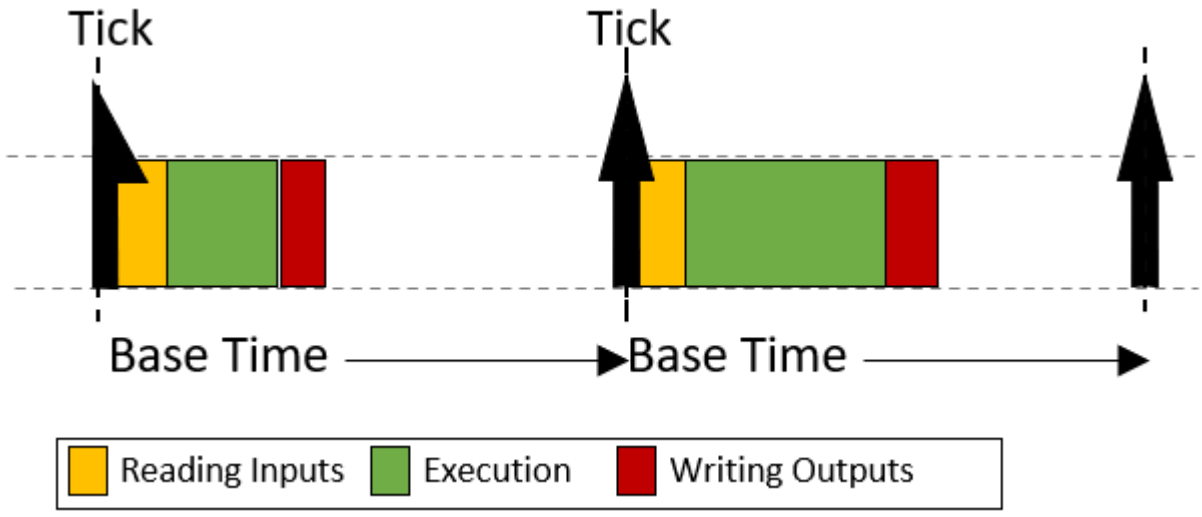
TwinCAT 3 运行时环境示意图

下图说明了与调度相关的 TwinCAT 3.1 运行时环境的结构。TwinCAT 3 运行时环境可实现实时执行用户模块。因此，实时驱动程序是 TwinCAT 3 运行时环境的一个重要组成部分，它在激活了 TwinCAT 的核上执行，并在那里处理调度。后者在单个核上独立进行。



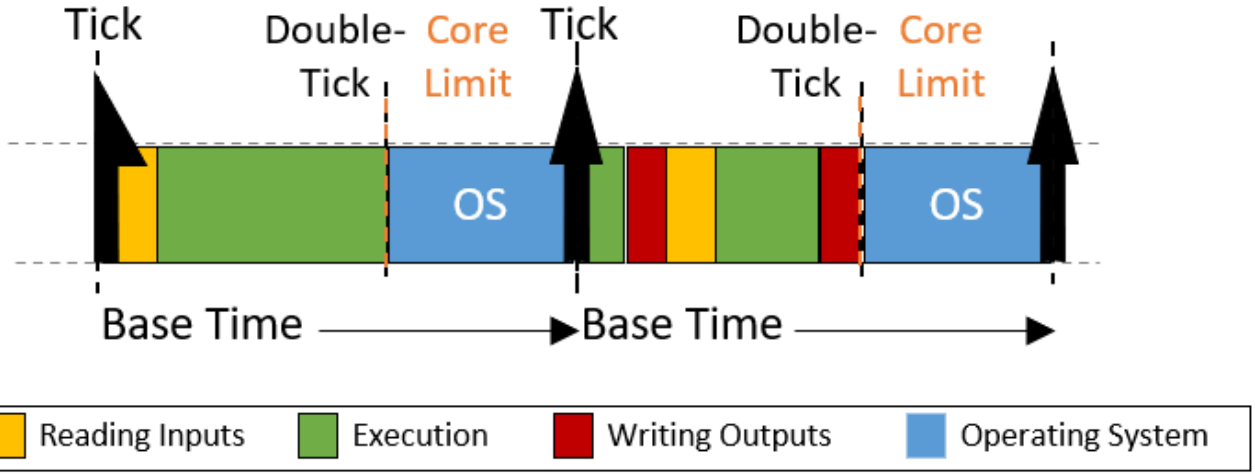
隔离核

如Real-time（实时系统）的调度 [▶ 7]中所述，TwinCAT 使用双时标程序在指定时间点切换回非实时模式。在实时模式和非实时模式之间切换时，会恢复之前的进程状态。这个过程和抢占式多任务处理 [▶ 8]类似。恢复状态需要一些时间，这取决于实时和非实时程序使用内存(特别是缓存)的密集程度。为消除这些瞬时效应，TwinCAT 3.1 Real-Time 允许将各核从客户操作系统中隔离出来。这样便不需要切换回之前的模式，通过避免与恢复“旧”进程状态相关的时间效应，为实时用户程序带来了更多的计算时间和更好的实时质量（减少抖动）。

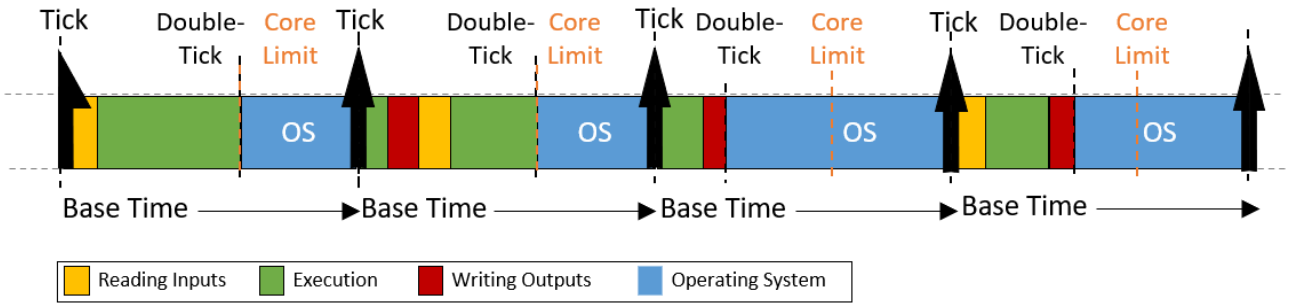


超过周期时的行为

如果超过了任务的规定周期，则在下一个周期中继续处理"旧的"循环。此外，任务超时计数器的计数也会递增。一旦完成旧循环/上一循环的处理，系统会立即尝试开始处理当前循环的任务。如果在当前循环内完成了这一操作，则会如上所示执行进一步处理。



如果直接紧随其后的第二个循环也超时（在这种情况下，系统是否仍在处理第一个循环或第二个循环是否开始并不重要），则当前处理任务完成，并且直到下一个可能的计划循环开始时才会处理下一任务。这意味着可能会丢失几个循环。超时计数器相应递增。

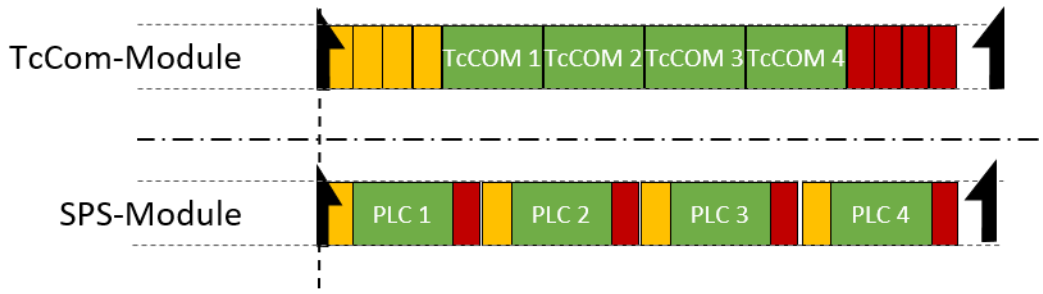


PLC 和"TcCom"运行模块的执行差异

TwinCAT 任务的处理与运行模块的执行有关，按照以下步骤进行：

1. 将输入复制到任务调用的运行模块的过程映像中。
2. 按照排序顺序（升序）执行模块。
3. 更新输出，这样可使输出相应可用。如果该任务驱动 EtherCAT 现场总线，则在输出过程映像时提供并发送帧。
4. 周期后更新：除其他事项外，当"I/O at task start（在任务开始输入/输出）"选项处于活动状态时，可将其用于触发周期更新。

如果运行模块被添加到某一任务中，它们会"登录"到任务的相应调用中。唯一例外是 PLC 运行时模块。为了与 TwinCAT 2 兼容，PLC 运行模块会直接更新输入和输出。两种行为的区别如下图所示：



每种情况都可以看见四个运行模块。标准 TwinCAT 3 运行模块登录到任务的相应方法调用。这意味着所有输入更新（黄色）和输出更新（红色）均由任务触发，并在任务处理开始或结束时直接相继进行。如果其中两个模块通过映射相互通信，它们在下一个周期之前不会收到当前值。

PLC 运行时模块独立执行输入和输出更新。如果两个 PLC 运行时相互通信，则执行的第二个运行模块会直接从第一个运行模块接收当前值。因此，在从第一个运行模块到第二个运行模块的通信方向上不存在周期偏移，但在另一个方向上存在这种偏移。

2.1 任务

Task（任务）是一个 runtime（运行时）对象，可由调度程序进行调度和触发。需要运行于该对象 context（上下文）下的 Functions（功能）或运行时对象必须注册到此对象。Task（任务）被指定了周期和优先级，调度程序以此调度任务执行（请参见 Real-time（实时系统）的调度 [▶ 7]）。此外，每个任务都指定了一个共享内存/地址空间（所有任务均可共同访问）以及一个额外堆栈内存。需要这个堆栈内存才能实现 function（功能）与 sub-function（子功能）的嵌套执行。这个堆栈也作为局部变量存储区。Task（任务）的状态取决于该堆栈以及机器寄存器（计算寄存器）的当前内容。如果 Task（任务）的 context 发生变化或者被更高优先级的任务打断，该状态则会保存并在再次执行时恢复。任务堆栈的大小可以在 TwinCAT 中定义（请参见实时系统的设置Settings（设置）选项卡一章）。

对于多任务顺序控制，（实时）系统提供通信和同步的功能（请参见 PLC 中的多任务数据访问同步一章）。

以下章节介绍相应的 Task（任务）类型。

2.1.1 TwinCAT 任务

TwinCAT 标准任务，可注册 TwinCAT Runtime 模块。

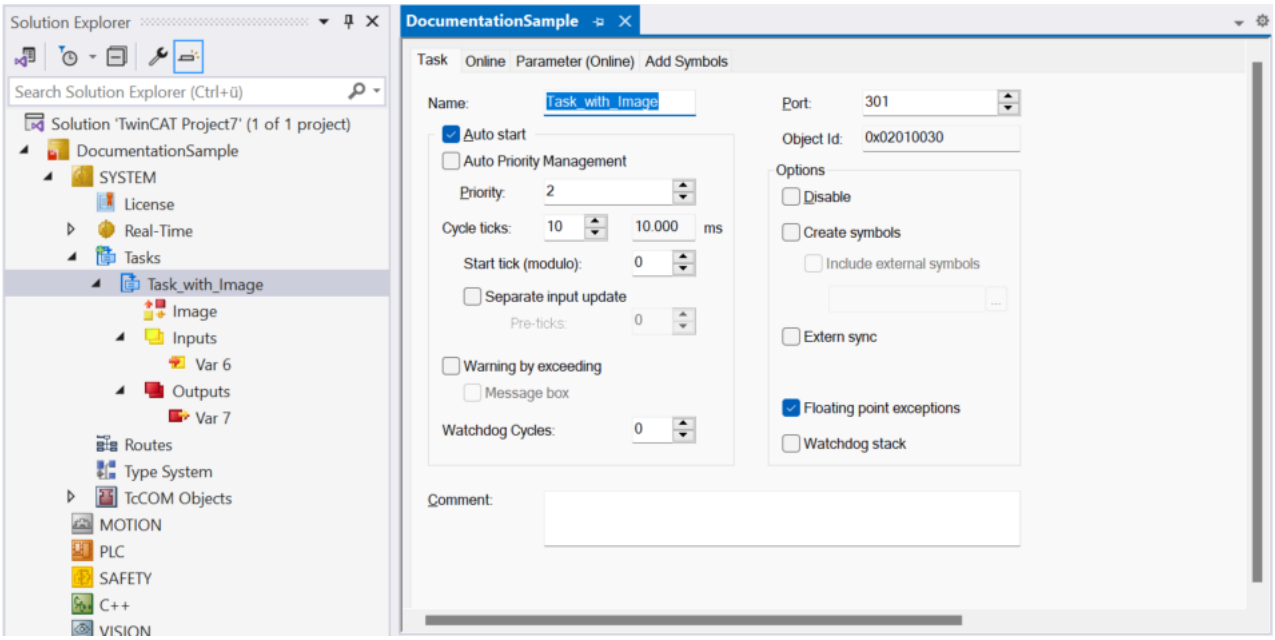
要在 PLC 中使用 Task，可将一个 Task 指定到一个 task reference（任务引用）（请参见创建引用任务）。

通常对于 TcCom 模块，可以通过它的 **context（上下文）**选项卡指定 Task（任务）。（请参见 TcCOM 模块处理一章中的 Context（上下文）项卡 [▶ 230]）。

在“TwinCAT 项目”文档中的 TwinCAT 任务章节中介绍了任务的设置。

2.1.2 TwinCAT Task with Image（带映像的 TwinCAT 任务）

与标准任务（请参见 TwinCAT 任务 [▶ 11]一章）相比，**TwinCAT Task with Image**还带有自己的过程映像。

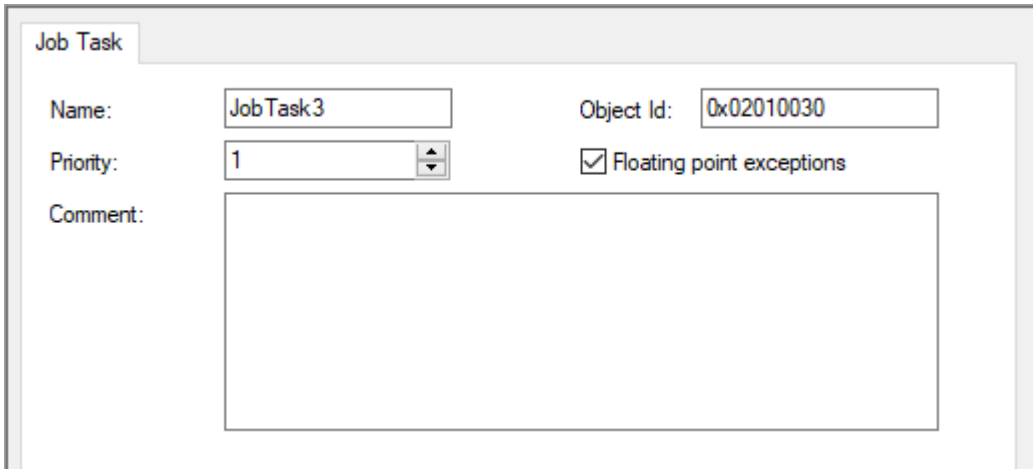


可在该过程映像中创建变量链接至其他过程映像（例如来自 EtherCAT 设备）。Task 根据设置的周期和优先级触发映射操作。因此可以无需 PLC 或其他 runtime 模块实现周期性的总线通信。过程映像中的变量可以从 TwinCAT Scope 或其它非实时应用通过 ADS 进行访问。

2.1.3 TwinCAT Job Task (Worker Task)

Job Task 是一种按需执行的任务。它由应用程序调用，而不是周期性执行。Job Task 可在 Tasks（任务）下或 Job Pool（工作池）中直接创建。如果您直接在 Tasks（任务）下创建 Job Task，可以从客户端应用程序直接向其传递任务（jobs）。

但是，如果您在 Job Pool（工作池）下创建 Job Task， 任务会就从应用程序传到 Job Pool（工作池）。然后，这样就会将该任务分配给池中下一个可用的 Job Task



名称	Job Task 的名称
Object ID（对象 ID)	Job Task 的对象 ID
优先级	Job Task 的优先级
Floating point exception	规定 TwinCAT 是否检查浮点异常。
注释	工作任务的可选注释

对于 calling task（调用任务）和 Job Task，应选中设置项 **Floating point exception**（浮点异常）。

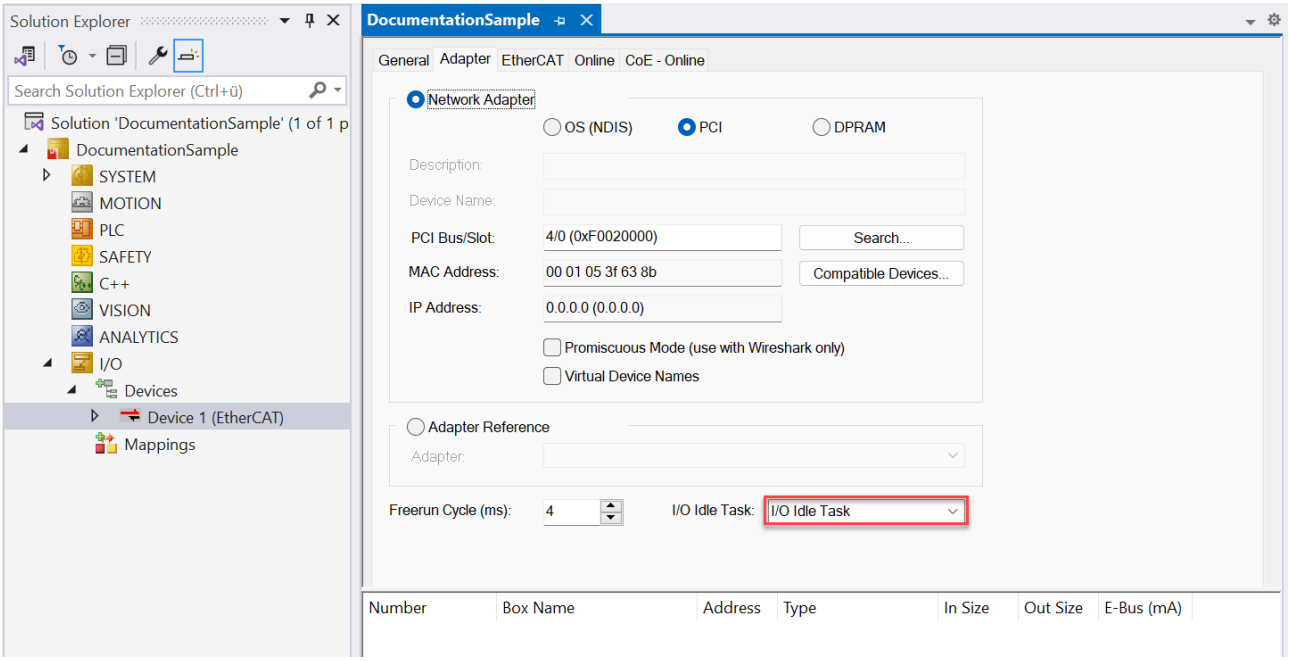
2.1.4 I/O Idle Task（闲时任务）

I/O Idle Task 处理现场总线的非周期性通信，因此，它负责：

- EtherCAT 设备的状态机
- CoE 和 SoE 通信
- 通过 AOE 读取或设置参数
- Mailbox（邮箱）通信
- EtherCAT 的非周期性诊断（例如，状态查询）

I/O Idle Task 是一个周期性任务，但不是用于周期性 I/O 通信（过程数据交换）。因此，在大多数情况下，合理的做法是将其优先级设置为在 PLC 和 Motion 任务之后执行。既然这也是非周期性通信，所以通常这些任务的周期超时并不重要。

自 TwinCAT 3.1.4026 版本起，已经可以使用多个 I/O Idle Task （每个 EtherCAT 主站一个）。I/O Idle Task 是通过 EtherCAT 主站的 adapter settings（适配器设置）来选择的。



2.1.5 PLC-AuxTask

PLC-AuxTask 用于 PLC 编辑器和 PLC runtime（运行时）之间的通信。包括 PLC 控制代码的下载和 Online Change（在线更改）以及调试（数值监控、断点设置等）。此外，PLC-AuxTask 还会处理 ADS 消息，它们从独立于开发环境（TcXaeShell）之外直接发送到 runtime（例如，来自 HMI）。

PLC-AuxTask 不是周期性任务，您应将其优先级设置为低于周期性任务。这样可确保周期性任务能够打断 PLC-AuxTask。在线更改时，新代码将通过 PLC-AuxTask 传输到目标系统，并生成对应的 symbols（符号）等。在线更改过程中，只有更换执行代码的“关键阶段”才受到保护，从而确保该过程不会被周期性任务打断。

如果多个 PLC runtime 模块使用相同的 PLC 任务，那么通过在线更改它们可能会相互影响。

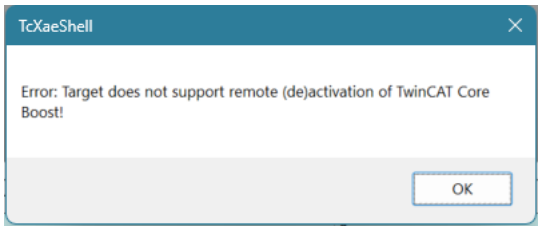
2.2 Core Boost

TwinCAT Core Boost

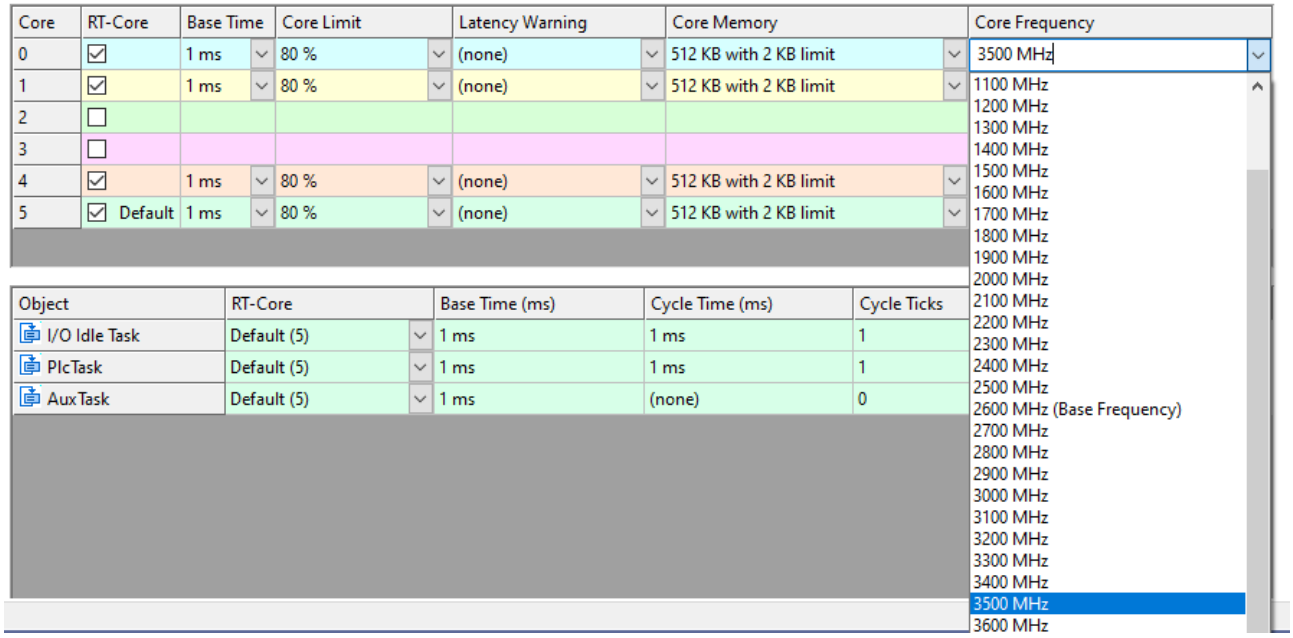
前提条件：开发环境和运行时环境都必须至少使用 TwinCAT 3.1.4026.6 版本。

如果给定目标系统支持并已激活 TwinCAT Core Boost 功能，则在按下 **Read from Target**（从目标读取）按钮后，此状态会在 **TwinCAT Core Boost** 选择框中显示。要激活或禁用该功能，必须使用 **Set on Target**（在目标上设置）按钮，在目标系统上激活这些设置。更改此设置后，您必须重新启动计算机。

如果给定目标系统不支持 TwinCAT Core Boost 功能，则在按下 **Set on Target**（在目标上设置）按钮后会出现以下消息：



如果已激活 TwinCAT Core Boost 设置，则可以为每个实时核定义一个时钟频率。如果没有为实时核定义时钟频率，则会自动选择 **Base Frequency**（基准频率）。



正确选择核的频率

根据系统中存储的各个核的温度限值或各个封装的功耗限值，TwinCAT 会自动监控各个核的时钟频率。如果超过这些限值，TwinCAT 就会相应地降低各个核的时钟频率（另请参见Core Boost（核的加速）选项卡一章）。如果 TwinCAT 被迫降低各个实时核的时钟频率，这可能会对在 TwinCAT 中设置的实时行为产生影响。然后在该实时核上执行的任务执行时间更长，可能因此导致周期超时。因此，在选择实时核的时钟频率时，应当确保 TwinCAT 不会长期运行在降频模式下。如果长期超过温度限值，系统可能会关机。

选择非实时核的时钟频率

非实时核是指未激活 TwinCAT 也不用于实时系统的计算机核，所以只有操作系统触发的进程在此执行。对于非实时核，时钟频率由操作系统根据需要自动选择。非实时核可提升到的最大时钟频率为基准频率。从第 12 代和第 13 代 Intel® 处理器开始，非实时核在必要时可以超频运行至核的加速频率。核的加速频率级别存储在系统中，因处理器类型的不同而有所差异。

可配置的时钟频率：

处理器	处理器代次	基准时钟	可配置的核加速时钟
Core i3-11100HE	第 11 代 Intel® Core™	2.40 GHz	4.00 GHz
Core i5-11500HE	第 11 代 Intel® Core™	2.60 GHz	4.10 GHz
Core i7-11850HE	第 11 代 Intel® Core™	2.60 GHz	4.20 GHz
Core i3-13100E	第 13 代 Intel® Core™	3.30 GHz	4.20 GHz
Core i5-13400E	第 13 代 Intel® Core™	2.40 GHz	4.10 GHz
Core i7-13700E	第 13 代 Intel® Core™	1.90 GHz	4.00 GHz
Core i9-13900E	第 13 代 Intel® Core™	1.80 GHz	3.90 GHz

2.3 核的内存

TwinCAT（3.1.4026 和 3.1.4024）允许实时任务能够在实时系统的框架内动态分配内存。由于实时任务无法直接从操作系统分配内存区块，因此 TwinCAT 提供了多种内存池。这些内存池是由操作系统预先分配的内存区块。在物理内存上这些内存区块被锁定，以避免在实时系统框架访问时发生分页。

如 一章所述，TwinCAT 有多个内存池。它们包括：

Executable RT Memory（RT 执行内存）	实时系统的执行内存，这是 TwinCAT 系统的内部需求，不直接对用户开放
Global RT Memory（RT 全局内存）	实时系统的全局内存，用于在实时系统框架内为 TwinCAT 模块（PLC 和 C++）进行内存分配
Core Memory（核的内存） <n>	用于为第 <n> 个实时核进行内存分配的实时系统内存
Global ADS Memory（ADS 全局内存）	用于 ADS 通信的数据内存。TwinCAT 各组件之间的 ADS 消息存放地址，均分配在这个内存。
Tc/OS 内存	Usermode runtime（用户模式运行时）的存池：在用户模式的运行时启动的时候进行分配。用户模式运行时在使用过程中，其他内存池均由此内存池提供支持。

在 TwinCAT 3 开发环境中，每个项目的 RT 全局内存大小可通过 **Router Memory**（路由内存）选项设置 。自 TwinCAT 3.1.4026 版本起，ADS 消息的数据内存已从“Router Memory（路由内存）”中分离出来，可作为独立的 ADS 全局内存。ADS 全局内存的大小由 RT 全局内存的大小决定，相当于其大小的 25%，最大为 32 MB。

TwinCAT 3.1.4026 版本还引入了另一个内存池，即 Core Memory（核的内存）。Core Memory 为每个实时核单独提供了一个专用的实时内存池。如果 Core Memory 配置给了一个实时核，就会首先尝试从这个内存池中提供该核所请求的内存。如果这个内存不够用，才会使用实时系统全局内存中的内存。

因此，Core Memory 使得来同时发生的自不同核的内存分配可以平行且相互独立地进行。而同时发生的来自不同核的访问 RT 全局内存的内存分配，必须依次处理。这可能导致在一个实时周期中出现多次等待内存分配的时间，造成性能瓶颈，例如在视觉应用中。

另请参见：

- Core Management（核的管理）

3 目标系统

当前正在使用 TwinCAT 开发环境 (TwinCAT XAE) 进行编程的控制器被称为目标系统。本章将介绍处理目标系统的重要基础知识。要理解 章节中基于这些内容的文档，还需要这些内容。

首先，必须在开发环境和控制器之间建立连接，才能对控制器进行编程。为此，可以使用各种通道。路由 [▶ 16]一章将对各个选项有更详细的说明。

如果控制器已经编程并在现场，而您想在不使用编程环境的情况下更新设备，则有必要了解存在哪些文件和文件夹、需要它们用来做什么以及如何才能最好地交换它们。章节文件夹和文件类型 [▶ 183] 和 固件更新 [▶ 186] 专门对这些主题进行了讨论。

重启 TwinCAT 时，可能还需要自动启动其他程序（例如外部 HMI）。自动启动程序 [▶ 188] 一章对此进行了说明。

如果网络中的多个控制器都在处理同一个进程，则有必要在收集和评估数据时校正各控制器的时间戳，使收集到的数据遵守准确的时间顺序。为此，可以相应校正各控制器的时间戳。校正时间戳 [▶ 189] 一章对此进行了介绍。

3.1 启动目录

根据目标系统的操作系统的不同，启动目录的标准存储路径如下所示。

Windows 10、Windows 11：

Kernel Mode Runtime（内核模式运行时）：

- < TC3.1.4026.0: C:\TwinCAT\3.1\Boot
- >=TC3.1.4026.0: C:\ProgramData\Beckhoff\TwinCAT\3.1\Boot

User mode runtime（用户模式运行时）：

C:\ProgramData\Beckhoff\TwinCAT\3.1\Runtimes\UmRT_Default\3.1\Boot

TwinCAT/BSD：

/usr/local/etc/TwinCAT/3.1/Boot

倍福 RT Linux®：

/etc/TwinCAT/3.1/Boot

3.2 路由

如“理论”一章早已描述过，TwinCAT 3 由工程环境 (XAE) 和运行时环境 (XAR) 组成。工程环境用于在现场对运行时环境进行配置和编程。然后，运行环境（目标系统）以硬实时的方式执行控制程序。不一定在同一台 PC/IPC 上运行的两个环境之间通过 ADS 协议建立连接（请参见 ADS [▶ 17] 章节）。必须输入路由，以便工程环境可以与目标系统通信。这意味着另一参与者在两个环境（工程环境和运行时环境）已知的情况下进行输入。

为考虑当前安全和连接技术方面的趋势和要求，您可以相应地保护 ADS 连接，或通过当前传输协议建立隧道。请参见 安全 ADS [▶ 171] 或 ADS-over-MQTT [▶ 160] 章节。

3.2.1 ADS

3.2.1.1 ADS 简介

ADS 定义

《自动化设备规范》描述了一个独立于设备和现场总线的接口，该接口规定了对ADS设备的访问类型。

ADS 接口允许：

- 与其他 ADS 设备通信
- ADS 设备的实现

ADS 设备通信

为了能够参与 ADS 通信（作为 ADS 客户端或可能作为 ADS 服务器），提供以下软件对象：

- ADS-DLL
  ，适用于 C/C++ 等
- ADS.NET [▶ 5] 组件
  ，适用于 VB.NET、Visual C# 等

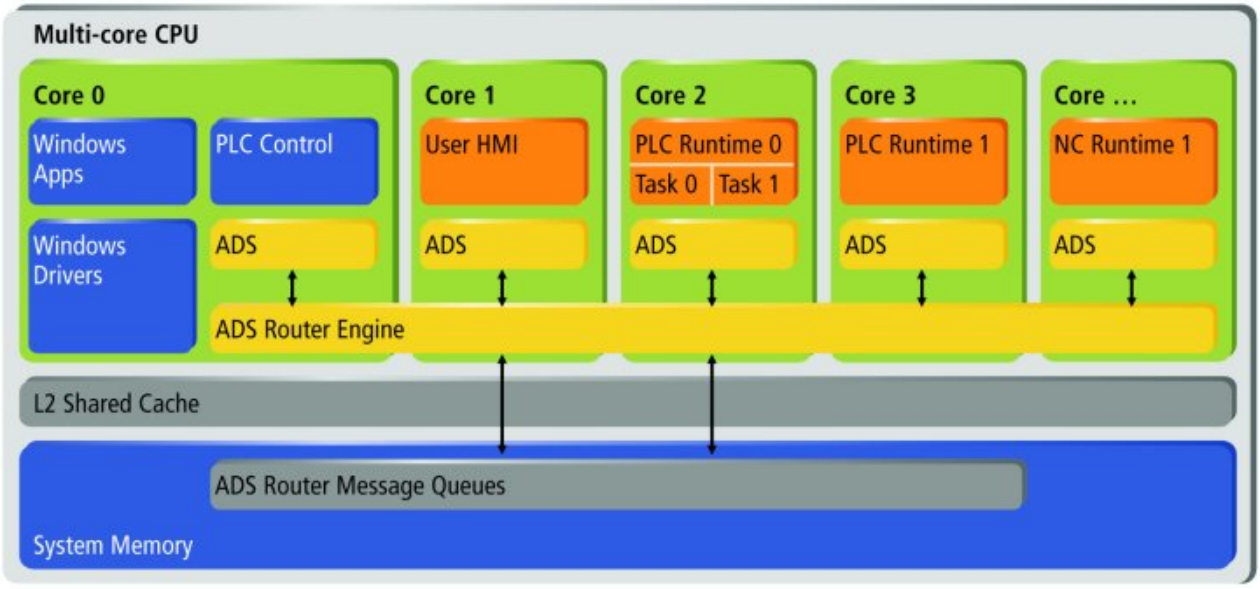
3.2.1.2 ADS 设备概念

TwinCAT 系统架构允许将软件的各个模块（如 TwinCAT PLC、用户 HMI.....）视为独立的设备：每个任务都有一个软件模块（“服务器”或“客户端”）。系统中的服务器是以软件形式执行工作的“设备”，其操作行为与硬件设备完全一样。因此，我们可以说在软件中实施了“虚拟”设备。“客户端”是请求“服务器”服务的程序，比如一个可视化工具，甚至是以程序形式存在的"编程设备"。因此，TwinCAT 具有可扩展性，因为总是有新的服务器和客户端用于凸轮轴控制器、示波器、PID 控制器等任务……

这些对象之间的消息由“消息路由器”通过一致的 ADS（**自动化设备规范**）接口进行交换。它管理和分配系统中以及通过 TCP/IP 连接进行的所有消息。

每个 TwinCAT 设备上都有 TwinCAT 消息路由器。
这样，所有 TwinCAT 服务器和客户端程序都可以交换命令和数据、发送消息、传输状态信息等……

下图显示了基于 ADS 的 TwinCAT 设备概念：



3.2.1.3 ADS 设备标识

ADS 设备的唯一标识借助两个标识符来实现：

- PortNr
- NetId

AMS 端口

TwinCAT 消息路由器中的 ADS 设备通过一个编号唯一标识，该编号称为 ADS 端口号。ADS 设备的端口号为指定和固定编号，而纯 ADS 客户端应用程序（如 HMI 系统）在首次访问消息路由器时会获分配可变端口号。

已分配以下 AMS 端口号：

AMS 端口	设备
1	ADS 路由器
2	AMS 调试器
10	TCom 服务器
11	TCom 服务器任务，实时上下文（RT context）
12	TCom 服务器，被动级别
20	TwinCAT 调试器
21	TwinCAT 调试器任务
30	授权服务器
100	记录器
110	事件记录器
120	EtherCAT 设备的应用程序
130	事件记录器用户模式 (V2)
131	实时事件记录器 (V2)
132	事件记录器发布服务器 (V2)
200	Ring 0 实时
290	Ring 0 追踪
300	Ring 0 IO
400	Ring 0 PLC（传统）
500	Ring 0 NC
501	Ring 0 NC SEC
511	Ring 0 NC SPP
520	NC 实例
550	Ring ISG
600	Ring 0 CNC
700	Ring 0 线路
800	Ring 0 TC2 PLC
801	TC2 PLC 运行时系统 1
811	TC2 PLC 运行时系统 2
821	TC2 PLC 运行时系统 3
831	TC2 PLC 运行时系统 4
850	Ring 0 TC3 PLC
851	TC3 PLC 运行时系统 1
852	TC3 PLC 运行时系统 2
853	TC3 PLC 运行时系统 3
854 - ⋯	TC3 PLC 运行时系统 4 - ⋯
900	凸轮控制器
950	凸轮设计工具
1000–1199	Ring 0 IO 端口
2000	Ring 0 用户
2500	Crestron 服务器
10000	系统服务
10201	TCP/IP 服务器
10300	系统管理器
10400	SMS 服务器
10500	Modbus 服务器
10502	AMS 记录器

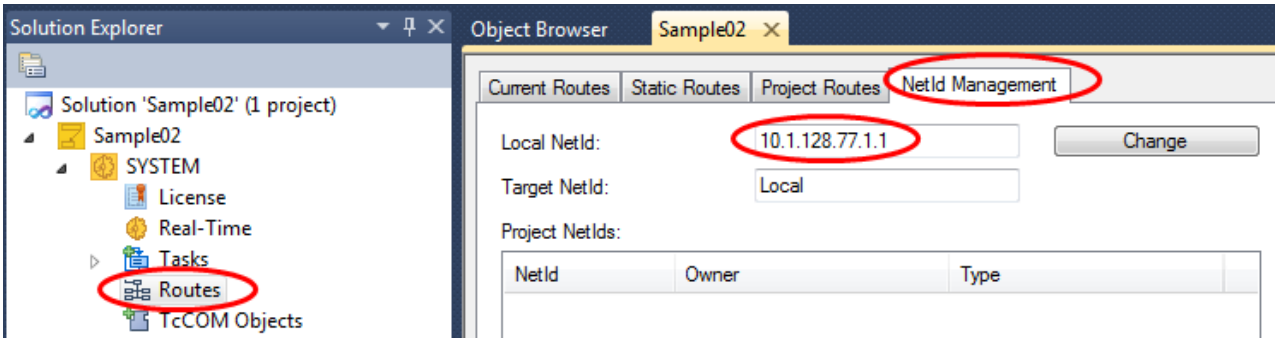
AMS 端口	设备
10600	XML 数据服务器
10700	自动配置
10800	PLC 控制
10900	FTP 客户端
11000	NC 控制系统
11500	NC 解释器
11600	GST 解释器
12000	追踪控制
13000	凸轮控制
14000	Scope Server
14100	状态监测
15000	正弦 CH1
16000	控制网
17000	OPC 服务器
17500	OPC 客户端
18000	邮件服务器
19000	虚拟 COM EL60xx
19100	管理服务器
19200	Miele@home 服务器
19300	CP-Link 3
19310	触摸锁
19500	视觉服务
19800	HMI 服务器
21372	数据库服务器
25000–25999	ADS 服务器保留端口范围
25013	FIAS 服务器
25014	Bang&Olufsen 服务器
26000–26999	客户私有端口范围
32768–65535	ADS 客户端保留端口范围

AMS NetId

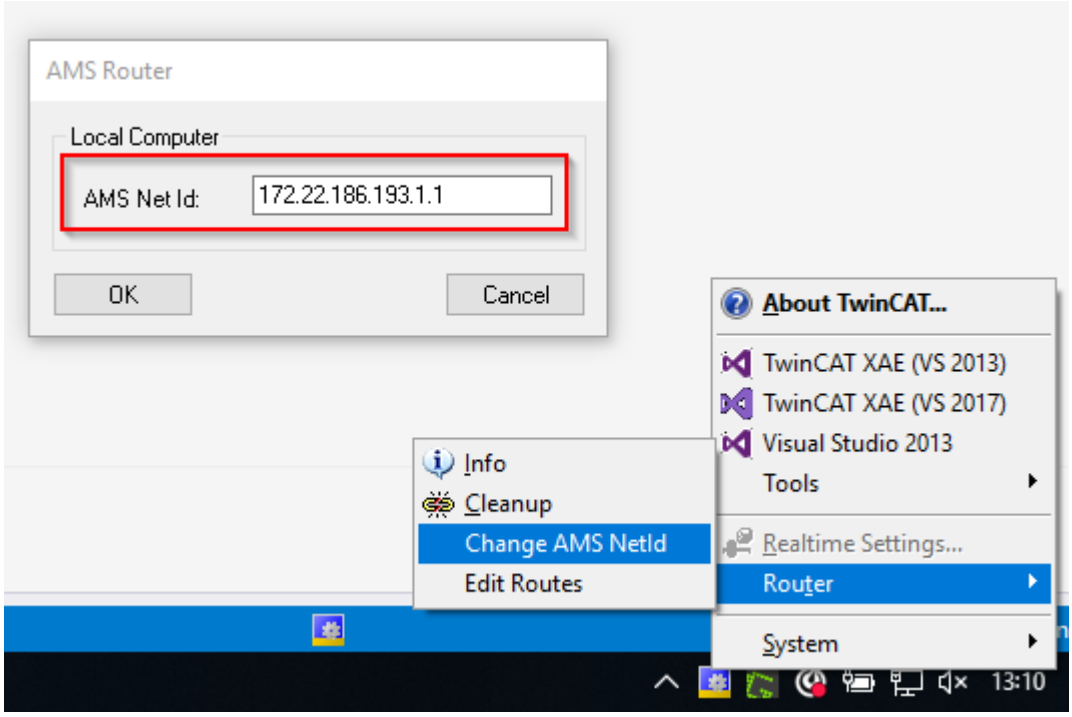
网络中的每台 TwinCAT 设备都由 AMS NetId 标识。AMS NetId 由六个八位字节组成。前四个八位字节可自行选定。最后两个八位字节（通常为 .1.1）用作现场总线或其他设备的子网掩码。AMS NetId 对所有通信伙伴来说都必须具有唯一性。

配置：

本地或远程 TwinCAT 设备的 AMS NetId 可在 TC3 项目的 SYSTEM\Routes\NetId 管理中设置。

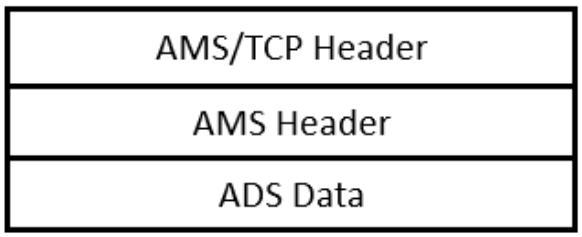


另外，也可以通过 TwinCAT SysTray 菜单中的路由器类别在本地配置 AMS NetId。更改 AMS NetId 后必须重新启动设备。



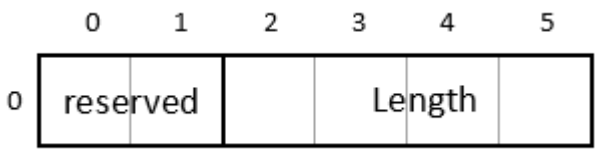
3.2.1.4 AMS/TCP 包

3.2.1.4.1 结构 AMS/TCP 包



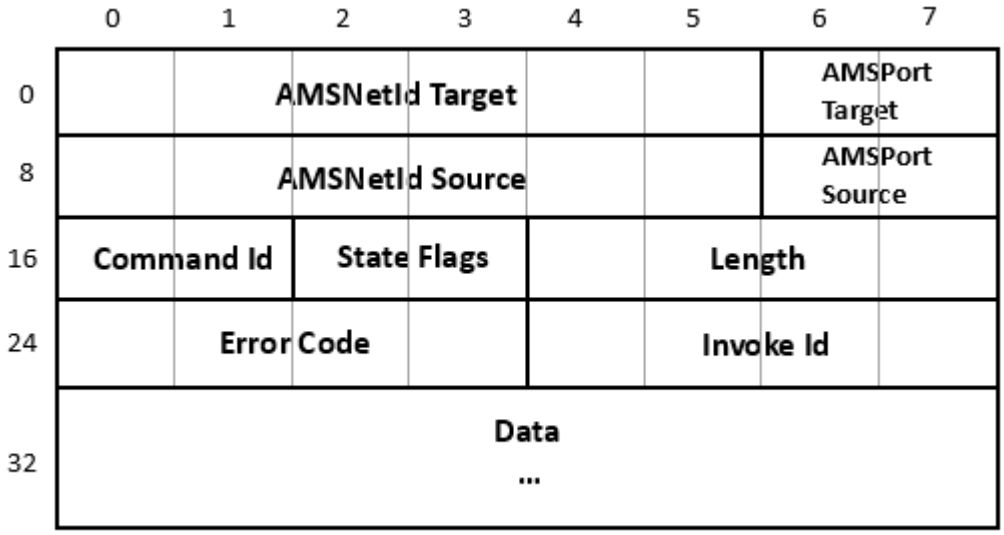
数据数组	大小	描述
AMS/TCP 报头	6 字节	包含数据包的长度。
AMS 报头	32 字节	AMS/TCP 报头包含发送器和接收器的地址。此外还有 AMS 错误代码、ADS 命令 ID 和其他一些信息。
ADS 数据	n 字节	ADS 数据范围包含单个 ADS 命令的参数。数据数组的结构取决于 ADS 命令。某些 ADS 命令不需要额外数据。

3.2.1.4.2 AMS/TCP 报头



数据数组	大小	描述
保留	2 字节	这些字节必须设置为 0。
长度	4 字节	该数组包含数据包的长度。它由 AMS 报头和所附 ADS 数据组成。单位是字节。

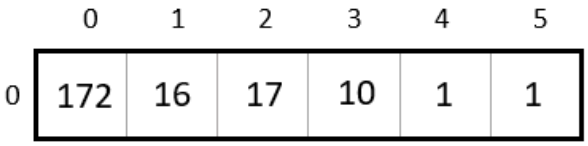
3.2.1.4.3 AMS 报头



数据数组	大小	描述
AMSNetId 目标	6 字节	这是数据包接收站的 AMSNetId。备注请参见下文 [▶ 22]。
AMSPort 目标	2 字节	这是数据包要接收站的 AMSPort。
AMSNetId 来源	6 字节	这包含数据包发出站的 AMSNetId。
AMSPort 来源	2 字节	这段信息包含了该站点发送该数据包的AMSPort。
命令 Id	2 字节	请参见下文 [▶ 23]。
状态旗标	2 字节	请参见下文 [▶ 23]。
数据长度	4 字节	数据范围的大小。单位为字节。
错误代码	4 字节	AMS 错误编号。请参见“ADS 返回代码”。
调用 Id	4 字节	可用的 32 位自由数组。通常，该数组用于发送一个 Id。有了这个 Id，就可以将收到的响应分配给之前发送的请求。
数据	n 字节	数据范围。数据范围包含考虑 ADS 命令的参数。

AMS Net Id

AMSNetId 由 6 个字节组成，并用于寻址发送器或接收器。例如，一个可能的 AMSNetId 是 172.16.17.10.1.1. 本例中的存储方式如下所示：



AMSNetId 是纯逻辑的，通常与 IP 地址无关。AMSNetId 在目标系统配置。为此，在 PC 上使用了 TwinCAT 系统控制器。如果使用其他硬件，请参见相关文档，了解有关 AMS NetId 设置的注意事项。

命令 Id

命令	描述
0x0000	无效
0x0001	ADS 读取设备信息 [▶ 23]
0x0002	ADS 读取 [▶ 24]
0x0003	ADS 写入 [▶ 25]
0x0004	ADS 读取状态 [▶ 25]
0x0005	ADS 写入控制 [▶ 26]
0x0006	ADS 添加设备通知 [▶ 26]
0x0007	ADS 删除设备通知 [▶ 27]
0x0008	ADS 设备通知 [▶ 28]
0x0009	ADS 读写 [▶ 29]

其他命令没有定义或不在内部使用。因此，命令 Id 只允许包含上述枚举值！

状态旗标

旗标	描述
0x0001	0：请求/1：响应
0x0004	ADS 命令

第一个位标志它是请求还是响应。第三个位必须设置为 1，以便与 ADS 命令交换数据。其他位没有定义或用于其他内部用途。

因此，其他位必须设置为 0！

旗标	描述
0x000x	TCP 协议
0x004x	UDP 协议

第 7 个位标志它是使用 TCP 还是 UDP 传输。

3.2.1.4.4 ADS 命令

3.2.1.4.4.1 命令概览

命令	描述
ADS 读取设备信息 [▶ 23]	读取 ADS 设备的名称和版本号。
ADS 读取 [▶ 24]	通过 ADS 读取，可从 ADS 设备读取数据
ADS 写入 [▶ 25]	通过 ADS 写入，可将数据写入 ADS 设备。
ADS 读取状态 [▶ 25]	读取 ADS 状态和 ADS 设备的设备状态。
ADS 写入控制 [▶ 26]	更改 ADS 状态和 ADS 设备的设备状态。
ADS 添加设备通知 [▶ 26]	在 ADS 设备中创建通知。
ADS 删除设备通知 [▶ 27]	在 ADS 设备中删除一个之前定义的通知。
ADS 设备通知 [▶ 28]	数据将从 ADS 设备独立传输到客户端
ADS 读写 [▶ 29]	通过 ADS 读写，数据将被写入 ADS 设备。此外，还可从 ADS 设备读取数据。

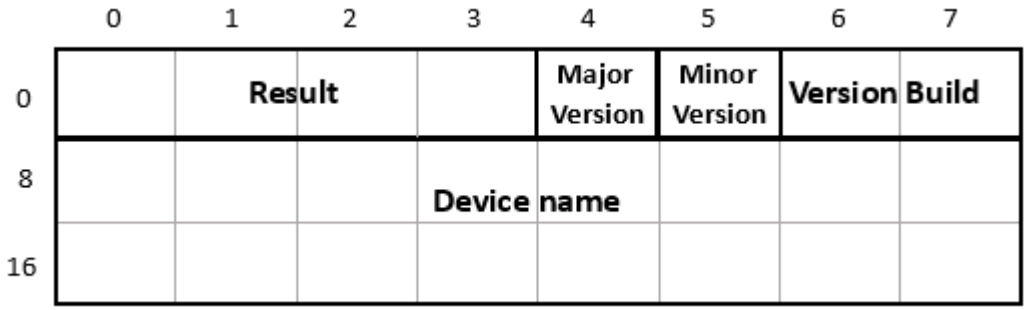
3.2.1.4.4.2 ADS 读取设备信息

读取 ADS 设备的名称和版本号。

请求

无需额外数据

响应

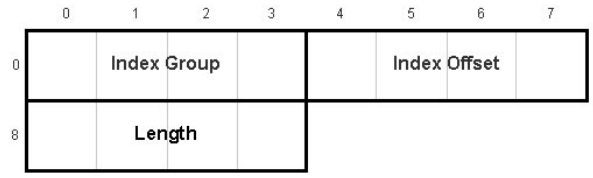


数据数组	大小	描述
结果	4 字节	ADS 错误编号。
主要版本	1 字节	主要版本号
次要版本	1 字节	次要版本号
版本构建	2 字节	构建号
设备名称	16 字节	ADS 设备的名称

3.2.1.4.4.3 ADS 读取

通过 ADS 读取，可从 ADS 设备读取数据。数据由索引组和索引偏移寻址

请求



数据数组	大小	描述
索引组	4 字节	应读取数据的索引组。
索引偏移	4 字节	应读取数据的索引偏移。
长度	4 字节	应读取数据的长度（以字节为单位）。

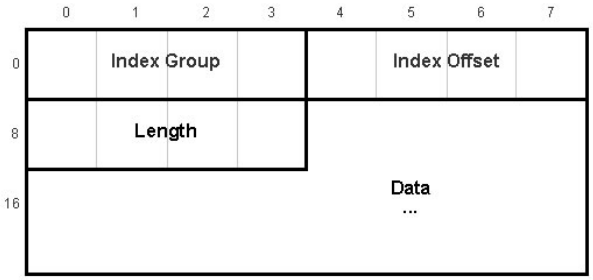
响应

数据数组	大小	描述
结果	4 字节	ADS 错误编号
长度	4 字节	传回数据的长度。
数据	n 字节	传回的数据。

3.2.1.4.4.4 ADS 写入

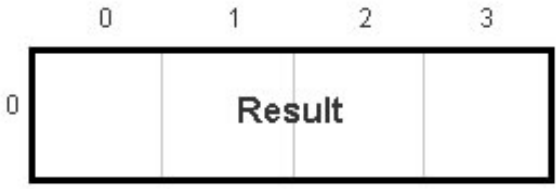
通过 ADS 写入，可将数据写入 ADS 设备。数据由索引组和索引偏移寻址

请求



数据数组	大小	描述
索引组	4 字节	应写入数据的索引组
索引偏移	4 字节	应写入数据的索引偏移
长度	4 字节	写入数据的长度（以字节为单位）
数据	n 字节	写入 ADS 设备的数据。

响应



数据数组	大小	描述
结果	4 字节	ADS 错误编号

3.2.1.4.4.5 ADS 读取状态

读取 ADS 状态和 ADS 设备的设备状态。

请求

无需额外数据

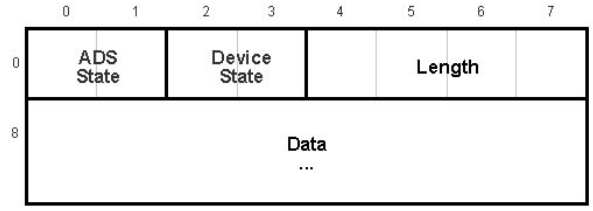
响应

数据数组	大小	描述
结果	4 字节	ADS 错误编号。
ADS 状态	2 字节	ADS 状态（请参见 ADS-DLL 的数据类型 ADSSTATE）。
设备状态	2 字节	设备状态

3.2.1.4.4.6 ADS 写入控制

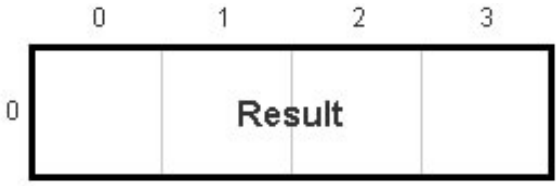
更改 ADS 状态和 ADS 设备的设备状态。此外，还可以向 ADS 设备发送数据，以传输更多信息。这些数据没有从当前的 ADS 设备（PLC、NC.....）进行分析

请求



数据数组	大小	描述
ADS 状态	2 字节	新 ADS 状态（请参见 ADS-DLL 的数据类型 ADSSTATE）。
设备状态	2 字节	新设备状态。
长度	4 字节	数据长度（以字节为单位）。
数据	n 字节	发送到 ADS 设备的附加数据

响应



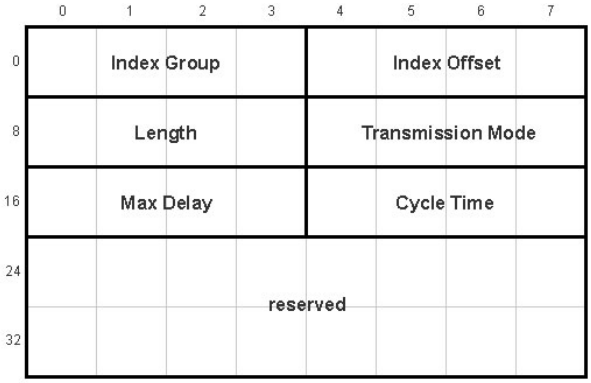
数据数组	大小	描述
结果	4 字节	ADS 错误编号。

3.2.1.4.4.7 ADS 添加设备通知

在 ADS 设备中创建通知。

注意：我们建议每个设备公布的通知不超过 550 个。否则，通过使用结构或 sum 命令来增加有效负载。

请求



数据数组	大小	描述
索引组	4 字节	每个通知应发送数据的索引组。
索引偏移	4 字节	每个通知应发送数据的索引偏移。
长度	4 字节	每个通知应发送数据的长度（以字节为单位）。
传输模式	4 字节	请参见 ADS-DLL 的 ADSTRANSMODE 结构描述。
最大延迟	4 字节	最迟在此时间之后，将调用 ADS 设备通知。单位是 1 ms。
循环时间	4 字节	ADS 服务器会检查值在该时间片内是否发生变化。单位是 1 ms
保留	16 字节	必须设置为 0

响应

数据数组	大小	描述
结果	4 字节	ADS 错误编号
通知句柄	4 字节	通知句柄

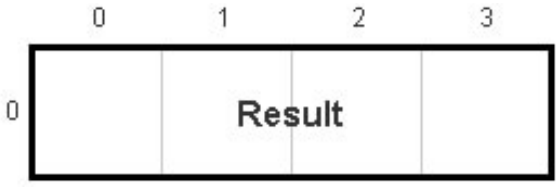
3.2.1.4.4.8 ADS 删除设备通知

在 ADS 设备中删除一个之前定义的通知。

请求

数据数组	大小	描述
通知句柄	4 字节	通知句柄。该句柄由 ADS 命令添加设备通知创建

响应



数据数组	大小	描述
结果	4 字节	ADS 错误编号

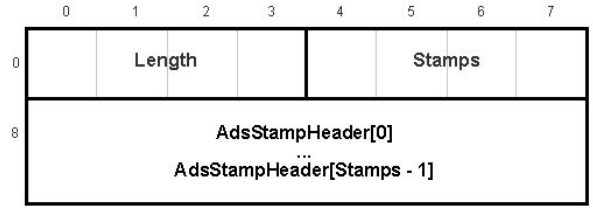
3.2.1.4.4.9 ADS 设备通知

数据将从 ADS 设备独立传输到客户端。

请求

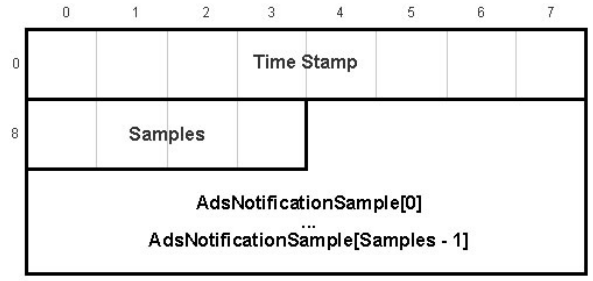
在设备通知处传输的数据是多重嵌套的。通知流包含一个数组，其中的元素类型为 AdsStampHeader。该数组同样包含 AdsNotificationSample 类型的元素。

AdsNotificationStream



数据数组	大小	描述
长度	4 字节	数据大小（以字节为单位）。
戳记	4 字节	AdsStampHeader [▶ 28] 类型的元素数量
AdsStampHeader	n 字节	包含 AdsStampHeader [▶ 28] 类型元素的数组

AdsStampHeader



数据数组	大小	描述
TimeStamp	8 字节	时间戳采用 Windows FILETIME 格式进行编码。也就是说，该值包含自 1.1.1601 以来的 100 纳秒间隔数。此外，也不考虑当地时间的变化。因此，时间戳表现为通用协调时间 (UTC)。
示例	4 字节	AdsNotificationSample [▶ 29] 类型的元素数量
AdsNotificationSample	n 字节	包含 AdsNotificationSample [▶ 29] 类型元素的数组

AdsNotificationSample

数据数组	大小	描述
通知句柄	4 字节	通知句柄。
样例大小	4 字节	数据范围的大小（以字节为单位）。
数据	n 字节	数据

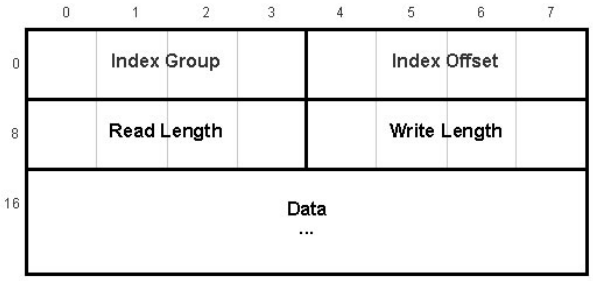
如果您的句柄失效，系统将会发送一个不带数据的提示通知作为建议。

3.2.1.4.4.10 ADS 读写

通过 ADS ReadWrite，数据将被写入 ADS 设备。此外，还可从 ADS 设备读取数据。

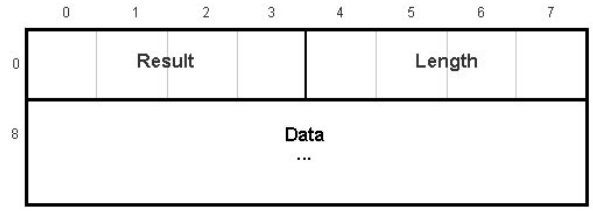
可读取数据由索引组和索引偏移寻址

请求



数据数组	大小	描述
索引组	4 字节	应写入数据的索引组。
索引偏移	4 字节	应写入数据的索引偏移
读取长度	4 字节	应读取数据的长度（以字节为单位）。
写入长度	4 字节	应写入数据的长度（以字节为单位）
数据	n 字节	写入 ADS 设备的数据。

响应



数据数组	大小	描述
结果	4 字节	ADS 错误编号
长度	4 字节	传回数据的长度。
数据	n 字节	传回的数据。

3.2.1.5 ADS 设备规范

3.2.1.5.1 一般信息

PLC 软件可以描述为虚拟现场单元（自动化设备），因为它是一个纯软件 PLC。因此，它为其他通讯伙伴（如其他虚拟现场单元或 Windows 程序）提供一个倍福 ADS（自动化设备规范）接口，通过该接口可以对其进行参数设置或查询。使用 ADS 标准可以统一访问 PLC，并将它整合到现有的虚拟现场单元中。

读取和写入操作通过两个数字在 PLC 接口（由 ADS 定义）上进行：索引组（16 位）和索引偏移（32 位）。关于 PLC 的 ADS 界面，我们将在下文中有关组和偏移指数的页面中进行详细介绍。

PLC 的“索引组”规范

ADS 单元的四个全局范围如下所示，PLC 作为索引组中的四个部分：

索引组（0x = hex）	索引组说明
0x00000000 0x00000FFF	保留
0x00001000	PLC ADS 参数范围
0x00002000	PLC ADS 状态范围
0x00003000	PLC ADS 单元功能范围
0x00004000	PLC ADS 服务（包括访问 PLC 内存范围（%M 字段）的服务） [▶ 30]
0x00006000 0x0000EFFF	为 PLC ADS 扩展保留
0x0000F000 0x0000FFFF	一般 TwinCAT ADS 系统服务（包括访问 PLC 物理输入和输出过程图的服务） [▶ 30]

3.2.1.5.2 PLC 服务规范

本节包括访问 PLC 内存范围（%M 字段）的服务。

索引组	索引偏移	访问	数据类型	描述	备注
0x00004020	0x00000000-0x0000FFFF	R/W	UINT8[n]	**READ_M - WRITE_M** PLC 内存范围（%M 字段）。偏移为字节偏移。	
0x00004021	0x00000000-0xFFFFFFFF	R/W	UINT8	**READ_MX - WRITE_MX** PLC 内存范围（%MX 字段）。索引偏移的低位字为字节偏移。索引偏移包含根据字节号 *8 + 位号计算出的位地址	
0x00004025	0x00000000	R	ULONG	**PLCADS_IGR_RMSIZE** 内存范围过程图的字节长度	
0x00004030	0x00000000-0xFFFFFFFF	R/W	UINT8	**PLCADS_IGR_RWRB** 保留数据范围。索引偏移是字节偏移	
0x00004035	0x00000000	R	ULONG	**PLCADS_IGR_RRSIZE** 保留范围的字节长度	
0x00004040	0x00000000-0xFFFFFFFF	R/W	UINT8	**PLCADS_IGR_RWDB** 数据范围。索引偏移为字节偏移。	
0x00004045	0x00000000	R	ULONG	**PLCADS_IGR_RDSIZE** 数据范围的字节长度	

3.2.1.5.3 ADS 系统服务规范

本节将介绍对每个 TwinCAT ADS 单元具有相同意义和作用的 ADS 服务。本节还包括访问 物理输入和输出PLC 过程图的服务。

索引 组	索引 偏移	访问	数据类型	描述
0x0000F003	0x00000000	读写	W： UINT8[n] R： UINT32	**GET_SYMHANDLE_BYNAME** 给写入数据中所包含的名称分配一个句柄（code word），并作为结果返回给调用者。
0x0000F004	0x00000000			保留。
0x0000F005	0x00000000-0xFFFFFFFF=symHandle	R/W	UINT8[n]	**READ_/WRITE_SYMVAL_BYHANDLE** 读取由 ,symHdl' 标识的变量值或为变量赋值。,symHdl' 必须首先由 GET_SYMHANDLE_BYNAME 服务确定。
0x0000F006	0x00000000	W	UINT32	**RELEASE_SYMHANDLE** 对于一个被查询的、有特定名称的 PLC变量，其写入数据中包含的代码（句柄）被释放。
0x0000F020	0x0001F400-0xFFFFFFFF	R/W	UINT8[n]	**READ_I - WRITE_I** 物理输入的 PLC 流程图（%I 字段）。偏移是字节偏移。
0x0000F021	0x000FA000-0xFFFFFFFF	R/W	UINT8	**READ_IX - WRITE_IX** 物理输入的 PLC 流程图（%IX 字段）。索引偏移包含位地址，其计算公式为基偏移 (0xFA000) + 字节号 + 8 + 位号
0x0000F025	0x00000000	R	ULONG	**ADSIGRP_IOIMAGE_RISIZEByte**物理输入的 PLC 流程图的字节长度。
0x0000F030	0x0003E800-0xFFFFFFFF	R/W	UINT8[n]	**READ_Q - WRITE_Q** 物理输出的 PLC 流程图（%Q 字段）。偏移是字节偏移。
0x0000F031	0x001F4000-0xFFFFFFFF	R/W	UINT8	**READ_QX - WRITE_QX** 物理输出的 PLC 流程图（%QX 字段）。索引偏移包含位地址，其计算公式为基偏移（0x1F4000）+ 字节号 *8 + 位号。
0x0000F035	0x00000000	R	ULONG	**ADSIGRP_IOIMAGE_ROSIZE** 物理输出 PLC 流程图的字节长度。

索引 组	索引 偏移	访问	数据类型	描述
0x0000F080	0x00000000- 0xFFFFFFFF= n （内部子命令数）n （最大值）= 500	读写	读取： n * ULONG[3] := IG1, IO1, Len1, IG2, IO2, Len2, ..., IG(n), IO(n), Len(n) 写入： n * ULONG + UINT8[Len1] + UINT8[Len2] + ..., + UINT8[Len(n)] := Result1, Result2, ..., Result(n), Data1, Data2, ..., Data(n)	**ADSIGRP_SUMUP_READ** 写入数据包含多个单独的 AdsReadReq(IG, IO, Len, Data) 子命令的列表。读取数据包含返回代码列表，后接请求数据。
0x0000F081	0x00000000- 0xFFFFFFFF= n （内部子命令数）n （最大值）= 500	读写	读取： (n * ULONG[3]) + UINT8[Len1] + UINT8[Len2] + ..., + UINT8[Len(n)] := **IG1, IO1, Len1,** **IG2, IO2, Len2,** **...,** **IG(n), IO(n), Len(n),** **Data1, Data2, ..., Data(n)** 写入： n * ULONG := **Result1, Result2, ...,** **Result(n)**	**ADSIGRP_SUMUP_WRITE** 写入数据包含多个单独的 AdsWriteReq(IG, IO, Len, Data) 子命令的列表。 读取数据包含返回代码列表。

索引 组	索引 偏移	访问	数据类型	描述
0x0000F082	0x00000000- 0xFFFFFFFF= n （内部子命令数）n （最大值）= 500	读写	读取： (n * ULONG[4]) + UINT8[WriteLen1] + UINT8[WriteLen2] + ..., + UINT8[WriteLen(n)] := **IG1, IO1, ReadLen1,** **WriteLen1,** **IG2, IO2, ReadLen2,** **WriteLen2,** **...,** **IG(n), IO(n), ReadLen(n), ...,** **WriteLen(n),** **WriteData1, WriteData2, ...,** **WriteData(n)** 写入： (n * ULONG[2]) + UINT8[ReturnLen1] + UINT8[ReturnLen2] + ..., + UINT8[ReturnLen(n)] := **Result1, ReturnLen1,** **Result2, ReturnLen2,** **...,** **Result(n), ReturnLen(n),** **Read**Data1, ReadData2, ..., ReadData(n)	**ADSIGRP_SUMUP_READWRITE** 写入数据包含多个单独的 AdsReadWriteReq(IG,IO,readLen,writeLen,Data) 子命令的列表。 读取数据包含返回代码和返回数据长度列表，后接请求数据。
0x0000F083	0x00000000- 0xFFFFFFFF= n （内部子命令数）n （最大值）= 500	读写	读取： n * ULONG[3] := IG1, IO1, Len1, IG2, IO2, Len2, ..., IG(n), IO(n), Len(n) 写入： n * ULONG + UINT8[Len1] + UINT8[Len2] + ..., + UINT8[Len(n)] := Result1, Result2, ..., Result(n), Data1, Data2, ..., Data(n)	**ADSIGRP_SUMUP_READEX** 写入数据包含多个单独的 AdsReadReq(IG, IO, Len, Data) 子命令的列表。读取数据包含返回代码列表，后接请求数据。

索引 组	索引 偏移	访问	数据类型	描述
0x0000F084	0x00000000- 0xFFFFFFFF= n （内部子命令数）n （最大值）= 500	读写	读取： n * ULONG[3] := IG1, IO1, Len1, IG2, IO2, Len2, ..., IG(n), IO(n), Len(n) 写入： n * ULONG + UINT8[Len1] + UINT8[Len2] + ..., + UINT8[Len(n)] := Result1, Result2, ..., Result(n), Data1, Data2, ..., Data(n)	**ADSIGRP_SUMUP_READEX2** 写入数据包含多个单独的 AdsReadReq(IG, IO, Len, Data) 子命令的列表。读取数据包含返回代码列表，后接请求数据。
0x0000F085	0x00000000- 0xFFFFFFFF= n （内部子命令数）n （最大值）= 500	读写	读取： (n * ULONG[3]) := **IG1, IO1, Len1,** **IG2, IO2, Len2,** **...,** **IG(n), IO(n), Len(n)** 写入： (n * ULONG) + UINT8[Len1] + UINT8[Len2] + ..., + UINT8[Len(n)] := **Result1, Result2, ...,** **Result(n),** **Handle1, Handle2,...,** **Handle(n)**	**ADSIGRP_SUMUP_ADDDEVNOTE** 写入数据包含多个单独的 AdsAddDeviceNotifications(IG, IO, Len, Data) 子命令的列表。读取数据包含返回代码列表，后接请求通知句柄。
0x0000F086	0x00000000- 0xFFFFFFFF= n （内部子命令数）n （最大值）= 500	读写	读取： **Handle1, Handle2,...,** **Handle(n)** 写入： (n * ULONG) + UINT8[Len1] + UINT8[Len2] + ..., + UINT8[Len(n)] := **Result1, Result2, ...,** **Result(n)**	**ADSIGRP_SUMUP_DELDEVNOTE** 写入数据包含多个句柄的列表。读取数据包含返回代码列表。

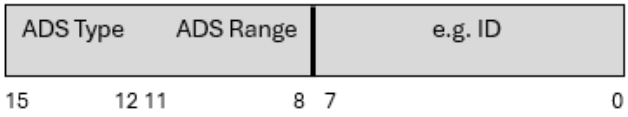
3.2.1.5.4 NC 的规范

本文档包含所有 TC3 特定修改和新功能。

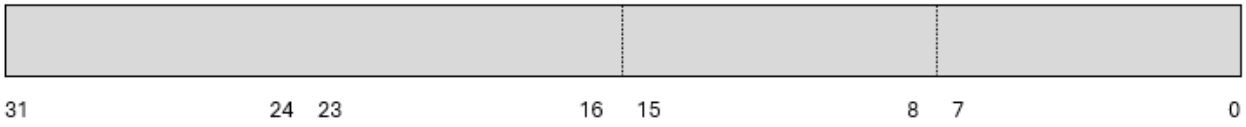
索引组（十六进制）	描述	备注
0x1000	Ring-0-Manager：参数 [▶ 37]	可选！
0x1100	Ring-0-Manager：状态 [▶ 38]	可选！
0x1200	Ring-0-Manager：功能 [▶ 38]	可选！
0x1300	Ring-0-Manager：循环性过程数据	未执行！
0x2000 + ID	具有相应 ID 的通道：参数 [▶ 39]	
0x2100 + ID	具有相应 ID 的通道：状态 [▶ 42]	
0x2200 + ID	具有相应 ID 的通道：功能 [▶ 43]	
0x2300 + ID	具有相应 ID 的通道：循环性过程数据 [▶ 45]	
0x3000 + ID	具有相应 ID 的组：参数 [▶ 46]	可选！
0x3100 + ID	具有相应 ID 的组：状态 [▶ 50]	可选！
0x3200 + ID	具有相应 ID 的组：功能 [▶ 55]	可选！
0x3300 + ID	具有相应 ID 的组：循环性过程数据	未执行！
0x4000 + ID	具有相应 ID 的轴：参数 [▶ 60]	
0x4100 + ID	具有相应 ID 的轴：状态 [▶ 72]	
0x4200 + ID	具有相应 ID 的轴：功能 [▶ 81]	
0x4300 + ID	具有相应 ID 的轴：循环性过程数据 [▶ 100]	
0x5000 + ID	具有相应 ID 的编码器：参数 [▶ 104]	可选！
0x5100 + ID	具有相应 ID 的编码器：状态 [▶ 108]	可选！
0x5200 + ID	具有相应 ID 的编码器：功能 [▶ 112]	可选！
0x5300 + ID	具有相应 ID 的编码器：循环性过程数据 [▶ 115]	可选！
0x6000 + ID	具有相应 ID 的控制器：参数 [▶ 119]	可选！
0x6100 + ID	具有相应 ID 的控制器：状态 [▶ 123]	可选！
0x6200 + ID	具有相应 ID 的控制器：功能 [▶ 126]	可选！
0x6300 + ID	具有相应 ID 的控制器：循环性过程数据	未执行！
0x7000 + ID	具有相应 ID 的驱动器：参数 [▶ 127]	可选！
0x7100 + ID	具有相应 ID 的驱动器：状态 [▶ 131]	可选！
0x7200 + ID	具有相应 ID 的驱动器：功能 [▶ 132]	可选！
0x7300 + ID	具有相应 ID 的驱动器：循环性过程数据 [▶ 133]	可选！
0x0A0**00** + **ID**	表具有相应 ID 的表 (n x m)：参数 [▶ 136] 0x0A000+ID 表示表 ID [1..255] 0x1A000+ID 表示表 ID [256..4095] . . . 0xFA000+ID 表示表 ID [3840..4095]	最大表数扩展至 4095（TC3.1 B4021 及以上版本）
0x**0**A1**00** + **ID**	具有相应 ID 的表 (n x m)：状态 [▶ 141] 0x0000A100+IDLowByte 表示表 ID [1..255] 0x0001A100+IdLowByte 表示表 ID [256..4095] … 0x000FA100+IdLowByte 表示表 ID [3840..4095] 0x000**n**A1**00**+**IdLowByte** 表示表 ID [1..4095] (TabID = n * 256 + IdLowByte)	

索引组（十六进制）	描述		备注
0x0A200 + ID	具有相应 ID 的表 (n x m)：功能 [▶ 142] 0x0000A100+IDLowByte 表示表 ID [1..255] 0x0001A100+IdLowByte 表示表 ID [256..4095] … 0x000FA100+IdLowByte 表示表 ID [3840..4095] 0x000nA100+IdLowByte 表示表 ID [1..4095] (TabID = n * 256 + IdLowByte)		
0x0A300 + ID	具有相应 ID 的表 (n x m)：循环性过程数据 0x0000A100+IDLowByte 表示表 ID [1..255] 0x0001A100+IdLowByte 表示表 ID [256..4095] … 0x000FA100+IdLowByte 表示表 ID [3840..4095] 0x000nA100+IdLowByte 表示表 ID [1..4095] (TabID = n * 256 + IdLowByte)		未执行!
0xF000 ... 0xFFFF	保留区域（TwinCAT 系统区域）		
IndexGroup:	IndexOffset:		
0xF081	0x00000000 ... 0xFFFFFFFF （n 个元素）	ADSIGRP_SUMUP_WRITE 读写命令在多个单独的 ADS 写入命令（如组请求）的写入数据中包含一个列表。 写入数据结构： [IdxGrp(1), IdxOff(1), WriteLen(1), ..., IdxGrp(n), IdxOff(n), WriteLen(n), WriteData(1), ..., WriteData(n)] 读取数据结构： [Error(1), ..., Error(n)]	
0xF082	0x00000000 ... 0xFFFFFFFF （n 个元素）	ADSIGRP_SUMUP_READWRITE 读写命令在多个单独的 ADS 读写命令（如组请求）的写入数据中包含一个列表。 写入数据结构： [IdxGrp(1), IdxOff(1),ReadLen(1), WriteLen(1), ..., IdxGrp(n), IdxGrp(n), ReadLen(n), WriteLen(n), WriteData(1), ..., WriteData(n)] 读取数据结构： [Error(1), ReadLen(1), ..., Error(n), ReadLen(n), ReadData(1), ..., ReadData(n)]	
0xF084	0x00000000 ... 0xFFFFFFFF （n 个元素）	ADSIGRP_SUMUP_READ (READEX2) 读写命令在多个单独的 ADS 写入命令（如组请求）的写入数据中包含一个列表。 写入命令结构： [IdxGrp(1), IdxOff(1), ReadLen(1), ..., IdxGrp(n), IdxGrp(n), ReadLen(n)] 读取命令结构： [Error(1), ReadLen(1), ..., Error(n), ReadLen(n), ReadData(1), ..., ReadData(n)]	

Index Group:



Index Offset:



3.2.1.5.4.1 Ring-0-Manager 规范

3.2.1.5.4.1.1 *Ring-0 参数的"索引偏移"规范（索引组 0x1000）*

索引偏移（十六进制）	访问	Ring-0-Manager	数据类型	物理单位	定义范围	描述	备注
0x00000010	读取	每个	UINT32	100 ns		循环时间 SAF 任务	
0x00000012	读取	每个	UINT32	100 ns		循环时间 SVB 任务	
0x00000014	读取	每个	INT32	ns		全局时间补偿漂移（针对 SAF 任务）	
0x00000020	读/写	每个	UINT16	1	0/1	循环数据一致性检查和 NC 设定点值校正	

3.2.1.5.4.1.2 *Ring-0 状态的"索引偏移"规范（索引组 0x1100）*

索引偏移（十六进制）	访问	Ring-0-Manager	数据类型	物理单位	定义范围	描述	备注
0x00000001	读取	每个	UINT32	1	0, 1...255	通道数量	
0x00000002	读取	每个	UINT32	1	0, 1...255	组的数量	
0x00000003	读取	每个	UINT32	1	0, 1...255	轴的数量	
0x00000004	读取	每个	UINT32	1	0, 1...255	编码器的数量	
0x00000005	读取	每个	UINT32	1	0, 1...255	控制器的数量	
0x00000006	读取	每个	UINT32	1	0, 1...255	驱动器的数量	
0x0000000A	读取	每个	UINT32	1	0, 1...255	表的数量（n x m）	
0x00000010	读取	每个	UINT32	1		循环时间错误计数器 SAF 任务（不可范围化）	保留！
0x00000014	读取	每个	UINT32	1		IO 循环时间错误计数器 SAF 任务（不可范围化）	保留！
0x00000020	读取	每个	UINT32	s		计算时间 SAF 任务（不可范围化）	保留！
0x00000031	读取	每个	UINT32[n]	1	0, 1...255	为系统中所有通道提供通道 ID	
0x00000032	读取	每个	UINT32[n]	1	0, 1...255	为系统中所有组提供组 ID	
0x00000033	读取	每个	UINT32[n]	1	0, 1...255	为系统中所有轴提供轴 ID	
0x00000034	读取	每个	UINT32[n]	1	0, 1...255	为系统中所有编码器提供编码器 ID	
0x00000035	读取	每个	UINT32[n]	1	0, 1...255	为系统中所有控制器提供控制器 ID	
0x00000036	读取	每个	UINT32[n]	1	0, 1...255	为系统中所有驱动器提供驱动器 ID	
0x0000003A	读取	每个	UINT32[n]	1	0, 1...255	为系统中所有表格提供表格 ID	
0x000001nn	读取	每个	UINT32	1	0, 1...255	为编码器 ID 提供相应的轴 IDnn = 编码器 ID	保留！
0x000002nn	读取	每个	UINT32	1	0, 1...255	为控制器 ID 提供相应的轴 IDnn = 控制器 ID	保留！
0x000003nn	读取	每个	UINT32	1	0, 1...255	为驱动器 ID 提供相应的轴 IDnn = 驱动器 ID	保留！

3.2.1.5.4.1.3 *适用于 Ring-0 功能的"索引偏移"规范（索引组 0x1200）*

索引偏移（十六进制）	访问	Ring-0-Manager	数据类型	物理单位	定义范围	描述	备注
0x00000020	写入	每个	VOID	1		清除循环时间错误计数器 SAF 和 SVB	保留！

3.2.1.5.4.2 通道规范

3.2.1.5.4.2.1 *通道参数的"索引偏移"规范（索引组 0x2000 + ID）*

索引偏移（十六进制）	访问	通道类型	数据类型	物理单位	定义范围	描述	备注
0x00000001	读取	每个	UINT32	1		通道 ID	
0x00000002	读取	每个	UINT8[30+1]	1		通道名称	
0x00000003	读取	每个	UINT32	1	枚举	通道类型 [▶ 142]	
0x00000004	读取	每个	UINT32	1	枚举	解释器类型 [▶ 142]	
0x00000005	读取	每个	UINT32	1		程序加载缓冲区大小（以字节为单位）	
0x00000006	读取	每个	UINT32	1		根据工作清单划分的程序编号	
0x00000007	读/写	每个	UINT32	1	枚举	设置加载日志模式 [▶ 143]	
0x00000008	读/写	每个	UINT32	1	枚举	设置追踪模式 [▶ 143]	
0x00000009	读/写	每个	UINT32	1		保留	
0x0000000A	读/写	每个	UINT32	1	0/1	在名为"TcNci.log"的日志文件中记录所有进给器操作记录	
0x0000000B	读/写	每个	UINT32	1	0/1	NC 记录器消息的通道特定级别 0：仅错误 1：所有 NC 消息	
0x00000010	读写	每个	**写入**				
			{				
			UINT32	1	0..159	M 函数的起始索引	
			UINT32	1	1..160	要读取的 M 函数数	
			}				
			读取 [n]				
			{				
			UINT8	1	0..159	M 函数的规则位掩码	
			INT32[10]	1	-1..159	要清除的 M 函数数	
			}				
0x00000011	写入	插补				写入 M 函数说明	仅在内部使用！
0x00000012	读/写	插补	LREAL64	1		G70 系数	
0x00000013	读/写	插补	LREAL64	1		G71 系数	
0x00000014	写入	插补	{			轴的用户自定义符号	尚未发布
			char[32]			用户符号（空字符结尾）	
			char[10]			系统符号（空字符结尾）	
			}				
0x00000015	读/写	插补	UINT16 resp. UINT32	1	0/1 默认：FALSE	激活默认 G 代码	从 TC3.1 B4014 起更新
0x00000021	读取	每个	UINT32	1		组 ID（仅对 3D 和 FIFO 通道显式）	
0x00000031	读/写	插补	UINT16	1		解释器的标准输出端口	保留功能，无标准！
0x00000032	读/写	插补	UINT16	1	0/1	笛卡尔工具偏移输入	保留功能，无标准！
0x00000040	读/写	插补	{			**解释器钩子的目标地址**	保留功能，无标准！
			char[6]			Ams Net ID	
			UINT16			端口	
			UINT32			索引组	
			UINT32			索引偏移	
			}				

索引偏移（十六进制）	访问	通道类型	数据类型	物理单位	定义范围	描述	备注
0x00000050	读/写	插补	UINT32	1	枚举	如果在半径补偿过程中识别到瓶颈，会做出反应 0：错误和中止 1：注意和故障排除 2：仅注意，不进行轮廓调制	
0x00000051	读/写	插补	UINT32	1	1..24	预期瓶颈检测	
0x00000052	读/写	插补	UINT32	1	0/1	倒角开启/关闭	保留功能，无标准！
0x00000053	读/写	插补	UINT32	1		激活以读取当前有效插补规则、零点漂移和旋转 0：关闭 1：打开	
0x00000054	读/写	插补	UINT32	1	0/1	回描开启/关闭	保留功能，无标准！
0x00000055	读/写	插补	UINT32[4]	1		为 UINT32 配置循环通道接口；最多可配置 4 个索引偏移。	
0x00000056	读/写	插补	UINT32[4]	1		为 LREAL 配置循环通道接口；最多可配置 4 个索引偏移。	
0x00010K0L	读/写	每个	REAL64	例如 mm	±MAX REAL64	零点漂移值（NPV）	
					[1..3]	轴索引 K=1 → X K=2 → Y K=3 → Z	
					[1..0xA]	L=1 → G54F L=2 → G54G L=3 → G55F ...	
0x0002ww00	读/写	每个	UINT16			工具编号：工具补偿值	
0x0003ww00	读/写	每个	UINT16		[1...50]	工具类型 ww = 工具 1...50	
0x0004wwnn	读/写	每个	REAL64		[1...14]	参数： nn = 索引 1...14	
0x000500gg	读/写	每个	REAL64	例如 mm	⩾ 0（值） [1...9] (g)	公差球面半径 gg = 通道组（默认：1）	

3.2.1.5.4.2.2 *通道状态的"索引偏移"规范（索引组 0x2100 + ID）*

索引偏移（十六进制）	访问	通道类型	数据类型	物理单位	定义范围	描述	备注
0x00000001	读取	每个	INT32	1	枚举	错误代码通道	
0x00000002	读取	每个	UINT32	1		通道中的组数	
0x00000003	读取	每个	UINT32	1	枚举	解释器状态 [▶ 143]	无法用示波器追踪！
0x00000004	读取	每个	UINT32	1	枚举	解释器/通道运行模式 [▶ 143]	
0x00000005	读取	每个	UINT32	1		当前加载的程序	
0x00000007	读取	每个	UINT8[...]	1		当前加载程序的程序名（100 个字符，空字符结尾）	最多 100 个字符，空字符结尾
0x00000008	读取	解释器	UINT32	1	[0,1]	解释器模拟模式 0：关闭（默认） 1：打开	无法用示波器追踪！
0x00000010	读取	解释器	UINT32	1		文本索引 如果解释器处于中止状态，则可在此读出当前文本索引	无法用示波器追踪！
0x00000011	读写	解释器	**写入**				无法用示波器追踪！
			UINT32	1		文本索引	
			读取				
			UINT8[..]	1		来自文本索引的 NC 零件程序行	
0x00000012	读取	解释器	{				
			UINT32	1		当前显示为 1：SAF 2：解释器 3：误差偏移	
			UINT32	1		文件偏移	
			UINT8[260]	1		路径 + 程序名称	
			}				
0x00000013	读取	解释器	UINT32[18]			显示当前有效的 G 代码	
0x00000014	读取	解释器	{			确定当前有效的零点漂移	
			UINT32	1		块计数器	
			UINT32			虚拟	
			LREAL[3]	1		零点漂移 G54...G57	
			LREAL[3]	1		零点漂移 G58	
			LREAL[3]	1		零点漂移 G59	
			}				
0x00000015	读取	解释器	{			确定当前有效的旋转	
			UINT32	1		块计数器	
			UINT32	1		虚拟	
			LREAL[3]	1		X、Y 和 Z 的旋转（以度数为单位）	
			}				
0x00000016	读取	解释器	UINT32	1	[0,1]	进给器信息	仅在内部使用！不标准
0x00000100	读取	每个	UINT32 [n]	1	[0, 1...255]	在通道编号中返回相应的轴 ID：[1...255] 轴 ID：[0, 1...255]	无法用示波器追踪！

3.2.1.5.4.2.3 *通道功能的"索引偏移"规范（索引组 0x2200 + ID）*

索引偏移（十六进制）	访问	通道类型	数据类型	物理单位	定义范围	描述	备注
0x00000001	写入	每个	UINT32	1		用程序编号加载 NC 程序	
0x00000002	写入	每个	VOID			启动解释器	
0x00000003	写入	每个	VOID			保留	
0x00000004	写入	每个	UINT8[...]			按名称加载 NC 程序。尽管可以，但标准的NC路径不是必需要提供的。也允许其他路径。	
0x00000005	写入	每个	UINT16	枚举	参见解释器运行模式附录 [▶ 143]	设置解释器/通道运行模式	
0x00000006	写入	解释器	UINT8[...]			设置子程序路径	
0x00000008	写入	解释器	UINT32	1		解释器模拟模式： 0：关闭（默认） 1：打开	尚未发布
0x0000000F	写入	每个	VOID			保留	
0x00000010	写入	每个	VOID			"重置"通道	
0x00000011	写入	每个	VOID			“停止”通道	
0x00000012	写入	每个	VOID			"重试"通道（重新启动通道）	
0x00000013	写入	每个	VOID			"跳过"通道（跳过任务/块）	
0x00000014/0x00000015	写入	每个	{			"启用回描"/"禁用回描"	保留功能，无标准！
			UINT32	1	>0	进给器方向： 1：向前 2：向后	
			UINT32	1	⩾ 0	条目索引	
			REAL64[3]	mm	±∞	主轴 X、Y、Z 的位置	
			REAL64[5]	mm	±∞	辅助轴 Q1、...、Q5 的位置	
			}				
0x00000020	写入	每个	VOID			"保存"零点偏移 量 (NPV)	
0x00000021	写入	每个	VOID			"加载"零点偏移量(NPV)	
0x00000022	写入	每个	VOID			保存”工具补偿	
0x00000023	写入	每个	VOID			“加载”工具补偿	
0x00000024	写入	插补	{			在给定文件中保存解释器快照	
			char[32]			TwinCAT\CNC 文件夹中的文件名	
			UINT32	1	0..1	掩码： 0x1: R-Parameters 0x2: Zeroshifts 0x4: Tool Desc	
			}				
0x00000025	写入	插补	{			将给定文件快照读入解释器	
			char[32]			TwinCAT\CNC 文件夹中的文件名	
			UINT32	1	0..1	掩码： 0x1: R-Parameters 0x2: Zeroshifts 0x4: Tool Desc	
			}				
0x00000026	写入	插补	VOID			将所有工具参数（包括类型和编号）设置为空	

索引偏移（十六进制）	访问	通道类型	数据类型	物理单位	定义范围	描述	备注
0x00000027	写入	插补	VOID			将所有零点偏移 量设置为空	
0x00000030	写入	每个	VOID			编程 解释器停止后重新启动（继续执行程序）解释器	
0x00000040	写入	每个	VOID			删除 NCI 中任何剩余行程的触发器事件	
0x00000041	写入	每个				为 特定事件保留	
0x00000050	写入	插补	VOID	1		在解释器中设置 ExecIdleInfo	保留功能，无标准！
0x00000051	写入	插补	UINT32	1		在解释器参数中设置块跳过掩码：SkippingMask	保留功能，无标准！
0x00000052	写入	内插法	UINT32	1		在解释器参数中设置 ItpOperationMode：OperationMode mask	保留功能，无标准！
0x00000053	写入	插补	VOID			在 NC 设备中设置 ScanningFlag	保留功能，无标准！
0x00000054	写入	插补				扫描位置	保留功能，无标准！
			double[8]			位置	
0x00000055	写入	插补				预留	
0x00000056	写入	插补	VOID			在中止状态中设置解释器	保留功能，无标准！
0x00000060	写入	插值	UINT16	1	0..159	手动重置快速 M 函数	

3.2.1.5.4.2.4 *循环通道处理数据的"索引偏移"规范（索引组 0x2300 + ID）*

索引偏移（十六进制）	访问	通道类型	数据类型	物理单位	定义范围	描述	备注
0x00000000	读取	每个 (PLC→NC)	{128 字节}		STRUCT 参见通道接口	通道结构 (PLC→NC) 备注：大小和对齐方式已更改。	当前 PLC 结构： NciChannelFromPlc plctonc_ncichannel_ref
0x00000001	读取	每个	UINT8[...] 最少 30 字节	1		解释器程序显示	无法用示波器追踪！
0x00000002	读/写	每个 (PLC→NC)	UINT32	%	[0...1000000]	速度超驰通道（通道中的轴）	1000000 = 100%
0x00000003	读/写	每个 (PLC→NC)	UINT32	%	[0...1000000]	速度超驰主轴	1000000 = 100%
0x00000080	读取	每次（NC→PLC）	{160 字节}		STRUCT 参见通道接口	通道结构 (NC→PLC) 备注：大小和对齐方式已更改。	当前 PLC 结构： NciChannelToPlc nctoplc_ncichannel_ref
0x10000000 +RegIndex	读/写	每个	REAL64	1	[0...999]	解释器的 R 参数	无法用示波器追踪！
0x20000001	读取	每个	UINT8[...] 最少 30 字节	1	[1...9]	组警示处理 (SAF) 的程序显示	无法用示波器追踪！

3.2.1.5.4.3 规范组

3.2.1.5.4.3.1 *组参数的"索引偏移"规范（索引组 0x3000 + ID）*

索引偏移（十六进制）	访问	组类型	数据类型	物理单位	定义范围	描述	备注
0x00000001	读取	每个	UINT32	1		组 ID	
0x00000002	读取	每个	UINT8[30+1]	1		组名称	
0x00000003	读取	每个	UINT32	1	枚举	组类型 [▶ 143]	
0x00000004	读取	每个	UINT32	µs		SAF 循环时间组	
0x00000005	读取	每个	UINT32	µs		SVB 循环时间组	
0x00000006	读/写	每个	UINT16	1	0/1	单块运行模式?	
0x0000000B	读取	每个	UINT32	1		SVB 表的大小（SVB 条目的最大数量）	
0x0000000C	读取	每个	UINT32	1		SAF 表的大小（SVB 条目的最大数量）	
0x00000010	读/写	每个	UINT32	1	[1,2...32] 默认：1	内部 SAF 循环时间除数（将内部 SAF 循环时间除以该系数）	例如，DXD 组
0x00000021	读取	通道：每个	UINT32	1		通道 ID	
0x00000022	读取	通道：每个	UINT8[30+1]	1		通道名称	
0x00000023	读取	通道：每个	UINT32	1	枚举	通道类型 [▶ 142]	
0x00000024	读取	通道：每个	UINT32	1	>0	通道中的数字	
0x00000500	读/写	DXD 组	INT32	枚举	[0, 1]	转向速度减小法 [▶ 143] 0：库仑散射 1：余弦定律 2：速度跳变	
0x00000501	读/写	DXD 组	REAL64	1	[0.0...1.0]	速度减小系数 C0 转换（连续，但既不是连续可微一次，也不是连续可微两次）	
0x00000502	读/写	DXD 组	REAL64	1	[0.0...1.0]	速度减小系数 C1 转换（连续和连续可微一次）	
0x00000503	读/写	DXD 组	REAL64	程度	[0.0...180.0]	分段转换临界角"低"（必须严格小于或等于减速角 C0）	
0x00000504	读/写	DXD 组	REAL64	程度	[0.0...180.0]	分段转换临界角"高"（必须严格小于或等于减速角 C0）	
0x00000505	读/写	DXD 组	REAL64	mm/s	≥ 0	最小速度，在分段转换时，即使速度可能降低，也不能低于最小速度。	注意：参数不会保存在解决方案中，也不会作为 NC 启动参数传输！
0x00000506	读/写	DXD 组	REAL64	例如 mm	[0.0...1000.0]	用于混合的公差球面半径	未执行！
0x00000507	读/写	DXD 组	REAL64	1		速度减小系数 C2 转换	
0x00000508	读/写	DXD 组	UINT16	1	0/1	可计算总剩余路径长度	从 TC3.1 B4020.40 起更新
0x00000509	读/写	DXD 组	UINT16	1	0/1 默认：1	主轴（X、Y、Z）软件限制位置监测的一般激活（请参见编码器参数）	
0x0000050A	读/写	DXD 组	UINT32	1	0/1	NCI 速率比类型 0：与内部降速有关（无迭代） 1：与原始外部（编程）速度有关 2：与内部降速有关（0 ... >100%）	

索引偏移（十六进制）	访问	组类型	数据类型	物理单位	定义范围	描述	备注
0x0000050C	读取	DXD 组	UINT32	1	[128 ... 1024] 默认：128	用户定义 NCI SAF 表条目最大数	从 TC3.1 B4014 起的启动参数有更新
0x00000510	读/写	DXD 组	REAL64	1	≥ 0	对于减小方法 VeloJump C0 转换的减小系数：X 轴	未执行！
0x00000511	读/写	DXD 组	REAL64	1	≥ 0	对于减小方法 VeloJump C0 转换的减小系数：Y 轴	未执行！
0x00000512	读/写	DXD 组	REAL64	1	≥ 0	对于减小方法 VeloJump C0 转换的减小系数：Z 轴	未执行！
0x00000513	读/写	DXD 组	LREAL64	1	]0.0..1.0[	辅助轴混合：如果有效路径速度小于编程速度与此系数相乘的结果，则插入一个准停并删除公差球	尚未发布
0x00000514	读/写	DXD 组	UINT32	1	[1 ⋯ 20] 默认：1	每个 NC 循环（从 SVB 到 SAF）传送作业的最大数	从 TC3.1 B4020.40 起更新
0x00000604	读/写	编码器组	REAL64	例如 mm/s	[0.0...1000.0]	速度窗口或静止窗口	基本单位 / s
0x00000605	读/写	编码器组	REAL64	s	[0.0...60.0]	静止窗口的滤波时间（以秒为单位）	
0x00000606	读/写	编码器组	REAL64	s	[0.0...60.0]	死区时间补偿主轴/从轴耦合（“角度预控制”）	
0x00000701	读取	FIFO 组	UINT32	1	[1...16]	FIFO 尺寸（m = 轴数） 注：FIFO 尺寸增至 16。	(n x m) FIFO 启动数据
0x00000702	读取	FIFO 组	UINT32	1	[1...10000]	FIFO 大小（长度）（n = FIFO 条目数）	(n x m) FIFO 启动数据
0x00000703	读取	FIFO 组	UINT32	1	[0, 1, 4]	FIFO 设定点生成器的插值类型 0: INTERPOLATIONTYPE_LINEAR（默认） 1: INTERPOLATIONTYPE_4POINT 4: INTERPOLATIONTYPE_CUBICSPLINE（带 6 个点）	从 TC3.1 B4020 起更新
0x00000704	读/写	FIFO 组	UINT32	1	[1, 2]	FIFO 设定点生成器的速率比类型 类型 1: OVERRIDETYPE_INSTANTANEOUS（默认） 类型 2: OVERRIDETYPE_PT2	
0x00000705	读/写	FIFO 组	REAL64	s	> 0.0	超驰变化的 P-T2 时间（T1=T2=T0）	

索引偏移（十六进制）	访问	组类型	数据类型	物理单位	定义范围	描述	备注
0x00000706	读/写	FIFO 组	REAL64	s	≥ 0.0	两个有序 FIFO 条目的时间差值（FIFO 条目时基）	
0x00000801	读写	运动学组	**写入**			计算位置的运动学正向转换（ACS -> MCS）	
			{				
			REAL64[8]	例如度数	±∞	ACS（轴坐标系）轴位置，最大尺寸：8	
			UINT32	1	≥ 0	保留	
			UINT32	1	≥ 0	保留	
			}				
			读取				
			{				
			REAL64[8]	例如 mm	±∞	MCS（机器坐标系）轴位置，最大尺寸：8	
			UINT32	1	≥ 0	保留	
			UINT32	1	≥ 0	保留	
			}				
0x00000802	读写	运动学组	**写入**			计算位置的运动学逆向转换 (MCS -> ACS)	
			{				
			REAL64[8]	例如 mm	±∞	MCS（机器坐标系）轴位置，最大尺寸：8	
			UINT32	1	≥ 0	保留	
			UINT32	1	≥ 0	保留	
			}				
			读取				
			{				
			REAL64[8]	例如度数	±∞	ACS（轴坐标系）轴位置，最大尺寸：8	
			UINT32	1	≥ 0	保留	
			UINT32	1	≥ 0	保留	
			}				

3.2.1.5.4.3.2 *组状态的"索引偏移"规范（索引组 0x3100 + ID）*

索引偏移（十六进制）	访问	组类型	数据类型	物理单位	定义范围	描述	说明
0x00000001	读取	每个	INT32	1	枚举	错误代码组	
0x00000002	读取	每个	UINT32	1		主轴数	
0x00000003	读取	每个	UINT32	1		从轴数	
0x00000004	读取	每个	UINT32	1	参见 枚举	SVB 组状态（状态）	
0x00000005	读取	每个	UINT32	1	参见 枚举	SAF 组状态（主状态）	
0x00000006	读取	每个	UINT32	1	参见 枚举	运动状态（状态）	
0x00000007	读取	每个	UINT32	1	参见 枚举	SAF 子组状态（子状态）	
0x00000008	读取	每个	UINT32	1	参见 枚举	引用状态（状态）	
0x00000009	读取	每个	UINT32	1	参见 枚举	耦合状态（状态）	无法用示波器追踪！
0x0000000A	读取	每个	UINT32	1	≥0	耦合表索引	无法用示波器追踪！
0x0000000B	读取	每个	UINT32	1	≥0	当前 SVB 条目/任务数	符号访问：'SvbEntries' (DXD)
0x0000000C	读取	每个	UINT32	1	≥0	当前 SAF 条目/任务数	符号访问：'SafEntries' (DXD)
0x0000000D	读取	每个	UINT32	1		当前块编号（仅对插值组有效）	符号访问：'BlockNumber' (DXD)
0x0000000E	读取	每个	UINT32	1	≥0	当前自由 SVB 条目/任务数	无法用示波器追踪！
0x0000000F	读取	每个	UINT32	1	≥0	当前自由 SAF 条目/任务数	无法用示波器追踪！
0x00000011	读取	每个	UINT16	1	0/1	紧急停止 (E-Stop) 激活?	无法用示波器追踪！
0x00000110	读取	PTP 组	{			内部 NC 信息（分辨率）	保留！
			REAL64	例如 mm	± ∞	外部结束位置	
			REAL64	例如 mm/s	>0	外部目标速度	
			REAL64	例如 mm/s^2	>0	外部加速度	
			REAL64	例如 mm/s^2	>0	外部减速度	
			REAL64	例如 mm/s^3	>0	外部加加速度	
			UINT32	1	>0	外部速率比类型	
			REAL64	例如 mm	± ∞	内部结束位置	
			REAL64	例如 mm/s	>0	内部目标速度（指的是100%）	
			REAL64	%	[0 ... 100]	内部实际速率比	
			REAL64	例如 mm/s^2	>0	内部加速度	
			REAL64	例如 mm/s^2	>0	内部减速度	
			REAL64	例如 mm/s^3	>0	内部加加速度	
			REAL64	例如 mm	>0	位置分辨率	
			REAL64	例如 mm/s	≥0	速度分辨率	
			REAL64	例如 mm/s^2	≥0	加速度分辨率	
			REAL64	例如 mm/s	≥0	加速度为零时的速度分辨率	
			}				

索引偏移（十六进制）	访问	组类型	数据类型	物理单位	定义范围	描述	说明
0x00000500	读取	DXD 组	REAL64	例如 mm	≥ 0	当前路径分段上的剩余路径（剩余弧长）	符号访问：SetPathRemLength
0x00000501	读取	DXD 组	REAL64	例如 mm	≥ 0	当前路径分段上的累积弧长	符号访问：'SetPathLength'
0x00000502	读取	DXD 组	REAL64	例如 mm/s	≥ 0	当前路径设置速度	符号访问：'SetPathVelo'
0x00000503	读取	DXD 组	REAL64	例如 mm/s^2	± ∞	当前路径设置加速度	符号访问：'SetPathAcc'
0x00000504	读取	DXD 组	REAL64	例如 mm/s^2	≥ 0	当前矢量集加速度的量	符号访问：'SetPathAbsAcc'
0x00000505	读取	DXD 组	REAL64	例如 mm/s	≥ 0	最大分段结束路径设置速度	符号访问：'SetPathVeloEnd'
0x00000506	读取	DXD 组	REAL64	例如 mm/s	≥ 0	分段最大路径设置速度	符号访问：'SetPathVeloMax'
0x00000507	读取	DXD 组	REAL64	例如 mm	≥ 0	基于当前弧长的当前相对制动距离	符号访问：'SetPathStopDist'
0x00000508	读取	DXD 组	REAL64	例如 mm	± ∞	安全距离 = 分段弧长 - 当前弧长 - 相对制动距离	符号访问：'SetPathSecurityDist'
0x00000509	读取	DXD 组	REAL64	1	0/1	分段转换	符号访问：'SetPathSegmentChange'
0x0000050A	读取	DXD 组	REAL64	%	[0 ... 100]	路径速度速率比	符号访问：'SetPathOverride'
0x00000511	读取	DXD 组	REAL64	例如 mm/s	≥ 0	实际路径速度的分量	符号访问：'ActPathAbsVelo'
0x00000512	读取	DXD 组	REAL64	例如 mm/s^2	± ∞	当前分段上的实际路径加速度	符号访问：'ActPathAcc'
0x00000513	读取	DXD 组	REAL64	例如 mm/s^2	≥ 0	当前分段上实际路径加速度的分量	符号访问：'ActPathAbsAcc'
0x00000514	读取	DXD 组	REAL64	例如 mm	± ∞	路径上切线方向的位置误差（用符号表示领先和滞后）	符号访问：'PathDiffTangential'
0x00000515	读取	DXD 组	REAL64	例如 mm	≥ 0	路径上正交方向的位置误差	符号访问：'PathDiffOrthogonal'
0x00000520	读取	DXD 组	REAL64	1	≥ 0	当前分段的覆盖弧长，规范化为 1.0。	
0x00000521	读取	DXD 组	REAL64	1	0/1	改变部分分段（公差球半径）	
0x00000522	读取	DXD 组	REAL64	1	≥ 0	到上一个几何条目或下一个准停的剩余路径总长度。参见组参数 0x508。	
0x00000523	读取	DXD 组	REAL64	1	≥ 0	当前分段的编程速度	
0x00000524	读取	DXD 组	REAL64	例如 mm	≥ 0	自程序启动后行进的路径距离（弧长）	TwinCAT 3.1 B4022.31 及以上 TwinCAT 3.1 B4024.0 及以上

索引偏移（十六进制）	访问	组类型	数据类型	物理单位	定义范围	描述	说明
0x00000530	读取	DXD 组	{			主轴 X、Y 和 Z 的当前或上一个 MCS 目标位置	
			REAL64	例如 mm	± ∞	目标位置 X 轴	
			REAL64	例如 mm	± ∞	目标位置 Y 轴	
			REAL64	例如 mm	± ∞	目标位置 Z 轴	
			}				
0x00000531	读取	DXD 组	{			辅助轴 Q1 至 Q5 的当前或上一个 MCS 目标位置	
			REAL64[5]	例如 mm	± ∞	轴 Q1 至 Q5 的目标位置	
			}				
0x00000532	读取	DXD 组	{			读取与当前 DC 时间相关的下 11 个分段的路径长度、H 参数和条目 ID	未广泛发布
			UINT32			DC 时间	
			UINT32			保留	
			PreViewTab[11]			11*24 字节	
			}				
			PreViewTab				
			{				
			REAL64	例如 mm		分段长度	
			UINT32	1		块号	
			UINT32	1		H 参数	
			UINT32	1		条目 ID	
			UINT32	1		保留	
			}				
0x0000054n	读取	DXD 组	REAL64	1	0/1	在辅助轴公差球内 n = 1..5 辅助轴数（非轴 ID）	
0x00000546	读取	DXD 组	REAL64[8]	例如 mm	± ∞	设置 3D 组 (3+5) 轴的位置阵列	TC3.1 B4022.17 及以上
0x00000547	读取	DXD 组	REAL64[8]	例如 mm	± ∞	3D 组 (3+5) 轴的实际位置阵列	TC3.1 B4022.17 及以上
0x00000548	读取	DXD 组	REAL64[8]	例如 mm	± ∞	作为 3D 组 (3+5) 轴阵列的位置差（设置值/实际值）或滞后误差	TC3.1 B4022.17 及以上
0x00000550	读取	DXD 组	{			读取 3D 组内的轴 ID：	
			UINT32	1	[0, 1...255]	X 轴 ID	
			UINT32	1	[0, 1...255]	Y 轴 ID	
			UINT32	1	[0, 1...255]	Z 轴 ID	
			}				
0x00000552	读取	DXD 组 FIFO 组 运动学组	{ UINT32[m] }	1	[0, 1...255]	组的轴分配： 第 1 轴 ID - 第 m 轴 ID m：带有主轴和辅助轴（X、Y、Z、Q1、Q2、Q3、Q4、Q5）或 FIFO 组或运动学组 ACS 轴的 3D 组的尺寸	

索引偏移（十六进制）	访问	组类型	数据类型	物理单位	定义范围	描述	说明
0x00000553	读取	运动学组	{			读取运动学组内的轴分配 (ID)：	
			UINT32[8]	1	[0, 1...255]	MCS 轴 ID（机器坐标系）	
			UINT32[8]	1	[0, 1...255]	ACS 轴 ID（轴坐标系）	
			UINT32	1	⩾ 0	保留	
			UINT32	1	⩾ 0	保留（新）	
			}				
0x0000056n	读取	DXD 组	REAL64	1	± ∞	公差球内辅助轴的当前位置误差（仅设置值侧） 仅适用于辅助轴 n = 1..5 辅助轴数（非轴 ID）	

3.2.1.5.4.3.3 *组功能的"索引偏移"规范（索引组 0x3200 + ID）*

索引偏移（十六进制）	访问	组类型	数据类型	物理单位	定义范围	描述	备注
0x00000001	写入	每个	VOID			重置组	
0x00000002	写入	每个	VOID			停止组	
0x00000003	写入	每个	VOID			清除组（缓冲/任务）	
0x00000004	写入	PTP 组、3D 组	{			紧急停止（E-stop）（通过控制斜坡紧急停止）	
			REAL64	例如 mm/s^2	⩾ 0.0	减速度（必须大于或等于原始减速度）	
			REAL64	例如 mm/s^3	⩾ 0.0	加加速度（必须大于或等于原始加加速度）	
			}				
0x00000005	写入	PTP 组	{			可参数化停止（使用控制斜坡）	保留功能，无标准！
			REAL64	例如 mm/s^2	⩾ 0.0	减速度	
			REAL64	例如 mm/s^3	⩾ 0.0	加加速度	
			}				
0x00000006	写入	PTP 组、3D 组	VOID			紧急停止 (E-Stop) 后的"步进"	
0x00000050	写入	PTP 组、3D 组	{			组的轴分配：	
			UINT32	1	[0, 1...255]	X 轴 ID	
			UINT32	1	[0, 1...255]	Y 轴 ID	
			UINT32	1	[0, 1...255]	Z 轴 ID	
			}				
0x00000051	写入	PTP 组、3D 组 FIFO 组	{			组的轴分配：	
			UINT32	1	[1...255]	轴 ID	
			UINT32	1	[0 ... (m-1)]	轴在组 m 中的位置索引：组尺寸（PTP：1；DXD：3，FIFO：16）	
			}				
0x00000052	写入	3D 组 FIFO 组	{ UINT32[m] }	1	[0, 1...255]	组的轴分配： 第一轴 ID、... 、M 轴 ID m：3D 组（X、Y、Z、Q1、Q2、Q3、Q4、Q5）或 FIFO 组的尺寸	
0x00000053	写入	3D 组 FIFO 组 运动学组	VOID			删除 3D 轴分配、FIFO 轴分配或运动学轴分配，并将轴返回到各自的 PTP 组	
0x00000054	写入	运动学组	{			运动学组的轴分配：	
			UINT32[8]	1	[0, 1...255]	MCS 轴 ID（机器坐标系）	
			UINT32[8]	1	[0, 1...255]	ACS 轴 ID（轴坐标系）	
			UINT32	1	⩾ 0	保留	
			UINT32	1	⩾ 0	保留（新）	
			}				
0x00000060	读写	3D 组		1		内部"进给组"命令（“进给器”）	执行命令！
0x00000061	读写	3D 组		1		内部"进给组"命令（“进给器”）	执行命令！
0x00000110	写入	1D 组	VOID			参见 1D 组（“校准”）	

索引偏移（十六进制）	访问	组类型	数据类型	物理单位	定义范围	描述	备注
0x00000111	写入	1D 组	{			新结束位置 1D 组	
			UINT32	枚举	参见附录	结束位置类型 [▶ 145]（参见附录）	
			REAL64	例如 mm	±∞	新结束位置（目标位置）	
			}				
0x0000011A	写入	1D 组	{			设置实际位置 1D 组	**使用时请注意！** 始终连接至 SAF 501 端口！
			UINT32	枚举	参见附录	实际位置类型 [▶ 145]（参见附录）	
			REAL64	例如 mm	±∞	轴的实际位置	
			}				
0x0000011B	写入	1D 组	UINT32	1	0/1	设置参考旗标（“校准旗标”）	**使用时请注意！**
0x00000120	写入	1D 组	{			启动 1D 组（标准启动）：	
			UINT32	枚举	参见附录	启动类型 [▶ 144]（参见附录）	
			REAL64	例如 mm	±∞	结束位置（目标位置）	
			REAL64	mm/s	≥ 0.0	所需速度	
			}				
0x00000121	写入	1D 组 (SERVO)	{			启动 1D 组（扩展启动）：	
			UINT32	枚举	参见附录	启动类型 [▶ 144]（参见附录）	
			REAL64	例如 mm	±∞	结束位置（目标位置）	
			REAL64	mm/s	≥0.0	所需速度	
			UINT32	1	0/1	标准加速度?	
			REAL64	mm/s^2	≥ 0.0	加速度	
			UINT32	1	0/1	标准减速度?	
			REAL64	mm/s^2	≥ 0.0	减速度	
			UINT32	1	0/1	标准加加速度?	
			REAL64	mm/s^3	≥ 0.0	加加速度	
			}				
0x00000122	写入	1D 组（MW 伺服）	{			启动 1D 组（特殊启动）：	保留启动功能，无标准！
			UINT32	枚举	参见附录	启动类型 [▶ 144]（参见附录）	
			REAL64	例如 mm	±∞	结束位置（目标位置）	
			REAL64	mm/s	≥0.0	所需启动速度	
			REAL64	例如 mm	±∞	新速度等级的位置	
			REAL64	mm/s	≥0.0	新结束速度级	
			UINT32	1	0/1	标准加速度?	
			REAL64	mm/s^2	≥0.0	加速度	
			UINT32	1	0/1	标准减速度?	
			REAL64	mm/s^2	≥0.0	减速度	
			UINT32	1	0/1	标准加加速度?	
			REAL64	mm/s^3	≥0.0	加加速度	
			}				
0x00000126	写入	1D 组	{			启动驱动输出:	
			UINT32	枚举	参见附录	输出类型 [▶ 153]（参见附录）	
			REAL64	例如 %	±∞	所需输出值（例如 %）	
			}				
0x00000127	写入	1D 组	VOID			停止驱动输出	

索引偏移（十六进制）	访问	组类型	数据类型	物理单位	定义范围	描述	备注
0x00000128	写入	1D 组	{			更改驱动输出：	
			UINT32	枚举	参见附录	输出类型 [▶ 153]（参见附录）	
			REAL64	例如 %	±∞	所需输出值（例如 %）	
			}				
0x00000130	写入	1D 组 (SERVO)	{			1D 区间补偿（伺服）：	
			UINT32	枚举	参见附录	补偿类型 [▶ 145]（附录）	
			REAL64	mm/s/s	⩾ 0.0	最大加速度增加	
			REAL64	mm/s/s	⩾ 0.0	最大减速度增加	
			REAL64	mm/s	⩾ 0.0	最大增速	
			REAL64	mm/s	⩾ 0.0	进程的基本速度	
			REAL64	例如 mm	±∞	需要补偿的路径差	
			REAL64	例如 mm	⩾ 0.0	补偿的路径距离	
			}				
0x00000131	写入	1D 组 SERVO	VOID			停止区间补偿（伺服）	
0x00000140 (0x00n00140)	写入	主轴/从轴耦合：1D 组（伺服）	{			主轴/从轴耦合（伺服）：	扩展功能：“飞锯”功能！锯切角度大于0.0度且小于或等于90.0度（平行锯切：90.0度）
			UINT32	枚举	参见附录	从轴类型/耦合类型 [▶ 145]（参见附录）	
			UINT32	1	[1...255]	主轴/组的轴 ID	
			UINT32	1	[0...8]	主轴的子索引 n（默认值：0）	
			UINT32	1	[0...8]	从轴的子索引 n（默认：0）	
			REAL64	1	[±1000000.0]	参数 1：线性：齿轮系数 FlySawVelo：保留 FlySaw：绝对同步位置主轴 [mm]	
			REAL64	1	[±1000000.0]	参数 2：线性：保留 FlySawVelo：保留 FlySawPos：绝对同步位置从轴 [mm]	
			REAL64	1	[±1000000.0]	参数 3：线性：保留 FlySawVelo：倾角，以 [DEGREE] 为单位 FlySawPos：倾角，以 [DEGREE] 为单位	
			REAL64	1	[±1000000.0]	参数 4：线性：保留 FlySawVelo：齿轮系数 FlySawPos：齿轮系数	
			}				
0x00000141	写入	主轴/从轴耦合：1D 组（伺服）	VOID			主轴/从轴解耦（伺服）	

索引偏移（十六进制）	访问	组类型	数据类型	物理单位	定义范围	描述	备注
0x00000142	写入	主轴/从轴参数 1D 组（伺服）	{			更改耦合参数（耦合）：	
			REAL64	1	[±1000000.0]	参数 1：线性：齿轮系数	
			REAL64	1	[±1000000.0]	参数 2：线性：保留	
			REAL64	1	[±1000000.0]	参数 3：线性：保留	
			REAL64	1	[±1000000.0]	参数 4：线性：保留	
			}				
0x00000144	写入	从轴停止 1D 组（伺服）	VOID			停止"飞锯"（伺服）	仅适用于"飞锯"
0x00000149	写入	从表 1D 组（伺服）	REAL64	1	±∞	设置单表耦合的从表缩放比例（伺服）	仅适用于单表从轴
0x00000150	写入	1D 组	VOID			停用完整的 1D 组/轴（禁用）	
0x00000151	写入	1D 组	VOID			激活完整的 1D 组/轴（启用）	
0x00000160	写入	1D 组	VOID			停用 1D 组的驱动输出（禁用）	
0x00000161	写入	1D 组	VOID			激活 1D 组的驱动输出（启用）	
0x00000362	写入	高速/低速组	UINT16	1	0/1	释放驻车制动? 0：自动激活（默认） 1：必须始终释放!	
0x00000701	写入	FIFO 组	VOID			启动 FIFO 组（FIFO 表必须事先填好）	(n*m)-FIFO
0x00000710	写入	FIFO 组	{ REAL64[x*m]}	例如 mm	±∞	写入 x 个 FIFO 条目（行）： (x*m) 值（一行或多行） n：FIFO 长度（行数） m：FIFO 尺寸（列数） 值范围 x：[1 ... n]	只能逐行进行!（整数倍）
0x00000711	写入	FIFO 组	{ REAL64[x*m]}	例如 mm	±∞	重写最后 x 个 FIFO 条目（行）： (x*m) 值（一行或多行） n：FIFO 长度（行数） m：FIFO 尺寸（列数） 值范围 x：[1 ... n]	只能逐行进行!（整数倍）
0x00000801	写入	运动学组	VOID			启动运动学组	保留功能，无标准!

3.2.1.5.4.4 规范轴

3.2.1.5.4.4.1 *轴参数的"索引偏移"规范（索引组 0x4000 + ID）*

索引偏移（十六进制）	访问	轴类型	数据类型	物理单位	定义范围	描述	说明
0x00n00000	读取	每个（所有轴参数的结构）	{			一般轴参数结构 (NC/CNC)， 还包含编码器、控制器和驱动器等子元素（参见 TcMc2.lib 中的 MC_ReadParameterSet） 注：尺寸和对齐方式有所改变。	从 TC3 起修改
			UINT32	1		轴 ID	
			STRING[30+1]	1		轴名称	
			UINT32	1	枚举	轴类型 [▶ 143]	
			. . .	. . .	. . .	. . .	
			}			1024 字节（而不是 512 字节）	
0x00000001	读取	每个	UINT32	1		轴 ID	
0x00000002	读取	每个	STRING[30+1] UINT8[. . .]	1		轴名称	任意字符数，从 TC3.1 版本 4022.32 或 4024.6 起
0x00000003	读取	每个	UINT32	1	枚举	轴类型 [▶ 143]	
0x00000004	读取	每个	UINT32	µs		循环时间轴 (SEC)	
0x00000005	读取	每个	STRING[10+1]	1		物理单位	
0x00000006	读/写	每个	REAL64	例如 mm/s		凸轮方向的参考速度	
0x00000007	读/写	每个	REAL64	例如 mm/s		同步方向的参考速度	
0x00000008	读/写	每个	REAL64	例如 mm/s		速度手动慢速	
0x00000009	读/写	每个	REAL64	例如 mm/s		速度手动快速	
0x0000000A	读/写	每个	REAL64	例如 mm/s	[0.0...1.0E20]	速度快速移位	
0x0000000F	读/写	每个	UINT16	1	0/1	位置范围监测?	
0x00000010	读/写	每个	REAL64	例如 mm	[0.0...1.0E6]	位置范围窗口	
0x00000011	读/写	每个	UINT16	1	0/1	运动监测?	
0x00000012	读/写	每个	REAL64	s	[0.0...600]	运动监测时间	
0x00000013	读/写	每个	UINT16	1	0/1	循环?	
0x00000014	读/写	每个	REAL64	例如 mm		循环距离 (±)	
0x00000015	读/写	每个	UINT16	1	0/1	目标位置监测?	
0x00000016	读/写	每个	REAL64	例如 mm	[0.0...1.0E6]	目标位置窗口	
0x00000017	读/写	每个	REAL64	s	[0.0...600]	目标位置监测时间	
0x00000018	读/写	每个	REAL64	例如 mm		正方向脉冲路径	
0x00000019	读/写	每个	REAL64	例如 mm		负方向脉冲路径	
0x0000001A	读/写	每个	UINT32	1	枚举 (≥0)	错误反应模式： 0：瞬时（默认） 1: 延迟 （例如 主轴/从轴耦合）	
0x0000001B	读/写	每个	REAL64	s	[0...1000]	错误延迟时间（如果选择延迟错误反应）	
0x0000001C	读/写	每个	UINT16	1	0/1	如果设备尚未准备好运行，是否通过实际值耦合从轴?	

索引偏移（十六进制）	访问	轴类型	数据类型	物理单位	定义范围	描述	说明
0x0000001D	读/写	每个	REAL64	例如 mm/s^2	[0, 0.01...1.0E10]	从设置值切换到实际值时的衰减剖面加速度： 默认：0（在这种情况下使用轴加速度的最小值，即 MIN(Acc, Dec)）	
0x0000001E	读/写	每个	UINT32	1	枚举 (≥0)	快速轴停止信号类型： 选择触发快速轴停止的信号类型（请参见 Drive->nStatus4 中的第 7 位） "0（SignalType_OFF）"、 "1（SignalType_RisingEdge）"、"2（SignalType_FallingEdge）"、"3（SignalType_BothEdges）"、"4（SignalType_HighActive）"、"5（SignalType_LowActive）"	
0x00000020	读/写	每个	UINT16	1	0/1	是否允许对从轴发出运动命令? 默认：FALSE	
0x00000021	读/写	每个	UINT16	1	0/1	是否允许对具有激活外部设定点发生器的轴发出运动命令? 默认：FALSE	
0x00000026	读/写	每个	UINT32	1		单位（位置、速度、时间）的解释 第 0 位：速度单位为 x/min，而非 x/s 第 1 位：以千分之一为基本单位的位置 第 2 位：模除位置显示	参见编码器！位数组
0x00000027	读/写	每个	REAL64	例如 mm/s	[>0...1.0E20]	最大允许速度	
0x00000028	读/写	每个	REAL64	例如 mm	[0.0...1.0E6]	运动监测窗口	
0x00000029	读/写	每个	UINT16	1	0/1	PEH 时间监测?	位置结束和准停
0x0000002A	读/写	每个	REAL64	s	[0.0...600]	PEH 监测时间	
0x0000002C	读/写	每个	REAL64	例如 mm	[-1000.0 ...1000.0]	间隙	
0x00000030	读取	每个	UINT16	1	[0,1]	持久性数据（PERSISTENT DATA），例如用于编码器的实际位置和参考状态?	启动参数，无法在线更改。
0x00000031	读取	每个	{ uint8[6] uint16 uint16 }10 字节	AmsAddr: AmsNetId, AmsPort No. Channel No	1	读取硬件 AMS 地址（AMS 网络 ID 和 AMS 端口号）和 EtherCAT 通道号（通信通道 0、1、2、3...）	

索引偏移（十六进制）	访问	轴类型	数据类型	物理单位	定义范围	描述	说明
0x00000031	读取	每个	{ uint8[6] uint16 uint16 // uint16 uint32 uint32 uint32 uint32 uint32 uint32 uint32 uint32 uint32 uint32 uint32[3] }64 字节	AmsAddr: AmsNetId, AmsPort No. Channel No Reserviert NcDriveID NcDriveIndex NcDrive Type NcEncID NcEncIndex NcEncType NcAxisID NcAxisType TcDrive ObjectId TcEncObjectId 保留	1	读取硬件 AMS 地址（AMS 网络 ID 和设备 AMS 端口号）和 EtherCAT 通道号（通信通道 0、1、2、3...） 由其他 NC 信息补充，如 NcDriveID、NcDriveType（参见附录）等……	从 TC3 起更新 DriveObjectId 和 EncObjectId，从 NC build 4437 起
0x00000033	读取	每个	{ UINT16 ApplRequestBit UINT16 ApplRequestType UINT32 ApplCmdNo UINT32 ApplCmdVersion 。.. } 1024 字节	 1 1 未执行 1	 0/1 ≥0 >0 ≥0	一般应用请求结构 (NC/NCI)，例如用于 ApplicationHoming 请求（请参见 TcMc2.lib 中的 MC_ReadApplicationRequest) 应用请求类型： 0：无（空闲） 1：归位	TC3 中的更改
0x00000051	读取	通道：每个	UINT32			通道 ID	
0x00000052	读取	通道：每个	STRING[30+1]			通道名称	
0x00000053	读取	通道：每个	UINT32	1	枚举	通道类型 [▶ 142]	
0x00000054	读取	组：每个	UINT32			组 ID	
0x00000055	读取	组：每个	STRING[30+1]			组名称	
0x00000056	读取	组：每个	UINT32	1	枚举	组类型 [▶ 143]	
0x00000057	读取	每个	UINT32			编码器数	
0x00000058	读取	每个	UINT32			控制器数	
0x00000059	读取	每个	UINT32			驱动器数	
0x0000005A	读取	每个	{			读取轴的所有子元素：	
			UINT32[9]	1	[0, 1...255]	轴编码器 ID	
			UINT32[9]	1	[0, 1...255]	轴控制器 ID	
			UINT32[9]	1	[0, 1...255]	轴驱动器 ID	
			} 108 字节				
0x000000F1	读/写	每个	REAL64	例如 mm/s^2	默认：1.0E5	最大允许加速度	从 TC 3.2 起更新
0x000000F2	读/写	每个	REAL64	例如 mm/s^2	默认：1.0E6	最大允许减速度	从 TC 3.2 起更新

索引偏移（十六进制）	访问	轴类型	数据类型	物理单位	定义范围	描述	说明
0x00000101	读/写	伺服	REAL64	例如 mm/s^2	[0.01...1.0E20]	加速度（默认数据集）	
0x00000102	读/写	伺服	REAL64	例如 mm/s^2	[0.01...1.0E20]	减速度（默认数据集）	
0x00000103	读/写	伺服	REAL64	例如 mm/s^3	[0.1...1.0E30]	加加速度（默认数据集）	
0x00000104	读/写	伺服	REAL64	s	[0.0 ... 1.0] 默认：0.0 s	设定点发生器的速度值和位置值之间的减速度时间（以秒为单位）	
0x00000105	读/写	伺服	UINT32	1	枚举 默认：类型 1	速度的速率比类型 [▶ 144]： 1：与内部降速有关（无迭代） 2：与原始外部起始速度有关（无迭代） 3：与内部降速有关（通过迭代方式进行优化） 4：与原始外部起始速度有关（通过迭代方式进行优化）	
0x00000106	读/写	伺服	REAL64	1	[0.0 ... 1.0E6] 默认：0.0	动态减速速度的最大允许步长变化DV = 系数 * 最小值（A+，A-） * DT	
0x00000107	读/写	伺服	UINT16	1	[0.1] 默认：1	激活辅助轴的加速度和加加速度限制（Q1 至 Q5）	
	读/写	伺服	REAL64	例如 mm	[0.0..1000.0]	辅助轴公差球面半径	
	读/写	伺服	REAL64	例如 mm	[0.0..10000.0]	减小公差球面时允许的最大位置偏差 仅适用于辅助轴	
0x0000010A	读/写	伺服	REAL64	例如 mm/s^2	[0.01 ... 1.0E20]	快速轴停止：加速度 （另见 快速轴停止信号类型）	
0x0000010B	读/写	伺服	REAL64	例如 mm/s^2	[0.01 ... 1.0E20]	快速轴停止：减速度 （另见 快速轴停止信号类型）	
0x0000010C	读/写	伺服	REAL64	例如 mm/s^3	[0.1 ... 1.0E30]	快速轴停止：加加速度 （另见 快速轴停止信号类型）	
0x0000010D	读/写	伺服	UINT32	1		在循环接口中作为"UserData"传递的轴状态的索引偏移。 0x00000000：停用 0x00010012：带位置偏置电压的编码器位置（无位置校正且无死区时间补偿） 0x00010014：DriveActVelo 0x00010017：MC_SetPosition 偏移	
0x00000201	读/写	步进电机	UINT32	1	枚举	运行模式步进电机	
0x00000202	读/写	步进电机	REAL64	例如 mm/STEP	[1.0E-6 ... 1000.0]	电机步进的距离缩放	
0x00000203	读/写	步进电机	REAL64	例如 mm/s	[0.0 ... 1000.0]	速度剖面的最小速度	

索引偏移（十六进制）	访问	轴类型	数据类型	物理单位	定义范围	描述	说明
0x00000204	读/写	步进电机	UINT32	1	[0 ... 100]	每个频率/速度阶跃的阶跃数	
0x00000205	读/写	步进电机	UINT32	1		作为同步脉冲的电机掩码	未执行!
0x00000301	读/写	高/低	REAL64	例如 mm	[0.0 ... 100000.0]	正方向爬行距离	
0x00000302	读/写	高/低	REAL64	例如 mm	[0.0 ... 100000.0]	负方向漂移距离	
0x00000303	读/写	高/低	REAL64	例如 mm	[0.0 ... 100000.0]	正方向制动距离	
0x00000304	读/写	高/低	REAL64	例如 mm	[0.0 ... 100000.0]	负方向制动距离	
0x00000305	读/写	高/低	REAL64	s	[0.0 ... 60.0]	正方向制动减速度	
0x00000306	读/写	高/低	REAL64	s	[0.0 ... 60.0]	负方向制动减速度	
0x00000307	读/写	高/低	REAL64	s	[0.0 ... 60.0]	从高速到低速的切换时间	
0x00000308	读/写	高/低	REAL64	例如 mm	[0.0 ... 100000.0]	漂移距离停止	
0x00000309	读/写	高/低	REAL64	s	[0.0 ... 60.0]	松开制动器的延迟时间	
0x0000030A	读/写	高/低	REAL64	s	[0.0 ... 60.0]	正方向脉冲时间	
0x0000030B	读/写	高/低	REAL64	s	[0.0 ... 60.0]	负方向脉冲时间	
编码器							
0x00n10001	读取	编码器：每个	UINT32	1	[1 ... 255]	编码器 ID n = 0：轴的标准编码器 > 0：轴的第 n 个编码器（可选）	
0x00n10002	读取	编码器：每个	STRING[30+1]	1	30 个字符	编码器名称	
0x00n10003	读取	编码器：每个	UINT32	1	枚举 (>0)	编码器类型 [▶ 147]	
0x00n10004	读/写	编码器：每个	UINT32	1	Byteoffset	输入地址偏移（I/O-输入-图像）	更改 I/O 地址
0x00n10005	读/写	编码器：每个	UINT32	1	Byteoffset	输出地址偏移（I/O-输出-图像）	更改 I/O 地址
0x00n10006	读/写	编码器：每个	REAL64	例如 mm/INC	[1.0E-12 ... 1.0E+30]	结果缩放因子（分子/分母） 注：从 TC3 起，缩放因子（scaling factor）由分子和分母两部分组成（默认：1.0）。	如果已发出启用控制器的信号，则不允许写入。
0x00n10007	读/写	编码器：每个	REAL64	例如 mm	[±1.0E+9]	位置偏移	如果已发出启用控制器的信号，则不允许写入。
0x00n10008	读/写	编码器：每个	UINT16	1	[0,1]	编码器计数方向	如果已发出启用控制器的信号，则不允许写入。
0x00n10009	读/写	编码器：每个	REAL64	例如 mm	[0.001 ... 1.0E+9]	模除因子	
0x00n1000A	读/写	编码器：每个	UINT32	1	参见 枚举 (>0)	编码器模式 [▶ 149]	
0x00n1000B	读/写	编码器：每个	UINT16	1	0/1	软限位最小监测?	
0x00n1000C	读/写	编码器：每个	UINT16	1	0/1	软限位最大监测?	
0x00n1000D	读/写	编码器：每个	REAL64	mm		软限位最小结束位置	
0x00n1000E	读/写	编码器：每个	REAL64	mm		软限位最大结束位置	
0x00n1000F	读/写	编码器：每个	UINT32	1	参见 枚举 (≥0) 在附录中	编码器评估方向 [▶ 149]（启用日志记录计数方向）	
0x00n10010	读/写	编码器：每个	REAL64	s	[0.0...60.0]	实际位置值的滤波时间（以秒为单位）(P-T1)	
0x00n10011	读/写	编码器：每个	REAL64	s	[0.0...60.0]	实际速度值的滤波时间（以秒为单位）(P-T1)	

索引偏移（十六进制）	访问	轴类型	数据类型	物理单位	定义范围	描述	说明
0x00n10012	读/写	编码器：每个	REAL64	s	[0.0...60.0]	实际加速度值的滤波时间（以秒为单位）(P-T1)	
0x00n10013	读/写	编码器：每个	STRING[10+1]	1		物理单位	未执行!
0x00n10014	读/写	编码器：每个	UINT32	1		单位（位置、速度、时间）的解释 第 0 位：速度单位为 x/min，而非 x/s 第 1 位：以千分之一为基本单位的位置	未执行! 位数组
0x00n10015	读取	编码器：每个	UINT32	INC	[0x0... 0xFFFFFFFF]	编码器掩码（以增量表示的编码器实际值的最大值 注：编码器掩码可以是任何数值（如 3600000）。与过去不同，它不再需要对应于连续的一系列二进制 $1(^{2}n-1)$。	只读参数 另请参见"编码器子掩码"参数
0x00n10016	读/写	编码器：每个	UINT16	1	0/1	实际位置校正（测量系统误差校正）?	
0x00n10017	读/写	编码器：每个	REAL64	s	[0.0...60.0]	实际位置校正的滤波时间（以秒为单位）(P-T1)	
0x00n10019	读/写	编码器：每个	UINT32	1	枚举 (>0)	编码器绝对尺寸系统 [▶ 149]	如果已发出启用控制器的信号，则不允许写入。
0x00n1001A	读取	编码器：每个	UINT32	1	枚举 (>0)	编码器位置初始化	未执行!
0x00n1001B	读/写	编码器：每个	REAL64	例如 mm	[≥0，模除因子/2]	模除启动的容差窗口	
0x00n1001C	读取	编码器：每个	UINT32	1	枚举 (>0)	编码器符号解释 [▶ 149]（数据类型）	
0x00n1001D	读取	编码器：每个	UINT16	1	0/1	增量或绝对编码器? 0：增量编码器类型 1：绝对编码器类型	
0x00n10023	读/写	编码器：每个	REAL64	例如 mm/INC	[1.0E-12 ... 1.0E+30]	缩放因子(scaling factor)的组成部分：分子 （=> 缩放因子(scaling factor)分子/缩放(scaling factor)因子分母）	从 TC3 起更新 如果已发出启用控制器的信号，则不允许写入。
0x00n10024	读/写	编码器：每个	REAL64	1	[1.0E-12 ... 1.0E+30]	缩放因子(scaling factor)的组成部分：分母 （=> 缩放因子(scaling factor)分子/缩放(scaling factor)因子分母） 默认：1.0	从 TC3 起更新 如果已发出启用控制器的信号，则不允许写入。

0x00n10025	读/写	编码器：每个	{
			real64
			real64
			}
				例如 mm/INC
				1
					[1.0E-12 ... 1.0E+30]
					[1.0E-12 ... 1.0E+30]
						缩放因子(scaling factor)的组成部分：分子
						缩放因子(scaling factor)的组成部分：分母
						（=> 缩放因子(scaling factor)分子/缩放因子(scaling factor)分母）
							从 TC3 起更新

索引偏移（十六进制）	访问	轴类型	数据类型	物理单位	定义范围	描述	说明
0x00n10030	读/写	编码器：每个	UINT32	1		内部编码器控制双字，用于指定运行模式和属性	从 TC3 起更新
0x00n10101	读/写	E: INC	UINT16	1	[0,1]	参考凸轮的反向搜索方向?	
0x00n10102	读/写	E: INC	UINT16	1	[0,1]	同步脉冲的反向搜索方向?	
0x00n10103	读/写	E: INC	REAL64	例如 mm	[±1000000.0]	参考位置	
0x00n10104	读/写	E: INC	UINT16	1	[0,1]	是否激活了参考凸轮和同步脉冲之间的距离监测?	未执行!
0x00n10105	读/写	E: INC	UINT32	INC	[0 ... 65536]	参考凸轮与同步脉冲之间的最小间隙（以增量为单位）	未执行!
0x00n10106	读/写	E: INC	UINT16	1	[0,1]	外部同步脉冲?	
0x00n10107	读/写	E: INC	UINT32	1	参见 枚举 (>0)	参考模式 [▶ 150]	
0x00n10108	读/写	E: INC	UINT32	1	[0x0000000F... 0xFFFFFFFF]二进制掩码：（^{2}n - 1）	编码器子掩码（编码器实际值绝对范围的最大值，以增量为单位） 例如，用作参考模式"软件同步"和 NC 保留数据"绝对（MODULO模除）"、"增量（SINGLETURN ABSOLUTE单转绝对）"的参考标记。 注 1：编码器子掩码必须小于或等于编码器掩码。 注 2：编码器掩码必须是编码器子掩码的整数倍。 注 3：编码器子掩码必须是连续的二进制 1 序列（^{2}n-1），例如 0x000FFFFF。	另请参见"编码器掩码"参数
0x00n10110	读/写	E： INC（编码器模拟）	REAL64	1	[0.0 ... 1000000.0]	模拟编码器噪声部分的缩放/权重	
控制器							
0x00n20001	读取	控制器：每个	UINT32	1	[1 ... 255]	控制器 ID n = 0：轴的标准控制器 > 0：轴的第 n 个控制器（可选）	
0x00n20002	读取	控制器：每个	STRING[30+1]	1	30 个字符	控制器名称	
0x00n20003	读取	控制器：每个	UINT32	1	枚举 (>0)	控制器类型 [▶ 146]	
0x00n2000A	读/写	控制器：每个		1	枚举 (>0)	控制器模式	
0x00n2000B	读/写	控制器：每个	REAL64	%	[0.0 ... 1.0]	速度预控制的权重（默认值：1.0 = 100）	
0x00n20010	读/写	控制器：每个	UINT16	1	0/1	位置滞后监测位置?	
0x00n20011	读/写	控制器：每个	UINT16	1	0/1	位置滞后监测速度?	
0x00n20012	读/写	控制器：每个	REAL64	例如 mm		最大滞后误差位置	
0x00n20013	读/写	控制器：每个	REAL64	s		最大滞后误差滤波时间位置	

索引偏移（十六进制）	访问	轴类型	数据类型	物理单位	定义范围	描述	说明
0x00n20014	读/写	控制器：每个	REAL64	例如 mm/s		最大滞后误差速度	
0x00n20015	读/写	控制器：每个	REAL64	s		最大滞后误差滤波时间速度	
0x00n20100	读/写	P/PID（位置、速度）	REAL64	1	[0.0...1.0]	控制器总输出的最大输出限制 (±)	（默认：0.5 == 50%）
0x00n20102	读/写	P/PID（位置）	REAL64	例如 mm/s/ mm	[0.0...1000.0]	比例增益 kp 或 kv 单位：基本单位 / s / 基本单位	位置控制
0x00n20103	读/写	PID（位置）	REAL64	s	[0.0 ... 60.0]	积分作用时间 Tn	位置控制
0x00n20104	读/写	PID（位置）	REAL64	s	[0.0 ... 60.0]	微分作用时间 Tv	位置控制
0x00n20105	读/写	PID（位置）	REAL64	s	[0.0 ... 60.0]	阻尼时间 Td	位置控制
0x00n20106	读/写	PP（位置）	REAL64	例如 mm/s/ mm	[0.0...1000.0]	超过限制速度时适用的额外比例增益，分别为 kp 或 kv，以百分比为单位。 单位：基本单位 / s / 基本单位	位置控制
0x00n20107	读/写	PP（位置）	REAL64	%	[0.0...1.0]	阈值速度（以百分比为单位），超过该速度时，适用额外的比例增益（分别为 kp 或 kv）	
0x00n20108	读/写	P/PID（加速度）	REAL64	s	[0.0 ... 100.0]	比例增益 ka	加速度预控制
0x00n2010D	读/写	P/PID	REAL64	mm	[0.0 ... 10000.0]	位置误差的"死区"（控制偏差） （适用于带速度或扭矩接口的 P/PID 控制器）	保留功能
0x00n2010F	读/写	P/PP/PID（位置） 从轴控制	REAL64	(mm/s) / mm	[0.0...1000.0]	从轴耦合差异控制： 比例增益 k_{cp}	从轴耦合差异控制
0x00n20110	读/写	P（位置）	UINT16	1	0/1	自动偏移校准：主动/被动	
0x00n20111	读/写	P（位置）	UINT16	1	0/1	自动偏移校准：保持模式	
0x00n20112	读/写	P（位置）	UINT16	1	0/1	自动偏移校准：衰减模式	
0x00n20114	读/写	P（位置）	REAL64	%	[0.0 ... 1.0]	自动偏移校准：预控制限制	
0x00n20115	读/写	P（位置）	REAL64	s	[0.1 ... 60.0]	自动偏移校准：时间常数	
0x00n20116	读/写	PID（位置）	REAL64	%	[0.0...1.0]	I 部分的最大输出限制 (±)，以百分比为单位（默认设置：0.1 = 10%）	
0x00n20117	读/写	PID（位置）	REAL64	%	[0.0...1.0]	D 部分的最大输出限制 (±)，以百分比为单位（默认设置：0.1 = 10%）	
0x00n20118	读/写	PID（位置）	UINT16	1	0/1	在主动定位过程中停用 I 部分（假设 I 部分激活）？（默认设置：0 = FALSE）	
0x00n20120	读/写	P/PID（位置）	REAL64	s	⩾0	PT-1 位置误差滤波值（位置控制偏差）	保留功能，无标准！
0x00n20202	读/写	P/PID（速度）	REAL64	1	[0.0...1000.0]	比例增益 kp 或 kv	速度控制
0x00n20203	读/写	PID（速度）	REAL64	s	[0.0 ... 60.0]	积分作用时间 Tn	速度控制
0x00n20204	读/写	PID（速度）	REAL64	s	[0.0 ... 60.0]	微分作用时间 Tv	速度控制

索引偏移（十六进制）	访问	轴类型	数据类型	物理单位	定义范围	描述	说明
0x00n20205	读/写	PID（速度）	REAL64	s	[0.0 ... 60.0]	阻尼时间 Td	速度控制
0x00n20206	读/写	PID（速度）	REAL64	%	[0.0...1.0]	I 部分的最大输出限制(±)，以百分比为单位（默认设置：0.1 = 10%）	速度控制
0x00n20207	读/写	PID（速度）	REAL64	%	[0.0...1.0]	D 部分的最大输出限制(±)，以百分比为单位（默认设置：0.1 = 10%）	速度控制
0x00n2020D	读/写	P/PID（速度）	REAL64	mm/s	[0.0 ... 10000.0]	速度误差的"死区"（控制偏差） （适用于带速度或扭矩接口的 P/PID 控制器）	保留功能
0x00n20220	读/写	P/PID（速度）	REAL64	s	≥0	PT-2 速度误差滤波值（速度控制偏差）	速度控制，非标准！
0x00n20221	读/写	P/PID（速度）	REAL64	s	≥0	PT-1 速度误差滤波值（速度控制偏差）	保留功能，无标准！
0x00n20250	读/写	P/PI（观测器）	UINT32	1	枚举 (>0)	扭矩接口中控制器的观测器模式 [▶ 146] 0：关闭（默认） 1：卢恩伯格（LUENBERGER）	
0x00n20251	读/写	P/PI（观测器）	REAL64	Nm / A	>0.0	电机： 转矩常数 K_T	
0x00n20252	读/写	P/PI（观测器）	REAL64	kg m^2	>0.0	电机： 转动惯量 J_M	
0x00n20253	读/写	P/PI（观测器）	REAL64	Hz	[100.0 ... 2000.0] 默认：500	带宽 f_0	
0x00n20254	读/写	P/PI（观测器）	REAL64	1	[0.0 ... 2.0] 默认：1.0	校正系数 k_c	
0x00n20255	读/写	P/PI（观测器）	REAL64	s	[0.0 ... 0.01] 默认：0.001	速度滤波（一阶）： 时间常量 T	
0x00n20A03	读/写	P/PID (MW)	REAL64	cm^2	[0.0 ... 1000000]	A 面气缸面积 $_{AA}$，单位为 cm^2	保留参数！
0x00n20A04	读/写	P/PID (MW)	REAL64	cm^2	[0.0 ... 1000000]	B 面气缸面积 $_{AB}$，单位为 cm^2	保留参数！
0x00n20A05	读/写	P/PID (MW)	REAL64	cm^3/s	[0.0 ... 1000000]	额定体积流量 $_{Qnom}$，单位为 cm^3/s	保留参数！
0x00n20A06	读/写	P/PID (MW)	REAL64	bar	[0.0 ... 1000000]	额定压力或阀压降，$_{Pnom}$，以bar为单位	保留参数！
0x00n20A07	读/写	P/PID (MW)	UINT32	1	[1 ... 255]	系统压力 Po 的轴 ID	保留参数！
驱动：							
0x00n30001	读取	驱动器：每个	UINT32	1	[1 ... 255]	驱动器 ID	
0x00n30002	读取	驱动器：每个	STRING[30+1]	1	30 个字符	驱动器名称	
0x00n30003	读取	驱动器：每个	UINT32	1	枚举 (>0)	驱动器类型 [▶ 152]	
0x00n30004	读/写	驱动器：每个	UINT32	1	Byteoffset	输入地址偏移（I/O-输入-图像）	更改 I/O 地址
0x00n30005	读/写	驱动器：每个	UINT32	1	Byteoffset	输出地址偏移（I/O-输出-图像）	更改 I/O 地址
0x00n30006	读/写	驱动器：每个	UINT16	1	[0,1]	电机极性	如果已发出启用控制器的信号，则不允许写入。
0x00n3000A	读/写	驱动器：每个	UINT32	1	枚举 (≥0)	驱动模式	

索引偏移（十六进制）	访问	轴类型	数据类型	物理单位	定义范围	描述	说明
0x00n3000B	读/写	驱动器：每个	REAL64	%	[-1.0 ... 1.0]	最小输出限制（输出限制）（默认设置：-1.0 = -100）	
0x00n3000C	读/写	驱动器：每个	REAL64	%	[-1.0 ... 1.0]	最大输出限制（输出限制）（默认设置：1.0 = 100%）	
0x00n3000D	读取	驱动器：每个	UINT32	INC		输出增量的最大数量（输出掩码）	
0x00n30010	读/写	驱动器：每个	UINT32	1		内部驱动控制双字，用于确定驱动运行模式	保留！
0x00n30011	读/写	每个	UINT32	1	≥ 5	内部驱动复位计数器（启用和复位的 NC 循环时间）	保留！
0x00n30101	读/写	D：伺服	REAL64	例如 mm/s	>0.0	参考输出的参考速度（速度预控制）	
0x00n30102	读/写	D：伺服	REAL64	%	[0.0 ... 5.0]	以百分比为单位的参考输出（默认设置：1.0 = 100%）	
0x00n30103	读取	D：伺服	REAL64	例如 mm/s	>0.0	100% 输出时产生的速度	
0x00n30104	读/写	D：伺服	REAL64	例如 mm/s	±∞	用于轴漂移校准（偏移校准）的速度偏移（DAC 偏移）	
0x00n30105	读/写	D：伺服（Sercos、Profi、Drive、AX200x、CANopen）	REAL64	1	[0.0 ... 100000000.0]	速度缩放（响应驱动器中权重的缩放因子（scaling factor））	适用于 Sercos、Profi Drive、AX200x、CANopen
0x00n30106	读/写	D: Profi Drive DSC	UINT32	0.001 * 1/s	≥ 0	Profibus/Profi Drive DSC：位置控制增益 Kpc	仅适用于 Profi Drive DSC
0x00n30107	读/写	D: Profi Drive DSC	REAL64	1	≥ 0.0	Profibus/Profi Drive DSC：用于计算 'XERR' 的缩放比例（默认：1.0）	仅适用于 Profi Drive DSC
0x00n30109	读/写	D：伺服（Sercos、CANopen）	REAL64	1	[0.0 ... 100000000.0]	位置缩放（响应驱动器中权重的缩放因子（scaling factor））	适用于 Sercos、CANopen
0x00n3010A	读/写	D：伺服（Sercos、Profi、Drive、AX200x、CANopen）	REAL64	1	[0.0 ... 100000000.0]	加速度缩放（响应驱动器中权重的缩放因子（scaling factor））	适用于 Sercos、Profi Drive、AX200x、CANopen
0x00n3010B	读/写	D：伺服（Sercos、Profi、Drive、AX200x、CANopen）	REAL64	1	[0.0 ... 100000000.0]	适用于"TorqueOffset"的扭矩缩放（旋转电机）或力缩放（直线电机）（响应驱动器中权重的缩放因子（scaling factor））（作为预控制的加法力矩）	适用于 Sercos、Profi Drive、AX200x、CANopen
0x00n3010C	读/写	D：伺服（Sercos、Profi、Drive、AX200x、CANopen）	REAL64	1	[0.0 ... 100000000.0]	适用于"SetTorque"的扭矩缩放（旋转电机）或力缩放（直线电机）（响应驱动器中重量的缩放因子（scaling factor））（例如 MC_TorqueControl，使用驱动器运行模式 CST）	适用于 Sercos、Profi Drive、AX200x、CANopen TC3.1 B4024.2 及以上

索引偏移（十六进制）	访问	轴类型	数据类型	物理单位	定义范围	描述	说明
0x00n30120	读/写	D: servo/ hydraulics/	UINT32	1	⩾ 0	表 ID （0：无表）	仅适用于 KL4xxx、M2400、通用型
0x00n30121	读/写	D：伺服/液压	UINT32	1	⩾ 0	插值类型 0：线性 2：样条	仅适用于 KL4xxx、M2400、通用型
0x00n30122	读/写	伺服/液压	REAL64	%	[-1.0 ... 1.0]	输出偏移（以百分比为单位） 注：根据特征评估进行操作！	仅适用于 KL4xxx、M2400、通用型
0x00n30151	读/写	D：伺服/非线性	REAL64	1	[0.0 ... 100.0]	象限补偿系数（I 和 III 象限之间的关系）	
0x00n30152	读/写	D：伺服/非线性	REAL64	1	[0.01 ... 1.0]	以百分比为单位的速度参考点 (1.0 = 100 %)	
0x00n30153	读/写	D：伺服/非线性	REAL64	1	[0.01 ... 1.0]	以百分比为单位的输出参考点 (1.0 = 100 %)	
0x00030301	读/写	D：步进电机	UINT8	1		位掩码：循环 1	
0x00030302	读/写	D：步进电机	UINT8	1		位掩码：循环 2	
0x00030303	读/写	D：步进电机	UINT8	1		位掩码：循环 3	
0x00030304	读/写	D：步进电机	UINT8	1		位掩码：循环 4	
0x00030305	读/写	D：步进电机	UINT8	1		位掩码：循环 5	
0x00030306	读/写	D：步进电机	UINT8	1		位掩码：循环 6	
0x00030307	读/写	D：步进电机	UINT8	1		位掩码：循环 7	
0x00030308	读/写	D：步进电机	UINT8	1		位掩码：循环 8	
0x00030310	读/写	D：步进电机	UINT8	1		位掩码：保持电流	

3.2.1.5.4.4.2 *轴状态的"索引偏移"规范（索引组 0x4100 + ID）*

索引偏移（十六进制）	访问	轴类型	数据类型	物理单位	定义范围	描述	说明
0x00n00000	读取	每个（轴数据的在线结构）	{			轴在线结构 (NC/CNC)	从 TwinCAT 3 开始发生变化，无法通过示波器观察！(NCAXISSTATE_ONLINESTRUCT)
			INT32	1		错误状态	
			INT32			预留	
			REAL64	例如 mm		实际位置	
			REAL64	例如度数		模轴实际位置	
			REAL64	例如 mm		设置位置	
			REAL64	例如度数		模轴设置位置	
			REAL64	例如 mm/s		可选：实际速度	
			REAL64	例如 mm/s		设置速度	
			UINT32	%	0...1000000	速度超驰 (1000000 == 100%)	
			UINT32			预留	
			REAL64	例如 mm		滞后误差位置	
			REAL64	例如 mm		最大负位置滞后的峰值保持值（位置）	
			REAL64	例如 mm		最大位置滞后的峰值保持值（位置）	
			REAL64	%		控制器输出（以百分比为单位）	
			REAL64	%		总输出（以百分比为单位）	
			UINT32	1	⩾ 0	轴状态双字	
			UINT32	1	⩾ 0	轴控制双字	
			UINT32	1	⩾ 0	从轴耦合状态（状态）	
			UINT32	1	0; 1,2,3...	轴控制环路索引	
			REAL64	例如 mm/s^2		实际加速度	
			REAL64	例如 mm/s^2		设置加速度	
			REAL64	例如 mm/s^3		设置加加速度（从 TwinCAT 3.1 B4013 起的新功能）	
			REAL64	例如，100% = 1000		设置扭矩或设置力 ("SetTorque")	
			REAL64	例如，100% = 1000		实际扭矩或实际力（从 TwinCAT 3.1 B4013 起的新功能）	
			REAL64	例如 %/s		设置扭矩变化或设置力变化（设置扭矩或设置力的时间导数）（从 TwinCAT 3.1 B4024.2 起）	
			REAL64	例如，100% = 1000		附加设定扭矩或附加设定力 ("TorqueOffset")（从 TwinCAT 3.1 B4024.2 起）	
			. . .				
			}			256 字节	
0x00000001	读取	每个	UINT32	1		轴状态错误代码	符号访问："ErrState''
0x00n00009	读取	每个	UINT32	1	⩾ 0	设置循环计数器（SAF 时间戳）	

索引偏移（十六进制）	访问	轴类型	数据类型	物理单位	定义范围	描述	说明
0x00n0000A	读取	每个	REAL64	例如 mm		设置位置	符号访问: "SetPos''
0x00n0000B	读取	每个	REAL64	例如度数		模轴设置位置	符号访问："SetPosModulo''
0x00n0000C	读取	每个	INT32	1		模轴设置旋转	
0x00n0000D	读取	每个	REAL64	1	[-1.0, 0.0, 1.0]	设置行进方向	
0x00n0000E	读取	每个	REAL64	例如 mm/s		设置速度	符号访问："SetVelo''
0x00n0000F	读取	每个	REAL64	例如 mm/s^2		设置加速度	符号访问："SetAcc''
0x00n00010	读取	每个	REAL64	例如 mm/s^3		设置加加速度（设置加速度的时间导数）	符号访问："SetJerk''
0x00n00011	读取	每个	REAL64	例如，分别为 Nm 或 N， 例如，100% = 1000		设置扭矩（旋转电机） 或 设置力（直线电机） ("SetTorque")	从 TwinCAT 3.1 B4022 起的新功能 符号访问: "SetTorque''
0x00n00012	读取	每个	REAL64	1		设置耦合系数（设置齿轮比）	
0x00n00013	读取	每个	REAL64	例如 mm		预期目标位置	
0x00n00014	读取	伺服	{			剩余行进时间和距离（伺服）：	始终连接至 SEC 501 端口!
			REAL64	s	≥ 0	剩余行进时间	
			REAL64	例如 mm	≥ 0	剩余距离	
			}				
0x00n00015	读取	每个	UINT32	1	≥ 0	设置命令编号	符号访问："CmdNo''
0x00n00016	读取	伺服	REAL64	s	≥ 0	最后运动命令的定位时间（起始 → 目标位置窗口）	
0x00n00017	读取	伺服	REAL64	%	[0.0…1.0] 1.0=100%	为速度设置速率比 注：最初仅针对 FIFO 组执行	从 TwinCAT 3.1 B4020 起的新功能
0x00000018	读写	伺服	**写入**			读取"停止信息"（停止距离、停止时间）	始终连接至 SEC 501 端口!
			REAL64	例如 mm/s^2	≥ 0	轴停止减速度	
			REAL64	例如 mm/s^3	≥ 0	轴停止加加速度	
			读取				
			REAL64	例如 mm	≥ 0	停止距离	
			REAL64	s	≥ 0	停止时间	
0x00n0001A	读取	每个	REAL64	例如 mm		未校正设置位置	
0x00n0001D	读取	每个	REAL64	1	[-1.0, 0.0, 1.0]	未校正设置行进方向	
0x00n0001E	读取	每个	REAL64	例如 mm/s		未校正设置速度	
0x00n0001F	读取	每个	REAL64	例如 mm/s^2		未校正设置加速度	
0x00000020	读取	每个	UINT32	1	参见 枚举	耦合状态（状态）	
0x00000021	读取	每个	UINT32	1	≥ 0	耦合表索引	

索引偏移（十六进制）	访问	轴类型	数据类型	物理单位	定义范围	描述	说明
0x00000022	读取	伺服主轴/从轴耦合 类型：线性，(&SPECIAL)	{			读取耦合参数（伺服）：	
			REAL64	1	[±1000000.0]	参数 1：线性：齿轮比	
			REAL64	1	[±1000000.0]	参数 2：线性：保留	
			REAL64	1	[±1000000.0]	参数 3：线性：保留	
			REAL64	1	[±1000000.0]	参数 4：线性：保留	
			}				
0x00000023	读取	伺服主轴/从轴耦合 类型：线性，(&SPECIAL)	REAL64	1	[±1000000.0]	读取齿轮比（伺服） 类型线性	
0x00000024	读取	伺服	UINT32	1	⩾ 0	活动轴控制电路的编号/索引（编码器、控制器和轴接口三合一）	
0x00000025	读取	伺服	UINT16	1	0/1	是否激活通过轴接口 PCLtoNC 指定外部设定点?	
0x00000026	读取	伺服主轴/从轴耦合 类型：同步	REAL64 [64]	1	±∞	读取从轴同步配置文件的特征值 类型：同步	从 TwinCAT 3 起修改
0x00000027	读写	伺服主轴/从轴耦合 类型：表格、MF	**写入**			读取"表耦合信息"	仅端口 500! 从 TwinCAT 3 起修改
			VOID 或 REAL64 或 dword、dword、real64	例如 mm	±∞	- 没有"当前信息"数据 - 某个"主轴位置"可选 - 适用于某个表 ID 和可选"主轴位置"（TC 3.1 B4017）	
			读取				
			REAL64 [32]		±∞	读取表耦合信息 [▶ 155]的结构	
0x00000028	读写	伺服主轴/从轴耦合 类型：MULTICAM (CamAddition)	**写入**			读取"多表耦合信息"（CamAddition）	仅端口 500!
			UINT32	1	⩾ 0	查询所涉及的表 ID	
			读取				
			96 字节			读取多表耦合信息 [▶ 155]的结构	
0x00000029	读取	伺服	UINT32	1		延迟错误反应时的延迟错误代码（错误预先警告）（参见位 ErrorPropagationDelayed）	
0x0000002A	读取	伺服	REAL64	例如 mm	±∞	从设置位置衰减到实际位置时的位置差（衰减部分）	
0x0000002B	读取	伺服	REAL64	例如 mm/s	±∞	从设置位置衰减到实际位置时的相对速度（衰减部分）	
0x0000002C	读取	伺服	REAL64	例如 mm/s ^2	±∞	从设置位置衰减到实际位置时的相对加速度（衰减部分）	
0x0000002D	读取	伺服	UINT32	1	⩾ 0	初始化命令计数器 (InitializeCommandCounter)	新
0x0000002E	读取	伺服	UINT32	1	⩾ 0	复位命令计数器 (ResetCommandCounter)	新

索引偏移（十六进制）	访问	轴类型	数据类型	物理单位	定义范围	描述	说明
0x00000030	读取	伺服	REAL64	例如 Nm/s 或 N/s	±∞	设置扭矩变化（旋转电机）或 设置力变化（直线电机） （设置扭矩或设置力的时间导数）	从 TwinCAT 3.1 B4024 起的新功能
0x00000031	读/写	伺服	REAL64	例如，分别为 Nm 或 N， 例如，100% = 1000		用于预控制的附加设定扭矩（旋转电机）或加性附加设定力（直线电机）。 ("TorqueOffset")	从 TwinCAT 3.1 B4024.2 起 符号访问："TorqueOffset''
0x00000040	读取	伺服	UINT32	1	⩾ 0	用于在数据不一致的情况下校正 NC 设定点的计数器 （通过 Idx-Group 0x1000 和 Idx-Offset 0x0020 激活）	从 TwinCAT 3.1 B4020 起的新功能
0x00000050	读取	每个	UINT32	1		设置行进阶段 (SWGenerator)	无法用示波器追踪！
0x00000051	读取	每个	UINT16	1		轴是否被禁用？	无法用示波器追踪！
0x00n00060	读/写	每个（在线设定点结构） 40 字节	{			简单轴设定点结构 (NC/CNC)	无法用示波器追踪！ TC 3.1 B4022.30 及以上
			REAL64	例如 mm		设置位置	
			REAL64	例如 mm/s		设置速度	
			REAL64	例如 mm/s^2		设置加速度/减速度	
			REAL64	1	[-1.0, 0.0, 1.0]	设置行进方向	
			REAL64	例如 mm/s^3		设置加加速度	
			}				
0x00n00060	读/写	每个（在线设定点结构） 56 字节	{			扩展轴设定点结构 (NC/CNC)	无法用示波器追踪！ TC 3.1 B4022.29 及以上
			REAL64	例如 mm		设置位置	
			REAL64	例如 mm/s		设置速度	
			REAL64	例如 mm/s^2		设置加速度/减速度	
			REAL64	1	[-1.0, 0.0, 1.0]	设置行进方向	
			REAL64	例如 mm/s^3		设置加加速度	
			REAL64	Nm 或 N 或 %		设置扭矩或设置力	
			REAL64	Nm/s 或 N/s 或 %/s		设置扭矩或设置力的时间导数（斜坡）	
			}				

索引偏移（十六进制）	访问	轴类型	数据类型	物理单位	定义范围	描述	说明
0x00n00061	读/写	每个（在线动态设定点结构） 32 字节	{			轴动态设定点结构 (NC/CNC)	TC 3.1 B4022.30 及以上
			REAL64	例如 mm/s		设置速度	
			REAL64	例如 mm/s^2		设置加速度/减速度	
			REAL64	1	[-1.0, 0.0, 1.0]	设置行进方向	
			REAL64	例如 mm/s^3		设置加加速度	
			}				
0x00n00061	读/写	每个（在线动态设定点结构） 48 字节	{			轴动态设定点结构 (NC/CNC)	TC 3.1 B4022.29 及以上
			REAL64	例如 mm/s		设置速度	
			REAL64	例如 mm/s^2		设置加速度/减速度	
			REAL64	1	[-1.0, 0.0, 1.0]	设置行进方向	
			REAL64	例如 mm/s^3		设置加加速度	
			REAL64	Nm 或 N 或 %		设置扭矩或设置力	
			REAL64	Nm/s 或 N/s 或 %/s		设置扭矩或设置力的时间导数（斜坡）	
			}				
0x00n00062	读/写	每个（在线扭矩设定点结构） 16 字节	{			扭矩设定点结构 (NC/CNC)	TC 3.1 B4022.30 及以上
			REAL64	Nm 或 N 或 %		设置扭矩或设置力	
			REAL64	Nm/s 或 N/s 或 %/s		设置扭矩或设置力的时间导数（斜坡）	
			}				
0x00000063	读写	仅适用于 SERCOS/SoE 和 CANopen/CoE	**写入**			读取活动"驱动运行模式"	从 TC 3.1 B4022（NC 4443）起更新 始终连接至 SEC 501 端口！
			UINT32	1		保留	
			UINT32	1		保留	
			读取				
			INT32	枚举 [▶ 153]（参见附录）	[0; 1, 2, 3, …] 特殊情况： ⩾ 100：SoE <0：CoE	当前活动"驱动运行模式"（通用模式）	
			UINT32	1		保留	
0x00n10002	读取	每个（编码器）	REAL64	例如 mm		实际位置（带有实际位置补偿值） n = 0：轴的标准编码器 > 0：轴的第 n 个编码器（可选）	符号访问："ActPos''
0x00n10003	读取	每个（编码器）	REAL64	例如度数		模轴实际位置	符号访问："ActPosModulo''
0x00n10004	读取	每个（编码器）	INT32	1		模轴实际旋转	
0x00n10005	读取	每个（编码器）	REAL64	例如 mm/s		可选：实际速度	符号访问："ActVelo''
0x00n10006	读取	每个（编码器）	REAL64	例如 mm/s^2		可选：实际加速度	符号访问："ActAcc''
0x00n10007	读取	每个（编码器）	INT32	INC		编码器实际增量	

索引偏移（十六进制）	访问	轴类型	数据类型	物理单位	定义范围	描述	说明
0x00n10008	读取	每个（编码器）	INT64	INC		软件 — 实际增量计数器	
0x00n10009	读取	每个（编码器）	UINT16	1	0/1	参考 标志（“校准 标志”）	
0x00n1000A	读取	每个（编码器）	REAL64	例如 mm		实际位置校正值（测量系统错误校正）	
0x00n1000B	读取	每个（编码器）	REAL64	例如 mm		无实际位置补偿值的实际位置	无法用示波器追踪！
0x00n10010	读取	每个（编码器）	REAL64	例如 mm/s		无实际位置补偿值的实际速度	
0x00n10012	读取	每个（编码器）	REAL64	例如 mm		未滤波实际位置（带有实际位置补偿值）	
0x00n10014	读取	编码器：SoE、CoE、MDP 742	REAL64	例如 mm/s		可选：实际驱动速度（直接从 SoE、CoE 或 MDP 742 驱动器传输）	从 TwinCAT 3.1 B4020.30 起的新功能
0x00n10015	读取	每个（编码器）	REAL64	例如 mm/s		可选：未滤波实际速度	
0x00n10017	读取		REAL64	例如 mm		读出 MC_SetPosition 偏移	
0x00n10018	读取	PTP 轴 编码器轴	UINT32	0/1	0/1	在 NC 编码器重新初始化已开始后，返回重新初始化的状态（索引组 0x4200+ID；索引偏移 0x00n0003B）。 n = 0：轴的标准编码器 n > 0：轴的第 n 个编码器（可选）	端口 501
0x00n10101	读取	INC（编码器）	REAL64	例如 mm		内部硬件锁存器启动和生效时间之间的位置差的反馈	无法用示波器追踪！
0x00n20001	读取	R：每个	INT32	1		控制器的错误状态 n = 0：轴的标准控制器 > 0：轴的第 n 个控制器（可选）	
0x00n20002	读取	R：每个	REAL64	例如 mm/s		控制器输出（以绝对单位表示）	符号访问："CtrlOutput''
0x00n20003	读取	R：每个	REAL64	%		控制器输出（以百分比为单位）	无法用示波器追踪！
0x00n20004	读取	R：每个	REAL64	V		控制器输出（以伏特为单位）	无法用示波器追踪！
0x00n2000D	读取	R：每个	REAL64	例如 mm		滞后误差位置（无死区时间补偿）	基本单位
0x00n2000F	读取	R：每个	REAL64	例如 mm		滞后误差位置（有死区时间补偿）	符号访问："PosDiff''
0x00n20010	读取	R：每个	REAL64	例如 mm		位置最大负滞后误差的峰值保持值	
0x00n20011	读取	R：每个	REAL64	例如 mm		位置最小正滞后误差的峰值保持值	
0x00n20012	读取	R：每个	REAL64	例如 mm/s		滞后误差速度	未执行！

索引偏移（十六进制）	访问	轴类型	数据类型	物理单位	定义范围	描述	说明
0x00n20021	读取	R：每个	REAL64	例如 mm		主轴与从轴位置滞后误差之间的差（偏差）（主轴滞后误差减去从轴滞后误差）	符号访问："PosDiffCouple''
0x00n20022	读取	R：每个	REAL64	例如 mm		主轴和从轴位置滞后误差之间最大负差的峰值保持值	基本单位
0x00n20023	读取	R：每个	REAL64	例如 mm		主轴和从轴位置滞后误差之间最大正差的峰值保持值	基本单位
0x00n20101	读取	R：P/PID（位置）	REAL64	例如 mm/s		控制器 P(比例) 部分（以绝对单位表示）	
0x00n20102	读取	R：PID（位置）	REAL64	例如 mm/s		控制器 I(积分) 部分（以绝对单位表示）	
0x00n20103	读取	R：PID（位置）	REAL64	例如 mm/s		控制器 D(微分) 部分（以绝对单位表示）	
0x00n20104	读取	R：PID（位置）	UINT16	1	0/1	I(积分) 部分限制激活?	
0x00n20105	读取	R：PID（位置）	UINT16	1	0/1	D(微分) 部分限制激活?	
0x00n20106	读取	R：PID（位置）	UINT16	1	0/1	I 部分 ARW 测量激活? ARW：防积分饱和	未执行!
0x00n20110	读取	R：PID（位置）	REAL64	例如 mm/s		控制器的加速度预控制 Yacc（以绝对单位表示） 注：功能取决于控制器类型!	加速度预控制
0x00n20111	读取	R: PP（位置）	REAL64	mm/s/mm	⩾0	内部插值比例增益 kp 或 kv	PP 控制器
0x00n20201	读取	R：P,PID（速度）	REAL64	例如 mm/s		控制器的速度部分	
0x00n20202	读取	R：P,PID（速度）	REAL64	%		控制器的速度部分（以百分比为单位）	无法用示波器追踪!
0x00n20203	读取	R：P,PID（速度）	REAL64	V		控制器的速度部分（以伏特为单位）	无法用示波器追踪!
0x00n20201	读取	R：P/PID（速度）	REAL64	例如 mm/s		控制器 P(比例) 部分（以绝对单位表示）	
0x00n20202	读取	R：P/ PID（速度）	REAL64	例如 mm/s		控制器 I(积分) 部分（以绝对单位表示）	
0x00n20203	读取	R：P/ PID（速度）	REAL64	例如 mm/s		控制器 D(微分) 部分（以绝对单位表示）	
0x00n20204	读取	R：P/ PID（速度）	UINT16	1	0/1	I(积分) 部分限制激活?	
0x00n20205	读取	R：P/ PID（速度）	UINT16	1	0/1	D(微分) 部分限制激活?	
0x00n20206	读取	R：P/ PID（速度）	UINT16	1	0/1	I(积分) 部分 ARW 测量激活?（ARW：积分抗饱和）	
0x00n2020A	读取	R：P/ PID（速度）	REAL64	例如 mm/s		速度控制器的总输入大小	
0x00n20A00	读取	R: PID (MW)	REAL64	%	[-1.0...1.0]	设置速度的位移（预控制）	保留参数!
0x00n20A01	读取	R: PID (MW)	REAL64	例如 mm/s		控制器的 P(比例) 部分以绝对单位或百分比表示（根据输出权重计算）	保留参数!

索引偏移（十六进制）	访问	轴类型	数据类型	物理单位	定义范围	描述	说明
0x00n20A02	读取	R: PID (MW)	REAL64	例如 mm/s		控制器的 I(积分) 部分以绝对单位或百分比表示（根据输出权重计算）	保留参数！
0x00n20A03	读取	R: PID (MW)	REAL64	例如 mm/s		控制器的 D(微分) 部分以绝对单位或百分比表示（根据输出权重计算）	保留参数！
0x00n20A04	读取	R: PID (MW)	UINT16	1	0/1	I(积分) 部分限制激活?	保留参数！
0x00n20A05	读取	R: PID (MW)	UINT16	1	0/1	D(微分) 部分限制激活?	保留参数！
0x00n20A06	读取	R: PID (MW)	UINT16	1	0/1	I(积分) 部分 ARW 测量激活? ARW：防积分饱和	保留参数！
0x00n20A10	读取	R: PID (MW)	REAL64	例如 mm/s		控制器的加速度预控制 Yacc（以绝对单位表示）	保留参数！
0x00n30001	读取	D：每个	INT32	1		驱动器的错误状态	
0x00n30002	读取	D：每个	REAL64	例如 mm/s		总输出（以绝对单位表示）	符号访问："DriveOutput""
0x00n30003	读取	D：每个	REAL64	%		总输出（以百分比为单位）	
0x00n30004	读取	D：每个	REAL64	V		总输出（以伏特为单位）	无法用示波器追踪！
0x00n30005	读取	D：每个	REAL64	例如 mm/s		最大负总输出的峰值保持值	
0x00n30006	读取	D：每个	REAL64	例如 mm/s		最大正总输出的峰值保持值	
0x00n30007	读取	D：每个	REAL64	例如 100% = 1000，例如 Nm 或 N		分别为实际扭矩或实际力（通常 100% = 1000）	从 TwinCAT 3.1 B4022 起 符号访问："ActTorque
0x00n30008	读取	D：每个	REAL64	例如 Nm/s 或 N/s	±∞	分别为实际扭矩变化或实际力变化 （分别为实际扭矩或实际力的时间导数）	从 TwinCAT 3.1 B4024 起
0x00n30013	读取	D：每个	REAL64	%		总输出以百分比为单位（基于非线性特征曲线！）	
0x00n30014	读取	D：每个	REAL64	V		总输出以伏特为单位（基于非线性特征曲线！）	无法用示波器追踪！
0x00n3011A	读取	D：伺服（Sercos、CANopen）	REAL64	例如 mm		可选输出滤波：滤波设置位置	新 适用于 Sercos、CANopen
0x00n3011E	读取	D：伺服（Sercos、CANopen）	REAL64	例如 mm/s		可选输出滤波：滤波设置速度	新 适用于 Sercos、CANopen
0x00n3011F	读取	D：伺服（Sercos、CANopen）	REAL64	例如 mm/s^2		可选输出滤波：滤波设置加速度/设置减速度	新 适用于 Sercos、CANopen

3.2.1.5.4.4.3 *轴功能的"索引偏移"规范（索引组 0x4200 + ID）*

索引偏移（十六进制）	访问	轴类型	数据类型	物理单位	定义范围	描述	说明
0x00000001	写入	每个	VOID			复位轴	也适用于 FIFO 轴！
0x00000002	写入	每个	VOID			停止轴	也适用于 FIFO 轴！
0x00000003	写入	每个	VOID			清除轴（任务）	也适用于 FIFO 轴！
0x00000004	写入	每个	{			紧急停止（使用控制斜坡）	仅适用于 PTP 轴！
			REAL64	例如 mm/s^2	> 0.0	减速度（必须大于或等于原始减速度）	
			REAL64	例如 mm/s^3	> 0.0	加加速度（必须大于或等于原始加加速度）	
			}				
0x00000005	写入	PTP 轴	{			可参数化停止（使用控制斜坡）	仅适用于 PTP 轴！ 保留功能，无标准！
			REAL64	例如 mm/s^2	> 0.0	减速度	
			REAL64	例如 mm/s^3	> 0.0	加加速度	
			}				
0x00000009	写入	PTP 轴	{			定向停止（定向结束位置）	仅适用于 PTP 轴！
			REAL64	例如度数	⩾ 0.0	模除结束位置（模除目标位置）	
			REAL64	例如 mm/s^2	> 0.0	减速度（目前未激活）	
			REAL64	例如 mm/s^3	> 0.0	加加速度（尚未执行）	
			}				
0x00000010	写入	每个	VOID			参考轴（“校准”）	
0x00000011	写入	每个	{			轴的新结束位置	从 TwinCAT 3 起修改
			UINT32	枚举	参见附录	结束位置类型 [▶ 145]（参见附录）	
			UINT32			预留（TwinCAT 3）	
			REAL64	例如 mm	±∞	新结束位置（目标位置）	
			}				
0x00000012	写入	每个	{			轴的新结束位置和新速度	
			UINT32	枚举	参见附录	命令类型 [▶ 145]（参见附录）	
			UINT32	枚举	参见附录	结束位置类型 [▶ 145]（参见附录）	
			REAL64	例如 mm	±∞	新结束位置（目标位置）	
			REAL64	例如 mm/s	⩾ 0.0	新最终速度（要求的行进速度）	
			REAL64	例如 mm	±∞	可选：激活新行进配置文件的切换位置	
			}				
0x00000015	写入	每个	{			用于主动定位的新动态参数	
			REAL64	例如 mm/s^2	> 0.0	加速度	
			REAL64	例如 mm/s^2	> 0.0	减速度	
			REAL64	例如 mm/s^3	> 0.0	可选：加加速度（尚未执行）	
			}				

索引偏移（十六进制）	访问	轴类型	数据类型	物理单位	定义范围	描述	说明
0x00000016	**ReadWrite**	每个伺服	**写入**（80 字节）			通用轴启动 (UAS)：通过整合单轴启动指令与在线参数修改功能，并配合"缓冲模式"（请参见 TcMc2.lib）实现动态控制流程的无缝衔接	始终连接至 SEC 501 端口！ 从 TwinCAT 3 起修改
			{				
			UINT32	枚举	参见附录	启动类型 [▶ 144]（参见附录）	
			UINT32	1	≥ 0	用于检查和运行模式的位掩码（默认值：0）	
			REAL64	例如 mm	±∞	结束位置（目标位置）	
			REAL64	例如 mm/s	≥ 0.0	所需速度 Vrequ	
			REAL64	例如 mm/s^2	≥ 0.0	可选：加速度	
			REAL64	例如 mm/s^2	≥ 0.0	可选：减速度	
			REAL64	例如 mm/s^3	≥ 0.0	可选：加加速度	
			UINT32	枚举	参见附录	缓冲模式 [▶ 144]（命令缓冲）	
			UINT32			预留（TwinCAT 3）	
			REAL64	例如 mm	±∞	可选：混合位置（命令混合位置）	
			REAL64	例如 mm/s	≥ 0.0	可选：初始分段速度 Vi (0 ≤ Vi ≤ Vrequ)	
			REAL64	例如 mm/s	≥ 0.0	可选：分段结束速度 Vf (0 ≤ Vf ≤ Vrequ)	
			}				
			读取				
			{				
			UINT16	1	≥ 0	命令编号（工作编号）	
			UINT16	1	≥ 0	命令状态	
			}				

索引偏移（十六进制）	访问	轴类型	数据类型	物理单位	定义范围	描述	说明
0x00000017	**ReadWrite**	伺服	**写入**（80 字节）			"主轴/从轴解耦"和"通用轴启动 (UAS)"： 合并从轴的解耦命令（IdxOffset: 0x00000041）和后续通用轴启动 (UAS)（IdxOffset: 0x00000016）	尚未发布！
			{				
			UINT32	枚举	参见附录	启动类型 [▶ 144]（参见附录）	
			UINT32	1	⩾ 0	检查和运行模式的位掩码（默认值：0）	
			REAL64	例如 mm	±∞	结束位置（目标位置）	
			REAL64	例如 mm/s	⩾ 0.0	所需速度 Vrequ	
			REAL64	例如 mm/s^2	⩾ 0.0	加速度	
			REAL64	例如 mm/s^2	⩾ 0.0	减速度	
			REAL64	例如 mm/s^3	⩾ 0.0	加加速度	
			UINT32	枚举	参见附录	缓冲模式 [▶ 144]（命令缓冲）	
			UINT32			预留（TwinCAT 3）	
			REAL64	例如 mm	±∞	可选：混合位置（命令混合位置）	
			REAL64	例如 mm/s	⩾ 0.0	可选：初始分段速度 Vi (0 ⩽ Vi ⩽ Vrequ)	
			REAL64	例如 mm/s	⩾ 0.0	可选：分段结束速度 Vf (0 ⩽ Vf ⩽ Vrequ)	
			}				
			读取				
			{				
			UINT16	1	⩾ 0	命令编号（工作编号）	
			UINT16	1	⩾ 0	命令状态	
			}				
0x00000018	写入	每个	VOID			运动命令释放轴锁 (TcMc2)	
0x00000019	写入	每个	UINT32	1	> 0	设置外部轴错误（运行时错误）	**使用时请注意！**
0x00n0001A	写入	每个	{			设置实际轴位置	**使用时请注意！** 也适用于 FIFO 轴！ 始终连接至 SEC 501 端口！ 从 TwinCAT 3 起修改
			UINT32	枚举	参见附录	实际位置类型 [▶ 145]（参见附录）	
			UINT32			预留（TwinCAT 3）	
			REAL64	例如 mm	±∞	轴的实际位置 n = 0：轴的标准编码器 n > 0：轴的第 n 个编码器（可选）	
			}				
0x00n0001B	写入	每个	UINT32	1	0/1	设置参考标志（“校准标志”） n = 0：轴的标准编码器 n > 0：轴的第 n 个编码器（可选）	**使用时请注意！** 也适用于 FIFO 轴！

索引偏移（十六进制）	访问	轴类型	数据类型	物理单位	定义范围	描述	说明
0x00n0001C	写入	伺服	{			仅设置实际轴位置，不干预设定位置（也适用于及运行中进程）	**使用时请注意！**
			UINT32	枚举	参见附录	实际位置类型 [▶ 145]（参见附录）	
			REAL64	例如 mm	±∞	轴的实际位置 n = 0：轴的标准编码器 > 0：轴的第 n 个编码器（可选） **使用时请注意！**	
			}				
0x00n0001D	写入	每个	{			驱动侧轴的实际值设置（假设位置接口和编码器偏移为零！）。 n = 0：轴的标准编码器 n > 0：轴的第 n 个编码器（可选）	**使用时请注意！** 仅适用于 CANopen！
			UINT32	枚举	参见附录	实际位置类型 [▶ 145]（参见附录）	
			REAL64	例如 mm	±∞	轴的实际位置	
			}				
0x00n0001E	写入	每个	{			实时设置新编码器缩放因子（scaling factor）（在轴的运动过程中）	**使用时请注意！** 始终连接至 SEC 501 端口！ 从 TwinCAT 3 起修改
			UINT16	枚举	1	编码器缩放因子（scaling factor）类型 1：绝对 2：相对	
			UINT16			ControlWord	
			UINT32			预留（TwinCAT 3）	
			REAL64	例如 mm/INC	[1.0E-8 ... 100.0]	新编码器缩放因子（scaling factor） n = 0：轴的标准编码器 n > 0：轴的第 n 个编码器（可选）	
			}				
0x00n0001F	写入	每个	{			实时设置实际轴位置（在轴的运动过程中）	**使用时请注意！** 始终连接至 SEC 501 端口！
			UINT32	枚举		用于实时设置实际值的位置类型 1：绝对 2：相对	
			UINT32	1		控制双字，例如用于"清除滞后误差"	
			REAL64			保留	
			REAL64	例如 mm	±∞	新实际轴位置	
			UINT32			保留	
			UINT32			保留	
			}				
0x00000020	写入	每个 1D 启动	{			标准轴启动：	从 TwinCAT 3 起修改
			UINT32	枚举	参见附录	启动类型 [▶ 144]（参见附录）	
			UINT32			预留（TwinCAT 3）	
			REAL64	例如 mm	±∞	结束位置（目标位置）	
			REAL64	例如 mm/s	≥0.0	所需速度	
			}				

索引偏移（十六进制）	访问	轴类型	数据类型	物理单位	定义范围	描述	说明
0x00000021	写入	每个 1D 启动	{			扩展轴启动（伺服）：	从 TwinCAT 3 起修改
			UINT32	枚举	参见附录	启动类型 [▶ 144]（参见附录）	
			UINT32			预留（TwinCAT 3）	
			REAL64	例如 mm	±∞	结束位置（目标位置）	
			REAL64	例如 mm/s	≥ 0.0	所需速度	
			UINT32	0/1	0/1	标准加速度?	
			UINT32			预留（TwinCAT 3）	
			REAL64	例如 mm/s^2	≥ 0.0	加速度	
			UINT32	0/1	0/1	标准减速度?	
			UINT32			预留（TwinCAT 3）	
			REAL64	例如 mm/s^2	≥ 0.0	减速度	
			UINT32	0/1	0/1	标准加加速度?	
			UINT32			预留（TwinCAT 3）	
			REAL64	例如 mm/s^3	≥ 0.0	加加速度	
			}				
0x00000022	写入	伺服 (MW)	{			特殊轴启动（伺服）：	保留启动功能，无标准! 从 TwinCAT 3 起修改
			UINT32	枚举	参见附录	启动类型 [▶ 144]（参见附录）	
			UINT32			预留（TwinCAT 3）	
			REAL64	例如 mm	±∞	结束位置（目标位置）	
			REAL64	mm/s	≥ 0.0	所需启动速度	
			REAL64	例如 mm	±∞	新速度等级的位置	
			REAL64	例如 mm/s	≥ 0.0	新结束速度等级	
			UINT32	0/1	0/1	标准加速度?	
			UINT32			预留（TwinCAT 3）	
			REAL64	例如 mm/s^2	≥ 0.0	加速度	
			UINT32	0/1	0/1	标准减速度?	
			UINT32			预留（TwinCAT 3）	
			REAL64	例如 mm/s^2	≥ 0.0	减速度	
			UINT32	0/1	0/1	标准加加速度?	
			UINT32			预留（TwinCAT 3）	
			REAL64	例如 mm/s^3	≥ 0.0	加加速度	
			}				
0x00000023	写入	伺服	{			启动外部设定点规范（通过循环轴接口 PLCtoNC 设置）	从 TwinCAT 3 起修改
			UINT32	枚举	1：绝对 2：相对	启动类型 [▶ 144]	
			UINT32			预留（TwinCAT 3）	
			REAL64	例如 mm	±∞	新结束位置（目标位置）（可选）！	
			REAL64			预留（TwinCAT 3）	
			}				
0x00000024	写入	伺服	VOID			停止/禁用外部设定点规范（循环轴接口 PLCtoNC）	

索引偏移（十六进制）	访问	轴类型	数据类型	物理单位	定义范围	描述	说明
0x00000025	写入	伺服	{			开始反向定位操作（伺服）：	从 TwinCAT 3 起修改
			UINT32	枚举	1	启动类型 [▶ 144]（默认：1）	
			UINT32			预留（TwinCAT 3）	
			REAL64	例如 mm	±∞	结束位置 1（目标位置）	
			REAL64	例如 mm	±∞	结束位置 2（目标位置）	
			REAL64	0/1	0/1	所需速度	
			REAL64	s	⩾ 0.0	空闲时间	
			}				
0x00000026	写入	每个	{			启动驱动输出	从 TwinCAT 3 起修改
			UINT32	枚举	参见附录	输出类型 [▶ 153]（参见附录）	
			UINT32			预留（TwinCAT 3）	
			REAL64	例如 %	±∞	所需输出值（例如 %）	
			}				
0x00000027	写入	每个	VOID			停止驱动输出	
0x00000028	写入	每个	{			更改驱动输出：	
			UINT32	枚举	参见附录	输出类型 [▶ 153]（参见附录）	
			REAL64	例如 %	±∞	所需输出值（例如 %）	
			}				
0x00000029	写入	每个	VOID			立刻采用当前速率比并保持不变，直至下一次速率比更改！	保留功能，无标准！

0x0000002A	写入	每个	{
			32 字节
			}
						计算和设置编码器偏移	保留功能，无标准！

0x0000002B	读写	每个	数据写入: s. UAS' 数据读取: s. 'UAS'			停止外部设定点发生器并保持连续不断的运动（'UAS'：通用轴启动）	保留功能，无标准！
0x0000002C	写入	每个	UINT32		⩾ 0	设置"归零状态"（供内部使用）	从 TwinCAT 3 起的新功能

索引偏移（十六进制）	访问	轴类型	数据类型	物理单位	定义范围	描述	说明
0x0000002D	读写	伺服	写入			将 NC 控制轴切换到"循环同步扭矩模式"(CST)，并为其设置扭矩设定点。	**谨防使用过程中的危险！**（* 参见表结尾）
			{				
			UINT32			扭矩轴启动类型： 0x3001：绝对 0x3002：相对	
			UINT32	1（位阵列）		内部控制掩码（位阵列）： 00000000_00000001（位 0）：使用手动扭矩进行初始化。 10000000_000000000（位 31）：在“ContinuousUpdate”刷新当前命令参数（fTorqueRamp、fVelocityLimitHigh、fVelocityLimitLow），不增加命令编号。	
			UINT32	0/1	0/1	模式： 0：默认（离散） 1：持续数据更新模式	
			UINT32	枚举	参见附录	缓冲模式 [▶ 144]只能进行中止操作	
			REAL64	Nm 或 %	[0.0 … 1.0E10]	扭矩目标值（符号值）	
			REAL64	Nm/s 或 %/s	[0.0 … 1.0E10]	扭矩变化速度	
			REAL64	例如 mm/s	[0.0 ... 1.0E10] 'VelocityLimitHigh' 必须大于或等于 'VelocityLimitLow'（两个值都可以是负数）。	高速度限制	
			REAL64	例如 mm/s	[0.0 … 1.0E10]	低速度限制	
			REAL64	例如 mm/s^2	[0.0 … 1.0E10]	加速度	
			REAL64	例如 mm/s^2	[0.0 … 1.0E10]	减速度	
			REAL64	Nm 或 %	[0.0 … 1.0E10]	可选：手动输入扭矩启动值（同步值）	
			}				
			读取				
			{				
			UINT16	1	>=0	命令编号（工作编号）	
			UINT16	1	>=0	命令状态	
			}				
0x0000002E						预留	
0x0000002F						预留	

索引偏移（十六进制）	访问	轴类型	数据类型	物理单位	定义范围	描述	说明
0x00000030	写入	伺服	{			启动区间补偿（伺服）	仅影响旧 TwinCAT 2 系统
			UINT32	枚举	参见附录	补偿类型 [▶ 145]（参见附录）	
			UINT32			预留（TwinCAT 3）	
			REAL64	例如 mm/s^2	≥ 0.0	最大加速度增加	
			REAL64	例如 mm/s^2	≥ 0.0	最大减速度增加	
			REAL64	例如 mm/s	> 0.0	最大增速	
			REAL64	例如 mm/s	> 0.0	过程的基本速度	
			REAL64	例如 mm	±∞	需要补偿的路径差	
			REAL64	例如 mm	> 0.0	补偿的路径距离	
			}				
0x00000030	读写	伺服将实际执行参数作为返回值返回	{ **读取+写入：**			启动区间补偿（伺服）注：仅包含在 'TcMc2.lib' 或 'Tc2_MC2.library' 中	从以下版本起更改 TwinCAT 2 211R3 TwinCAT 3
			UINT32	枚举	参见附录	补偿类型 [▶ 145]（参见附录）	
			UINT32			预留（TwinCAT 3）	
			REAL64	例如 mm/s^2	≥ 0.0	=> 最大加速度增量 <= 返回实际执行的加速度增量（在 'TcMc2.lib' 或 'Tc2_MC2.library' 中新增加）	
			REAL64	例如 mm/s^2	≥ 0.0	=> 最大减速度增量 <= 返回实际执行的减速度增量（在 'TcMc2.lib' 或 'Tc2_MC2.library' 中新增加）	
			REAL64	例如 mm/s	> 0.0	=> 要求的最大增速 <= 返回实际执行的增速	
			REAL64	例如 mm/s	> 0.0	过程的基本速度	
			REAL64	例如 mm	±∞	=> 需要补偿的路径差 <= 返回实际执行的路径差	
			REAL64	例如 mm	> 0.0	=> 需要补偿的最大距离 <= 返回实际执行的距离	
			UINT32	1	≥ 0	<= 返回警告 ID（例如 0x4243）	
			UINT32			预留（TwinCAT 3）	
			}				
0x00000031	写入	伺服	VOID			停止区间补偿（伺服）	

索引偏移（十六进制）	访问	轴类型	数据类型	物理单位	定义范围	描述	说明
0x00000032	写入	伺服	{			通过速度跳变（伺服）启动反转操作： （可用于确定速度阶跃响应）	从 TwinCAT 3 起修改
			UINT32	枚举	1	启动类型 [▶ 144]（默认：1）	
			UINT32			预留（TwinCAT 3）	
			REAL64	例如 mm/s	±∞	所需速度 1 （允许为负值）	
			REAL64	例如 mm/s	±∞	所需速度 2 （允许为负值）	
			REAL64	s	> 0.0	速度 1 和 2 的行进时间	
			REAL64	s	≥ 0.0	空闲时间	
			UINT32	1	0, 1,2,3...	可选：重复次数， 默认"0"：无时间限制	
			UINT32			预留（TwinCAT 3）	
			}				
0x00000033	写入	伺服	{			正弦振荡序列 - 用作单一正弦波振荡（正弦波发生器） - 用作正弦波振荡序列（例如，用于波特图）	从 TwinCAT 3 起修改
			UINT32	枚举	1	启动类型 [▶ 144]（固定为启动类型 1）	
			UINT32			预留（TwinCAT 3）	
			REAL64	例如 mm/s	> 0.0	基本振幅（例如 2.5 mm/s）	
			REAL64	Hz	[0.0 .... 10.0]	基频（例如 1.953125 Hz）	
			REAL64	例如 mm/s	≥ 0.0	开始时的启动振幅（例如 0.0 mm/s）	
			REAL64	例如 mm/REV	> 0.0	进给恒定电机（每电机转一圈）（例如 10.0 mm/REV）	
			REAL64	Hz	≥ 1.0	频率范围：启动频率（例如 20.0 Hz）	
			REAL64	Hz	≤ 1/(2*dT)	频率范围：停止频率（例如 500.0 Hz）	
			REAL64	s	> 0.0	步进时长（例如 2,048s）	
			UINT32	1	[1 ... 200]	测量次数（步进循环）（例如 20）	
			UINT32	1		尚未使用的并行测量数（如 1）！	
			}				

索引偏移（十六进制）	访问	轴类型	数据类型	物理单位	定义范围	描述	说明
0x00000034	写入	伺服	{			定相 - 开始定相 - 停止定相	
			UINT32	枚举	1	定相类型： 1: 绝对 2: 相对 4096：停止	
			UINT32	1	⩾ 0	控制掩码 位 0：持续更新	
			UINT32	1	⩾ 0	主轴 ID （用于多主轴）	
			UINT32			保留	
			REAL64	例如 mm	±∞	相移	
			REAL64	例如 mm/s	> 0.0	速度	
			REAL64	例如 mm/s^2	⩾ 0.0	加速度	
			REAL64	例如 mm/s^2	⩾ 0.0	减速度	
			REAL64	例如 mm/s^3	⩾ 0.0	加加速度	
			REAL64[4]			保留	
			UINT32			保留	
			UINT32	1	枚举	缓冲模式 （未执行）	
			REAL64	例如 mm	±∞	混合位置 （未执行）	
			}				
0x00n0003B	写入	PTP 轴 编码器轴	VOID			触发 NC 编码器重新初始化，确保 IO 值有效 n = 0：轴的标准编码器 n > 0：轴的第 n 个编码器（可选）	**谨防使用过程中的危险！** 一定不能启用控制器（位置跳转） 轴状态索引偏移 0x00n10018 可以用于读出 NC 编码器重新初始化是否已完成。 端口 501

索引偏移（十六进制）	访问	轴类型	数据类型	物理单位	定义范围	描述	说明
0x00000040 (0x00n00040)	写入	主轴/从轴耦合（伺服）	{			主轴/从轴耦合（伺服）：	“飞锯”扩展！ 角度 >0.0 和 £90.0 度（平行锯：90.0 度）
			UINT32	枚举	参见附录	从轴类型 [▶ 145]/耦合类型（参见附录）	
			UINT32	1	[1...255]	主轴/组的轴 ID	
			UINT32	1	[0...8]	主轴的子索引 n（默认：值：0）	
			UINT32	1	[0...8]	从轴的子索引 n（默认：0）	
			REAL64	1	[±1000000.0]	参数 1：线性：齿轮比 FlySawVelo：保留 FlySaw：主轴绝对同步位置 [mm]	
			REAL64	1	[±1000000.0]	参数 2：线性：保留 FlySawVelo：保留 FlySawPos：从轴绝对同步位置 [mm]	
			REAL64	1	[±1000000.0]	参数 3：线性：保留 FlySawVelo：倾斜角（以[度]为单位） FlySawPos：倾斜角（以[度]为单位）	
			REAL64	1	[±1000000.0]	参数 4：线性：保留 FlySawVelo：齿轮比 FlySawPos：齿轮比	
			}				

索引偏移（十六进制）	访问	轴类型	数据类型	物理单位	定义范围	描述	说明
0x00000040 (0x00n00040)	写入	主轴/从轴耦合（伺服）	{			主轴/从轴耦合（伺服）：	多主轴耦合 (MC_GearInMultiMaster) 版本 V1 和 V2 从 TwinCAT 3 起修改
			UINT32	枚举	参见附录	从轴类型 [▶ 145]/耦合类型（参见附录）	
			UINT32	1	[1...255]	主轴/组的轴 ID	
			UINT32	1	[1...8]	主轴的子索引 n（默认：值：0）	
			UINT32	1	[1...8]	从轴的子索引 n（默认：0）	
			UINT32	1	[0...255]	轴 ID 主轴 2	
			UINT32	1	[0...255]	轴 ID 主轴 3	
			UINT32	1	[0...255]	轴 ID 主轴 4	
			UINT32	1	[0...255]	保留（轴 ID 主轴 5）	
			UINT32	1	[0...255]	保留（轴 ID 主轴 6）	
			UINT32	1	[0...255]	保留（轴 ID 主轴 7）	
			UINT32	1	[0...255]	保留（轴 ID 主轴 8）	
			UINT32			预留（TwinCAT 3）	
			REAL64	例如 mm/s^2		从轴的最大加速度/减速度	
			UINT32	1	⩾ 0	控制掩码，之前未使用（配置文件的检查和运行模式）	
			UINT32			预留（TwinCAT 3）	
			扩展 V2（可选）：				
			REAL64	例如 mm/s^2	⩾ 0.0	从轴的最大减速度	
			REAL64	例如 mm/s^3	⩾ 0.0	从轴的最大加加速度	
			REAL64	例如 mm/s	⩾ 0.0	从轴的最大速度	
			REAL64			保留	
			REAL64			保留	
			} 64 或 104 字节				
0x00000041	写入	主轴/从轴解耦（伺服）	VOID			主轴/从轴解耦（伺服）	
0x00000041	写入	主轴/从轴解耦，具有可配置跟随功能（伺服）	{			主轴/从轴解耦，具有可配置跟随功能（例如，新结束位置、新速度、停止、E-stop）（伺服）	尚未发布! 从 TC3 起修改
			UINT32	枚举	参见附录	解耦类型 [▶ 146]（参见附录）	
			UINT32			预留（TwinCAT 3）	
			REAL64	例如 mm	±∞	可选：新结束位置	
			REAL64	例如 mm/s	> 0.0	可选：新要求速度	
			REAL64	例如 mm/s^2	⩾ 0.0（0：默认）	可选：新结束位置、新速度和紧急停止 (E-stop) 的加速度	
			REAL64	例如 mm/s^2	⩾ 0.0（0：默认）	可选：新结束位置、新速度和紧急停止 (E-stop) 的减速度	
			REAL64	例如 mm/s^3	⩾ 0.0（0：默认）	可选：新结束位置、新速度和紧急停止 (E-stop) 的加加速度	
			}				

索引偏移（十六进制）	访问	轴类型	数据类型	物理单位	定义范围	描述	说明
0x00000042	写入	主轴/从轴耦合 类型：线性 (&SPECIAL)	{			更改耦合参数（耦合）：	
			REAL64	1	[±1000000.0]	参数 1：线性：齿轮比	
			REAL64	1	[±1000000.0]	参数 2：线性：保留	
			REAL64	1	[±1000000.0]	参数 3：线性：保留	
			REAL64	1	[±1000000.0]	参数 4：线性：保留	
			}				
0x00000043	写入	主轴/从轴表耦合 类型：表格	{			更改表耦合参数（伺服）：	
			REAL64	mm	±∞	从轴位置偏移	
			REAL64	mm	±∞	主轴位置偏移	
			}				
0x00000043	写入	主轴/从轴表耦合 类型：表格 和 "运动功能"	{			更改表耦合参数（伺服）：	也适用于"运动功能"
			REAL64	mm	±∞	从轴位置偏移	
			REAL64	mm	±∞	主轴位置偏移	
			REAL64	1	±∞ (<> 0.0)	从轴位置缩放	
			REAL64	1	±∞ (<> 0.0)	主轴位置缩放	
			}				
0x00000043	写入	主轴/从轴表耦合 类型：表格	{			更改表耦合参数（伺服）：	
			REAL64	mm	±∞	从轴位置偏移	
			REAL64	mm	±∞	主轴位置偏移	
			REAL64	1	±∞ (<> 0.0)	从轴位置缩放	
			REAL64	1	±∞ (<> 0.0)	主轴位置缩放	
			REAL64	例如 mm	±∞	绝对主轴激活位置	
			}				
0x00000044	写入	从轴停止（伺服）	VOID			停止"飞锯"（伺服）	仅适用于"飞锯"

索引偏移（十六进制）	访问	轴类型	数据类型	物理单位	定义范围	描述	说明
0x00000045 (0x00n00045)	写入	主轴/从轴表耦合（伺服）	{			主轴/从轴表耦合（伺服）：	
			UINT32	枚举	参见附录	从轴类型/耦合类型 [▶ 145]（参见附录）	
			UINT32	1	[1...255]	主轴的轴 ID	
			UINT32	1	[0...8]	主轴的子索引 n（默认：值：0）	
			UINT32	1	[0...8]	从轴的子索引 n（默认：0）	
						独表区间	
			REAL64	mm	±∞	从轴位置偏移（类型：TABULAR）	
			REAL64	mm	±∞	主轴位置偏移（类型：TABULAR）	
			UINT32	1	[0,1]	绝对从轴位置（类型：TABULAR）	
			UINT32	1	[0,1]	绝对主轴位置（类型：TABULAR）	
			UINT32	1	[1...255]	耦合表的表 ID（类型：TABULAR）	
						多表区间	
			UINT16	1	[0...8]	表数量（类型：MULTITAB） 注：该插补类型被误用在独表中	
			UNIT16	1	[0...8]	配置文件表的数量（类型：MULTITAB）	
			UNIT32[8]	1	[1...255]	耦合表的表 ID（类型：MULTITAB）	
			}				
0x00000046	写入	主轴/从轴多表	UINT32	1	[1...255]	校正表激活，校正表 ID	
0x00000046	写入	主轴/从轴多表	{			激活校正表	从 TwinCAT 3 起修改
			UINT32	1	[1...255]	校正表 ID	
			UINT32			预留（TwinCAT 3）	
			REAL64	例如 mm	±∞	绝对主轴激活位置	
			}				
0x00000047	写入	主轴/从轴多表	UINT32	1	[1..255]	循环结束时停用配置文件表，当前单循环配置文件表的表 ID	
0x00000048	读写	主轴/从轴多表	写入：UINT32	1	[1..255]	读取末次校正偏移量：校准表ID	
			读取：REAL32	例如 mm	±∞	根据对应的表格ID退出校正表，来确定偏移量	
0x00000049	写入	主轴/从轴表耦合 类型：表格	REAL64	1	±∞	更改从轴表列的从轴表缩放因子（scaling factor）（默认：1.0）	

索引偏移（十六进制）	访问	轴类型	数据类型	物理单位	定义范围	描述	说明
0x0000004A(0x00n0004A)	写入	主轴/从轴通用表耦合（伺服）	{			主轴/从轴独表耦合（伺服）：	从 TwinCAT 3 起修改
			UINT32	枚举	参见附录	从轴类型/耦合类型 [▶ 145]（参见附录）	
			UINT32	1	[1...255]	主轴的轴 ID	
			UINT32	1	[0...8]	主轴的子索引 n（默认：值：0）	
			UINT32	1	[0...8]	从轴的子索引 n（默认：0）	
			UINT32	1	1...255]	耦合表的表 ID（类型：TABULAR）	
			UINT32	1		表插值类型	
			REAL64	mm	±∞	从轴位置偏移（类型：TABULAR）	
			REAL64	mm	±∞	主轴位置偏移（类型：TABULAR）	
			REAL64	mm	±∞	从轴位置缩放（类型：TABULAR）	
			REAL64	mm	±∞	主轴位置缩放（类型：TABULAR）	
			UINT32	1	[0,1]	从轴绝对位置?（类型：TABULAR）	
			UINT32	1	[0,1]	主轴绝对位置?（类型：TABULAR）	
			UINT32	枚举	参见附录	更改的激活类型： 0: '瞬时的（instantaneous）'（默认） 1: '在主凸轮位置' 2: '在主轴位置' 3: '下一个周期'	
			UINT32			预留（TwinCAT 3）	
			REAL64	mm	±∞	激活位置	
			UINT32	枚举	参见附录	主轴缩放类型： 0：用户定义（默认） 1：使用自动偏移缩放 2：关闭	
			UINT32	枚举	参见附录	从轴缩放类型： 0：用户定义（默认） 1：使用自动偏移缩放 2：关闭	
			}				

索引偏移（十六进制）	访问	轴类型	数据类型	物理单位	定义范围	描述	说明
0x0000004A(0x00n0004A)	写入	主轴/从轴通用表耦合（伺服）	{			主轴/从轴独表耦合（伺服）：	从 TwinCAT 3 起修改
			UINT32	枚举	参见附录	从轴类型/耦合类型 [▶ 145]（参见附录）	
			UINT32	1	[1...255]	主轴的轴 ID	
			UINT32	1	[0...8]	主轴的子索引 n（默认：值：0）	
			UINT32	1	[0...8]	从轴的子索引 n（默认：0）	
			UINT32	1	1...255]	耦合表的表 ID（类型：TABULAR）	
			UINT32	1		表插值类型	
			REAL64	mm	±∞	从轴位置偏移（类型：TABULAR）	
			REAL64	mm	±∞	主轴位置偏移（类型：TABULAR）	
			REAL64	mm	±∞	从轴位置缩放（类型：TABULAR）	
			REAL64	mm	±∞	主轴位置缩放（类型：TABULAR）	
			UINT32	1	[0,1]	从轴绝对位置？（类型：TABULAR）	
			UINT32	1	[0,1]	主轴绝对位置？（类型：TABULAR）	
			UINT32	枚举	参见附录	更改的激活类型： 0: '瞬时的（instantaneous）'（默认） 1: '在主凸轮位置' 2: '在主轴位置' 3: '下一个周期'	
			UINT32			预留（TwinCAT 3）	
			REAL64	mm	±∞	激活位置	
			UINT32	枚举	参见附录	主轴缩放类型： 0：用户定义（默认） 1：使用自动偏移缩放 2：关闭	
			UINT32	枚举	参见附录	从轴缩放类型： 0：用户定义（默认） 1：使用自动偏移缩放 2：关闭	
			MultiCam 扩展：				
			UINT32	枚举	参见附录	凸轮运行模式	
			UINT32	1	[1⋯255]	参考表 ID	
			BYTE[104]			预留（TwinCAT 3）	
			}				

索引偏移（十六进制）	访问	轴类型	数据类型	物理单位	定义范围	描述	说明
0x0000004B(0x00n0004B)	写入	主轴/从轴通用飞锯（伺服）	{			主轴/从轴同步耦合（伺服）：	从 TwinCAT 3 起修改
			UINT32	枚举	参见附录	从轴类型/耦合类型（参见附录）	
			UINT32	1	[1...255]	主轴的轴 ID	
			UINT32	1	[0...8]	主轴的子索引 n（默认：值：0）	
			UINT32	1	[0...8]	从轴的子索引 n（默认：0）	
			REAL64	1	±∞ (<> 0.0)	齿轮比	
			REAL64	mm	±∞	主轴同步位置	
			REAL64	mm	±∞	从轴同步位置	
			REAL64	mm/s	⩾ 0.0	从轴速度（可选）	
			REAL64	mm/s^2	⩾ 0.0	从轴加速度（可选）	
			REAL64	mm/s^2	⩾ 0.0	从轴减速度（可选）	
			REAL64	mm/s^3	⩾ 0.0	从轴加加速度（可选）	
			UINT32	1	⩾ 0	位掩码（默认值：0）	
			UINT32			预留（TwinCAT 3）	
			}				
0x0000004D(0x00n0004D)	写入	主轴/从轴表耦合 类型： TABULAR 和 MF	{			修改表格里的缩放比例（伺服）：	从 TwinCAT 3 起修改
			UINT32	枚举	参见附录	更改的激活类型 0: '瞬时的（instantaneous）'（默认） 1: '在主凸轮位置' 2: '在主轴位置' 3: '下一个周期'	
			UINT32			预留（TwinCAT 3）	
			REAL64	例如 mm	±∞	激活位置	
			UINT32	枚举	参见附录	主轴缩放类型 0：用户定义（默认） 1：使用自动偏移缩放 2：关闭	
			UINT32	枚举	参见附录	从轴缩放类型 0：用户定义（默认） 1：使用自动偏移缩放 2：关闭	
			REAL64	例如 mm	±∞	主轴位置偏移	
			REAL64	例如 mm	±∞	从轴位置偏移	
			REAL64	1	±∞ (<> 0.0)	主轴位置缩放	
			REAL64	1	±∞	从轴位置缩放	
			MultiCam 可选扩展：				
			UINT32	1	⩾ 0	凸轮表 ID	
			UINT32			预留（TwinCAT 3）	
			}				
0x00000050	写入	每个	VOID			停用整个轴（禁用）	
0x00000051	写入	每个	VOID			激活整个轴（启用）	

索引偏移（十六进制）	访问	轴类型	数据类型	物理单位	定义范围	描述	说明
0x00000052	写入	伺服	{			在有/无外部设定点规范的情况下，更改活动轴控制回路（编码器、控制器和轴接口三合一）：	从 TwinCAT 3 起修改
			UINT32	1	⩾ 0	轴控制回路的编号/索引 （默认值：0）	
			UINT32	枚举	参见附录 (>0)	同步行为的切换类型 [▶ 156] 1: '标准'	
			REAL64	1	±∞	切换同步值（可选）	
			UINT32	0/ 1	0/1	通过轴接口的方式指定外部设定点? 注：目前尚未使用!	
			UINT32			预留（TwinCAT 3）	
			}				
0x00000060	写入	每个	VOID			关闭驱动输出（禁用）	
0x00000061	写入	每个	VOID			激活驱动输出（启用）	
0x00000062	写入	高/低	UINT16	1	0/1	释放保持制动器? 0：自动激活（默认） 1：必须始终释放 注： 复位轴时重置为'0'!	
0x00000063	写入	仅适用于 SERCOS/SoE 和 CANopen/ CoE	{			激活"驱动运行模式"（如位置速度、扭矩等）	从 TC 3.1 B4022（NC 4443）起更新 始终连接至 SEC 501 端口!
			INT32	枚举 [▶ 153]（参见附录）	[0; 1, 2, 3, …] 特殊情况： ⩾ 100：SoE < 0：CoE	新"驱动运行模式"（通用模式）	
			UINT32	1	0	保留	
			UINT32	1	0	保留	
			UINT32	1	0	保留	
			}				
0x00000070	写入	每个	VOID			将轴从例如 3D 组等返回其自身的 PTP 组	

* 以下警告与索引偏移 0x0000002D 有关：

⚠ 危险

有由于轴意外运动而造成生命危险或严重伤害或财产损失的风险

使用功能块时，轴将切换到 CST 模式。使用功能块后（尤其是出错情况下），轴可能仍处于 CST 模式。当释放轴时，这可能会导致突然和意外运动（尤其是升降轴）。

- 确保不存在风险评估中所界定的危险。
- 通过功能块 MC_ReadDriveOperationMode 检查当前运行模式。
- 如果轴未处于与位置相关的运行模式 (CSV/CSP)，则应在启用之前转换：
  – 直接使用MC_WriteDriveOperationMode转换为所需的位置相关运行模式 (CSV/CSP)，或者
  — 直接使用MC_Halt / MC_Stop 转换为所需的位置相关运行模式 (CSV/CSP)（TwinCAT 3.1.4024.40 及以上）
  将轴间接切换到位置相关运行模式的其他功能块只能在有限范围内做到这一点，因此不能用于刻意改变运行模式。

⇨ 随后，有必要再次检查轴是否真的处于位置相关运行模式 (CSV/CSP)，如果不是，则需要中止并进行错误处理。

3.2.1.5.4.4.4 *循环轴过程数据的"索引偏移"规范（索引组 0x4300 + ID）*

索引偏移（十六进制）	访问	轴类型	数据类型	物理单位	定义范围	描述	说明
0x00n00000	读/写	每个 (PLC→NC)	{128 字节}		STRUCT 参见轴接口	轴结构 (PLC→NC) n = 0：标准轴接口 n > 0: 第 n 个轴接口（可选）	写入命令仅为可选项! 考虑安全方面! PLCTONC_AXIS_REF
0x00n00001	读/写	每个 (PLC→NC)	UINT32	1	>0	控制双字	写入命令仅为可选项! 可进行符号访问! "ControlDWord"
0x00n00002	读/写	每个 (PLC→NC)	UINT16	1	0/1	控制器启用	无法用示波器追踪!
0x00n00003	读/写	每个 (PLC→NC)	UINT16	1	0/1	进给启用加	无法用示波器追踪!
0x00n00004	读/写	每个 (PLC→NC)	UINT16	1	0/1	进给启用减	无法用示波器追踪!
0x00n00007	读/写	每个 (PLC→NC)	UINT16	1	0/1	参考凸轮	无法用示波器追踪!
0x00n00021	读/写	每个 (PLC→NC)	UINT32	%	0...1000000	速度超驰 (1000000 == 100%)	写入命令仅为可选项! 可进行符号访问! "OverrideV"
0x00n00022	读/写	每个 (PLC→NC)	UINT32	1	枚举	运行模式轴	写入命令仅为可选项!
0x00n00025	读/写	每个 (PLC→NC)	REAL64	例如 mm		实际位置校正值（测量系统错误校正）	写入命令仅为可选项!
0x00n00026	读/写	每个 (PLC→NC)	REAL64	例如 mm/s		外部控制器组件（位置控制器组件）	写入命令仅为可选项!
0x00n00027	读/写	每个 (PLC→NC)	{			外部设定点生成	写入命令仅为可选项! 从 TC3 起修改
			REAL64	例如 mm	±∞	外部设置位置	
			REAL64	例如 mm/s	±∞	外部设置速度	
			REAL64	例如 mm/s^2	±∞	外部设置加速度	
			INT32	1	+1, 0, -1	外部设置行进方向	
			UINT32			保留 (TC3)	
			REAL64			保留 (TC3)	
			}				
0x00n00080	读取	每个 (PLC→NC)	{256 字节}		STRUCT 参见轴接口	轴结构 (NC→PLC) 注：尺寸和对齐方式有所改变 n = 0：标准轴接口 n > 0: 第 n 个轴接口（可选）	从 TC3.NCTOPLC_AXIS_REF 起更改
0x00n00071	读取	每个 (PLC→NC)	UINT8	1	>0	状态双字：字节 1	
0x00n00072	读取	每个 (PLC→NC)	UINT8	1	>0	状态双字：字节 2	
0x00n00073	读取	每个 (PLC→NC)	UINT8	1	>0	状态双字：字节 3	
0x00n00074	读取	每个 (PLC→NC)	UINT8	1	>0	状态双字：字节 4	
0x00n00081	读取	每个 (PLC→NC)	UINT32	1	>0	状态双字（完整）	可进行符号访问! "StateDWord"

索引偏移（十六进制）	访问	轴类型	数据类型	物理单位	定义范围	描述	说明
0x00n00082	读取	每个 (PLC→NC)	UINT16	1	0/1	轴已准备好运行	无法用示波器追踪！
0x00n00083	读取	每个 (PLC→NC)	UINT16	1	0/1	轴已被引用	无法用示波器追踪！
0x00n00084	读取	每个 (PLC→NC)	UINT16	1	0/1	轴处于保护运行模式（例如从轴）	无法用示波器追踪！
0x00n00085	读取	每个 (PLC→NC)	UINT16	1	0/1	轴处于快速模式	无法用示波器追踪！
0x00n00088	读取	每个 (PLC→NC)	UINT16	1	0/1	轴 I/O 数据无效	无法用示波器追踪！
0x00n00089	读取	每个 (PLC→NC)	UINT16	1	0/1	轴处于错误状态	无法用示波器追踪！
0x00n0008A	读取	每个 (PLC→NC)	UINT16	1	0/1	轴向较大值移动	无法用示波器追踪！
0x00n0008B	读取	每个 (PLC→NC)	UINT16	1	0/1	轴向较小值移动	无法用示波器追踪！
0x00n0008C	读取	每个 (PLC→NC)	UINT16	1	0/1	轴处于逻辑静止状态（仅考虑设定点） （位置控制器？）	无法用示波器追踪！
0x00n0008D	读取	每个 (PLC→NC)	UINT16	1	0/1	轴被引用	无法用示波器追踪！
0x00n0008E	读取	每个 (PLC→NC)	UINT16	1	0/1	轴处于位置窗口中	无法用示波器追踪！
0x00n0008F	读取	每个 (PLC→NC)	UINT16	1	0/1	轴处于目标位置（达到目标位置）	无法用示波器追踪！
0x00n00090	读取	每个 (PLC→NC)	UINT16	1	0/1	轴具有恒定速度或旋转速度	无法用示波器追踪！
0x00n0009A	读取	每个 (PLC→NC)	UINT16	1	0/1	运行模式未执行（忙）	无法用示波器追踪！
0x00n0009B	读取	每个 (PLC→NC)	UINT16	1	0/1	轴有指令，正在执行指令	无法用示波器追踪！
0x00n000B1	读取	每个 (PLC→NC)	UINT32	1	≥0	轴错误代码	
0x00n000B2	读取	每个 (PLC→NC)	UINT32	1	枚举	轴的运动状态（主轴状态 [▶ 153]/从轴状态 [▶ 153]）	可进行符号访问！ "AxisState"
0x00n000B3	读取	每个 (PLC→NC)	UINT32	1	枚举	轴的运行模式（修订版 NC）	
0x00n000B4	读取	每个 (PLC→NC)	UINT32	1	枚举	轴参考状态	可进行符号访问！ "HomingState"
0x00n000B5	读取	每个 (PLC→NC)	UINT32	1	枚举	轴耦合状态	可进行符号访问！ "CoupleState"
0x00n000B6	读取	每个 (PLC→NC)	UINT32	1	≥0	轴的 SVB 条目/任务（PRE 表）	
0x00n000B7	读取	每个 (PLC→NC)	UINT32	1	≥0	轴的 SAF 条目/任务（EXE 表）	
0x00n000B8	读取	每个 (PLC→NC)	UINT32	1	≥0	轴 ID	

索引偏移（十六进制）	访问	轴类型	数据类型	物理单位	定义范围	描述	说明
0x00n000B9	读取	每个 (PLC→NC)	UINT32	1	≥0	运行模式状态双字：位 0：位置范围监测激活? 位 1：目标位置窗口监测激活? 位 2：循环距离激活? 位 3：物理运动监测激活? 位 4：PEH 时间监测激活? 位 5：间隙补偿激活? 位 6：延迟错误反应模式激活? 位 7：模除运行模式激活（模除轴）? 位 16：跟随错误监测位置激活? 位 17：跟随错误监测速度激活? 位 18：最小结束位置监测激活? 位 19：最大结束位置监测激活? 位 20：实际位置校正激活?	
0x00n000BA	读取	每个 (PLC→NC)	REAL64	例如 mm		实际位置（计算出的绝对值）	
0x00n000BB	读取	每个 (PLC→NC)	REAL64	例如 mm		模除实际位置	
0x00n000BC	读取	每个 (PLC→NC)	INT32	1		模除旋转	
0x00n000BD	读取	每个 (PLC→NC)	REAL64	例如 mm/s		实际速度（可选）	
0x00n000BE	读取	每个 (PLC→NC)	REAL64	例如 mm		跟随误差位置	
0x00n000BF	读取	每个 (PLC→NC)	REAL64	例如 mm		设置位置	
0x00n000C0	读取	每个 (PLC→NC)	REAL64	例如 mm/s		设置速度	
0x00n000C1	读取	每个 (PLC→NC)	REAL64	例如 mm/s^2		设置加速度	
0x00n10000	读/写	编码器：每个（NC→IO）	{40 字节 }		STRUCT 参见编码器 IO 接口	编码器输出结构（NC→IO，40 字节）NCENCODERSTRUCT_OUT2	写入命令仅为可选项！考虑安全方面！
0x00n10080	读取	编码器：每个 (IO→NC)	{40 字节 }		STRUCT 参见编码器 IO 接口	编码器输入结构（IO→NC，40 字节）NCENCODERSTRUCT_IN2	
0x00n30000	读/写	驱动器：每个（NC→IO）	{40 字节 }		STRUCT 参见驱动器 IO 接口	驱动器-输出-结构（NC→IO，40 字节）NCDRIVESTRUCT_OUT2	写入命令仅为可选项！考虑安全方面！
0x00n30080	读取	驱动器：每 (IO→NC)	{40 字节 }		STRUCT 参见驱动器 IO 接口	驱动器-输入-结构（NC→IO，40 字节）NCDRIVESTRUCT_IN2	

3.2.1.5.4.5 编码器规范

3.2.1.5.4.5.1 *编码器参数的"索引偏移"规范（索引组 0x5000 + ID）*

索引偏移（十六进制）	访问	组类型	数据类型	物理单位	定义范围	描述	说明
0x00000001	读取	每个	UINT32	1	[1 ... 255]	编码器 ID	
0x00000002	读取	每个	UINT8[30+1]	1	30 个字符	编码器名称	
0x00000003	读取	每个	UINT32	1	参见 枚举 (>0)	编码器类型 [▶ 147]	
0x00000004	读/写	每个	UINT32	1	Byteoffset	输入地址偏移（IO-输入-图像）	更改 I/O 地址
0x00000005	读/写	每个	UINT32	1	Byteoffset	输出地址偏移（IO-输出-图像）	更改 I/O 地址
0x00000006	读/写	每个	REAL64	例如 mm/INC	[1.0E-12 ... 1.0E+30]	结果缩放因子（scaling factor）（分子/分母） 注：从 TC3 起，缩放因子（scaling factor）由分子和分母两部分组成（默认：1.0）。	如果已发出启用控制器的信号，则不允许写入。
0x00000007	读/写	每个	REAL64	例如 mm	[±1.0E+9]	位置偏移	如果已发出启用控制器的信号，则不允许写入。
0x00000008	读/写	每个	UINT16	1	[0,1]	编码器计数方向	如果已发出启用控制器的信号，则不允许写入。
0x00000009	读/写	每个	REAL64	例如 mm	[0.001 ... 1.0E+9]	模除因子	
0x0000000A	读/写	每个	UINT32	1	参见 枚举 (>0) 在附录中	编码器模式 [▶ 149]	
0x0000000B	读/写	每个	UINT16	1	0/1	软结束最小监测?	
0x0000000C	读/写	每个	UINT16	1	0/1	软结束最大监测?	
0x0000000D	读/写	每个	REAL64	mm		最小软结束位置	
0x0000000E	读/写	每个	REAL64	mm		最大软结束位置	
0x0000000F	读/写	每个	UINT32	1	参见 枚举 (≥0) 在附录中	编码器评估方向 [▶ 149]（启用日志记录计数方向）	
0x00000010	读/写	每个	REAL64	s	[0.0...60.0]	实际位置值的滤波时间（以秒为单位）(P-T1)	
0x00000011	读/写	每个	REAL64	s	[0.0...60.0]	实际速度值的滤波时间（以秒为单位）(P-T1)	
0x00000012	读/写	每个	REAL64	s	[0.0...60.0]	实际加速度值的滤波时间（以秒为单位）(P-T1)	
0x00000013	读/写	每个	UINT8[10+1]	1		物理单位	未执行!
0x00000014	读/写	每个	UINT32	1		单位（位置、速度、时间）的解释 第 0 位：速度单位为 x/min，而非 x/s 第 1 位：以千分之一为基本单位的位置	未执行! 位数组
0x00000015	读/写	每个	UINT32	INC	[0x0... 0xFFFFFFFF]	编码器掩码（以增量表示的编码器实际值的最大值） 注：编码器掩码可以是任何数值（如 3600000）。与过去不同，它不再需要对应于连续的一系列二进制 $1(^2n-1)$。	必须禁用轴才能进行写入访问。 另请参见"编码器子掩码"参数
0x00000016	读/写	每个	UINT16	1	0/1	实际位置校正（测量系统误差校正）?	
0x00000017	读/写	每个	REAL64	s	[0.0...60.0]	实际位置校正的滤波时间（以秒为单位）(P-T1)	
0x00000018	读/写	每个	UINT32	1	[0x0... 0xFFFFFFFF]	原始增量值的滤波掩码（0x0：全通）	

索引偏移（十六进制）	访问	组类型	数据类型	物理单位	定义范围	描述	说明
0x00000019	读/写	每个	UINT32	1	参见 枚举 (≥0) 在附录中	编码器绝对尺寸系统 [▶ 149]	如果已发出启用控制器的信号，则不允许写入。
0x0000001A	读/写	每个	UINT32	1	参见 枚举 (≥0)	编码器位置初始化	未执行!
0x0000001B	读/写	每个	REAL64	例如 mm	[≥0，模除因子/2]	模除启动的容差窗口	
0x0000001C	读取	每个	UINT32	1	参见 枚举 (≥0)	编码器符号解释 [▶ 149]（数据类型）	
0x0000001D	读取	每个	UINT16	1	0/1	增量或绝对编码器? 0：增量编码器类型 1：绝对编码器类型	
0x00000020	读/写	每个	UINT32	1	参见 枚举 (≥0)	编码器死区时间补偿模式 0：关闭（默认） 1：开（带速度） 2：开（带速度和加速度）	
0x00000021	读/写	每个	UINT32	1		用于编码器死区时间补偿的控制双字（32 位）： 位 0 = 0：相对 I/O 时间（默认） 位 0 = 1：绝对 I/O 时间	
0x00000022	读/写	每个	INT32	ns	[±1.0E+9]	用于编码器死区时间补偿的参数化时间移位之和（通常为正数值）	
0x00000023	读/写	每个	REAL64	例如 mm/INC	[1.0E-12 ... 1.0E+30]	缩放因子(scaling factor)的组成部分：分子 （=> 缩放因子(scaling factor)分子/缩放(scaling factor)因子分母）	从 TC3 起更新 如果已发出启用控制器的信号，则不允许写入。
0x00000024	读/写	每个	REAL64	1	[1.0E-12 ... 1.0E+30]	缩放因子(scaling factor)的组成部分：分母 （=> 缩放因子(scaling factor)分子/缩放(scaling factor)因子分母） 默认：1.0	从 TC3 起更新 如果已发出启用控制器的信号，则不允许写入。
0x00000025	读/写	每个	{ real64 real64 }16 字节	例如 mm/INC 1	[1.0E-12 ... 1.0E+30] [1.0E-12 ... 1.0E+30]	缩放因子(scaling factor)的组成部分：分子 缩放因子(scaling factor)的组成部分：分母 （=> 缩放因子(scaling factor)分子/缩放(scaling factor)因子分母）	从 TC3 起更新
0x00000030	读/写	每个	UINT32	1		内部编码器控制双字，用于指定运行模式和属性	从 TC3 起更新
0x00000101	读/写	INC	UINT16	1	[0,1]	参考凸轮的反向搜索方向?	
0x00000102	读/写	INC		1	[0,1]	同步脉冲的反向搜索方向?	

索引偏移（十六进制）	访问	组类型	数据类型	物理单位	定义范围	描述	说明
0x00000103	读/写	INC	REAL64	例如 mm	[±1.0E+9]	参考位置	
0x00000104	读/写	INC	UINT16	1	[0,1]	是否激活了参考凸轮和同步脉冲之间的距离监测?	未执行!
0x00000105	读/写	INC	UINT32	INC	[0 ...65536]	参考凸轮与同步脉冲之间的最小间隙（以增量为单位）	未执行!
0x00000106	读/写	INC	UINT16	1	[0,1]	外部同步脉冲?	
0x00000107	读/写	INC	UINT32	1	参见 枚举 (>0)	参考模式（同步条件） [▶ 150]	
0x00000108	读/写	INC	UINT32	1	[0x0000000F... 0xFFFFFFFF]二进制掩码：（^{2}n - 1）	编码器子掩码（编码器实际值绝对范围的最大值，以增量为单位） 例如，用作参考模式"软件同步"和 NC 保留数据"绝对（MODULO模除）"、"增量（SINGLETURN ABSOLUTE单转绝对）"的参考标记。 注 1：编码器子掩码必须小于或等于编码器掩码。 注 2：编码器掩码必须是编码器子掩码的整数倍。 注 3：编码器子掩码必须是连续的二进制 1 序列 (^{2}n-1)，例如 0x000FFFFF。	新 另请参见参数"编码器掩码"
0x00000109	读/写	INC	UINT32	1	参见 枚举 (≥0)	回零传感器源 [▶ 150] 设置参考凸轮的数字输入源。	
0x00000110	读/写	INC（编码器模拟）	REAL64	1	[0.0 ... 1000000.0]	模拟编码器噪声部分的缩放/权重	

3.2.1.5.4.5.2 *编码器状态的"索引偏移"规范（索引组 0x5100 + ID）*

索引偏移（十六进制）	访问	组类型	数据类型	物理单位	定义范围	描述	说明
0x00000001	读取	每个	INT32			误差状态编码器	
0x00000002	读取	每个	REAL64			实际位置（带有实际位置补偿值）	可进行符号访问！ActPos
0x00000003	读取	每个	REAL64			模除实际位置	可进行符号访问！ActPosModulo
0x00000004	读取	每个	INT32			模除实际旋转	
0x00000005	读取	每个	REAL64			可选：实际速度	基本单位 / s 可进行符号访问！ActVelo
0x00000006	读取	每个	REAL64			可选：实际加速度	基本单位 / s^2 可进行符号访问！ActAcc
0x00000007	读取	每个	INT32			编码器实际增量	
0x00000008	读取	每个	INT64			软件 — 实际增量计数器	
0x00000009	读/写	每个	UINT16			参考旗标（“校准旗标”）	
0x0000000A	读取	每个	REAL64			实际位置校正值（测量系统误差校正）	
0x0000000B	读取	每个	REAL64			无实际位置补偿值的实际位置	
0x0000000C	读取	每个	REAL64	例如 mm		死区时间补偿导致的实际位置补偿值	
0x0000000D	读取	每个	REAL64	s		编码器死区时间补偿的时间位移总和（参数化和可变死区时间）注：系统中规定的死区时间为正值。	
0x0000000E	读取	每个	REAL64	例如 mm		内部位置偏移作为基准期（模除范围）减值的校正值	
0x00000010	读取	每个	REAL64	例如 mm/s		无实际位置补偿值的实际速度	
0x00000012	读取	每个	REAL64	例如 mm		未滤波实际位置（带有实际位置补偿值）	
0x00000013	读取	每个	REAL64	例如 mm		滤波后的实际位置（带实际位置校正值的偏移，无死区时间补偿）	
0x00000014	读取	类型SoE、CoE、MDP 742	REAL64	例如 mm/s		可选：实际驱动速度（直接从 SoE、CoE 或 MDP 742 驱动器传输）	基本单位 / s 从 TC3.1 B4020.30 起更新
0x00000015	读取	每个	REAL64	例如 mm/s		可选：未滤波实际速度	基本单位 / s
0x00000016	读取	每个	**读取**（16 字节 * N）			读取实际位置缓冲	
			{				
			UINT32	ns	⩾0	带有 32 位的 DcTimeStamp	
			UINT32			保留	
			REAL64	例如 mm	±∞	相关时间戳的实际位置	
			} [N]				
0x00000017	读取		REAL64	例如 mm		读出 MC_SetPosition 偏移	
0x00000101	读取	INC	REAL64	例如 mm		读回硬件锁存器从激活到生效之间的位置差	无法用示波器追踪！

索引偏移（十六进制）	访问	组类型	数据类型	物理单位	定义范围	描述	说明
0x00000200	**读取 写入**	**功能组"TouchProbeV2"： - SERCOS/SoE - EtherCAT/ CoE (CANopen DS402) - SoftDrive (TCom)， - MDP 511 (EL5101, EL5151, EL5021, EL7041, EL7342)**	**写入**（24 字节）			读取"接触式探头"状态（外部锁存器状态）	仅适用于 SAF 端口 501
			{				
			UINT32	1	[1,2,3,4]	探头单元（探头 1、2、3、4）	
			UINT32[5]			保留	
			}				
			读取（64 字节）				
			{				
			UINT32	1	[0/1]	接触式探头上升沿激活?	
			UINT32	1	[0/1]	接触式探头上升沿有效?	
			REAL64	例如 mm		接触式探头上升沿位置值	
			UINT32	1	⩾0	接触式探头上升沿计数器（连续模式）	
			UINT32			保留	
			UINT32	1	[0/1]	接触式探头下降沿激活?	
			UINT32	1	[0/1]	接触式探头下降沿有效?	
			REAL64	例如 mm		接触式探头下降沿位置值	
			UINT32	1	⩾0	接触式探头下降沿计数器（连续模式）	
			UINT32[5]			保留	
			}				
0x00000201	读取	**KL5101、SERCOS、AX2xxx、ProviDrive**	UINT16	1	[0,1]	"外部锁存器功能"激活? 或 "接触式探头功能"激活? （边沿独立? ）	无法用示波器追踪!
0x00000201	读取	**CANopen**	UINT32[4]	1	[0,1]	"外部锁存器功能 1 至 4"激活? 或 "接触式探头功能 1 至 4"激活?	无法用示波器追踪!
0x00000202	读取	**KL5101、SERCOS、AX2xxx、ProviDrive**	UINT16	1	[0,1]	外部锁存器值有效? 或 接触式探头锁定? （边沿独立? ）	另请参见轴接口 NcToPlc（状态双字）
0x00000202	读取	**CANopen**	UINT32[4]	1	[0,1]	外部锁存器值 1 至 4 有效? 或 接触式探头 1 至 4 锁定?	另请参见轴接口 NcToPlc（状态双字）
0x00000203	读取	**KL5101、SERCOS、AX2xxx、ProviDrive**	UINT32	INC		外部/接触式探头硬件增量锁存器值	
0x00000204	读取	**KL5101、SERCOS、AX2xxx、ProviDrive**	UINT64	INC		外部/接触式探头软件增量锁存器值	
0x00000205	读取	**KL5101、SERCOS、AX2xxx、ProviDrive**	REAL64	例如 mm		外部/接触式探头位置锁存器值	基本单位

索引偏移（十六进制）	访问	组类型	数据类型	物理单位	定义范围	描述	说明
0x00000205	读取	**CANopen**	REAL64[4]	例如 mm		外部接触式探头值/位置锁存器值	基本单位
0x00000206	读取	**KL5101、SERCOS、AX2xxx、ProviDrive**	UINT32	INC		硬件增量锁存器值差（NewLatch - LastLatch）	无法用示波器追踪！
0x00000207	读取	**KL5101、SERCOS、AX2xxx、ProviDrive**	UINT64	INC		软件增量锁存器值差（NewLatch - LastLatch）	无法用示波器追踪！
0x00000208	读取	**KL5101、SERCOS、AX2xxx、ProviDrive**	REAL64	例如 mm		位置锁存器值差（NewLatch - LastLatch）	无法用示波器追踪！基本单位
0x00000210	读取	**KL5101、AX2xxx、ProviDrive**	UINT16	1	[0,1]	上升沿"外部锁存器功能"激活? 或 上升沿"接触式探头功能"激活?	无法用示波器追踪！
0x00000210	读取	**CANopen**	UINT16[4]	1	[0,1]	上升沿"外部锁存器功能"激活? 或 上升沿"接触式探头功能"激活?	无法用示波器追踪！
0x00000211	读取	**KL5101、AX2xxx、ProviDrive**	UINT16	1	[0,1]	下降沿"外部锁存器功能"激活? 或 下降沿"接触式探头功能"激活?	无法用示波器追踪！
0x00000211	读取	**CANopen**	UINT16[4]	1	[0,1]	下降沿"外部锁存器功能"激活? 或 下降沿"接触式探头功能"激活?	无法用示波器追踪！
0x00000212	读取	**CANopen**	UINT16	1	[0,1]	“接触式探头 1”输入信号的状态	无法用示波器追踪！ TC3.1 B4024.11 及以上
0x00000213	读取	**CANopen**	UINT16	1	[0,1]	“接触式探头 2”输入信号的状态	无法用示波器追踪！ TC3.1 B4024.11 及以上

3.2.1.5.4.5.3 *编码器功能的"索引偏移"规范（索引组 0x5200 + ID）*

索引偏移（十六进制）	访问	组类型	数据类型	物理单位	定义范围	描述	备注
0x0000001A	写入	每个	{			设置实际位置编码器/轴	基本单位
			UINT32	枚举	参见附录	实际位置类型 [▶ 145]（参见附录）	
			REAL64	mm	±∞	编码器/轴的实际位置 **使用时请注意！**	
			}				
0x0000001B	写入	每个	VOID			重新初始化实际编码器位置 注：对参考系统"绝对多圈量程（带单个溢出）"和"绝对单圈量程（带单个溢出）"有效。	从 TC3 起更新
0x00000200	写入	功能组"TouchProbeV2": - SERCOS/SoE, - EtherCAT/CoE (CANopen DS402) - SoftDrive (TCom), - MDP 511 (EL5101, EL5151, EL5021, EL7041, EL7342)	{			激活"接触式探头"（外部锁存器）	仅适用于 SAF 端口 501
			UINT32	1	[1,2,3,4]	探头单元（探头 1、2、3、4）	
			UINT32	1	[0,1]	信号边沿（0=上升沿，1=下降沿）	
			UINT32	1	[1,2]	探头模式（1=单，2=连续，...）	
			UINT32	1	[1,2,3,4; 128,129]	信号源（1=输入 1，2=输入 2，...）	
			UINT32			保留	
			UINT32			保留	
			} 24 字节				
0x00000201	写入	KL5101、SERCOS、AX2xxx、PROFIDrive	VOID			激活"外部锁存器"或激活"测量探头功能"（通常为上升沿）	
0x00000201	写入	CANopen	UINT32[4]			激活"外部锁存器"1 至 4 或激活"测量探头功能"1 至 4（通常为上升沿）	
0x00000202	写入	KL5101、SERCOSAX2xxx、PROFIDrive	VOID			激活"外部锁存器"或激活"测量探头功能"（下降沿）	
0x00000202	写入	CANopen	UINT32[4]			激活"外部锁存器"1 至 4 或激活"测量探头功能"1 至 4（下降沿）	
0x00000205	写入	功能组"TouchProbeV2": - SERCOS/SoE, - EtherCAT/CoE (CANopen DS402) - SoftDrive (TCom), - MDP 511 (EL5101, EL5151, EL5021, EL7041, EL7342)	{			停用"接触式探头"（外部锁存器）	仅适用于 SAF 端口 501
			UINT32	1	[1,2,3,4]	探头单元（探头 1、2、3、4）	
			UINT32	1	[0,1]	信号边沿（0=上升沿，1=下降沿）	
			UINT32			保留	
			UINT32			保留	
			UINT32			保留	
			UINT32			保留	
			} 24 字节				
0x00000205	写入	KL5101、SERCOS、AX2xxx、PROFIDrive	VOID			停用"外部锁存器"或停用"测量探头功能"	
0x00000205	写入	CANopen	UINT32[4]			停用"外部锁存器"或停用"测量探头功能"	

索引偏移（十六进制）	访问	组类型	数据类型	物理单位	定义范围	描述	备注
0x00000210	写入	KL5101、SERCOS、AX2xxx、PROFIDrive	REAL64	例如 mm	±∞	设置"外部锁存器事件"和"外部锁存器位置"	仅适用于 EtherCAT：

3.2.1.5.4.5.4 *循环编码器过程数据的"索引偏移"规范（索引组 0x5300 + ID）*

索引偏移（十六进制）	访问	组类型	数据类型	物理单位	定义范围	描述	备注
0x00000000	读/写	每个（NC→IO）	{		STRUCT 参见编码器接口	编码器-输出-结构（NC→IO，40 字节）NCENCODERSTRUCT_OUT2	写入命令仅为可选项！ 考虑安全方面！
			INT32	INC	⩾ 0	nDataOut1	
			INT32	INC	⩾ 0	nDataOut2	
			UINT8	1	⩾ 0	nCtrl1	
			UINT8	1	⩾ 0	nCtrl2	
			UINT8	1	⩾ 0	nCtrl3	
			UINT8	1	⩾ 0	nCtrl4	
			INT32	INC	⩾ 0	nDataOut3	
			INT32	INC	⩾ 0	nDataOut4	
			INT32	INC	⩾ 0	nDataOut5	
			INT32	INC	⩾ 0	nDataOut6	
			UINT8	1	⩾ 0	nCtrl5	
			UINT8	1	⩾ 0	nCtrl6	
			UINT8	1	⩾ 0	nCtrl7	
			UINT8	1	⩾ 0	nCtrl8	
			INT32	1	⩾ 0	保留	
			INT32	1	⩾ 0	保留	
			} 40 字节				
0x00000000	读/写	每个(NC→IO)，可选 64 位编码器接口（例如 MDP513，带 64 位）	{		STRUCT 参见编码器接口	可选 ENCODER-OUTPUT-STRUCTURE（NC→IO，80 字节）NCENCODERSTRUCT_OUT3	写入命令仅为可选项！ 考虑安全方面！ 从 TC3 起更新
			UINT64	INC	⩾ 0	nDataOut1	
			UINT64	INC	⩾ 0	nDataOut2	
			UINT64	INC	⩾ 0	nDataOut3	
			UINT64	INC	⩾ 0	nDataOut4	
			UINT64	INC	⩾ 0	nDataOut5	
			UINT64	INC	⩾ 0	nDataOut6	
			UINT64	INC	⩾ 0	nDataOut7	
			UINT64	INC	⩾ 0	nDataOut8	
			UINT16	1	⩾ 0	nCtrl1	
			UINT16	1	⩾ 0	nCtrl2	
			UINT16	1	⩾ 0	nCtrl3	
			UINT16	1	⩾ 0	nCtrl4	
			UINT16	1	⩾ 0	nCtrl5	
			UINT16	1	⩾ 0	nComCtrl	
			INT32	1	⩾ 0	保留	
			} 80 字节				

索引偏移（十六进制）	访问	组类型	数据类型	物理单位	定义范围	描述	备注
0x00000001	写入	每个 (NC→IO)	{		STRUCT 参见编码器接口	按位访问 编码器-输出-结构（NC→IO，40 字节）NCENCODERSTRUCT_OUT2	写入命令仅为可选项！ 考虑安全方面！
			UINT32	1	[0 … 39]	ByteOffset 输出结构中的相对地址偏移 [0.39]。 例如：要写入"nControl1"，ByteOffset 必须为 8。	
			UINT32	1	[0x00000000... 0xFFFFFFFF]	BitSelectMask (BSM) 掩码在 DWORD 中定义允许写入的位。零位受保护，不受影响。	
			UINT32	1	[0x00000000... 0xFFFFFFFF]	数值 只有 BSM 等于 1 的值位才会被重写。	
			}				
0x00000080	读取	每次（IO→NC）	{		STRUCT 参见编码器接口	编码器-输入-结构（IO→NC，40 字节）NCENCODERSTRUCT_IN2	
			INT32	INC	⩾ 0	nDataIn1	
			INT32	INC	⩾ 0	nDataIn2	
			UINT8	1	⩾ 0	nState1	
			UINT8	1	⩾ 0	nState2	
			UINT8	1	⩾ 0	nState3	
			UINT8	1	⩾ 0	nState4（Bit0：WcState，Bit1：InputToggle）	
			INT32	INC	⩾ 0	nDataIn3	
			INT32	INC	⩾ 0	nDataIn4	
			INT32	INC	⩾ 0	nDataIn5	
			INT32	INC	⩾ 0	nDataIn6	
			UINT8	1	⩾ 0	nState5	
			UINT8	1	⩾ 0	nState6	
			UINT8	1	⩾ 0	nState7	
			UINT8	1	⩾ 0	nState8	
			INT32	[ns]	⩾ 0	nDcInputTime（用于死区时间补偿的绝对/相对 DcInputShift）	
			INT32	1	⩾ 0	保留	
			} 40 字节				

索引偏移（十六进制）	访问	组类型	数据类型	物理单位	定义范围	描述	备注
0x00000080	读取	每个（NC→IO），可选 64 位编码器接口（例如，MDP513，带 64 位）	{		STRUCT 参见编码器接口	可选编码器-输入-结构（IO→NC，80 字节）NCENCODERSTRUCT_IN3	从 TC3 起更新
			UINT64	INC	≥ 0	nDataIn1	
			UINT64	INC	≥ 0	nDataIn2	
			UINT64	INC	≥ 0	nDataIn3	
			UINT64	INC	≥ 0	nDataIn4	
			UINT64	INC	≥ 0	nDataIn5	
			UINT64	INC	≥ 0	nDataIn6	
			UINT64	INC	≥ 0	nDataIn7	
			UINT64	INC	≥ 0	nDataIn8	
			UINT16	1	≥ 0	nState1	
			UINT16	1	≥ 0	nState2	
			UINT16	1	≥ 0	nState3	
			UINT16	1	≥ 0	nState4	
			UINT16	1	≥ 0	nState5	
			UINT16	1	≥ 0	nComState （Bit0：WcState，Bit1：InputToggle）	
			INT32	[ns]	≥ 0	nDcInputTime（用于死区时间补偿的绝对/相对 DcInputShift）	
			} 80 字节				

3.2.1.5.4.6 控制器规范

3.2.1.5.4.6.1 *控制器参数的"索引偏移"规范（索引组 0x6000 + ID）*

索引偏移（十六进制）	访问	控制器类型	数据类型	物理单位	定义范围	描述	备注
0x00000001	读取	每个	UINT32	1	[1 ... 255]	控制器 ID	
0x00000002	读取	每个	UINT8[30+1]	1	30 个符号	控制器名称	
0x00000003	读取	每个	UINT32	1	参见 枚举 (>0)	控制器类型 [▶ 146]	
0x0000000A	读/写	每个	UINT32	1	参见 枚举 (>0)	控制器模式	默认： 标准
0x0000000B	读/写	每个	REAL64	%	[0.0 ... 1.0]	速度预控制的权重（标准值：1.0 = 100%）	
0x00000010	读/写	每个	UINT16	1	0/1	跟随错误监测位置?	
0x00000011	读/写	每个	UINT16	1	0/1	跟随错误监测速度?	
0x00000012	读/写	每个	REAL64	mm	[0.0…1.0E.6]	最大跟随误差位置	
0x00000013	读/写	每个	REAL64	s	[0.0...600]	最大跟随误差时间位置	
0x00000014	读/写	每个	REAL64	mm/s	[0.0…1.0E.6]	最大跟随误差速度	
0x00000015	读/写	每个	REAL64	s	[0.0…1.0E.6]	最大跟随误差时间速度	
0x00000021	读/写	每个	REAL64	1	[0.0...1000000.0]	主轴和从轴之间位置差的缩放因子（scaling factor）（乘数）（在同一坐标系中转换）	保留功能，无标准！
0x00000100	读/写	P/PID（位置、速度）	REAL64	1	[0.0...1.0]	控制器总输出的最大输出限制 ( )	（标准值：0.5 == 50%）
0x00000102	读/写	P/PID（位置）	REAL64	mm/s/mm	[0.0...1000.0]	比例放大系数 k_p 或 k_v	基础单位 / s / 基础单位位置控制
0x00000103	读/写	PID（位置）	REAL64	s	[0.0 ... 60.0]	积分作用时间 Tn	位置控制
0x00000104	读/写	PID（位置）	REAL64	s	[0.0 ... 60.0]	微分作用时间 Tv	位置控制
0x00000105	读/写	PID（位置）	REAL64	s	[0.0 ... 60.0]	阻尼时间 Td	位置控制
0x00000106	读/写	PP（位置）	REAL64	mm/s/mm	[0.0...1000.0]	在超过速度限制时，增加比例放大系数 kp或 kv，这个系数是以百分比的形式来表示的。	基础单位 / s / 基础单位位置控制
0x00000107	读/写	PP（位置）	REAL64	%	[0.0...1.0]	阈值水平速度（以百分比表示），超过该值时，将适用额外的比例放大系数 kp 或 kv。	（标准值：0.01 == 1%）
0x00000108	读/写	P/PID（加速度）	REAL64	s	[0.0 ... 100.0]	比例放大系数 ka	加速度预控制
0x0000010A	读/写	每个	UINT32	1	枚举	额定速度最大斜率滤波（加速度受限）： 0: Off, 1: Velo, 2: Pos+Velo	保留功能，无标准！
0x0000010B	读/写	每个	REAL64	mm/s^2		额定速度最大斜率的滤波值（最大加速度）	保留功能，无标准！
0x0000010D	读/写	P/PID	REAL64	mm	[0.0 ... 10000.0]	位置误差（位置偏差）的“死区” （适用于带速度或扭矩接口的 P/PID 控制器）	保留功能
0x0000010F	读/写	P/PP/PID（位置） 从轴控制器	REAL64	(mm/s) / mm	[0.0...1000.0]	从轴耦合控制器： 主轴和从轴之间位置偏差的比例增益 k_{cp}	从轴耦合控制器
0x00000110	读/写	P（位置）	UINT16	1	0/1	自动偏移校准： 主动/被动	
0x00000111	读/写	P（位置）	UINT16	1	0/1	自动偏移校准： 保持模式	
0x00000112	读/写	P（位置）	UINT16	1	0/1	自动偏移校准： 衰减模式	

索引偏移（十六进制）	访问	控制器类型	数据类型	物理单位	定义范围	描述	备注
0x00000114	读/写	P（位置）	REAL64	%	[0.0 ... 1.0]	自动偏移校准： 预控制限制	（标准值：0.05 == 5%）
0x00000115	读/写	P（位置）	REAL64	s	[0.1 ... 60.0]	自动偏移校准： 时间常量	
0x00000116	读/写	PID（位置）	REAL64	%	[0.0...1.0]	I 部分最大输出限制 ( )，以百分比为单位（默认设置：0.1 == 10 %）	
0x00000117	读/写	PID（位置）	REAL64	%	[0.0...1.0]	D 部分最大输出限制 ( )，以百分比为单位（默认设置：0.1 == 10 %）	
0x00000118	读/写	PID（位置）	UINT16	1	0/1	在主动定位过程中关闭 I 部分（只要 I 部分激活）？ （默认设置：0 = FALSE）	
0x00000120	读/写	P/PID（位置）	REAL64	s	≥0	PT-1 位置误差（位置差）滤波时间	保留功能，无标准！
0x00000202	读/写	P/PID（速度）	REAL64	1	[0.0...1000.0]	比例放大系数 k_p 或 k_v	速度控制
0x00000203	读/写	PID（速度）	REAL64	s	[0.0 ... 60.0]	积分作用时间 T_n	速度控制
0x00000204	读/写	PID（速度）	REAL64	s	[0.0 ... 60.0]	微分作用时间 T_v	速度控制
0x00000205	读/写	PID（速度）	REAL64	s	[0.0 ... 60.0]	阻尼时间 T_d	速度控制
0x00000206	读/写	PID（速度）	REAL64	%	[0.0...1.0]	I 部分最大输出限制（），以百分比为单位（默认设置：0.1 == 10 %）	速度控制
0x00000207	读/写	PID（速度）	REAL64	%	[0.0...1.0]	D 部分最大输出限制（），以百分比为单位（默认设置：0.1 = 10%）	速度控制
0x0000020D	读/写	P/PID（速度）	REAL64	mm/s	[0.0 ... 10000.0]	速度误差（速度偏差）的"死区" （适用于带速度或扭矩接口的 P/PID 控制器）	保留功能
0x00000220	读/写	P/PID（速度）	REAL64	s	≥0	PT-2 速度误差（速度差）滤波时间	速度控制，无标准！
0x00000221	读/写	P/PID（速度）	REAL64	s	≥0	PT-1 速度误差（速度差）滤波时间	保留功能，无标准！
0x00000250	读/写	P/PI（观测器）	UINT32	1	参见 枚举 (≥0)	带扭矩接口控制器的观测器模式 [▶ 146] 0：关闭（默认） 1：卢恩伯格（LUENBERGER）	
0x00000251	读/写	P/PI（观测器）	REAL64	Nm / A	>0.0	电机： 扭矩常数 K_T	
0x00000252	读/写	P/PI（观测器）	REAL64	kg m^2	>0.0	电机： 转动惯量 J_M	
0x00000253	读/写	P/PI（观测器）	REAL64	Hz	[100.0 ... 2000.0] 默认：500	带宽 f_0	
0x00000254	读/写	P/PI（观测器）	REAL64	1	[0.0 ... 2.0] 默认：1.0	校正系数 k_c	
0x00000255	读/写	P/PI（观测器）	REAL64	s	[0.0 ... 0.01] 默认：0.001	速度滤波（1. 阶）： 滤波时间常数 T	
0x00000A03	读/写	PID (MW)	REAL64	cm^2	[0.0 ...1000000]	A 面气缸面积 $_{AA}$，单位为 cm^2	

索引偏移（十六进制）	访问	控制器类型	数据类型	物理单位	定义范围	描述	备注
0x00000A04	读/写	PID (MW)	REAL64	cm^2	[0.0 ...1000000]	B 面气缸面积 $_{AB}$，单位为 cm^2	
0x00000A05	读/写	PID (MW)	REAL64	cm^3/s	[0.0 ...1000000]	额定体积流量 $_{Qnenn}$，单位为 cm^3/s	
0x00000A06	读/写	PID (MW)	REAL64	bar	[0.0 ...1000000]	额定压力或阀门减压 P_{nenn}，单位为 bar	
0x00000A07	读/写	PID (MW)	UINT32	1	[1 ... 255]	系统压力 $_{Po}$ 的轴 ID	

3.2.1.5.4.6.2 *控制器状态的"索引偏移"规范（索引组 0x6100 + ID）*

索引偏移（十六进制）	访问	控制器类型	数据类型	物理单位	定义范围	描述	说明
0x00000001	读取	每个	INT32			错误状态控制器	
0x00000002	读取	每个	REAL64	例如 mm/s		控制器输出（以绝对单位表示）	基本单位 / s 可以进行符号访问！ "CtrlOutput"
0x00000003	读取	每个	REAL64	%		控制器输出（以百分比为单位）	无法用示波器追踪！
0x00000004	读取	每个	REAL64	V		控制器输出（以伏特为单位）	无法用示波器追踪！
0x0000000D	读取	每个	REAL64	mm		跟随误差位置（无死区时间补偿）	基本单位
0x0000000E	读取	每个	REAL64	mm		跟随误差位置（无设置位置校正）	基本单位
0x0000000F	读取	每个	REAL64	mm		跟随误差位置（带设置位置校正和死区时间补偿）	基本单位 可以进行符号访问！ "PosDiff"
0x00000010	读取	每个	REAL64	mm		位置最大负跟随误差的峰值保持值	基本单位
0x00000011	读取	每个	REAL64	mm		位置最小正跟随误差的峰值保持值	基本单位
0x00000012	读取	每个	REAL64	mm/s		跟随误差速度	基本单位 / s
0x00000021	读取	每个	REAL64	mm		主轴和从轴的跟随误差之间的差（偏差）（主轴误差减去从轴误差）	基本单位 可通过轴进行符号访问！ "PosDiffCouple
0x00000022	读取	每个	REAL64	mm		主轴和从轴位置跟随误差之间最大负差的峰值保持值	基本单位
0x00000023	读取	每个	REAL64	mm		主轴和从轴位置跟随误差之间最大正差的峰值保持值	基本单位
0x00000101	读取	P/PID（位置）	REAL64	例如 mm/s		控制器的 P 部分（以绝对单位表示）	
0x00000102	读取	PID（位置）	REAL64	例如 mm/s		控制器的 I 部分（以绝对单位表示）	
0x00000103	读取	PID（位置）	REAL64	例如 mm/s		控制器的 D 部分（以绝对单位表示）	
0x00000104	读取	PID（位置）	UINT16	1	0/1	I 部分限制激活?	
0x00000105	读取	PID（位置）	UINT16	1	0/1	D 部分限制激活?	
0x00000106	读取	PID（位置）	UINT16	1	0/1	I 部分 ARW 测量激活?	ARW：防积分饱和
0x0000010F	读取	P/PP/PID（速度）	REAL64	例如 mm/s		自动偏移补偿比例（以绝对单位表示）	新
0x00000110	读取	PID（位置）	REAL64	例如 mm/s		控制器的加速度预控制 Y_{acc}（以绝对单位表示） 注：功能取决于控制器类型！	加速度预控制
0x00000111	读取	PP（位置）	REAL64	mm/s/mm	⩾0	内部插值比例增益 kp 或 kv	PP 控制器

索引偏移（十六进制）	访问	控制器类型	数据类型	物理单位	定义范围	描述	说明
0x0000011A 0x0000011B 0x0000011C 0x0000011D 0x0000011E 0x0000011F 0x00000120 0x00000121 0x00000122 0x00000123 0x00000124	读取	P（位置）	uint32 real64 real64 real64 real64 real64 real64 real64 real64 real64 real64	1 mm mm/s mm/s mm/s^2 mm mm mm/s mm/s^2 mm/s mm/s^2		设置速度滤波： InternalPhase InternalPosSollError! TestVeloSoll InternalLimitedVeloSoll InternalAccSollRel InternalPosSollRel PosSollCorrected! VeloSollCorrected! AccSollCorrected! TestVeloSollCorrected TestAccSollCorrected	列表！保留功能， 无标准！
0x00000201	读取	P,PID（速度）	REAL64	例如 mm/s		控制器的速度部分	基本单位 / s
0x00000202	读取	P,PID（速度）	REAL64	%		控制器的速度部分（以百分比为单位）	无法用示波器追踪！
0x00000203	读取	P,PID（速度）	REAL64	V		控制器的速度部分（以伏特为单位）	无法用示波器追踪！
0x00000201	读取	P/PID（速度）	REAL64	例如 mm/s		控制器的 P 部分（以绝对单位表示）	
0x00000202	读取	P/PID（速度）	REAL64	例如 mm/s		控制器的 I 部分（以绝对单位表示）	
0x00000203	读取	P/PID（速度）	REAL64	例如 mm/s		控制器的 D 部分（以绝对单位表示）	
0x00000204	读取	P/PID（速度）	UINT16	1	0/1	I 部分限制激活?	
0x00000205	读取	P/PID（速度）	UINT16	1	0/1	D 部分限制激活?	
0x00000206	读取	P/PID（速度）	UINT16	1	0/1	I 部分 ARW 测量激活?	ARW：防积分饱和
0x0000020A	读取	P/PID（速度）	REAL64	例如 mm/s		速度控制器的总输入大小	
0x00000250	读取	P/PI（观测器）	REAL64	例如 mm		观测器：位置差（实际位置 — 观测器位置）	
0x00000251	读取	P/PI（观测器）	REAL64	例如 mm		观测器：位置	
0x00000252	读取	P/PI（观测器）	REAL64	例如 mm/s		观测器：速度 2（用于 P 部分）	
0x00000253	读取	P/PI（观测器）	REAL64	例如 mm/s		观测器：速度 1（用于 I 部分）	
0x00000254	读取	P/PI（观测器）	REAL64	例如 mm/s^2		观测器：加速度	
0x00000255	读取	P/PI（观测器）	REAL64	A		观测器：电机实际电流	
0x00000256	读取	P/PI（观测器）	UINT16	1	0/1	观测器：I 部分限制激活?	
0x00000A00	读取	PID (MW)	REAL64	%	[-1.0...1.0]	计算设置速度（预控制）（以百分比为单位）	
0x00000A01	读取	PID (MW)	REAL64	例如 mm/s		控制器的 P 部分（以绝对单位或百分比表示）（根据输出权重计算）	
0x00000A02	读取	PID (MW)	REAL64	例如 mm/s		控制器的 I 部分（以绝对单位或百分比表示）（根据输出权重计算）	
0x00000A03	读取	PID (MW)	REAL64	例如 mm/s		控制器的 D 部分（以绝对单位或百分比表示）（根据输出权重计算）	
0x00000A04	读取	PID (MW)	UINT16	1	0/1	I 部分限制激活?	

索引偏移（十六进制）	访问	控制器类型	数据类型	物理单位	定义范围	描述	说明
0x00000A05	读取	PID (MW)	UINT16	1	0/1	D 部分限制激活?	
0x00000A10	读取	PID（位置）	REAL64	例如 mm/s		控制器的加速度预控制 Y_{acc}（以绝对单位表示）	加速度预控制

3.2.1.5.4.6.3 *控制器功能的"索引偏移"规范（索引组 0x6200 + ID）*

索引偏移（十六进制）	访问	控制器类型	数据类型	物理单位	定义范围	描述	备注

3.2.1.5.4.7 规范器驱动

3.2.1.5.4.7.1 *驱动参数的"索引偏移"规范（索引组 0x7000 + ID）*

索引偏移（十六进制）	访问	驱动器类型	数据类型	物理 单位	定义范围	描述	说明
0x00000001	读取	每个	UINT32	1	[1 ... 255]	驱动器 ID	
0x00000002	读取	每个	UINT8[30+1]	1	30 个字符	驱动器名称	
0x00000003	读取	每个	UINT32	1	参见 枚举 (>0)	驱动器类型 [▶ 152]	
0x00000004	读/写	每个	UINT32	1	Byteoffset	输入地址偏移（IO-输入-图像）	更改 I/O 地址
0x00000005	读/写	每个	UINT32	1	Byteoffset	输出地址偏移（IO-输出-图像）	更改 I/O 地址
0x00000006	读/写	每个	UINT16	1	[0,1]	电机极性	如果已发出启用控制器的信号，则不允许写入。
0x0000000A	读/写	每个	UINT32	1	参见 枚举 (>0)	驱动模式	默认： 1 = STANDARD
0x0000000B	读/写	每个	REAL64	%	[-1.0 ... 1.0]	最小输出限制（输出限制）（默认设置：-1.0 == -100）	
0x0000000C	读/写	每个	REAL64	%	[-1.0 ... 1.0]	最大输出限制（输出限制）（默认设置：1.0 == 100%）	
0x0000000D	读取	每个	UINT32	INC		输出增量的最大数量（输出掩码）	
0x00000010	读/写	每个	UINT32	1		内部驱动控制双字，用于确定驱动运行模式	保留！
0x00000011	读/写	每个	UINT32	1	≥ 5	内部驱动复位计数器（启用和复位的 NC 循环时间）	保留！
0x00000020	读/写	每个	UINT32	1	参见 枚举 (≥0) 参见附录	驱动死区时间补偿模式 0：关闭（默认） 1: 开（有速度） 2: 开（有速度和加速度）	
0x00000021	读/写	每个	UINT32	1		用于驱动死区时间补偿的控制双字（32 位）： 位 0 = 0：相对 IO 时间（默认） 位 0 = 1：绝对 IO 时间	
0x00000022	读/写	每个	INT32	ns	[±1.0E+9]	用于驱动死区时间补偿的参数化时间偏移之和（通常为正数值）	
0x00000031	读/写	每个	REAL64	例如 %/INC	[-1.0E+30 ... 1.0E+30]	驱动器实际扭矩值的缩放因子（scaling factor） （或分别为力或电流的实际值） 例如，AX5xxx: 0.1 => ±100%	从 TC3.1 起更新
0x00000032	读/写	每个	REAL64	s	[0.0 ... 60.0]	P-T1 实际扭矩值滤波时间 （或分别为力或电流的实际值）	从 TC3.1 起更新
0x00000033	读/写	每个	REAL64	s	[0.0 ... 60.0]	实际扭矩值时间导数的 P-T1 滤波时间 （或分别为力或电流的实际值）	从 TC3.1 起更新
0x00000101	读/写	伺服	REAL64	例如 mm/s	>0.0	参考输出的参考速度（速度预控制）	基本单位 / s
0x00000102	读/写	伺服	REAL64	%	[0.0 ... 5.0]	参考输出（以百分比为单位）	

索引偏移（十六进制）	访问	驱动器类型	数据类型	物理 单位	定义范围	描述	说明
0x00000103	读取	伺服	REAL64	例如 mm/s	>0.0	100% 输出时产生的速度	基本单位 / s
0x00000104	读/写	伺服	REAL64	例如 mm/s	±∞	轴漂移校准（偏移校准）的速度偏移（DAC 偏移）	基本单位 / s
0x00000105	读/写	伺服（Sercos、Profi Drive、AX200x、CANopen）	REAL64	1	[0.0 ... 100000000.0]	速度缩放（响应驱动器中权重的缩放因子（scaling factor））	适用于 Sercos、Profi Drive、AX200x、CANopen
0x00000106	读/写	Profi Drive DSC	UINT32	0.001 * 1/s	⩾ 0	Profibus/Profi Drive DSC：位置控制增益 Kpc	仅适用于 Profi Drive DSC
0x00000107	读/写	Profi Drive DSC	REAL64	1	⩾ 0.0	Profibus/Profi Drive DSC：用于计算"XERR"的缩放比例（默认：1.0）	仅适用于 Profi Drive DSC
0x00000109	读/写	伺服	REAL64	1	[0.0 ... 100000000.0]	位置缩放（响应驱动器中权重的缩放因子（scaling factor））	适用于 Sercos、CANopen
0x0000010A	读/写	伺服	REAL64	1	[0.0 ... 100000000.0]	加速度缩放（响应驱动器中权重的缩放因子（scaling factor））	适用于 Sercos、Profi Drive、AX200x、CANopen
0x0000010B	读/写	伺服	REAL64	1	[0.0 ... 100000000.0]	适用于"TorqueOffset"的扭矩缩放（旋转电机）或力缩放（直线电机）（响应驱动器中权重的缩放因子（scaling factor））（作为预控制的加法力矩）	适用于 Sercos、Profi Drive、AX200x、CANopen
0x0000010C	读/写	伺服	REAL64	1	[0.0 ... 100000000.0]	适用于"SetTorque"的扭矩缩放（旋转电机）或力缩放（直线电机）（响应驱动器中重量的缩放因子（scaling factor））（例如 MC_TorqueControl，使用驱动器运行模式 CST）	适用于 Sercos、Profi Drive、AX200x、CANopen TC 3.1 B4024.2 及以上
0x0000010D	读/写	伺服（Sercos、CANopen）	REAL64	s	[0.0 ... 1.0]	驱动速度输出的阻尼时间	适用于 Sercos、CANopen
0x0000010E	读/写	伺服（Sercos、CANopen）	REAL64	s	[0.0 ... 1.0]	驱动加速度输出的阻尼时间	适用于 Sercos、CANopen
0x0000010F	读/写	伺服（Sercos、CANopen）	REAL64	s	[0.0 ... 1.0]	驱动扭矩输出或力输出的阻尼时间	适用于 Sercos、CANopen
0x00000120	读/写	伺服/液压/	UINT32	1	⩾ 0	表 ID （0：无表）	仅适用于 KL4xxx、M2400、通用型
0x00000121	读/写	伺服/液压	UINT32	1	⩾ 0	插值类型 0：线性 2：样条	仅适用于 KL4xxx、M2400、通用型
0x00000122	读/写	伺服/液压	REAL64	%	[-1.0 ... 1.0]	输出偏移（以百分比为单位） 注： 根据特征评估进行操作！	仅适用于 KL4xxx、M2400、通用型

索引偏移（十六进制）	访问	驱动器类型	数据类型	物理 单位	定义范围	描述	说明
0x00000151	读/写	伺服/非线性	REAL64	1	[0.0 ... 100.0]	象限补偿系数（I 和 III 象限之间的关系）	
0x00000152	读/写	伺服/非线性	REAL64	1	[0.01 ... 1.0]	速度参考点（以百分比为单位）（1.0 == 100）	
0x00000153	读/写	伺服/非线性	REAL64	1	[0.01 ... 1.0]	输出参考点（以百分比为单位）（1.0 == 100%）	
0x00000301	读/写	步进电机	UINT8			位掩码：循环 1	
0x00000302	读/写	步进电机	UINT8			位掩码：循环 2	
0x00000303	读/写	步进电机	UINT8			位掩码：循环 3	
0x00000304	读/写	步进电机	UINT8			位掩码：循环 4	
0x00000305	读/写	步进电机	UINT8			位掩码：循环 5	
0x00000306	读/写	步进电机	UINT8			位掩码：循环 6	
0x00000307	读/写	步进电机	UINT8			位掩码：循环 7	
0x00000308	读/写	步进电机	UINT8			位掩码：循环 8	
0x00000310	读/写	步进电机	UINT8			位掩码：保持电流	

3.2.1.5.4.7.2 *驱动器状态的"索引偏移"规范（索引组 0x7100 + ID）*

索引偏移（十六进制）	访问	驱动器类型	数据类型	物理单位	定义范围	描述	说明
0x00000001	读取	每个	INT32			错误状态驱动器	
0x00000002	读取	每个	REAL64	例如 mm/s		总输出（以绝对单位表示）	基本单位 / s可进行符号访问! "DriveOutput"
0x00000003	读取	每个	REAL64	%		总输出（以百分比为单位）	
0x00000004	读取	每个	REAL64	V		总输出（以伏特为单位）	无法用示波器追踪!
0x00000005	读取	每个	REAL64	例如 mm/s		最大负总输出的峰值保持值	基本单位 / s
0x00000006	读取	每个	REAL64	例如 mm/s		最大正总输出的峰值保持值	基本单位 / s
0x00000007	读取	每个	REAL64	例如 100% = 1000，例如 Nm 或 N		分别为实际扭矩或实际力（通常 100% = 1000）	TC3.1 B4022 及以上 可进行符号访问! "ActTorque"
0x00000008	读取	每个	REAL64	例如 Nm/s 或 N/s	±∞	分别为实际扭矩变化或实际力变化 （分别为实际扭矩或实际力的时间导数）	TC3.1 B4024 及以上
0x0000000C	读取	每个	REAL64	例如 mm		设置由于死区时间补偿产生的驱动输出的位置校正值	
0x0000000D	读取	每个	REAL64	s		驱动死区时间补偿的时移总和 （参数化和可变死区时间） 注：系统中规定的死区时间为正值。	
0x00000013	读取	每个	REAL64	%		总输出（以百分比为单位）（基于非线性特征曲线！）	
0x00000014	读取	每个	REAL64	V		总输出（以伏特为单位）（基于非线性特征曲线！）	无法用示波器追踪!
0x0000011A	读取	伺服（Sercos、CANopen）	REAL64	例如 mm		可选输出滤波： 滤波设置位置	新 适用于 Sercos、CANopen
0x0000011E	读取	伺服（Sercos、CANopen）	REAL64	例如 mm/s		可选输出滤波： 滤波设置速度	新 适用于 Sercos、CANopen
0x0000011F	读取	伺服（Sercos、CANopen）	REAL64	例如 mm/s^2		可选输出滤波： 滤波设置加速度/设置减速度	新 适用于 Sercos、CANopen
0x00000200	读写		**读取：**			读取数字输入 1 至 8 的状态	TC3.1 B4024.12 及以上 仅适用于 SAF-Port 501!
			UINT32	1	0/1	所选输入的状态	
			写入：				
			UINT32	1	[1…8]	选择输入 1 至 8	

3.2.1.5.4.7.3 *驱动器功能的"索引偏移"规范（索引组 0x7200 + ID）*

索引偏移（十六进制）	访问	驱动器类型	数据类型	物理单位	定义范围	描述	备注
0x00000102	写入	伺服	{			移除和删除特征驱动表	仅适用于 SAF 端口 501!
			ULONG	1	>0	表 ID S.A. 轴功能，带索引偏移 0x00000012	
			}				

3.2.1.5.4.7.4 *循环驱动过程数据的"索引偏移"规范（索引组 0x7300 + ID）*

索引偏移（十六进制）	访问	驱动器类型	数据类型	物理单位	定义范围	描述	备注
0x00000000	读/写	每个（NC→IO）	{		STRUCT 参见驱动器接口	驱动器-输出-结构（NC→IO，40 字节）NCDRIVESTRUCT_OUT2	写入命令仅为可选项! 考虑安全方面!
			INT32	INC	⩾ 0	nOutData1	
			INT32	INC	2^31	nOutData2	
			UINT8	1	⩾ 0	nControl1	
			UINT8	1	⩾ 0	nControl2	
			UINT8	1	⩾ 0	nControl3	
			UINT8	1	⩾ 0	nControl4	
			INT32	INC	⩾ 0	nOutData3	
			INT32	INC	⩾ 0	nOutData4	
			INT32	INC	⩾ 0	nOutData5	
			INT32	INC	⩾ 0	nOutData6	
			UINT8	1	⩾ 0	nControl5	
			UINT8	1	⩾ 0	nControl6	
			UINT8	1	⩾ 0	nControl7	
			UINT8	1	⩾ 0	nControl8	
			INT32	1	⩾ 0	保留	
			INT32	1	⩾ 0	保留	
			}				
0x00000001	写入	每个（NC→IO）	{		STRUCT 参见驱动器接口	逐位访问驱动器-输出-结构（NC→IO，40 字节）NCDRIVESTRUCT_OUT2	写入命令仅为可选项! 考虑安全问题
			UINT32	1	[0 … 39]	ByteOffset 输出结构中的相对地址偏移 [0.39]。 例如：要写入"nControl1"，ByteOffset 必须为 8。	
			UINT32	1	[0x00000000... 0xFFFFFFFF]	BitSelectMask (BSM) 掩码在 DWORD 中定义允许写入的位。零位受保护，不受影响。	
			UINT32	1	[0x00000000... 0xFFFFFFFF]	数值 只有 BSM 等于 1 的值位才会被重写。	
			}				

索引偏移（十六进制）	访问	驱动器类型	数据类型	物理单位	定义范围	描述	备注
0x00000080	读取	每次（IO→NC）	{		STRUCT 参见驱动器接口	驱动器-输入-结构（IO→NC，40 字节）NCDRIVESTRUCT_IN2	
			INT32	INC	≥ 0	nInData1	
			INT32	INC	≥ 0	nInData2	
			UINT8	1	≥ 0	nStatus1	
			UINT8	1	≥ 0	nStatus2	
			UINT8	1	≥ 0	nStatus3	
			UINT8	1	≥ 0	nStatus4	
			INT32	INC	≥ 0	nInData3	
			INT32	INC	≥ 0	nInData4	
			INT32	INC	≥ 0	nInData5	
			INT32	INC	≥ 0	nInData6	
			UINT8	1	≥ 0	nStatus5	
			UINT8	1	≥ 0	nStatus6	
			UINT8	1	≥ 0	nStatus7	
			UINT8	1	≥ 0	nStatus8	
			INT32	1	≥ 0	保留	
			INT32	1	≥ 0	保留	
			}				

3.2.1.5.4.8 表规范

3.2.1.5.4.8.1 *表参数的"索引偏移"规范（索引组 0xA000 + ID）*

索引偏移（十六进制）	访问	表类型	数据类型	物理单位	定义范围	描述	说明
0x00000001	读取	每个	UINT32	1	[1 ... 255]	表 ID	
0x00000002	读取	每个	UINT8[30+1]	1	30 个字符	表格名称	
0x00000003	读取	每个	UINT32	1	参见 枚举 (>0)	表格子类型 [▶ 154]	
0x00000004	读取	每个	UINT32	1	参见 枚举 (>0)	表主要类型 [▶ 154]	
0x00000010	读取	每个	UINT32	1	[0... 16777216]	行数 (n)	
0x00000011	读取	每个	UINT32	1	[0... 16777216]	列数 (m)	
0x00000012	读取	每个	UINT32	1	≥0	元素总数 (n*m)	
0x00000013	读取	等距表格	REAL64	例如 mm	≥0.0	步长（位置增量）（等距表）	基本单位
0x00000014	读取	循环表	REAL64	例如度数	≥0.0	主轴周期（循环表）	基本单位
0x00000015	读取	循环表	REAL64	例如度数	≥0.0	每个主轴周期（循环表）的从轴差值	基本单位
0x0000001A	读/写	“Motion Function”（运动规则）	{			表格数据在线更改激活生效的类型（仅限 MF）	从 TC3 起修改
			UINT32	枚举	参见附录	Activation mode（激活模式） 0: 'instantaneous（立即）'（默认） 1: 'master cam pos（凸轮主轴位置）.' 2: 'master' axis pos（主轴的位置）.' 3: 'next cycle（下一个周期）' 4: 'next cycle once（下个周期，只循环一次）' 5: 'as soon as possible（尽快）' 6: 'off（关闭）' 7: 'delete queued data（删除待处理的数据）'	
			UINT32			保留 (TC3)	
			REAL64	例如 mm	± ∞	激活位置	
			UINT32	枚举	参见附录	主轴scaling type（缩放类型） 0：user defined（用户定义）（默认） 1：scaling with auto offset（自动偏移缩放比例） 2：off（关闭）	
			UINT32	枚举	参见附录	从轴scaling type（缩放类型） 0：user defined（用户定义）（默认） 1：scaling with auto offset（自动偏移缩放比例） 2：off（关闭）	
			}				

索引偏移（十六进制）	访问	表类型	数据类型	物理单位	定义范围	描述	说明
0x00000020	读/写	每个	{			写入单个值 [n,m]：	
			UINT32	1	[0 ... 16777216]	第 n 行	
			UINT32	1	[0 ... 16777216]	第 m 列	
			REAL64	例如 mm	± ∞	单个值	
			}				
0x00000021	读写	每个	*REAL64	例如 mm	± ∞	读取指定主轴位置的从轴位置（根据表格中的“原始值”）	
0x00000022	读写	“Motion Function”（运动规则）	**写入**			将“Motion Function”读作“点云”	只能逐行进行！（整数倍） TC3 中的更改
			{				
			UINT 16	1	0 /1	是否确认采用一致性数据?	
			UINT16	1	位掩码（≥0）	选择位掩码（列数 m 为主轴位置加上位数）： 位 0：Pos 位置（从轴） 位 1：Velo 速度（从轴） 位 2：Acc 加速度（从轴） 位 3：Jerk 加加速度（从轴）	
			UINT32			保留 (TC3)	
			REAL64	例如 mm	± ∞	Startposition（主轴）	
			REAL64	例如 mm	> 0.0	步长	
			}				
			读取				
			{				
			REAL64[x*m]	例如 mm	± ∞	从主轴起始位置读取 x 行：（x*m）个值（一行或多行）	
			}				
0x00000023	读写	每个	**写入**			读取指定主轴位置的从轴值（根据表格中的“原始值”）	
			REAL64	例如 mm	± ∞	主轴位置	
			读取				
			{				
			REAL64	例如 mm	± ∞	从轴位置	
			REAL64	mm/s	± ∞	从轴速度	
			REAL64	mm/s^2	± ∞	从轴加速度	
			}				

索引偏移（十六进制）	访问	表类型	数据类型	物理单位	定义范围	描述	说明
0x00000024	读写	每个	**写入**			计算给定从轴位置的主轴位置	
			REAL64	例如 mm	± ∞	从轴位置	
			REAL64	例如 mm		主轴的起始位置	
			REAL64	例如 mm		至凸轮表从轴位置的偏移量	
			REAL64	1		凸轮表从轴位置的缩放比例	
			REAL64	例如 mm		至凸轮表主轴位置的偏移量	
			REAL64	1		凸轮表主轴位置的缩放比例	
			REAL64			主轴的位置精度（默认：1.0E-3）	
			REAL64[5]			预留（40 字节）	
			读取				
			{				
			UINT32			主轴有位置的下限	
			UINT32			主轴有位置的上限	
			REAL64	例如 mm		主轴绝对位置的下限	
			REAL64	例如 mm		主轴绝对位置的上限	
			REAL64	例如 mm		凸轮表主轴位置的下限	
			REAL64	例如 mm		凸轮表主轴位置的上限	
			REAL64	例如 mm		从轴位置偏移的下限	
			REAL64	例如 mm		从轴位置偏移的上限	
			REAL64[5]			预留（40 字节）	
			}				
0x00000050	读/写	每个	REAL64 [64]	1	± ∞	表格中的特征值 [▶ 156]	
0x00000050	读写	每个	**写入**			读取与额定主轴速度相关的表格中的特征值	从 TC3 起修改
			REAL64 [64]	...	± ∞	可选额定主轴参考速度“fMasterVeloNom”（标准化为 => 1.0 mm/s）， 其余元素不作评估	
			读取				
			REAL64 [64]	...	± ∞	读取表格中的特征值 [▶ 156]	

索引偏移（十六进制）	访问	表类型	数据类型	物理单位	定义范围	描述	说明
0x00000115	写入	单调线性，单调循环，	{			设置/更改表缩放比例：	
			REAL64	1	[± 1000000.0]	表格的原始比例系数	
			REAL64	例如 mm	[± 1000000.0]	主轴列位置偏移	
			REAL64	1	[± 1000000.0]	主轴列缩放比例	
			REAL64	例如 mm	[± 1000000.0]	从轴列位置偏移	
			REAL64	1	[± 1000000.0]	从轴列缩放比例	
			REAL64	例如 mm	[± 1000000.0]	范围下限（起始位置）	
			REAL64	例如 mm	[± 1000000.0]	范围上限（结束位置）	
			}				
0x01000000 + n（从起始行计第 n 行）	读/写 [<=16777216]	每个	{ REAL64[x*m] }	例如 mm	± ∞	从第 n 行读/写 x 行：（x*m）个值（一行或多行），n 的取值范围：[0 ... 16777216]	只能逐行进行！（整数倍）
0x02000000 + m（从起始列计第 m 列）	读/写 [<=16777216]	每个	{ REAL64[x*n] }	例如 mm	± ∞	从第 m 列读/写 x 列：（x*n）个值（一列或多列），m 的取值范围：[0 ... 16777216]	只能逐列进行！（整数倍）
0x05000000 + n（从起始行计第 n 行）	读/写 [<=16777216]	“Motion Function（运动函数）”（运动规则）数据：STRUCT [x*m]	{			从第 n 行读/写 x 行：（x*m）个结构体（一行或多行）n 的取值范围：[0 ... 16777216]	只能逐行进行！（整数倍） 从 TC3 起修改
			UINT32	1		Point 的绝对索引号（不作评估）	
			UINT16	枚举		函数类型 1：多项式 1 15：多项式 5	
			UINT16	枚举		点类型 0：默认 1：忽略	
			INT32	1		最后一个 Point 的相对地址索引号 （默认：1）	
			UINT32			保留 (TC3)	
			REAL64	mm		主轴位置	
			REAL64	mm		从轴位置	
			REAL64	mm/s		从轴速度	
			REAL64	mm/s^2		从轴加速度	
			REAL64	mm/s^3		从轴加加速度	
			}				

索引偏移（十六进制）	访问	表类型	数据类型	物理单位	定义范围	描述	说明
0x06000000 + m（从起始列计第 m 列）	读/写 [<=16777216]	“Motion Function（运动函数）”（运动规则）数据：STRUCT [x* n]	{			从第 m 列读/写 x 列：（x*n）个结构（一列或多列），m 的取值范围：[0 ... 16777216]	只能逐列进行！（整数倍） 从 TC3 起修改
			UINT32	1		Point 的绝对索引号（不作评估）	
			UINT16	枚举		函数类型 1：多项式 1 15：多项式 5	
			UINT16	枚举		点类型 0：默认 1：忽略	
			INT32	1		最后一个 Point 的相对地址索引号 （默认：1）	
			UINT32			保留 (TC3)	
			REAL64	mm		主轴位置	
			REAL64	mm		从轴位置	
			REAL64	mm/s		从轴速度	
			REAL64	mm/s^2		从轴加速度	
			REAL64	mm/s^3		从轴加加速度	
			}				

3.2.1.5.4.8.2 *表状态的"索引偏移"规范（索引组 0xA100 + ID）*

索引偏移（十六进制）	访问	表类型	数据类型	物理单位	定义范围	描述	备注
0x0000000A	读取	每个	INT32	1	⩾ 0	“用户计数器”（表用户数量）	无法用示波器追踪！

3.2.1.5.4.8.3 *表功能的"索引偏移"规范（索引组 0xA200 + ID）*

索引偏移（十六进制）	访问	表类型	数据类型	物理单位	定义范围	描述	备注
0x00010000	写入	每个	{			生成尺寸为 (n*m) 的表格：	表类型：1、2、3、4 尺寸：至少 2x1
			UINT32	1	参见 枚举 (>0)	表类型 [▶ 154]（参见附录）	
			UINT32	1	[2...16777216]	行数量	
			UINT32	1	[1...16777216]	列数量	
			}				
0x00010001	写入	阀图	{			生成尺寸为 (n*m) 的阀图表：	表类型：1,3 尺寸：至少 2x1
			UINT32	1	参见 枚举 (>0)	表类型 [▶ 154]（参见附录）	
			UINT32	1	[2...16777216]	行数量	
			UINT32	1	[1...16777216]	列数量	
			}				
0x00010010	写入	"运动函数"（运动定律）	{			生成尺寸为 (n*m) 的"运动函数"表：	表类型：3、4 尺寸：至少 2x1
			UINT32	1	参见 枚举 (>0)	表格类型（参见附录）	
			UINT32	1	[2...16777216]	行数量	
			UINT32	1	[1...16777216]	列数量	
			}				
0x00020000	写入	每个	VOID			删除尺寸为 (n*m) 的表	表类型：1,2,3,4
0x00030000	写入	每个	VOID			初始化表格 初始化不再需要手动进行了，因为现在在以下情况下会自动进行初始化：a) 与表格关联时 b 耦合；b) 选择从位置时（参见表格参数）	

3.2.1.5.4.9 附录

枚举通道类型

定义	通道类型
1	标准
2	解释器
3	FIFO
4	运动转换

枚举解释器类型

定义	解释器类型
0	未定义
1	NC 解释器 DIN 66025（GST）
2	NC 解释器 DIN 66025（经典方言）

枚举解释器运行模式

定义	解释器/通道运行模式
0x0	默认（停用其他模式）
0x1	NC 核中的单块模式（块执行任务/SAF）
0x1000	保留
0x2000	保留
0x4000	解释器中的单块模式

枚举插值加载日志模式

定义	加载日志模式
0	关闭加载程序日志
1	仅来源
2	来源与编译

枚举插值追踪模式

定义	追踪模式
0	关闭追踪
1	追踪行数
2	追踪源

枚举解释器状态

已移至：解释器的系统管理器界面 — 解释器元素

枚举组类型

定义	组类型
0	未定义
1	PTP 组 + x 从轴
2	1D-Group + x 从轴
3	2D-Group + x 从轴
4	3D-Group + x 从轴
5	高速/低速 + x 从轴
6	低成本步进电机（数字 IO） + x 从轴
7	表组 + x 从轴
9	编码器组 + x 从轴
11	FIFO 组 + x 从轴
12	运动转换组 + x 从轴

枚举曲线速度减小法

已移至：解释器的系统管理器界面 — 组元素

枚举轴类型

定义	轴类型
0	未定义
1	连续轴（伺服）
2	离散轴（高速/低速）
3	连续轴（步进电机）
5	编码器轴
6	连续轴（带运行模式开关，用于位置/压力控制）
7	时基发生器
100	

枚举步进电机运行模式

定义	步进电机运行模式
0	未定义
1	2 相励磁（4 个循环）
2	1-2 相励磁（6 个循环）
3	动力部分

PTP 轴枚举超驰类型（速率比）

定义	超驰类型
1	减少 旧版本，已被"(3) 减少（迭代）"替代
2	原始 旧版本，已被"(4) 原始（迭代）"替代
3	减少（迭代） 默认值：速率比值与特殊情况下会在内部降低的速度有关。这导致从 0% 到 100% 的整个速率比范围速度成正比（=> 线性关系）。
4	原始（迭代） 速率比值始终是指用户编程的速度。但是，如果无法驱动此速度，则会得出一个最大速率比值，超过此值无法达到更高速度（=> 限制）。

枚举组/轴启动类型

定义	组/轴启动类型
0	未定义
1	绝对启动
2	相对启动
3	连续正启动
4	连续负启动
5	模除启动（旧）
261	模除从最短距离启动
517	模除在正方向上启动（带模除公差窗口）
773	模除在负方向上启动（带模除公差窗口）
4096	停止并锁定（轴锁定以执行运动命令）
8192	停止（无运动锁定）

通用轴启动 (UAS) 的枚举命令缓冲类型（缓冲模式）

定义	缓冲模式
0	终止（默认）（瞬时，终止当前运动并删除任何缓冲命令）
1	缓冲 （存储在命令缓冲中，在主动运动后执行）
18	低混合 （缓冲，无停止，以两个命令的最低速度运行通过中间目标位置）
19	混合上一个 （缓冲，无停止，以主动命令的速度运行通过中间目标位置）
20	混合下一个 （缓冲，无停止，以缓冲命令的速度运行通过中间目标位置）
21	混合高 （缓冲，无停止，以两个命令的最高速度运行通过中间目标位置）

枚举结束位置类型（新结束位置）

定义	结束位置类型
0	未定义
1	绝对位置
2	相对位置
3	连续正位置
4	连续负位置
5	模除位置

新速度下新结束位置的枚举命令类型（新结束位置和/或新速度）

定义	新速度下新结束位置的命令类型
0	未定义
1	位置（瞬时）
2	速度（瞬时）
3	位置和速度（瞬时）
9	位置（切换位置）
10	速度（切换位置）
11	位置和速度（切换位置）

枚举实际位置类型（设置实际位置）

定义	实际位置类型
0	未定义
1	绝对位置
2	相对位置
5	模除位置

枚举补偿类型（区间补偿或叠加）

定义	补偿类型
0	未定义
1	veloreduction_additivemotion 减小最大速度 VelocityDiff。补偿行程生效路径由长度和距离组成。
2	veloreduction_limitedmotion 减小最大速度 VelocityDiff。补偿行程生效路径由长度参数定义。
3	LENGTHREDUCTION_ADDITIVEMOTION 最大可用路径减小并由长度和距离组成。系统会尽量利用最大速度。VelocityDiff.
4	LENGTHREDUCTION_LIMITEDMOTION 最大可用路径减小并受到长度参数的限制。系统会尽量利用最大速度。VelocityDiff.

枚举从轴类型

定义	从轴类型
0	未定义
1	线性
2	飞锯（速度、加加速度受限配置文件）
3	飞锯（位置和速度、加加速度受限配置文件）
5	同步发生器（速度、加加速度受限配置文件）
6	同步发生器（位置和速度、加加速度受限配置文件）
10	表格
11	多表格
13	“运动功能”（MF）
15	线性，带有循环性齿轮系数变化（加速度限制的斜坡滤波）
100	特定

枚举从轴去耦类型（用于后续轴命令）

定义	从轴解耦类型（用于后续轴命令）
0	停止、E-stop 或 P-stop（默认） (STOP)
1	定向停止 (O-stop) （ORIENTEDSTOP）
2	将任何加速度减至 0（无动力），并继续移动到无限远的目标位置 (ENDLESS)
3	以新请求速度继续移动到无限远的目标位置 (endless_newvelo)
4	新结束位置 （NEWPOS）
5	新结束位置和新请求速度 (NEWPOSANDVELO)
6	逻辑解耦并立即停止轴，无速度斜坡 (INSTANTANEOUSSTOP)

枚举控制器类型

定义	控制器类型
0	未定义
1	P 控制器（标准） （位置）
2	PP 控制器（带 ka） （位置）
3	PID 控制器（带 ka） （位置）
5	P 控制器 （速度）
6	PI 控制器 （速度）
7	高速/低速控制器 （位置）
8	步进电机控制器 （位置）
9	SERCOS 控制器 （驱动器中的位置）
10	保留
11	保留
12	保留
13	保留
14	TCom 控制器（软驱动） （驱动器中的位置）

枚举控制器观测器模式

定义	控制器观测器模式
0	未启用观测器（默认）
1	"卢恩伯格（Luenberger）"观测器（经典观测器设计）

枚举编码器类型

定义	编码器类型
0	未定义
1	模拟编码器 （增量）
2	M3000 编码器（单圈/多圈） （绝对）
3	M31x0 / M2000 编码器 （增量）
4	MDP 511 编码器：EL7041、EL7342、EL5101、EL5151、EL2521、EL5021、IP5101 （增量）
5	MDP 500/501 编码器：EL5001、IP5009、KL5001（SSI） （绝对）
6	MDP 510 编码器：KL5051、KL2502-30K 编码器（BiSSI） （增量）
7	KL30xx 编码器（模拟） （绝对）
8	SERCOS 和 EtherCAT SoE（位置） （增量）
9	SERCOS 和 EtherCAT SoE（位置和速度） （增量）
10	二进制编码器 (0/1) （增量）
11	M2510 编码器 （绝对）
12	FOX50 编码器 （绝对）
14	AX2000 (Lightbus) （增量）
15	Provi-Drive MC（Simodrive 611U） （增量）
16	通用编码器（可变位掩码） （增量）
17	NC 后面板 （增量）
18	特殊 CANopen 类型（如 Lenze Drive 9300） （增量）
19	MDP 513 (DS402)：CANopen 和 EtherCAT CoE（AX2xx-B1x0/B510、EL7201） （增量）
20	AX2xx-B900（以太网） （增量）
21	KL5151 编码器 （增量）
24	IP5209 编码器 （增量）
25	KL2531/KL2541 编码器（步进电机） （增量）
26	KL2532/KL2542 编码器（DC 电机），KL2535/KL2545（PWM 电流端子） （增量）
27	时基编码器（时基发生器） （增量）

定义	编码器类型
28	TCom 编码器（软驱动） （增量）

编码器模式

定义	编码器模式
0	未定义
1	确定位置
2	确定位置和速度
3	确定位置、速度和加速度

枚举编码器评估方向（日志记录计数方向）

定义	编码器评估方向（日志记录计数方向）
0	正负计数方向评估（默认配置，即与先前状态兼容）
1	仅按正计数方向进行评估
2	仅按负计数方向进行评估
3	既不按正计数方向也不按负计数方向进行评估（评估受阻）

不适用于所有类型的编码器；仅适用于 KL5101、KL5151、KL2531、KL2541、IP5209、通用编码器等。

	编码器类型		
编码器评估方向（日志记录计数方向）	KL5101, ...	通用编码器	其他类型
0：正和负	√	√	—
1：仅正	√	√	—
2：仅负	√	√	—
3：受阻	√	√	—

枚举编码器符号解释（数据类型）

定义	编码器实际增量的符号解释（数据类型）
0	未定义（默认配置，即与先前状态兼容）
1	无符号：编码器实际增量的无符号解释
2	有符号：编码器实际增量的有符号解释

目前仅适用于 KL30xx/KL31xx

枚举编码器绝对尺寸系统

定义	编码器绝对尺寸系统
0	INC：带下溢和上溢偏移的增量绝对尺寸系统（默认设置，即与先前状态兼容）
1	ABS：无下溢和上溢偏移的绝对尺寸系统（不允许编码器下溢或上溢）
2	ABS MODULO：有条件的绝对尺寸系统，因为它有下溢和上溢偏移（绝对值通过模运算（无限循环）继续）

不适用于所有编码器类型；仅适用于 Profi Drive MC、M3000、KL5001/EL5001、IP5009、SERCOS、UNIVERSAL 等。

增量式编码器枚举参考模式

定义	参数文本	增量式编码器参考模式
0	默认值	未定义（默认分配，即与先前状态兼容）
1	仅归位传感器（PLC 凸轮或数字输入）	锁存器事件：关闭 PLC 凸轮（负边沿）
2	硬件同步（反馈参考脉冲）	锁存器事件：硬件同步脉冲（零轨）
3	硬件锁存器 1（位置 边沿）	锁存器事件：带正边沿的外部硬件锁存器（带正边沿的测量探头或各自地即时测量）
4	硬件锁存器 1（负 边沿）	锁存器事件：带负边沿的外部硬件锁存器（带负边沿的测量探头或各自地即时测量）
5	软件同步	锁存器事件：合成模拟软件同步脉冲（软件零轨）；前提条件：每电机转速绝对值，例如旋转变压器！
6	硬件锁存器 1（位置 边沿），驱动器已定义	锁存器事件：驱动器中定义的具有正边沿的硬件锁存器事件（例如 SoftDrive）
7	硬件锁存器 1（负边沿），驱动器已定义	锁存器事件：驱动器中定义的具有负边沿的硬件锁存器事件（例如 SoftDrive）
20	应用程序（PLC 代码）	用户特定的引用实现（PLC 代码）：通过 ApplicationRequest 位向 PLC 发送用户请求信号

	: 锁存器事件					
编码器类型	**0：未定义**	**1：PLC 凸轮（负边沿）**	**2：硬件同步脉冲（零/C 轨）**	**3：带正边沿的外部硬件锁存器**	**4：带负边沿的外部硬件锁存器**	**5：软件同步脉冲（软件零轨）**
AX2xxx-B200 (Lightbus)	—	√	√	√	√	√（仅旋转变压器）
AX2xxx-B510 (CANopen)	—	√	—	—	—	√（仅旋转变压器）（请参见"参考掩码"参数）
AX2xxx-B1x0（EtherCAT）	—	√	√	√	√	√（仅旋转变压器）（固定 20 位）
AX2xxx-B900 （以太网）	—	√	√	√	√	√（仅旋转变压器）
Sercos	—	√	√	√（特定于 AX5xxx 执行）	√	√（请参见"参考掩码"参数）
Profi Drive	—	√	√	√	√	√
KL5101 IP5109	—	√	√	√	√	√
KL5111	—	√	√	—	—	√
KL5151	—	√	√	√	√	√（无意义）
IP5209	—	√	√	—	—	√（无意义）
CANopen （如 Lenze）	—	√	—	√（输入 E1）	√（输入 E2）	√（仅旋转变压器）（固定 16 位）
其他 类型	—	—	—	—	—	—

枚举归位传感器源

该参数设置参考凸轮（归位传感器）的数字输入源。同时确定信号是高电平有效还是低电平有效。

定义	参数文本	归位传感器源
0	默认：PLC 凸轮 (MC_Home)	参考凸轮由 PLC 提供。输入 MC_Home 功能块的 bCalibrationCam。
1	数字输入 1（高电平有效），设备相关映射	驱动器->输入->nState8.bit0 或 MDP703/733 设备的 E1 ，例如 7031,7041,7201,7411
2	数字输入 2（高电平有效），设备相关映射	驱动器->输入->nState8.bit1 或 MDP703/733 设备的 E2 ，例如 L7031,7041,7201,7411
3	数字输入 3（高电平有效）	驱动器->输入->nState8.bit2
4	数字输入 4（高电平有效）	驱动器->输入->nState8.bit3
5	数字输入 5（高电平有效）	驱动器->输入->nState8.bit4
6	数字输入 6（高电平有效）	驱动器->输入->nState8.bit5
7	数字输入 7（高电平有效）	驱动器->输入->nState8.bit6
8	数字输入 8（高电平有效）	驱动器->输入->nState8.bit7
9	数字输入 1（低电平有效），设备相关映射	驱动器->输入->nState8.bit2
10	数字输入 2（低电平有效），设备相关映射	驱动器->输入->nState8.bit0 或 MDP703/733 设备的 E1 ，例如 L7031、7041、7201、7411
11	数字输入 3（低电平有效）	驱动器->输入->nState8.bit1 或 MDP703/733 设备的 E2 ，例如 L7031、7041、7201、7411
12	数字输入 4（低电平有效）	驱动器->输入->nState8.bit2
13	数字输入 5（低电平有效）	驱动器->输入->nState8.bit3
14	数字输入 6（低电平有效）	驱动器->输入->nState8.bit4
15	数字输入 7（低电平有效）	驱动器->输入->nState8.bit5
16	数字输入 8（低电平有效）	驱动器->输入->nState8.bit6

数字输入 [1-8]

使用与 NC 程序相连的数字输入。为此，在过程映像（Drive->Inputs->nState8）中定义了一个带有 8 个数字输入的通用驱动器状态字节，可作为归位传感器的信号源。因此，必须手动将要使用的数字输入映射到该字节中的所需位置。

数字输入 1 和 2 可能因使用的硬件而异。对于 MDP703/733 硬件（例如 EL7031、EL7041、EL7201、EL7411），则使用端子模块的直接数字输入 E1 和 E2，它们位于端子模块 Drive.nState2 字节的第 3 位 (E1) 和第 4 位 (E2) 位置。在这种情况下，Drive.nState8 的两个低位不会分配。

枚举驱动器类型

定义	驱动器类型
0	未定义
1	模拟伺服驱动器：M2400 DAC 1 （模拟）
2	模拟伺服驱动器：M2400 DAC 2 （模拟）
3	模拟伺服驱动器：M2400 DAC 3 （模拟）
4	模拟伺服驱动器：M2400 DAC 4 （模拟）
5	MDP 252 驱动器：模拟伺服驱动器：KL4xxx、KL2502-30K （模拟）
6	MDP 252 驱动器：模拟伺服驱动器（非线性）：KL4xxx、KL2502-30K （模拟）
7	高速/低速驱动器 （数字）
8	步进电机驱动器 （数字）
9	SERCOS-Drive （数字）
10	MDP 510 驱动器：KL5051（BiSSI 接口） （数字）
11	AX2000 (Lightbus) （数字）
12	Provi-Drive MC（Simodrive 611U） （数字）
13	通用驱动器 （模拟）
14	NC 后面板 （模拟）
15	特殊 CANopen 类型（如 Lenze Drive 9300） （数字）
16	MDP 742 (DS402)：CANopen 和 EtherCAT CoE（AX2xx-B1x0/B510） （数字）
17	AX2xx-B900 驱动器（以太网） （数字）
20	KL2531/KL2541 编码器（步进电机） （数字）
21	KL2532/KL2542 编码器（DC 电机），KL2535/KL2545 编码器（PWM 电流端子） （数字）
22	TCom 驱动器（软驱） （数字）
23	MDP 733 驱动器：配置文件 MDP 733（EL7332、EL7342、EP7342） （数字）
24	MDP 703 驱动器：配置文件 MDP 703（EL7031、EL7041、EP7041） （数字）

枚举驱动器-输出-启动类型

定义	枚举驱动器-输出-启动类型
0	未定义
1	输出值（以百分比为单位）
2	输出为速度，例如 m/min

枚举驱动器运行模式

定义	驱动器运行模式（独立于驱动器的通用运行模式）
0	默认模式 （如果模式已知，则重新激活 NC 默认运行模式）
1（标准类型）	扭矩控制
2（标准类型）	速度控制，带反馈 1
3（标准类型）	速度控制，带反馈 2
4（标准类型）	位置控制，带反馈 1（滞后少）
5（标准类型）	位置控制，带反馈 2（滞后少）
6（CANopen/CoE 特定）	扭矩控制，带换相角
17（超采样类型）	使用动态容器进行扭矩控制
18（超采样类型）	速度控制，带1个反馈，使用动态容器
19（超采样类型）	速度控制，带2个反馈，使用动态容器
20（超采样类型）	位置控制，带1个反馈（滞后少），使用动态容器
21（超采样类型）	位置控制，带2个反馈（滞后少），使用动态容器
38（CANopen/CoE 特定）	IO 驱动器控制的归位模式（适用于第三方设备）
100（Sercos/SoE 特定）	Sercos/SoE 主要运行模式 0（参见 S-0-0032）
101（Sercos/SoE 特定）	Sercos/SoE 次要运行模式 1（参见 S-0-0033）
102（Sercos/SoE 特定）	Sercos/SoE 次要运行模式 2（参见 S-0-0034）
103（Sercos/SoE 特定）	Sercos/SoE 次要运行模式 3（参见 S-0-0035）
104（Sercos/SoE 特定）	Sercos/SoE 次要运行模式 4（参见 S-0-0284）
105（Sercos/SoE 特定）	Sercos/SoE 次要运行模式 5（参见 S-0-0285）
106（Sercos/SoE 特定）	Sercos/SoE 次要运行模式 6（参见 S-0-0286）
107（Sercos/SoE 特定）	Sercos/SoE 次要运行模式 7（参见 S-0-0287）

主轴的枚举运动阶段/运动状态

定义	运动阶段/运动状态（内部和外部设定点生成之间的区别）
内部设定点生成	
0	设定点发生器未激活 (INACTIVE)
1	设定点发生器激活 (RUNNING)
2	速率比为零（OVERRIDE_ZERO）
3	匀速 (PHASE_VELOCONST)
4	加速度阶段 (PHASE_ACCPOS)
5	减速度阶段 (PHASE_ACCNEG)
外部设定点生成：	
41	外部设定点生成激活 (EXTSETGEN_MODE1)
42	内部和外部设定点生成激活 (EXTSETGEN_MODE2)

从轴的枚举运动阶段/运动状态

定义	运动阶段/运动状态
0	从轴发生器未激活（INACTIVE）
11	从轴处于运动前阶段（PRE-PHASE）
12	从轴正在同步 (SYNCHRONIZING)
13	从轴已同步并同步运动 (SYNCHRON)

目前只适用于同步发生器类型的从轴

枚举表主要类型

定义	表主要类型
1	(n*m) 平板形凸轮表（凸轮系统）
10	(n*m) 特征曲线表（特性）（例如液压阀特征曲线） 仅支持非循环表格子类型（1、3）！
16	(n*m) "运动功能"表 (MF) 仅支持非等距表格子类型（3、4）！

枚举表格子类型

定义	表格子类型
1	(n*m) 主轴位置等距且主轴配置文件无循环延拓（等距线性）的表格
2	(n*m) 主轴位置等距且主轴配置文件无循环延拓（等距循环）的表格
3	(n*m) 主轴位置非等距但严格单调递增且主轴配置文件无循环延拓（单调线性）的表格
4	(n*m) 主轴位置非等距但严格单调递增且主轴配置文件循环延拓（单调循环）的表格

枚举表插值类型

定义	参考点之间的表插值类型
0	线性插值（NC_INTERPOLATIONTYPE_LINEAR）（标准）
1	4 点插值 (NC_INTERPOLATIONTYPE_4POINT)（仅适用于等距表类型）
2	所有参考点的三次样条插值（"全局样条"）（NC_INTERPOLATIONTYPE_SPLINE
3	通过 n 个插值点进行滑动三次样条插值（"局部样条"）(NC_INTERPOLATIONTYPE_SLIDINGSPLINE)

枚举表运行模式

定义	用于添加、交换和删除表的表运行模式
0	（默认）
1	加法 — 添加另一个表格
2	交换 — 用新表格替换现有表格
3	删除 — 删除现有表格

表格（凸轮）耦合信息的结构

表		（凸轮）耦合信息
nTableID;	1.	凸轮表 ID
nTableMainType;	2.	例如：CAMMING, CHARACTERISTIC, MOTIONFUNCTION
nTableSubType;	3.	例如：EQUIDIST_LINEAR、EQUIDIST_CYCLE、NONEQUIDIST_LINEAR、NONEQUIDIST_CYCLE
nInterpolationType;	4.	例如， LINEAR, 4POINT, SPLINE
nNumberOfRows;	5.	行/元素数量
nNumberOfColumns;	6.	列数
fMasterCamStartPos	7.	主轴凸轮启动位置（表中第一点）
fSlaveCamStartPos	8.	从轴凸轮启动位置（表中第一点）
fRawMasterPeriod;	9.	主轴周期/循环（原始值，未缩放）
fRawSlaveStroke;	10.	每个主轴周期/循环的从轴差（原始值，未缩放）
fMasterAxisCouplingPos	11.	当从轴与主轴耦合后，凸轮原点的总绝对主动偏移量
fSlaveAxisCouplingPos	12.	当从轴与主轴耦合后，凸轮原点的总绝对从动偏移量
nMasterAbsolute	13.	主轴绝对位置 (0/1)
nSlaveAbsolute	14.	从轴绝对位置 (0/1)
fMasterOffset;	15.	主轴总偏移
fSlaveOffset;	16.	从轴总偏移
fMasterScaling;	17.	主轴总缩放
fSlaveScaling;	18.	从轴总缩放
fSumOfSlaveStrokes	19.	到"fActualMasterAxisPos"为止的从轴冲程总和
fSumOfSuperpositionDistance	20.	叠加距离总和（位置补偿偏移）
fActualMasterAxisPos;	21.	实际主轴设定点（绝对）
fActualSlaveAxisPos;	22.	实际从轴设定点（绝对）
fActualMasterCamPos;	23.	主轴凸轮实际设定点
fActualSlaveCamPos;	24.	主轴凸轮实际设定点
nSlaveStateDWord	25.	从轴状态 DWORD（参见 AxisRef）
...	...	...

特征值的结构

特征值		
fMasterVeloNom;	1.	主轴额定速度（标准化： => 1.0）
fMasterPosStart;	2.	主轴启动位置
fSlavePosStart;	3.	从轴启动位置
fSlaveVeloStart;	4.	从轴启动速度
fSlaveAccStart;	5.	从轴启动加速度
fSlaveJerkStart;	6.	从轴启动加加速度
fMasterPosEnd;	7.	主轴结束位置
fSlavePosEnd;	8.	从轴结束位置
fSlaveVeloEnd;	9.	从轴结束速度
fSlaveAccEnd;	10.	从轴结束加速度
fSlaveJerkEnd;	11.	从轴结束加加速度
fMPosAtSPosMin;	12.	从轴最小位置时的主轴位置
fSlavePosMin;	13.	从轴最小位置
fMPosAtSVeloMin;	14.	从轴最小速度时的主轴位置
fSlaveVeloMin;	15.	从轴最小速度
fMPosAtSAccMin;	16.	从轴最小加速度时的主轴位置
fSlaveAccMin;	17.	从轴最小加速度
fSVeloAtSAccMin;	18.	从轴最小加速度时的从轴速度
fSlaveJerkMin;	19.	从轴最小加加速度
fSlaveDynMomMin;	20.	从轴最小动态动量（尚不支持！）
fMPosAtSPosMax;	21.	从轴最大位置时的主轴位置
fSlavePosMax;	22.	从轴最大位置
fMPosAtSVeloMax;	23.	从轴最大速度时的主轴位置
fSlaveVeloMax;	24.	从轴最大速度
fMPosAtSAccMax;	25.	从轴最大加速度时的主位置
fSlaveAccMax;	26.	从轴最大加速度
fSVeloAtSAccMax;	27.	从轴最大加速度时的从轴速度
fSlaveJerkMax;	28.	从轴最大加加速度
fSlaveDynMomMax;	29.	从轴最小动态动量（尚不支持！）
fSlaveVeloMean;	30.	从轴平均绝对速度
fSlaveAccEff;	31.	从轴有效加速度
nCamTableID;	32.	凸轮表 ID
nNumberOfRows;	33.	行数/条目数，例如点数
nNumberOfColums;	34.	列数（通常为 1 或 2）
nCamTableType;	35.	凸轮表类型（10=EQUIDIST、11=NONEQUIDIST、22=MOTIONFUNC、23=CHARACTERISTIC）
nPeriodic;	36.	线性或循环/周期
nReserved	37.	保留

枚举轴控制环路开关类型

定义	轴控制环路开关类型
0	未定义
1	简单切换（类似于轴复位） （标准）
2	通过控制器的 I/D 部分切换/同步至内部初始值（无加加速度/平滑）
3	通过控制器的 I/D 部分切换/同步至参数化初始值

3.2.2 AmsNAT

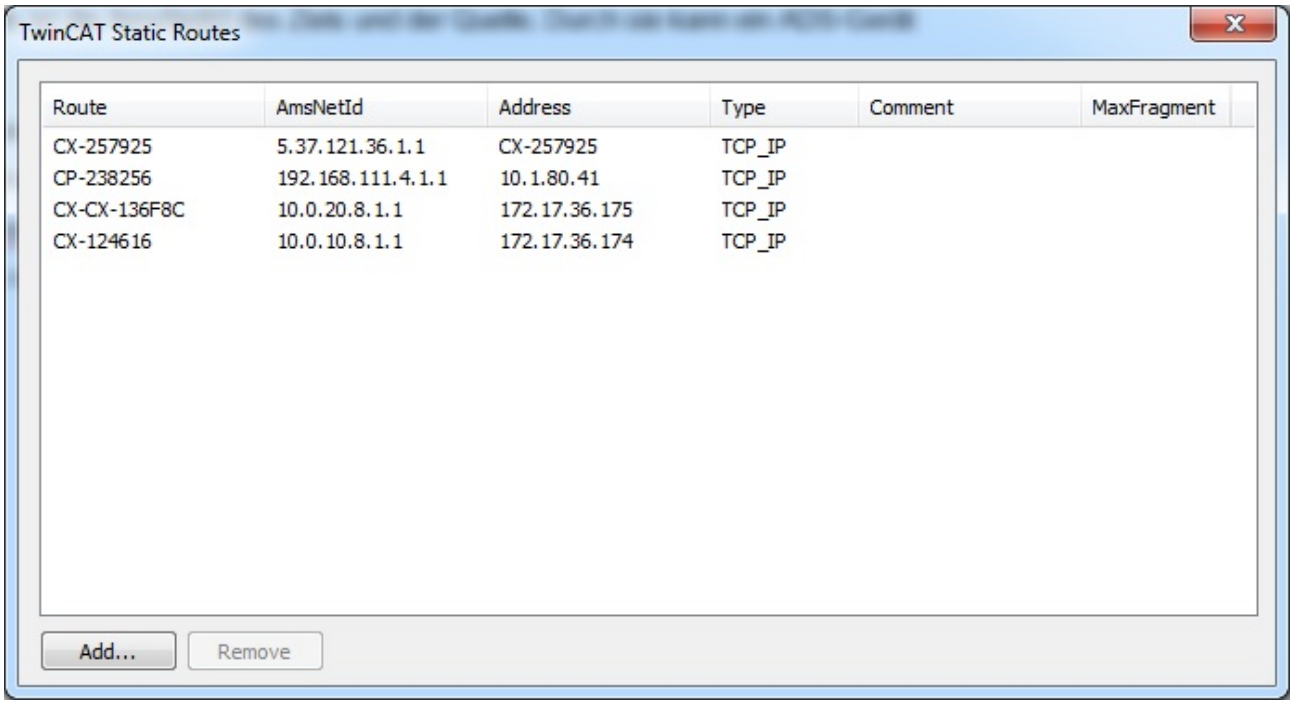
3.2.2.1 简介

为了更好地理解 AmsNAT 功能，了解 ADS 和 AMS 之间的区别以及什么是 ADS 路由非常重要。

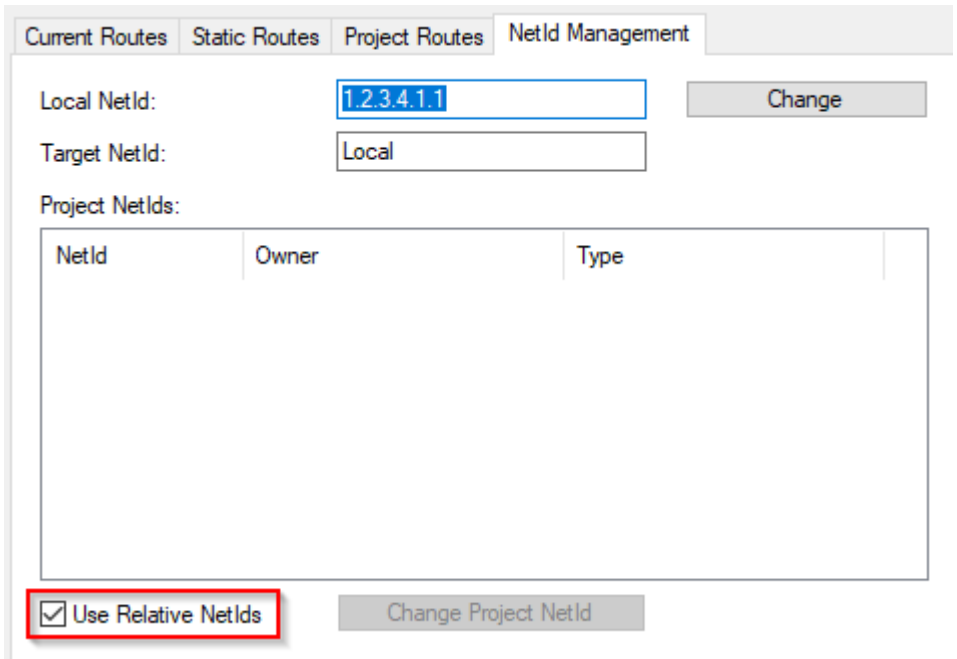
ADS（自动化设备规范）是 TwinCAT 的通信协议，规定了两个 ADS 设备之间的交互。例如，它定义了可以在另一台 ADS 设备上执行哪些操作、执行时需要哪些参数以及执行后发送哪些返回值。

AMS（自动化信息规范）规定了 ADS 数据的交换。通信协议的一个主要组成部分是 AmsNetId。这在源设备和目标设备的 AMS/ADS 包中指定。可以使用 AmsNetId 对 ADS 设备进行显式寻址。

必须在 TwinCAT 中设置两个设备之间的**路由**，这样它们才能进行通信。该路由在两端配置，通常包含路由名称、AmsNetId 和通信伙伴的地址以及连接类型。下图显示了 TwinCAT 系统中的新路由配置和现有路由概览。

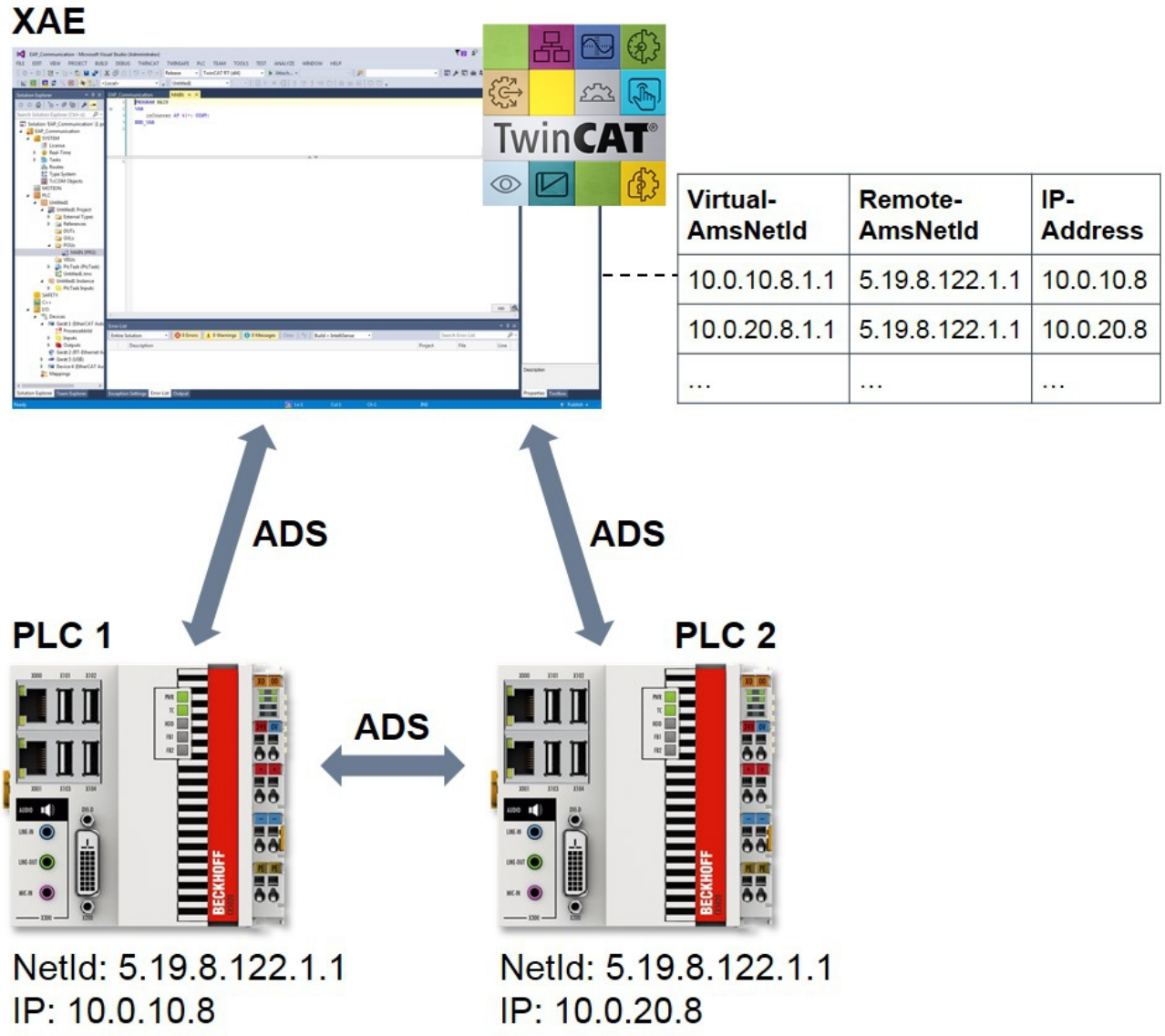


如果要扫描目标上的硬件，则必须使用相对的 NetID：



3.2.2.2 一般说明

AmsNAT 功能可使 XAE 系统建立通往具有相同 AmsNetId 的两个或多个控制器的路由（图 2）。除此之外，AmsNAT 还提供了一种解决方案，具有相同 AmsNetId 的不同 ADS 设备可以通过 ADS 相互通信。虚拟 AmsNetIds 与 AmsNAT 一起使用。虚拟 AmsNetId 是连接 ADS 设备的唯一地址，在通信过程中会被目标系统的真实 AmsNetId 取代。这意味着 AmsNAT 功能可确保在通过 ADS 进行的所有通信中替换目标系统的 AmsNetId。



附图 1: 与具有相同 NetId 的 TwinCAT 系统/在 TwinCAT 系统之间进行通信

3.2.2.3 动力

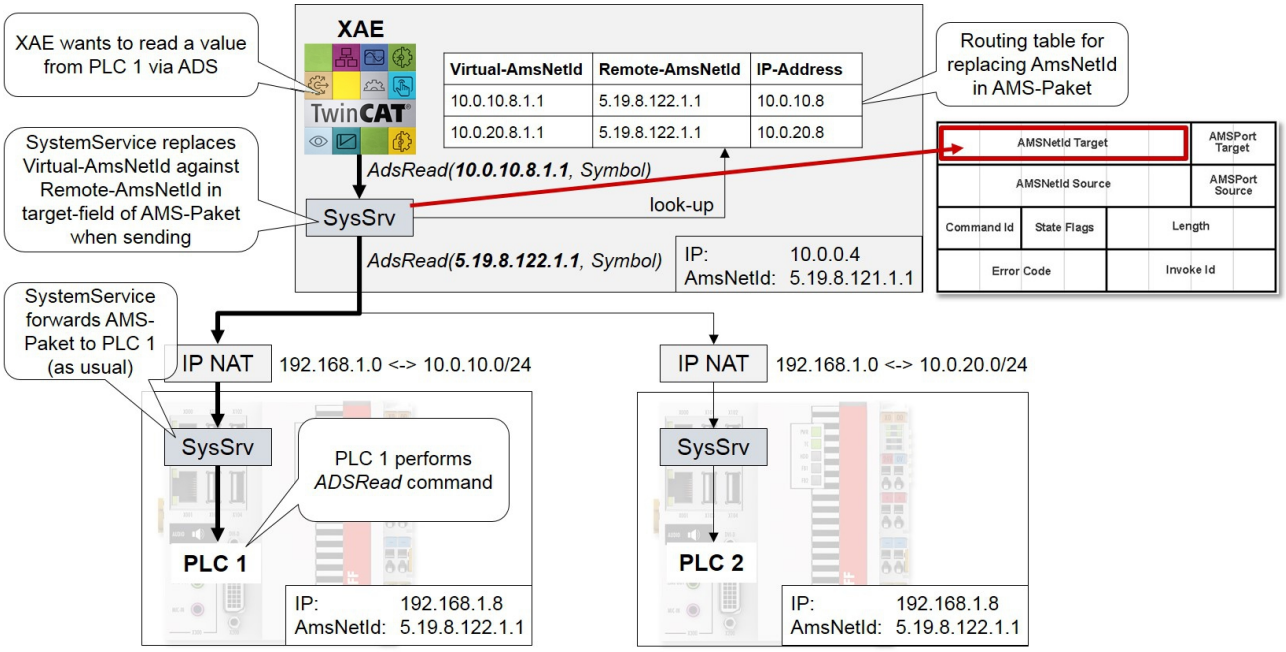
在系列机械工程中，一个常见的应用是克隆（即制作 1:1 的副本）控制器。在使用 TwinCAT 时，这样做的结果是所有克隆实例都拥有相同的 AmsNetId。起初这不是问题。但是，如果克隆实例要与同一工程系统并行连接，或者要通过 ADS 相互通信，这起初是不可能的，因为 AmsNetId 并非唯一。AmsNAT 功能恰恰消除了这一限制，因为系统使用的是虚拟 AmsNetIds。轻松即可完成配置。

AmsNAT 功能可用于 ADS 设备的任何路由。这提供了高度灵活性，AmsNetIds 不再需要根据机器计算机调整，从而大大减少了配置所需的时间和工作。

3.2.2.4 运作

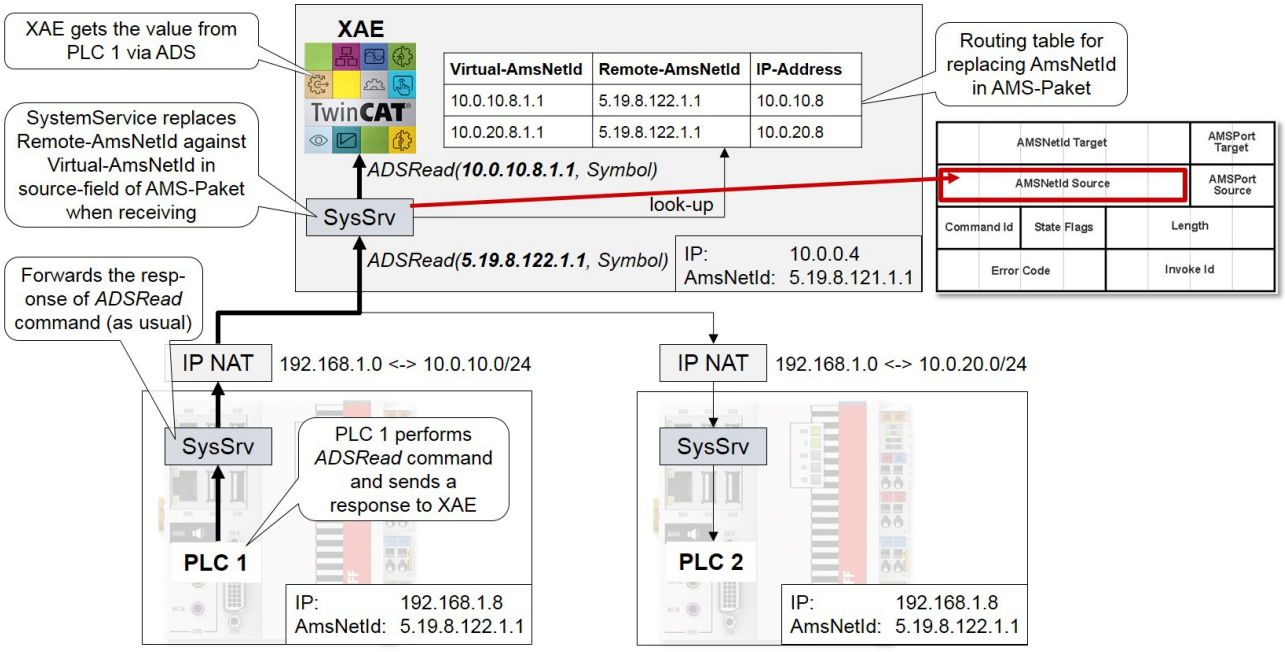
下面将根据一个典型的应用来解释 AmsNAT 的功能。在应用案例中，一个 TwinCAT 工程系统和两个 TwinCAT 运行时存在相同的 AmdNetId 和 IP 地址。配置如图 3 所示。工程系统将向 PLC 1 发送 AdsRead 命令，并期望得到相应响应。由于两个运行时都拥有一个相同的 IP 地址，因此额外使用了两个 IP NAT。它们的任务是实现明确无误寻址。为此，根据通信方向，本地 IP 地址的前三个位置将被全局 IP 地址的前三个位置取代，反之亦然。

在应用示例的第一步，工程系统向 PLC 1 发送 AdsRead 命令。由于该 AmdNetId 是虚拟的，因此 TwinCAT 系统服务会借助其路由表将其替换为远程 AmsNetId 5.19.8.122.1.1。这是系统中存在的真实 AmsNetId。它被输入 AMS 包的"AmsNetId 目标"字段。



附图 2: 使用 AmsNAT 发送 AdsRead 命令的序列

PLC 1 的 TwinCAT 系统服务不加修改地分程传递 AMS 包。PLC 1 执行 AdsRead 命令，然后向工程系统发送相应的响应。图 4 显示了响应的通信序列。



附图 3: 向 AdsRead 命令发送响应的序列

对于响应，PLC 1 的 TwinCAT 系统服务首先会原封不动地转发这个 AMS 数据包。AMS 数据包会继续传递，最终到达工程系统的 TwinCAT 系统服务。由于在 AMS 数据包的"AmsNetId 来源"字段中输入了 PLC 1 的真实 AmsNetId，因此必须根据路由表将其替换为虚拟 AmsNetId。然后，工程系统就可以明确分配和处理响应了。

使用 AmsNAT 功能时，传输的数据不会改变，只会改变 AMS 报头。因此应注意，如果配置数据包含 AmsNetId，这可能会导致使用虚拟 AmsNetId。可以使用相对 AmsNetId 来设计 I/O 设备工程。在这种情况下，AmsNetId 的最后两个字符会被考虑在内，而前四个字符会被忽略。

3.2.2.5 配置

要配置 AmsNAT，请打开文件 StaticRoutes.xml，该文件位于 TwinCAT 安装目录的 TwinCAT\3.1\Target 路径下。在该文件中，如下所示为每个路由定义属性"RemoteNetId"。

```
<?xml version="1.0" encoding="UTF-8"?>
<TcConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="http://www.beckhoff.com/schemas/2015/12/TcConfig">
   <RemoteConnections>
      <Route>
         <Name>CX-111111</Name>
         <Address>10.0.10.8</Address>
         <NetId RemoteNetId="5.19.8.122.1.1">10.0.10.8.1.1</NetId>
         <Type>TCP_IP</Type>
      </Route>
      <Route>
         <Name>CX-222222</Name>
         <Address>10.0.20.8</Address>
         <NetId RemoteNetId="5.19.8.122.1.1">10.0.20.8.1.1</NetId>
         <Type>TCP_IP</Type>
      </Route>
   </RemoteConnections>
</TcConfig>
```

分配给远程 ADS 设备的实际 AmsNetId 用属性"RemoteNetId"指定。不一定要具有唯一性。在配置了 AmsNAT 功能的 TwinCAT 系统中，只知道在 <NetId> 字段中定义的目标系统的 AmsNetId。

重新启动 TwinCAT 系统服务，以激活 AmsNAT 功能的预设配置。为此，请将 TwinCAT 系统从运行模式切换到配置模式。如果 TwinCAT 已进入配置模式，请重新打开以加载所做设置。

3.2.3 ADS-over-MQTT

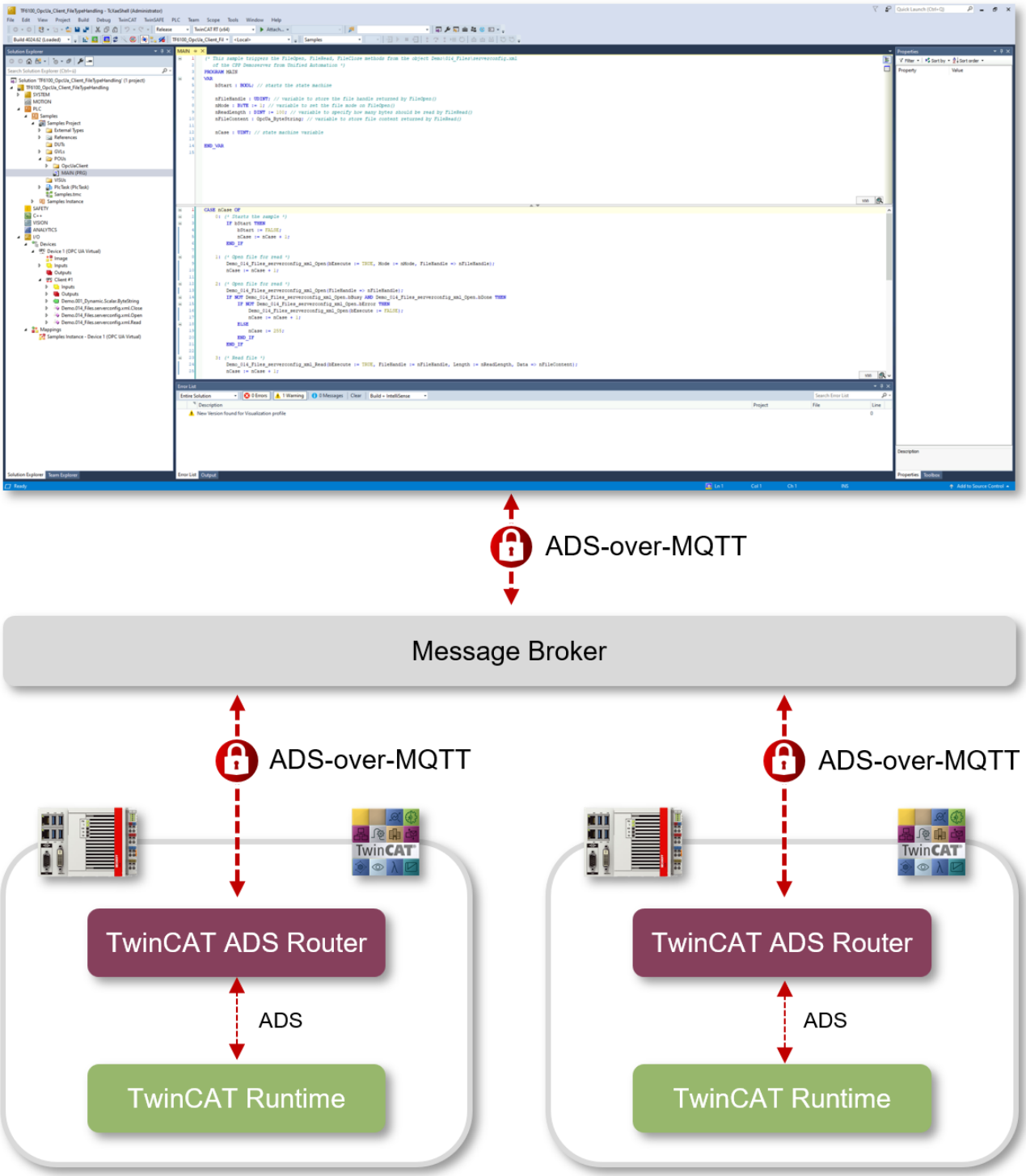
3.2.3.1 概述

倍福 ADS（自动化设备规范）是倍福为工业自动化系统中的高效数据交换开发的通信协议。它是将设备和软件集成到倍福基于 PC 的控制技术中的重要组成部分。

从 ADS 协议的角度来看，ADS-over-MQTT 是一个额外的传输通道，可用于传输 ADS。通过 MQTT 消息代理对通信进行解耦可带来许多优势，尤其是在集成其他 ADS 应用程序时，其扩展性和灵活性方面表现尤为突出。在传输层可采用 TLS 等安全机制来确保通信连接的安全性。

通过 ADS-over-MQTT，整个数据交换对 ADS 应用程序而言是透明的，因为 ADS 路由器只需感知并维护 MQTT 传输通道的相关信息。特别是，这也使现有应用程序可以更轻松地实现改造升级。

ADS-over-MQTT 的主要典型用例是远程维护和远程诊断场景，即 TwinCAT 工程环境（TwinCAT XAE）需要与一个或多个控制器连接，以进行远程调试。下图说明了此处创建的架构。



不过，ADS-over-MQTT 还有许多其他用例，尤其在整合多个分布式 PLC 系统时更具优势。

本文件概述了使用可能性，并从技术上说明了如何通过 MQTT 消息代理配置“虚拟 ADS 网络”。

基于 MQTT 的 ADS 网络的优势

- **子网、基于 NAT 的网络和防火墙：**
  在传统 ADS 设置中，传入 TCP/IP 连接在两个方向上均被使用。因此，在正常情况下，设备必须定位在同一网络中。在具有不同子网的分布式系统中，这需要复杂配置，以便使 ADS 路由可用。在基于 MQTT 的 ADS 网络中，设备只使用一个传出 TCP/IP 连接。这样，上层网络中的代理就可以在所有设备之间进行代理。由于传出连接，因此可以使用一般防火墙，无需注册传入端口。
- **访问控制：**
  创建适当的路由后，就可以在传统 ADS 设置中执行双向通信。设备 A 对设备 B 的访问也允许设备 B 访问设备 A。基于 MQTT 的 ADS 网络可以配置为设备 A 可以访问 B，但不能反向配置。

- **安全/加密：**
  TwinCAT 与代理之间的通信可以通过 TLS（使用证书或 PreSharedKey (PSK)）加密。在这种情况下，传输 MQTT 协议是加密的，因此 ADS 协议可以在载荷中以不加密的形式传输。
- **改造：**
  ADS-over-MQTT 对 ADS 应用程序而言是透明的，这意味着无需对其进行更改。

注意

ADS 访问意味着完全访问

正如安全公告 2017-01 中所述，ADS 提供对设备的完全访问。
安全 ADS 为通信提供授权和加密；因此，它代表了一种传输加密。因此，如果存在 ADS 路由，那么就存在完全访问。
OPC UA 等解决方案可提供专用的、与角色相关的单个文件访问权限。

3.2.3.2 安装

3.2.3.2.1 系统要求

以下系统要求适用于本产品的安装和操作。

技术数据	描述
操作系统	Windows 10 Windows Server 2022 TwinCAT/BSD TwinCAT/Linux®
目标平台	PC 架构（x86、x64、Arm®）
TwinCAT 版本	TwinCAT 3（版本 4022.0 及以上）
TwinCAT 安装级别	TwinCAT 3 XAE、XAR、ADS
所需 TwinCAT 授权	---
支持 TwinCAT 3 Usermode Runtime	是，另请参见 配置文件 [▶ 166]

MQTT 消息代理

要使用该产品，需要一个 MQTT 消息代理，通过它可以建立连接。任何 MQTT 消息代理均可使用，如 Mosquitto 或 HiveMQ。此外，还可以使用托管云服务，如 AWS IoT Core。

Mosquitto 消息代理插件

目前，所提供的 Mosquitto 消息代理插件仅适用于 Windows 操作系统。

3.2.3.2.2 安装

ADS-over-MQTT 功能是 TwinCAT 基础安装的固定组成部分，无论是 XAE、XAR 还是 ADS 安装环境均默认包含，无需在 TwinCAT 端进一步安装。要安装 MQTT 消息代理，请查阅您的消息代理软件文档。

3.2.3.3 技术简介

3.2.3.3.1 快速入门

本文档文章旨在帮助您快速入门，初步了解如何使用本产品。按照以下步骤操作，即可建立与 MQTT 消息代理的 ADS-over-MQTT 连接，并发送和接收 ADS 消息。

安装消息代理

本文档假设您已在本地安装并运行 MQTT 消息代理。例如，我们在此处使用的是 Mosquitto 消息代理，不过，您可以使用任何消息代理。

概述

为了进行功能演示，两个 TwinCAT 设备要与消息代理建立 ADS-over-MQTT 连接。然后，我们在第一个设备上使用 TwinCAT XAE（工程环境），通过 ADS-over-MQTT 路由与第二个系统建立连接。为了使安装方案尽可能简单，两个设备应处在同一网络上。

设备 1：

- Mosquitto 消息代理
- TwinCAT 3 XAE

设备 2：

- TwinCAT 3 XAE 或 XAR

请注意设备 1 的 IP 地址或主机名。在后一种情况下，请确保您的网络中的域名解析功能正常。

消息代理设置

自 2.x 版本起，Mosquitto 消息代理默认提供的配置需设置安全措施，以确保消息代理的安全运行。在下文中，我们将向您展示如何修改 Mosquitto 消息代理配置，以便与该代理建立非安全通信连接。不过，此操作仅限在可信操作环境中用于测试目的。对于生产环境操作，我们建议采用安全的消息代理配置。

1. 在您的系统上安装 Mosquitto 消息代理。
2. 备份 Mosquitto 安装目录中的 mosquitto.conf 文件。它通常位于 C:\Program Files\mosquitto 下。
3. 使用您所选的文本编辑器打开 mosquitto.conf 文件，并删除现有内容。添加以下内容，并保存文件。

```
listener 1883
allow_anonymous true
```

4. 通过相应的 Windows 服务重启 Mosquitto 消息代理，或者通过控制台或 mosquitto.exe 进行手动重启。

⇨ 现在，您已将 Mosquitto 消息代理配置为在端口 1883 上监听传入的客户端连接，并且无需任何安全措施（既不需要用户身份验证，也不需要客户端证书）。

设备 1 上的 Windows 防火墙

请在设备 1 的 Windows 防火墙中启用标准 MQTT 端口 1883/tcp 作为传入端口，以便设备 2 可以与消息代理建立连接。

现在，您可以开始配置 TwinCAT。

配置第一个 TwinCAT 设备

现在，该设备上的 TwinCAT ADS 路由器应建立一条通往本地消息代理的路由。创建一个新的 XML 文件，例如使用记事本进行创建，并插入以下内容。

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<TcConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="http://www.beckhoff.com/schemas/2015/12/TcConfig">
<RemoteConnections>
  <Mqtt>
    <Address Port="1883">127.0.0.1</Address>
    <Topic>VirtualAmsNetwork1</Topic>
    </Mqtt>
  </RemoteConnections>
</TcConfig>
```

将此 XML 文件以任何名称保存在以下目录中，并重新启动 TwinCAT，以便更改生效。

```
\TwinCAT\3.1\Target\Routes
```

配置第二个 TwinCAT 设备

该设备上的 TwinCAT ADS 路由器应建立一条通往设备 1 消息代理的路由。创建一个新的 XML 文件，例如使用记事本进行创建，并插入以下内容。

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<TcConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="http://www.beckhoff.com/schemas/2015/12/TcConfig">
<RemoteConnections>
  <Mqtt>
    <Address Port="1883">%IPAddress%</Address>
    <Topic>VirtualAmsNetwork1</Topic>
    </Mqtt>
  </RemoteConnections>
</TcConfig>
```

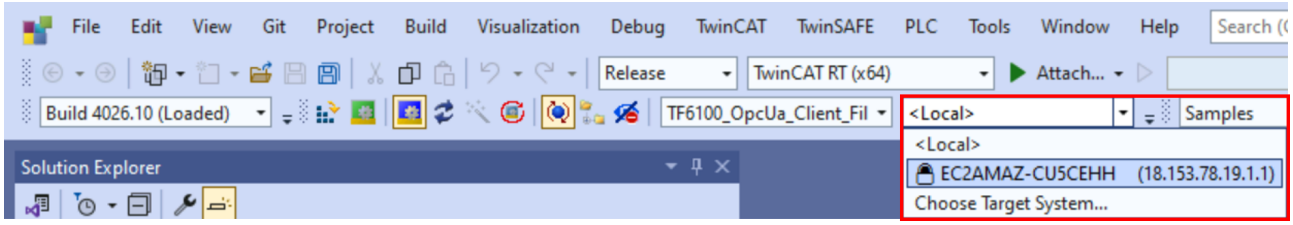
将 %IPAddress% 条目替换为设备 1 的 IP 地址或主机名。将此 XML 文件以任何名称保存在以下目录中，并重新启动 TwinCAT，以便更改生效。

```
\TwinCAT\3.1\Target\Routes
```

连接工程环境

现在，在设备 1 上启动 TwinCAT XAE，并创建一个新的 TwinCAT 项目。或者，您也可以打开一个现有项目。在本教程中，我们只需使用工具栏与远程 ADS 设备建立连接。

现在，您可以在工具栏中找到 TwinCAT 设备 2，并与其建立连接。

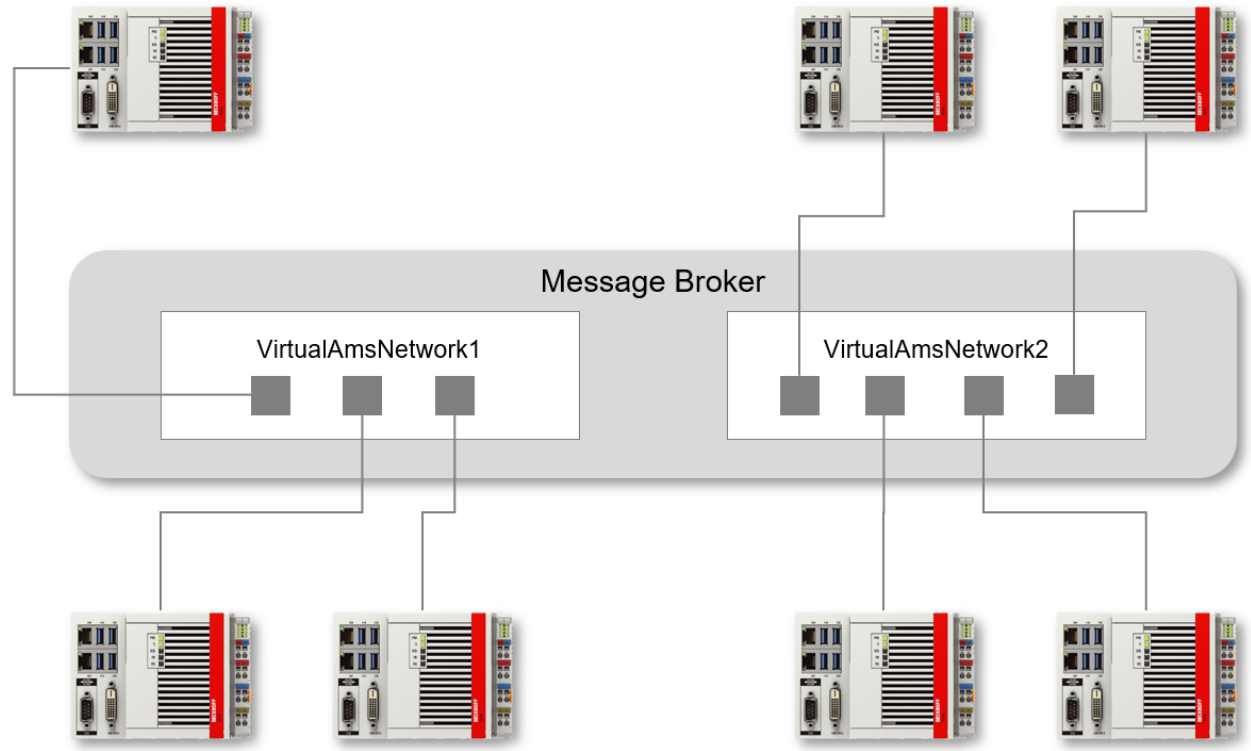


后续步骤

在成功建立与消息代理的 ADS-over-MQTT 连接后，我们建议您将 Mosquitto 消息代理重置为默认设置。为此，请备份您在前面步骤中创建的 mosquitto.conf 文件。该文件与代理文档共同构成了后续操作的良好基础，例如，可通过配置客户端/服务器证书和用户身份验证，建立安全的消息代理运行环境。

3.2.3.3.2 虚拟网络

在配置 ADS-over-MQTT 时，可以定义所谓的“虚拟网络”。这样可以将 ADS 设备部署到独立的网络中。只有同一网络内的设备才能相互通信。在 MQTT 层面上，这种分发部署以共同的基本主题为基础。



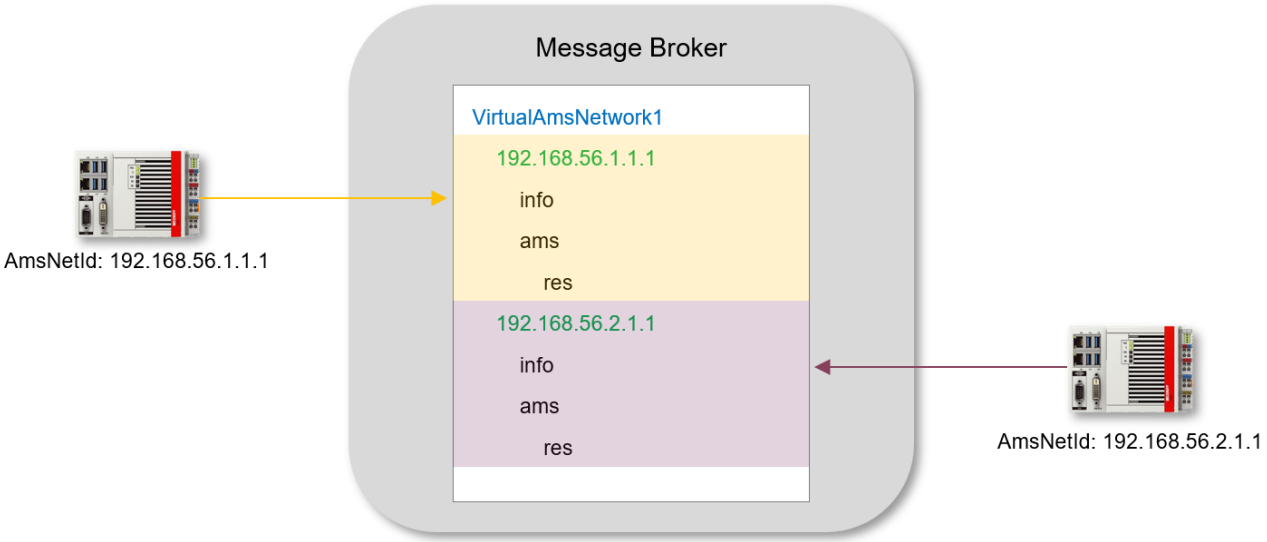
使用 ADS-over-MQTT 配置文件 [▶ 166] 在客户端层面可以定义这些虚拟网络，如有需要，还可使用 Mosquitto 消息代理的 TcMqttPlugin [▶ 169] 分配访问权限。

3.2.3.3.3 通信流

为了实现通过 ADS-over-MQTT 的数据交换与设备发现功能，所有 ADS 设备需在消息代理上采用统一的主题结构。主题结构取决于配置的虚拟网络 [▶ 164]名称和相应设备的 AMS Net ID。

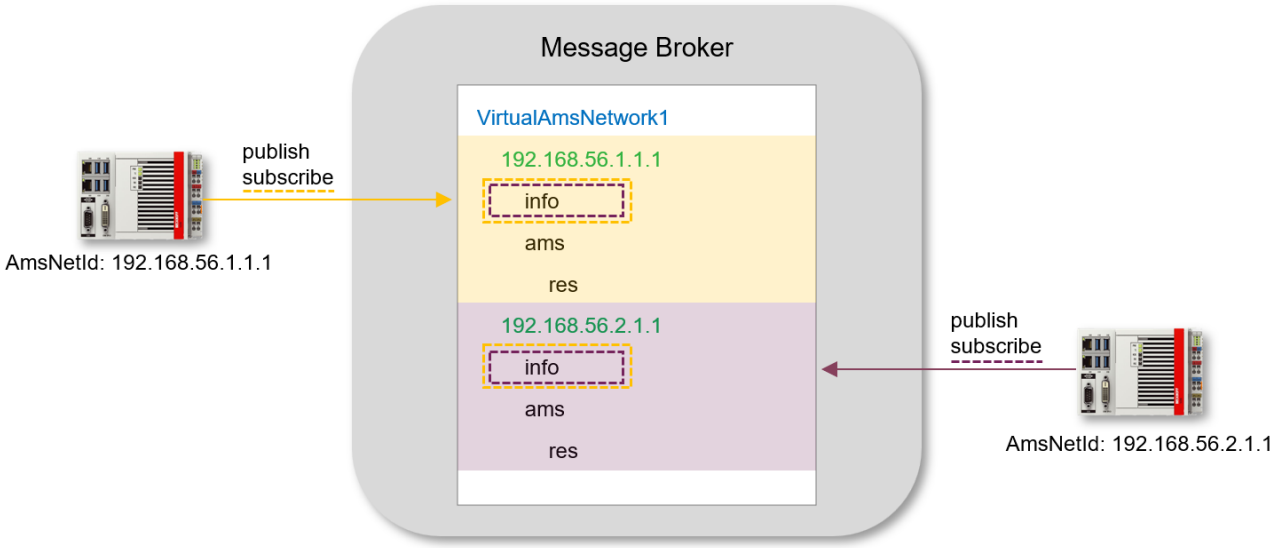
类型	主题
发现	<NetworkName>/<AmsNetId>/info
通信	<NetworkName>/<AmsNetId>/ams <NetworkName>/<AmsNetId>/ams/res

因此，每个 ADS 设备在消息代理上都有自己的“主题区域”。下图说明了这种关系。



发现

连接的 TwinCAT ADS 路由器向其发现主题发送包含设备信息的保留消息，同时订阅主题 <NetworkName>/+/info，以便了解所有其他连接路由器的情况。



发送至发现主题的消息包含一个 XML 结构，其中包含设备信息，例如，主机名和使用的 TwinCAT 版本：

```
<info>
  <online name="EC2AMAZ-2RRSQS6" osVersion="10.0.20348" osPlatform="2"
tcVersion="3.1.4026.10">true</online>
</info>
```

如果消息代理不支持保留消息，则可在 ADS-over-MQTT 配置文件 [▶ 166]中加以考虑。在这种情况下将执行通信握手而非保留消息：新连接的设备会向自身发现主题注册，所有其他连接设备则通过各自的发现主题响应其他消息。这样可以确保即使使用不支持保留消息的消息代理，设备也能找到彼此。其劣势在于，在需要频繁重连的大型操作环境中，消息量会显著增加。

通信

TwinCAT ADS 路由器在建立连接后会立即订阅其通信主题（<NetworkName>/<AmsNetId>/ams/#）。发送至该路由器的 ADS 命令将传输至 <NetworkName>/<AmsNetId>/ams，而响应则通过<NetworkName>/<AmsNetId>/ams/res 主题接收。

3.2.3.3.4 配置文件

TwinCAT ADS 路由器通过 XML 文件进行配置，以便与消息代理建立 ADS-over-MQTT 连接。该配置文件以任何名称存储在以下目录中：

```
\TwinCAT\3.1\Target\Routes
```

在示例 [▶ 171]一章中，您可以找到适用于下述所有用例的示例配置文件。

激活新的 ADS-over-MQTT 配置

只有在 TwinCAT ADS 路由器初始化时，新的或更改的 ADS-over-MQTT 配置才会被采用。例如，在 TwinCAT 从 RUN 到 CONFIG 或 CONFIG 到 CONFIG 的转换过程中就会发生这种情况。

路径信息

请确保配置文件中的任何路径详情均使用了操作系统的正确拼写。

TwinCAT 3 Usermode Runtime

您可以在 TwinCAT 3 Usermode Runtime 中操作 ADS-over-MQTT。您只需考虑 Usermode Runtime 的基本路径。以交付状态为例，路由器配置文件的基本目录定义如下：

%ProgramData%\Beckhoff\TwinCAT\3.1\Runtimes\UmRT_Default\3.1\Target\Routes\

除此之外，ADS-over-MQTT 特定主题的其他路径详情（参见 技术简介 [▶ 162]）将予以保留。

基本配置

基本配置始终包含以下要素：
消息代理的地址、其 TCP 端口和虚拟网络 [▶ 164]的名称。
通过 <Address> 节点可以指定代理的地址及其可以访问的 TCP 端口。虚拟网络通过 <Topic> 节点进行定义。
在下面的示例中，与本地（127.0.0.1）安装的消息代理和 TCP 端口 1883 建立了连接。
“VirtualAmsNetwork1”被定义为虚拟网络。

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<TcConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="http://www.beckhoff.com/schemas/2015/12/TcConfig">
 <RemoteConnections>
  <Mqtt>
   <Address Port="1883">127.0.0.1</Address>
   <Topic>VirtualAmsNetwork1</Topic>
  </Mqtt>
 </RemoteConnections>
</TcConfig>
```

NoRetain

设备发现的通信流 [▶ 165]可以通过 NoRetain 属性进行自定义。当设置 NoRetain = true 时，设备搜索功能不再基于保留消息。而是采用握手机制来识别所有连接的 ADS 设备。

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<TcConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="http://www.beckhoff.com/schemas/2015/12/TcConfig">
<RemoteConnections>
    <Mqtt NoRetain="true">
        <Address Port="1883">mybroker.someurl.com</Address>
        <Topic>VirtualAmsNetwork1</Topic>
    </Mqtt>
</RemoteConnections>
</TcConfig>
```

单向

通过单向属性可以阻止传入的 ADS 消息进入本系统。一个典型的用例可以是工程系统，例如，它应该能够通过 ADS 连接到运行时系统，但反之则不成立。

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<TcConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="http://www.beckhoff.com/schemas/2015/12/TcConfig">
<RemoteConnections>
    <Mqtt Unidirectional="true">
        <Address Port="1883">mybroker.someurl.com</Address>
        <Topic>VirtualAmsNetwork1</Topic>
    </Mqtt>
</RemoteConnections>
</TcConfig>
```

TLS

<TLS> 节点可用于定义通过 TLS 确保传输通道安全的设置。可提供各种连接选项，例如，配置客户端证书或 PSK。我们的示例 [▶ 171]介绍了所有可能的配置方案。

TLS IgnoreCn

IgnoreCn 属性可用于禁止验证服务器证书中的 CommonName（CN）。

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<TcConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="http://www.beckhoff.com/schemas/2015/12/TcConfig">
<RemoteConnections>
    <Mqtt>
        <Address Port="8883">mybroker.someurl.com</Address>
        <Topic>VirtualAmsNetwork1</Topic>
        <Tls IgnoreCn="false">
            <Ca>C:\TwinCAT\3.1\Target\Certificates\mybroker\CA.pem</Ca>
        </Tls>
    </Mqtt>
</RemoteConnections>
</TcConfig>
```

TLS PSK

使用带有预共享密钥（PSK）的 TLS 时，您可以将 PSK 指定为十六进制编码的 64 个字符字符串，也可以使用 Sha256（Identity + Pwd）在 TwinCAT 内部进行转换。在后一种情况下，您可以使用 IdentityCaseSensitive 属性来指定 TwinCAT 在计算时应将身份转换为 UpperCase。我们的示例 [▶ 171]介绍了两种可能的配置方案。

用户

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<TcConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="http://www.beckhoff.com/schemas/2015/12/TcConfig">
 <RemoteConnections>
  <Mqtt>
   <Address Port="1883">127.0.0.1</Address>
   <Topic>VirtualAmsNetwork1</Topic>
   <User>CX-12345</User>
  </Mqtt>
 </RemoteConnections>
</TcConfig>
```

<User> 元素指定了一个身份，可在 TcMqttPlugin [▶ 169] 中用于配置 ADS 设备之间的访问权限。不过，身份通常是通过其他方式定义的，例如，客户端证书的 CommonName（CN）。

作为选项，<Mqtt> 元素可以包含一个 ClientId 属性，以指定 MQTT ClientID。否则，它将由 <User> 和任意字符串组成。

3.2.3.3.5 启用/禁用

您可以在运行时启用或禁用 ADS-over-MQTT 路由，而无需重新启动 TwinCAT 系统。例如，您可以仅在检修时启用路由，并将此功能分配给按键开关，或通过 HMI 将其与用户输入连接起来。

通过 TwinCAT 系统服务中的 ADS 接口执行启用/禁用功能。您可以在路由配置文件中使用特殊属性来定义在启动 TwinCAT 后路由应该具有的默认状态。

路由配置文件

在 ADS-over-MQTT 路由的配置文件中，您可以指定在启动 TwinCAT 后路由应该具有的默认状态。该属性（“已禁用”）在 <Mqtt> 节点中指定。如果赋值为“true”，则启动 TwinCAT 后不会自动建立 ADS-over-MQTT 路由，必须通过 ADS 接口才能建立该路由。

```
<?xml version="1.0"?>
<TcConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:noNamespaceSchemaLocation="http://www.beckhoff.com/schemas/2015/12/TcConfig">
    <RemoteConnections>
        <Mqtt Disabled="true">
            <Name>MyBroker</Name>
            <Address Port="1883">myMessageBrokerAddress</Address>
            <Topic>VirtualAmsNetwork1</Topic>
        </Mqtt>
    </RemoteConnections>
</TcConfig>
```

ADS 接口

ADS 接口定义如下。

```
ADS-Kommando: AdsWriteRequest
ADS-Port: 10000
IndexGroup: 808
IndexOffset: siehe unten
Data: Identifizierung der Route, siehe unten
```

IndexOffset 指明您想要永久还是暂时启用/禁用某个路由。在这种情况下，“永久”意味着即使在 TwinCAT 重新启动后也会保留启用/禁用状态。提供以下选项：

IndexOffset	名称	描述
1	ADS_ROUTE_DISABLE	永久禁用路由。
2	ADS_ROUTE_ENABLE	永久启用路由。
3	ADS_ROUTE_DISABLE_TMP	暂时禁用路由。
4	ADS_ROUTE_ENABLE_TMP	暂时启用路由。

AdsWriteRequest 的数据区指定了要寻址的 ADS-over-MQTT 路由。这是通过在字符串中定义来实现的。格式如下，相当于配置文件的 <Address> 和 <Topic> 节点的组合。此处固定了关键词“MQTT”。

```
MQTT:<Address>:<Topic>
```

或者，您也可以在配置文件中添加 <name> 节点，然后使用该节点进行引用。此处使用的名称在所有 ADS 路由中必须是唯一的。在上述示例中，您可以在 AdsWriteRequest 的数据区中使用以下字符串：

```
MQTT:MyBroker
```

PLC 示例

相应的示例 [▶ 171]可供下载。

3.2.3.3.6 安全

确保通信安全有多种选择。为此，可以使用基于 X.509 证书的 TLS 连接或预共享密钥（PSK）。建议使用 TLS 确保通信安全，尤其是通过不可信网络（如互联网）进行通信时。在 配置文件 [▶ 166] 和示例 [▶ 171]两章中，您可以找到通过 TLS 运行 ADS-over-MQTT 的说明和示例配置文件。

代理本身必须在可信环境中运行，这是因为代理上的所有消息都没有进行加密或者安全保护。

入侵虚拟 ADS 网络

即使设备和代理之间的通信是通过 TLS 加密进行的，设备之间也没有安全保障。ADS 命令以未加密形式存在于代理上。
如果设备被入侵，攻击者可以通过获得的权限执行所有 ADS 命令。这些命令还包括文件读取操作或启动进程的操作。

采用两种方法可以在单独的 ADS 设备之间配置访问权限：

- 通过访问控制列表 [▶ 170]配置访问权限（以 Mosquitto 为例）
- 通过插件 [▶ 169]配置访问权限（仅适用于 Mosquitto）

3.2.3.3.6.1 Mosquitto 插件

一个专门为 Mosquitto 消息代理开发的插件，以便在各个 TwinCAT ADS 路由器之间定义访问权限。
另请注意操作该插件的系统要求 [▶ 162]。

该插件随 TwinCAT 3.1 版本 4024 下的 TwinCAT 安装一起交付。版本 4026 需要安装相应的软件包（TwinCAT.XAE.MqttPlugin）。该插件安装在以下目录中，可在 Mosquitto 配置中引用。

```
\TwinCAT\AdsApi\TcMqttPlugin
```

该插件有 32 位和 64 位版本，具体取决于您使用的 Mosquitto 消息代理的版本。然后，该插件在 Mosquitto 消息代理配置中的集成方式如下：

```
auth_plugin <Path>TcMqttPlugin.dll
auth_opt_xml_file <Path>MyACL.xml
```

MyACL.xml 文件包含对代理本身的访问配置，以及连接的 TwinCAT ADS 路由器之间的通信配置。该配置将在下一章节中进行详细介绍。

配置

插件提供配置虚拟 AMS 网络的选项。为此，要为每个目标设备指定哪个设备可以访问哪个其他设备。与传统的 ADS 路由不同，这些连接是定向的：因此，目标也无权访问源。

```
<TcMqttAclConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="C:\TwinCAT\3.1\Config\Modules\TcMqttAclConfig.xsd">
  <Ams>
    <Topic>VirtualAmsNetwork1</Topic>
    <User>
      <Name>EngineeringStation</Name>
    </User>
    <User>
      <Name>CX-12345</Name>
      <Access>EngineeringStation</Access>
    </User>
    <User>
      <Name>CX-56789</Name>
      <Access>EngineeringStation</Access>
    </User>
  </Ams>
</TcMqttAclConfig>
```

Ams 网络的名称在 <Ams> 节点中定义。它用于识别网络使用的 MQTT 主题。单个 <User> 元素描述设备。这些元素有一个 <Name> 属性，用于描述与之建立连接的 MQTT 身份。该身份可以通过各种 TLS 机制进行传输，例如，通过 TLS-PSK 身份标识或客户端证书的 CommonName（CN）。我们此处的示例 [▶ 171]介绍了可能的配置方案。

具有访问权限的设备通过 <Access> 元素定义。在上述示例中，“EngineeringStation”身份可以访问两个 CX 设备，但 CX 设备既不能访问“EngineeringStation”，也不能相互访问。

配置文件会循环重载，因此无需重新启动代理。

对要注册的 AmsNetId 的限制

通过这种配置，每个有效连接的设备都可以拥有一个任意的 AmsNetId，因此从 ADS 的角度看也就拥有了一个身份。可根据需要进一步限制：

```
<User>
  <Name>CX-56789</Name>
  <Access>EngineeringStation</Access>
  <NetId>192.168.56.1.1.1</NetId>
</User>
```

只要指定了至少一个 NetId，就只能从该列表中注册一个 NetId。另一种解决方案是在客户端证书的 CommonName（CN）中输入 NetId。

3.2.3.3.6.2 Mosquitto ACL

许多消息代理允许配置访问控制列表（ACL），以限制客户端与某些主题的交互。下一章将以 Mosquitto 消息为例介绍此配置。此处描述的授予访问权限的程序与 TcMqttPlugin [▶ 169] 不同。这指定了哪些其他 ADS 设备可以访问 ADS 设备。而（Mosquitto）ACL 则恰恰相反，因为此处指定了允许 ADS 设备访问的其他 ADS 设备。

概述

Mosquitto 消息代理允许配置访问控制列表，该列表被定义为一个单独的配置文件，可在代理的主配置中进行引用。该配置条目示例如下：

```
acl_file C:\Program Files\mosquitto\mosquitto.acl
```

您还可以在我们的示例 [▶ 171]中找到可供下载的完整配置文件。

在 ACL 文件中，您可以定义发布和订阅某些主题的授权，并为每个用户分别指定。用户的访问权限始终通过下面一行引入：

```
user <username>
```

随后，读写权限根据以下方案定义：

```
topic [read|write|readwrite] <topicName>
```

ADS-over-MQTT 的配置

对于 ADS-over-MQTT，必须根据通信流 [▶ 165]确保两件事：所有 ADS 设备可以访问发现主题，以及通过通信主题进行发送/接收。
ADS 设备必须始终拥有对自己主题的读取/写入访问权限。设备可接收其他 ADS 设备的发现主题的读取权限。要交换 ADS 消息，ADS 设备必须拥有对设备通信主题的读取/写入访问权限。ADS 设备在消息代理上由自己的身份标识。例如，该身份可以来自 PSK 或与客户端证书的 CommonName（CN）相对应。下面的配置说明了这些关系。

```
user <identity>
topic readwrite <VirtualAmsNetworkName>/<OwnAmsNetId>/#
topic read <VirtualAmsNetworkName>/+/info
topic readwrite <VirtualAmsNetworkName>/+/ams/#
```

如果要拒绝 ADS 设备访问其他设备，必须确保目标 AmsNetId 的主题没有写入权限。

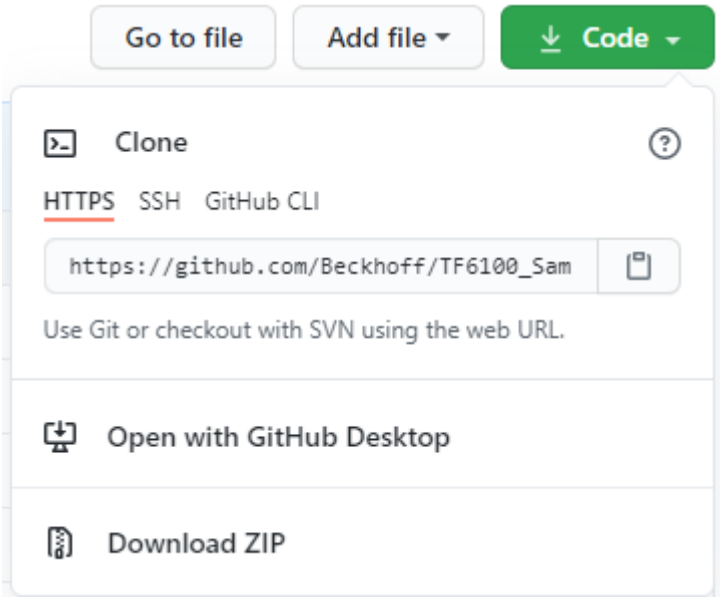
我们熟悉的 TcMqttPlugin 选项，即一个 ADS 设备只能注册一个 AmsNetId，也可以通过 Mosquitto ACL 实现。为此，条目 <OwnAmsNetId> 必须精确替换为预先设定好的唯一一个 AmsNetId。如果 ADS 设备可以注册任意 AmsNetId，则必须为 <OwnAmsNetId> 设置通配符（#）。

下面以两个 ADS 设备之间通信的访问权限条目为例：

```
user EngineeringStation
topic readwrite VirtualAmsNetwork1/18.153.78.19.1.1/#
topic read VirtualAmsNetwork1/+/info
topic readwrite VirtualAmsNetwork1/+/ams/#
user CX-12345
topic readwrite VirtualAmsNetwork1/3.120.15.8.1.1/#
topic read VirtualAmsNetwork1/+/info
topic readwrite VirtualAmsNetwork1/+/ams/#
```

3.2.3.4 示例

本产品的示例代码和配置可从 GitHub 上的相应存储库获取：https://github.com/Beckhoff/ADS-over-MQTT_Samples。您可以克隆存储库或下载包含示例的 ZIP 文件。



3.2.4 安全 ADS

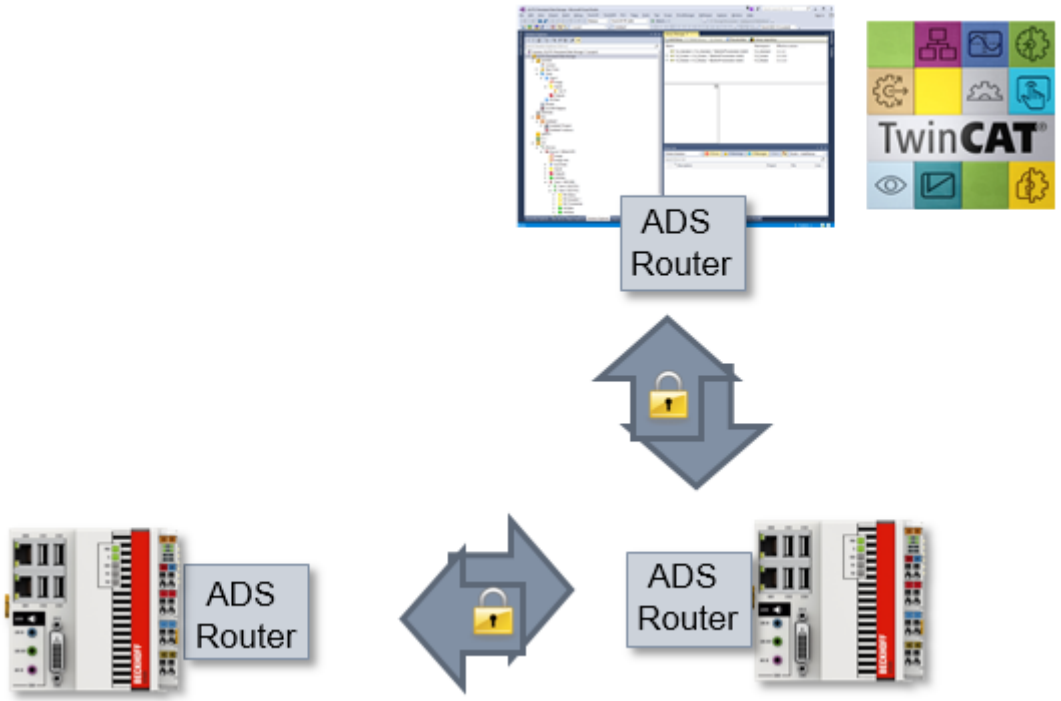
3.2.4.1 一般说明

TwinCAT 3.1 版本 4024.0 及以上
此处描述的功能从 TwinCAT 3.1 及以上版本可用。4024.0.

从 ADS 协议的角度来看，安全 ADS 是一种额外的传输通道。通过安全连接传输的 ADS 命令与通过其他通信协议传输的完全相同。

为此，在一个 TwinCAT 路由器与另一个 TwinCAT 路由器之间建立了通过 TLSv1.2 加密的连接。

由于在 TwinCAT 路由器内部实施，因此无需修改现有应用程序。只需对所用路由进行参数设置，即可使用加密连接。



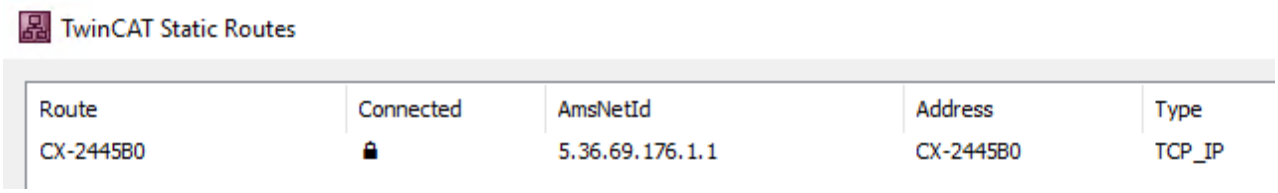
本文件说明了安全 ADS 的不同选项，特别是在提供密钥方面。

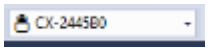
检测安全 ADS 路由

TwinCAT 将显示带有锁定图标的安全 ADS 路由。

它在适当的点显示：

- 系统的路由概览



- 在 XAE 工程环境中选择目标系统时：
- 

3.2.4.2 限制

TwinCAT 3.1 版本 4024.0 及以上

此处描述的功能从 TwinCAT 3.1 及以上版本可用。4024.0.

- 安全 ADS 仅在 ADS 路由器之间可用。

- 与所有其他 ADS 连接一样，安全 SDS 连接代表连接系统的完全访问权限，安全公告 2017-01 中也对此进行了描述。
  每个系统可通过单向 [▶ 174]ADS 路由配置这种访问。

3.2.4.3 要求

TwinCAT 3.1 版本 4024.0 及以上

此处描述的功能从 TwinCAT 3.1 及以上版本可用。4024.0.

- 安全 ADS 是 TC1000 的一个组件，使用时无需支付授权费用。
- 使用的设备需要网络通信。传入的安全 ADS 通过 TCP 端口 8016 通信。
- 可能需要为 TLS 加密生成和签署适当的证书。

3.2.4.4 技术简介

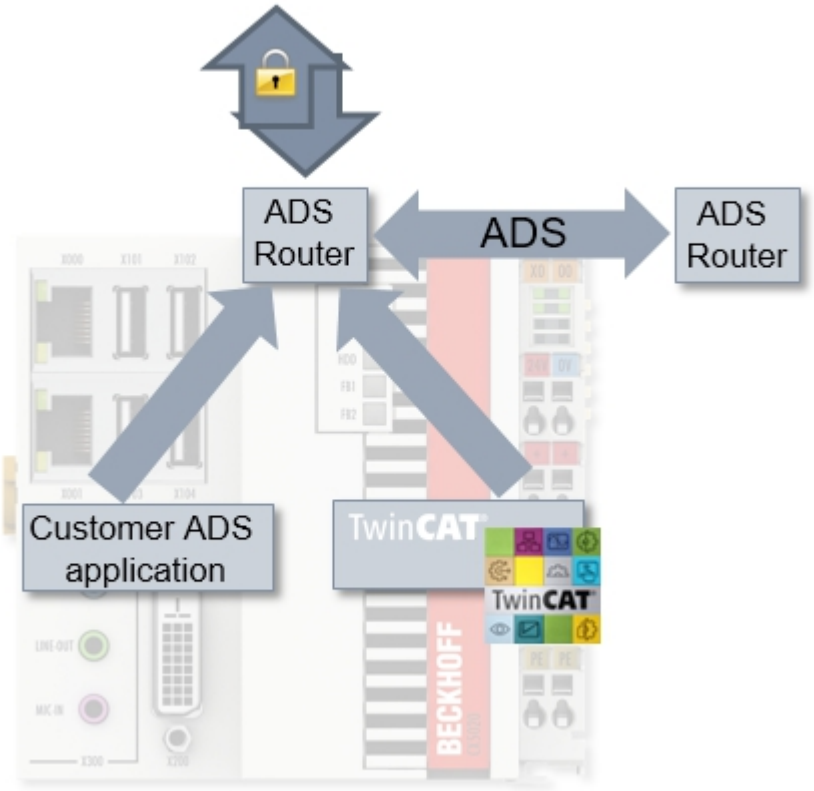
本节将介绍基本运行模式，与具体配置无关。

安全 ADS 为我们熟悉的 ADS 协议引入了额外的通信渠道。程序可以使用这一功能，而无需针对新通信通道进行调整。

因此，从安全角度来看，这是一种传输加密，但不是组件之间的端到端加密，因为设备上本地运行的所有应用程序都可以共同使用这种加密连接 — 这与 ADS 路由也是完全一样的。

本地实现

安全 ADS 是 ADS 路由器的一部分，也在此处进行配置。ADS 路由器与另一个 TwinCAT 路由器建立加密连接，并使其可供应用程序使用。因此，必须注意 ADS 设备本身不以加密形式通信应用程序，而是在路由器之



间进行。

透明改造

在 TwinCAT 路由器内实现安全 ADS 使应用程序的改造成为可能。所有 ADS 应用程序（客户端和服务器）（也包括客户编写的应用程序）都无需重新编译。

ADS 应用程序使用 ADS 路由来识别通信伙伴。该 ADS 路由独立于传输通道，并在 TwinCAT 路由器中进行描述。

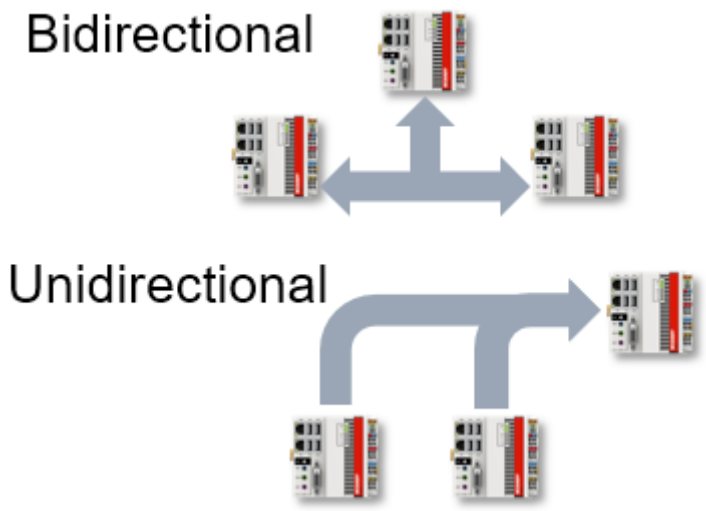
如果使用的路由切换为安全 ADS 连接，ADS 流量将以加密形式传输。

3.2.4.4.1 定向 ADS 通信

ADS 路由的特性之一是可以定向。在安全 ADS 范围内对该属性进行了补充，但一般可用于路由。

一旦在网络层面打开 后，ADS路由会被两边的ADS应用程序用来进行通信。这种行为非常高效，但可能并不可取。例如，工程计算机（XAE）在正常情况下应该可以访问运行时（XAR）系统，但 XAR 系统没有必要通过 ADS 访问 XAE 系统。
因此，可以对这一方向进行限制，即相应的系统（示例中的 XAE）不通过该路由接受任何 ADS 请求命令。

配置 [▶ 175]一章介绍了限制属性的程序。



3.2.4.4.2 服务器

一旦需要，两台设备就会立即建立正常的 ADS 路由。
路由一旦建立，就会双向使用。
作为安全 ADS 的扩展提供服务器配置。这种配置是设置特定路由的基础。

```
<TcConfig>
  <RemoteConnections>
    <Server>
      …
    </Server>
  </RemoteConnections>
</TcConfig>
```

对于 PSK [▶ 178] 和客户提供的证书 [▶ 179]，这用于在一端存储初始配置。
在设置特定路由时，会检查服务器条目是否存在权限。如果存在，将设置正常路由。

3.2.4.4.3 密钥交换

安全 ADS 提供三种提供加密所需密钥的方法，在此介绍一下它们的优缺点。

它们的共同点是，在获取机密（预共享密钥、证书）方面，相关设备必须隔离。如果这些机密被泄露，就必须重新设置系统，以恢复整个系统的完整性。

自签名证书（SSC）

首次启动时（如安装后），TwinCAT 会生成一个自签名证书。

使用这些证书的好处是，它们是在本地生成和提供的。不过，为了建立信任基础，必须在所有通信设备之间对证书进行检查。

因此，这些证书适用于初始调试，也适用于系统结构或授权访问实体不发生变化的静态机器。

从 TwinCAT 4024.0 起，这些证书在使用时将作为标准配置提供。配置 [▶ 176]一章将介绍如何使用它们来建立 ADS 路由。

证书有效期

生成证书的固定有效期为 2000 年 1 月 1 日至 2061 年 1 月 1 日。从安全的角度来看，这个时间过长，这意味着必须采取组织措施来满足安全需求。通过超长有效期，倍福可确保即使在本地系统中设置了错误时间等情况下也不会发生通信故障。

如果不需要这种行为，可以生成并使用自己的证书（请参见客户提供的证书）。

预共享密钥 (PSK)

预共享密钥可以存储在 TwinCAT 系统中。这些信息用于在建立连接时授权传入 ADS 路由。

由于预共享密钥必须经过配置，因此特别适合授予维护人员等访问权限。预共享密钥可与特定人员绑定。

预共享密钥没有像证书那样的有效期。它们还直接存储在文件中，因此不会以哈希值的形式存储（密码通常是这种情况）。因此，它们不受直观保护。

配置 [▶ 178]一章介绍了如何在通信两端使用预共享密钥。

客户提供的证书（带证书的 CA）

安全 ADS 还为客户提供了生成和管理自己证书的选择。

因此，动态配置尤其容易映射，因为可以有一个共同的证书颁发机构（CA）。所有信任该 CA 的设备都能以加密形式相互通信，无需进一步配置，即使它们之前从未遇到过对方。

配置 [▶ 179]一章介绍了如何将这些证书集成到 TwinCAT 中。

注意

证书到期

证书有有效期。必须采取组织措施，在证书到期前更换证书。

3.2.4.5 配置

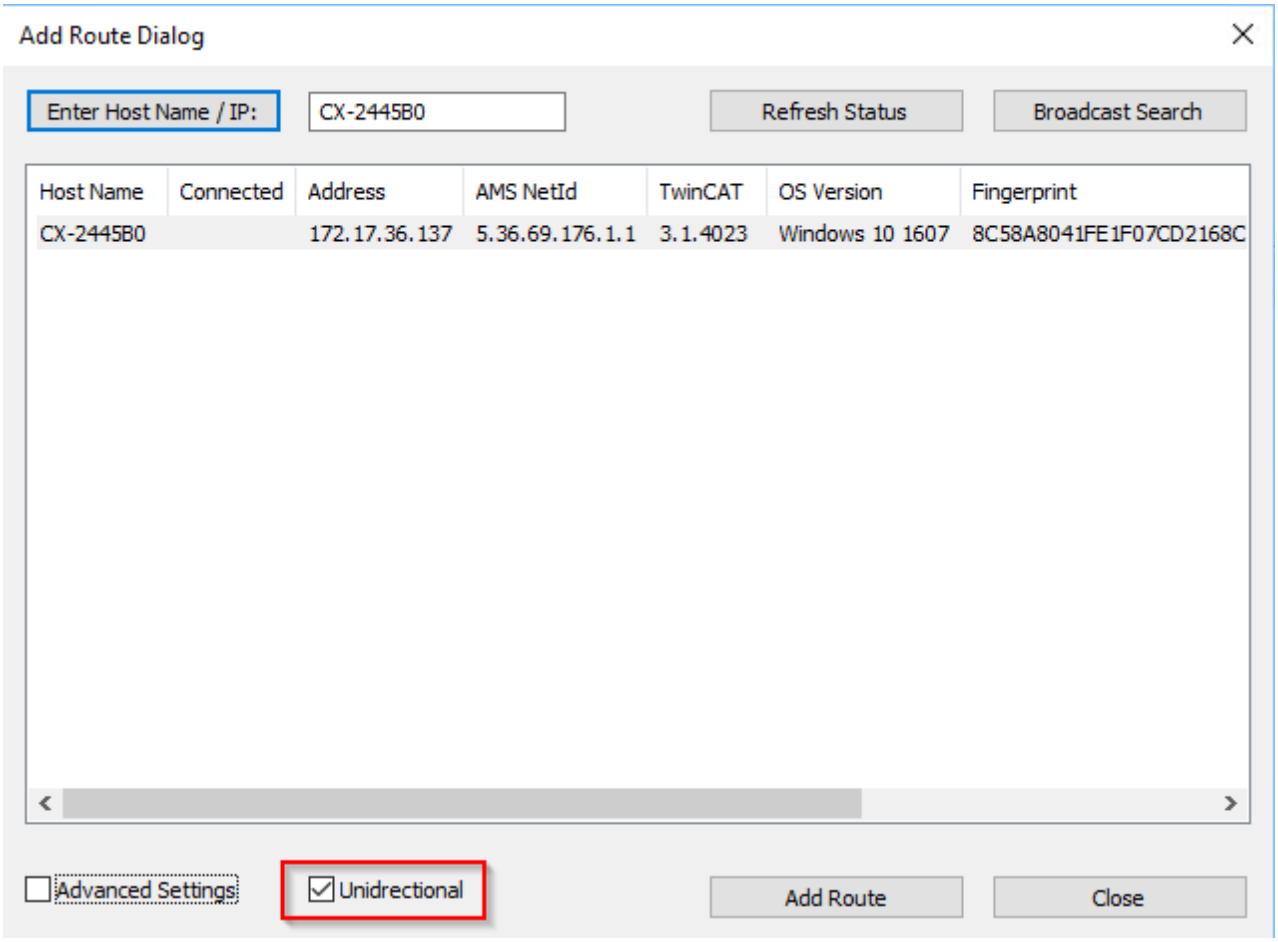
安全 ADS 提供三种提供加密所需密钥的方式。在此，我们将对这些配置进行单独描述。

虽然服务器与 路由配置在三种方式中进行了描述，但定向 ADS 连接 [▶ 175]独立说明。

3.2.4.5.1 定向 ADS 通信

在创建路由时，可使用**单向**复选框配置定向 ADS 通信。

如果设置了该复选框，则 TwinCAT 将不接受通过相关路由从对面目标系统调用的任何 ADS 命令。TwinCAT 本身发送 ADS 命令调用（请求）并接收响应。



在 XML 配置中，可通过属性 `Unidirectional="true"` 进行该设置：

```
<RemoteConnections>
<Route Unidirectional="true">
<Name>CX-123456</Name>
<Address>CX-123456</Address>
<NetId>5.36.69.176.1.1</NetId>
<Type>TCP_IP</Type>
<Flags>128</Flags>
<Tls>
…
</Tls>
</Route>
</RemoteConnections>
```

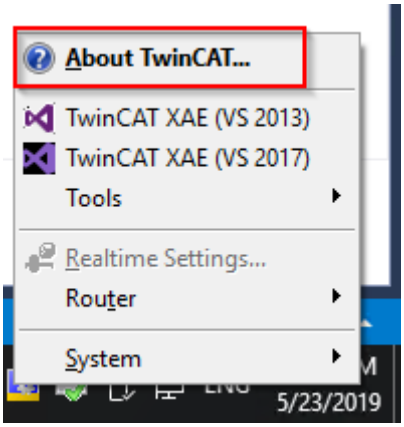
3.2.4.5.2 自签名证书（SSC）

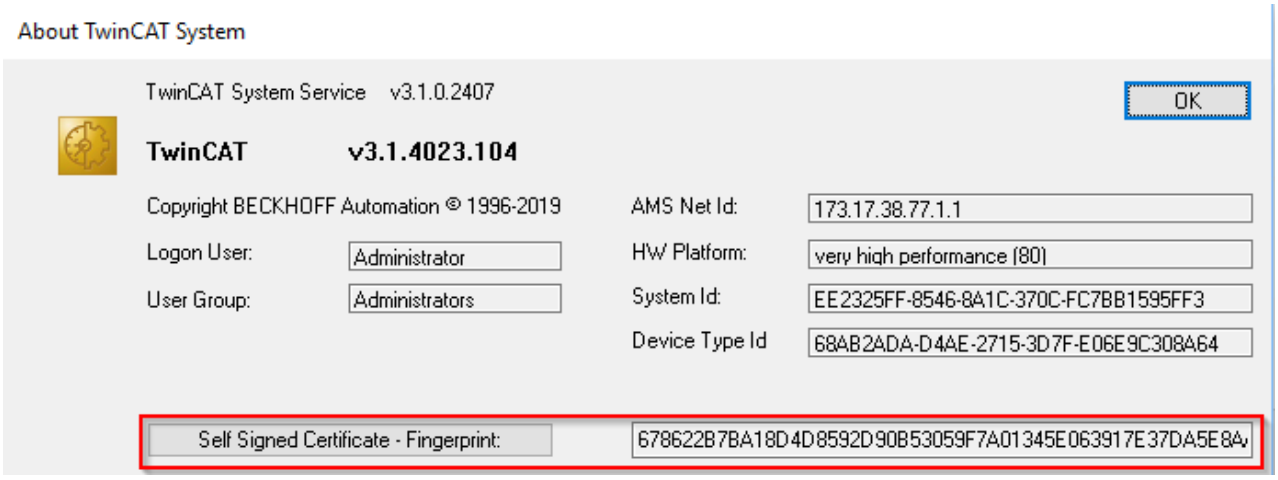
设置连接时，自签名证书需要检查通信设备，因为不会自动存在信任基础。

在 TwinCAT 中，这种检查通过对立系统的指纹来实现。

显示系统上的 SSC 指纹

您自己系统的"指纹"将显示在**关于 TwinCAT** 对话框中：



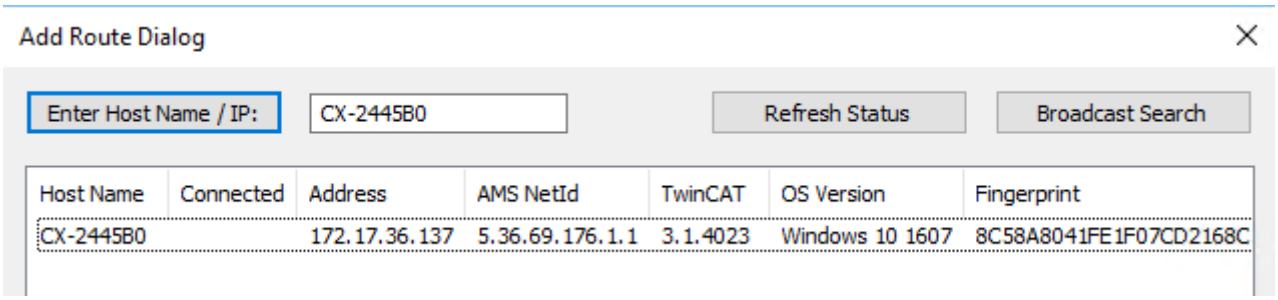


按钮**自签名证书 — 指纹：**可将右侧列出的指纹复制到剪贴板。

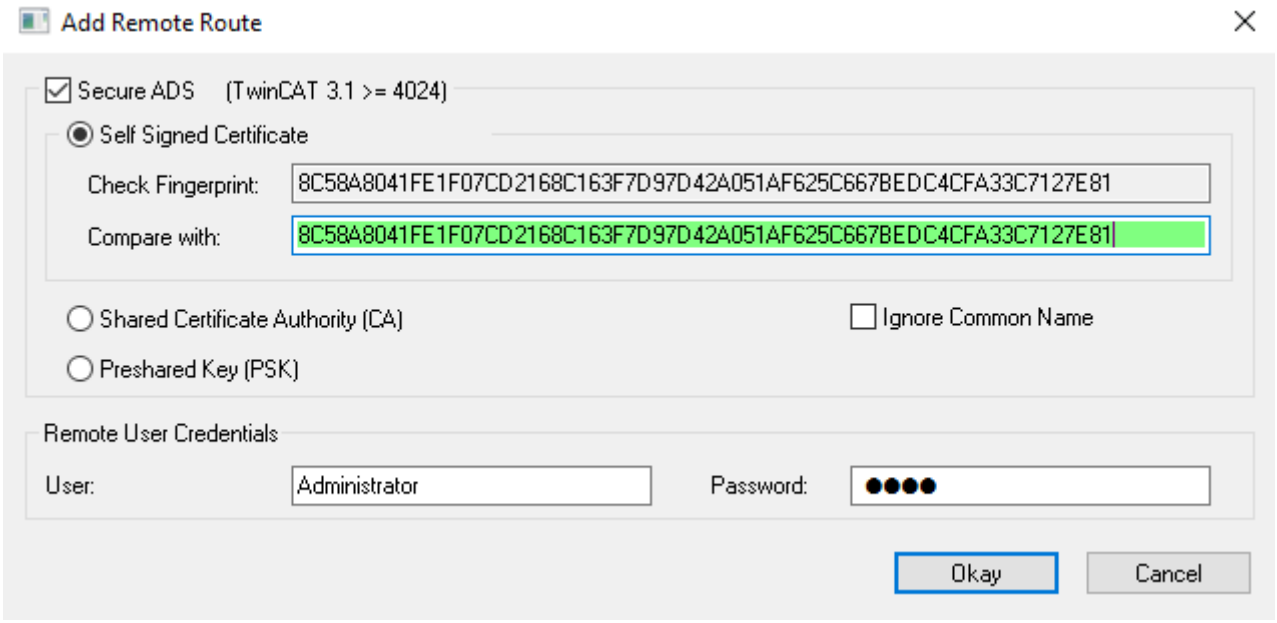
CE 系统不存在此对话框。指纹可以在这里的文件 `\Hard Disk\TwinCAT\3.1\Target\TcSelfSigned.xml` 中显示。

建立连接

指纹显示仅供参考，并且在发现后加密不安全：



指纹的最后检查在设置路由时进行：



例如，可以通过复制和粘贴使用**比较**字段进行检查：如果输入了相同的指纹，则该字段显示为绿色，否则显示为红色。

因此，例如 RDP 连接，可以通过**自签名证书 — 指纹**按钮将系统的指纹复制到剪贴板，并在此输入。

为了使目标系统接受路由建立，需要使用具有相应有效管理员权限的系统登录。
这些登录数据已加密传输。

使用 CE 系统时，即使在创建路由时选择了 **IP 地址**，也始终使用 TwinCAT 3.1 4024.5 输入主机名。因此，如果使用的网络没有正常运行的主机名查询功能，则必须通过文件 `\Hard Disk\TwinCAT\3.1\Target\StaticRoutes.xml` 中的 IP 地址手动更改主机名。

3.2.4.5.3 预共享密钥 (PSK)

预共享密钥在一端设置为服务器，在另一端设置为身份验证和授权。

将预共享密钥设置为服务器

预共享密钥通常用于服务器连接。
配置通过路由配置中的一个条目进行。

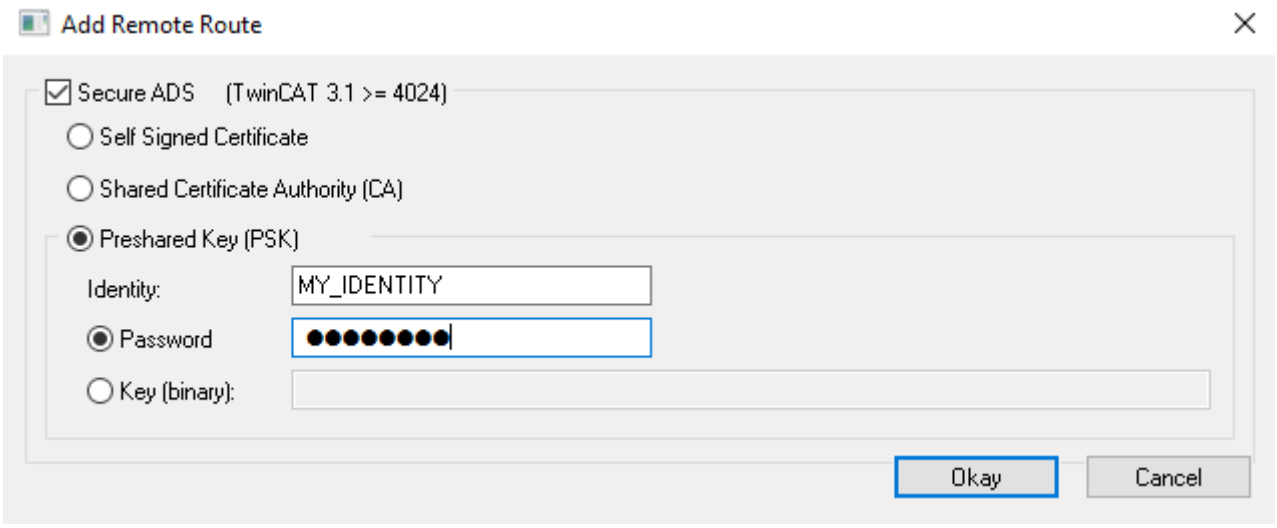
为此，可以在文件 C:\TwinCAT\3.x\Target\StaticRoutes.xml 中添加以下条目：

```
<?xml version="1.0"?>
<TcConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<RemoteConnections>
<Server>
<Tls>
<Psk>
<Identity>MY_IDENTITY</Identity>
<Pwd>MySecret</Pwd>
</Psk>
<Psk>
<Identity>MY_IDENTITY2</Identity>
<Pwd>MyOtherSecret</Pwd>
</Psk>
</Tls>
</Server>
</RemoteConnections>
</TcConfig>
```

保存的更改将在初始化 TwinCAT 路由器时被接受，例如在 RUN->CONFIG 或 CONFIG->CONFIG 转换过程中。

使用预共享密钥服务器

添加路由时，会选择**预共享密钥 (PSK)** 条目并输入相应的凭证。



如果成功，目标系统中就会存储一个特定路由，并用于未来建立连接。

3.2.4.5.4 客户提供的证书（带证书的 CA）

客户提供的证书通过路由配置中的一个条目进行配置。

为此，可以在文件 C:\TwinCAT\3.x\Target\StaticRoutes.xml 中添加以下条目：

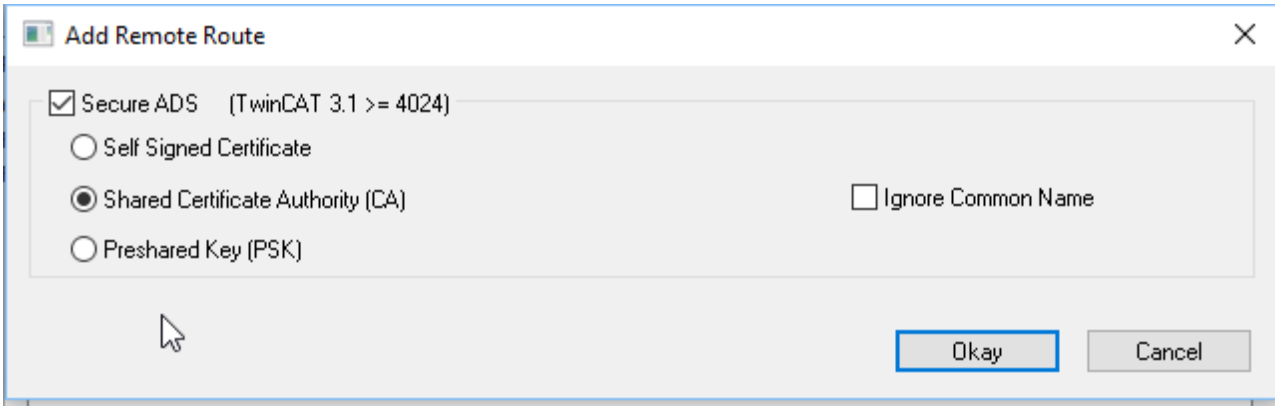
```
<?xml version="1.0"?>
<TcConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<RemoteConnections>
<Server>
<Tls IgnoreCn="true"> <!--see below-->
  <Ca>C:\TwinCAT\3.1\Target\CACerts\rootCA.pem</Ca>
  <Cert>C:\TwinCAT\3.1\Target\CACerts\ipc.crt</Cert>
                <Key>C:\TwinCAT\3.1\Target\CACerts\ipc.key</Key>
              </Tls>
          </Server>
</RemoteConnections>
</TcConfig>
```

保存的更改将在初始化 TwinCAT 路由器时被接受，例如在 RUN->CONFIG 或 CONFIG->CONFIG 转换过程中。

这些证书是 X.509 证书，可通过 OpenSSL 等生成。如果密钥（XML 元素`<Key>`）需要密码保护，可以通过 XML 元素 `<KeyPwd>` 来指定。支持 .der 和 .pem 格式。

证书的"CommonName"必须与建立连接时使用的名称一致（XML 元素 `<Name>`）。可以使用 `IgnoreCn="true"` 选项禁用该行为。

如果两端都有共同 CA 的合适证书，则无需其他信息即可使用此对话框创建路由：



如 服务器 [▶ 174] 下所述，两端都会因此创建一条特定路由。

3.2.4.5.5 停用 ADS

- 未加密的 ADS 通过 TCP 端口 48898 (0xBF02) 传输
- 发现（"广播搜索"）通过 UDP 端口 48899 (0xBF03) 传输

可以在防火墙中阻止这两个端口。

目标系统可根据要使用的端口进行配置。

以下密钥见于 KEY_LOCAL_MACHINE\SOFTWARE\[WOW6432Node\]Beckhoff\TwinCAT3\System 下：

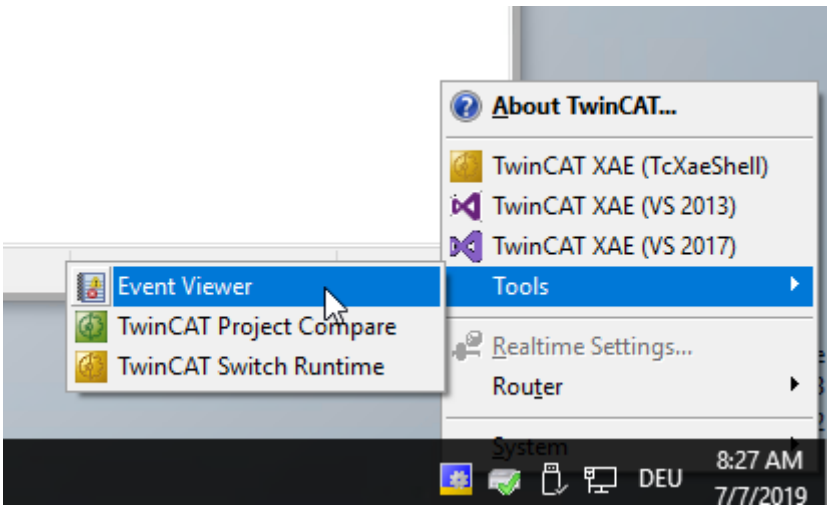
ADS 端口		
DisableAdsTcpListening	REG_DWORD	1 = 防止为未加密的 ADS 打开 TCP 端口 0xBF02。
DisableAdsTlsListening	REG_DWORD	1 = 防止为安全 ADS 打开 TCP 端口 8016
DisableAdsDiscovery	REG_DWORD	1 = 防止为 ADS 发现（"广播搜索"）打开 UDP 端口 0xBF03

此外，还可通过 StaticRoutes.xml 文件使用属性 `SecureOnly="True"`。因此，ADS 端口 0xBF02 保持打开，但不允许通过该端口进一步进行 ADS 通信。

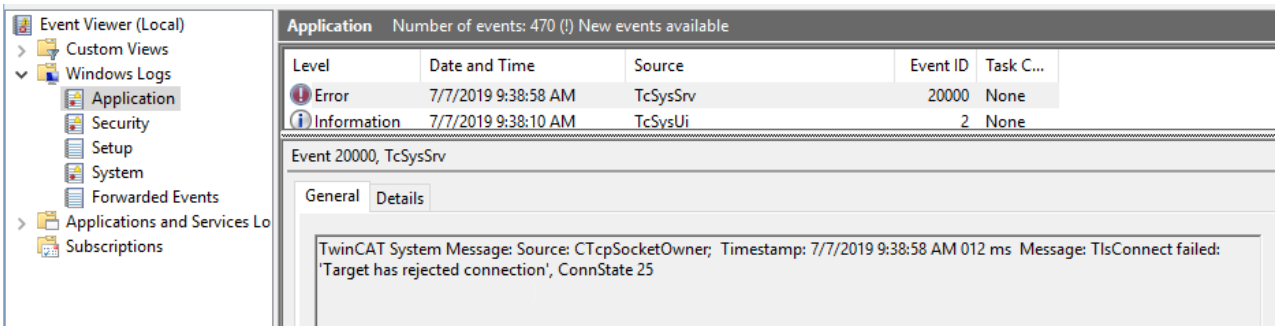
```
<RemoteConnections SecureOnly="True">
```

3.2.4.5.6 日志记录

安全 ADS 会将连接建立失败的信息写入 Windows 事件日志，该日志可通过 TwinCAT 系统托盘图标查看。



这些消息可在 **Windows 日志 > 应用程序**类别下找到：



3.2.4.6 示例

3.2.4.6.1 客户提供的证书（带证书的 CA）

此时，证书通过 Open SSL 生成，可用于安全 ADS 连接。

这些说明并不代表证书创建和处理的全面建议。特别是必须遵守有效期，这就需要采取组织措施，以确保在有效期到期前进行更换（在这种情况下：CA 为 3600 天，相关证书为 360 天）。

在本例中，将生成一个证书颁发机构 (CA)，为通信两端（此处称为 IPC 和 CX）签署证书。

调用参数的详细含义可通过 `openssl help` 查看。

✓ 已安装 OpenSSL 并可通过命令行使用。

1. 为稍后将被信任的证书颁发机构生成密钥。
   ```
   openssl genrsa -out rootCA.key 2048
   ```
2. 生成有效期为 3600 天的证书。所有者信息通过参数"-subj"添加。
   ```
   openssl req -x509 -new -nodes -key rootCA.key -sha256 -subj "/C=DE/ST=NRW/
   L=Verl/O=Bk/OU=TCPM/CN=RootCA" -days 3600 -out rootCA.pem
   ```
3. 为 IPC 生成密钥
   ```
   openssl genrsa -out ipc.key 2048
   ```
4. 为该密钥生成证书签名请求 (CSR)：
   请注意：建立连接时，必须使用指定为 CN 的地址（此处为 IP 地址）作为名称。或者，必须用 IgnoreCN 对路由进行参数设置。
   ```
   openssl req -out ipc.csr -key ipc.key -subj "/C=DE/ST=NRW/L=Verl/O=Bk/
   OU=TCPM/CN=192.168.2.1" -new
   ```
5. 与 CA 签署有效期为 360 天的 IPC CSR
   ```
   openssl x509 -req -in ipc.csr -CA rootCA.pem -CAkey rootCA.key
   -CAcreateserial -out ipc.crt -days 360 -sha256
   ```
   ⇨ 现在可以使用以下文件在 IPC 上设置路由：rootCA.pem、ipc.key 和 ipc.pem
6. 为 CX 生成密钥
   ```
   openssl genrsa -out cx.key 2048
   ```
7. 为该密钥生成证书签名请求 (CSR)：
   请注意：建立连接时，必须使用指定为 CN 的地址（此处为 IP 地址）作为名称。或者，必须用 IgnoreCN 对路由进行参数设置。
   ```
   openssl req -out cx.csr -key cx.key -subj "/C=DE/ST=NRW/L=Verl/O=Bk/OU=TCPM/
   CN=192.168.2.2" -new
   ```
8. 与 CA 签署有效期为 360 天的 IPC CSR
   ```
   openssl x509 -req -in cx.csr -CA rootCA.pem -CAkey rootCA.key -CAcreateserial
   -out cx.crt -days 360 -sha256
   ```
   ⇨ 现在可以使用这些文件在 CX 上设置路由：rootCA.pem、cx.key 和 cx.pem

⇨ 可以使用该路由。

3.3 文件夹和文件类型

3.3.1 TwinCAT PLC 项目文件

3.3.1.1 Port_xxx.app

3.3.1.2 Port_xxx.autostart

3.3.1.3 Port_xxx.cid

3.3.1.4 Port_xxx.crc

3.3.1.5 Port_xxx.occ

3.3.1.6 Port_xxx.oce

3.3.1.7 Port_xxx.ocm

3.3.1.8 Port_xxx_boot.tizip

3.3.1.9 Port_xxx_act.tizip

3.3.1.10 Port_xxx.bootdata

3.3.1.11 Port_xxx.bootdata-old

3.3.1.12 PLC_Name.tpzip

3.3.1.13 PLC_Name.tmc

3.3.1.14 PLC_Name.tpy

3.3.2 TwinCAT C++ 项目文件

文件	描述	更多信息
工程/XAE		
*.sln	Visual Studio Solution 文件，承载 TwinCAT 和非 TwinCAT 项目	
*.tsproj	TwinCAT 项目，所有嵌套 TwinCAT 项目的集合，如 TwinCAT C++ 或 TwinCAT PLC 项目	
_Config/	文件夹包含属于 TwinCAT 项目的其他配置文件（*.xti）。	参见菜单“工具| 选项| TwinCAT| XAE-Environment| 文件设置”
_Deployment/	用于编译 TwinCAT C++ 驱动程序的文件夹	
*.tmc	TwinCAT 模块类文件（基于 XML）	请参见 TwinCAT 模块类编辑器 (TMC)
*.rc	资源文件	请参见
.vcxproj.	Visual Studio C++ 项目文件	
*ClassFactory.cpp/.h	该 TwinCAT 驱动程序的类工厂	
*Ctrl.cpp/.h	上传和删除 TwinCAT UM 平台的驱动程序	
*Driver.cpp/.h	上传和删除 TwinCAT RT 平台的驱动程序	
*Interfaces.cpp/.h	TwinCAT COM 接口类的声明	
*W32.cpp./.def/.idl		
*.cpp/.h	驱动程序中的每个 TwinCAT 模块都有一个 C++/报头文件。在此处插入用户代码。	
Resource.h	*.rc 文件必需	
TcPch.cpp/.h	用于创建预编译报头	
%TC_INSTALLPATH%\Repository\<Vendor>\<PrjName>\<Version>\<Platform>*.tmx	通过 TcLoader 加载的编译驱动程序。 TwinCAT 3.1.4024.x： C:\TwinCAT\3.1\Repository\C++ Module Vendor\Untitled1\0.0.0.1\TwinCAT RT *\Unititled1.tmx 从 TwinCAT 3.1.4026.x 起： C:\ProgramData\Beckhoff\TwinCAT\3.1\Repository\C++ Module Vendor\Untitled1\0.0.0.1\TwinCAT RT *\Unititled1.tmx	请参见 版本控制的 C++ 项目
%TC_INSTALLPATH%\CustomConfig\Modules*	已发布的 TwinCAT C++ 项目通常为 TwinCAT 3.1.4024.x： C:\TwinCAT\3.1\CustomConfig\Modules* 从 TwinCAT 3.1.4026.x 起： C:\ProgramFiles (x86)\Beckhoff\TwinCAT\3.1\Config\Modules*	请参见
运行时/XAR		
%TC_BOOTPRJPATH%\CurrentConfig*	当前配置设置 Windows： TwinCAT 3.1.4024.x： C:\TwinCAT\3.1\Boot 从 TwinCAT 3.1.4026.x 起： Windows：C:\ProgramData\Beckhoff\TwinCAT\3.1\Boot	

文件	描述	更多信息
	TwinCAT/BSD：/usr/local/etc/TwinCAT/3.1/Boot	
%TC_DRIVERAUTOINSTALLPATH% *.sys/pdb	经编译的平台特定驱动程序，通过操作系统加载。 Windows： TwinCAT 3.1.4024.x： C:\TwinCAT\3.1\Driver\AutoInstall（已由系统加载） 从 TwinCAT 3.1.4026.x 起： <not available> 请迁移至基于 TMX 的 C++ 的项目 TwinCAT/BSD®： <not available>	
%TC_INSTALLPATH% \Boot\Repository\<Vendor>\<PrjName>\<Version>*.tmx	经编译的平台特定驱动程序通过 TcLoader 加载。 Windows： TwinCAT 3.1.4024.x： C:\TwinCAT\3.1\Boot\Repository\C++ Module Vendor\Untitled1\0.0.0.1\Untitled1.tmx 从 TwinCAT 3.1.4026.x 起： C:\ProgramData\Beckhoff \TwinCAT\3.1\Boot\Repository\C++ Module Vendor\Untitled1\0.0.0.1\Untitled1.tmx TwinCAT/BSD：/usr/local/etc/TwinCAT/3.1/Boot\/Repository\C++ Module Vendor\Untitled1/0.0.0.1/Untitled1.tmx	
%TC_BOOTPRJPATH% \TM\OBJECTID.tmi	TwinCAT 模块实例文件 功能说明驱动程序的变量 文件名为 ObjectID.tmi Windows： TwinCAT 3.1.4024.x： C:\TwinCAT\3.1\Boot\TMI\OTCID.tmi 从 TwinCAT 3.1.4026.x 起： Windows：C:\ProgramData\Beckhoff \TwinCAT\3.1\Boot\TMI\OTCID.tmi TwinCAT/BSD：/usr/local/etc/TwinCAT/3.1/Boot/TMI/OTCID.tmi	
临时文件		
*.sdf	IntelliSense 数据库	
.suo/.v12.suo	用户特定文件和 Visual Studio 特定文件	
*.tsproj.bak	从 tsproj 自动生成备份文件	
ipch/	为预编译报头创建中间目录	

3.3.3 TwinCAT 项目文件

3.3.3.1 CurrentConfig.xml

3.3.3.2 CurrentConfig.tszip

3.3.4 PLC HMI 文件

3.3.4.1 Port_xxx.textlistname.txt

3.3.4.2 Port_xxx Folder

3.3.5 PLC HMI 文件（目标可视化）

3.3.5.1 tc3plchmi.ini

3.3.6 PLC HMI Web 文件

3.3.6.1 port_xxx.imagepoolcollection.csv

3.3.6.2 webvisu.cfg.json

3.3.6.3 webvisu.htm

3.3.6.4 webvisu.js

3.4 固件更新

3.4.1 概述

如果没有 TwinCAT 3 开发环境 (XAE) 可用，可以通过文件拷贝的方式更新 TwinCAT PLC 系统或整个 TwinCAT 系统的启动数据。

- 执行 PLC 更新 [▶ 186]
- 执行 C++ 更新 [▶ 187]
- 对整个机器进行更新 [▶ 188]
- 克隆机器 [▶ 188]

关于各种文件的说明以及它们在相关项目（项目目录）中的存储位置信息以及有关机器（TwinCAT 启动目录）的信息，请参见文件夹和文件类型 [▶ 183]部分。

3.4.2 执行 PLC 更新

✓ TwinCAT 版本 TC3.1.4022.0 或更高版本

✓ 通过创建（或重新创建）PLC 项目，已为机器平台生成了boot data（启动数据）。创建（或重新创建）项目时不需要与目标系统连接。

✓ 自上次更新以来，过程映像和硬件配置均未发生变化。

✓ 在 PLC 项目设置中激活了**Symbolic Mapping（符号映射）**选项。

1. 将 PLC 项目的启动数据，即所有文件和文件夹，从 ..\<Solution name>\<Project name>_Boot\<Platform>\Plc\ 文件夹中复制。
2. 用复制的启动数据替换机器的 TwinCAT PLC 启动目录中的启动数据。
   ⇨ < TC3.1.4026.0: C:\TwinCAT\3.1\Boot\Plc
   ⇨ >=TC3.1.4026.0: C:\ProgramData\Beckhoff\TwinCAT\3.1\Boot\Plc
3. 重新启动机器的 TwinCAT 系统。

⇨ 更新 TwinCAT PLC 系统的启动数据，从而更新 PLC 运行时本身。

源代码更新

如果除了启动数据外，您在 Runtime（运行时）系统上还存储了 PLC 项目的源代码，那么您也可以在文件级更新时将文件夹 ..\<Solution name>\<Project name>_Boot\<Platform>\CurrentConfig\ 中的归档文件夹复制到Runtime（运行时）系统上启动目录的文件夹 CurrentConfig。

< TC3.1.4026.0: C:\TwinCAT\3.1\Boot\CurrentConfig

>=TC3.1.4026.0: C:\ProgramData\Beckhoff\TwinCAT\3.1\Boot\CurrentConfig

3.4.3 执行 C++ 更新

运行时数据可以通过文件拷贝的方式从一台机器传输到另一台，前提是这两台机器来自同一平台，并与同等硬件设备连接。

以下步骤功能说明了将二进制配置从一台机器（“源”）传输到另一台机器（“目标”）的简单程序。

1. 清空源设备上的启动文件夹。
   ⇨ < TC3.1.4026.0: C:\TwinCAT\3.1\Boot
   ⇨ >=TC3.1.4026.0: C:\ProgramData\Beckhoff\TwinCAT\3.1\Boot
2. 在源机器上创建（或启用）模块。
3. 将启动文件夹从源机器转移到目标设备。
   该文件夹还包含含有必要 TMX 文件的存储库。该文件夹在源机器和目标设备上的存放路径如下所示。
   ⇨ < TC3.1.4026.0: C:\TwinCAT\3.1\Boot
   ⇨ >=TC3.1.4026.0: C:\ProgramData\Beckhoff\TwinCAT\3.1\Boot
4. 对于 TwinCAT 驱动程序项目 (.sys)：传输 MYDRIVER.sys 驱动程序，必要时也传输 PDB 文件。
   ⇨ < TC3.1.4026.0: C:\TwinCAT\3.1\Driver\AutoInstall\MYDRIVER.sys
   ⇨ >=TC3.1.4026.0: C:\ProgramData\Beckhoff\TwinCAT\3.1\Driver\AutoInstall\MYDRIVER.sys
5. 对于 TwinCAT 驱动程序项目 (.sys)，如果驱动程序是新安装在设备上的：
   TwinCAT 必须执行一次注册。通过 SysTray（右键单击 -> **系统** -> **启动/重启**）将 TwinCAT 切换到运行模式。
   或者也可以使用此指令（将“%1”替换为驱动程序名称）：
   ⇨ < TC3.1.4026.0：
   ```
   sc create %1 binPath= c:\TwinCAT\3.1\Driver\AutoInstall\%1.sys type=
   kernel start= auto group= "file system" DisplayName= %1 error= normal
   ```
   ⇨ >=TC3.1.4026.0：
   ```
   sc create %1 binPath= C:
   \ProgramData\Beckhoff\TwinCAT\3.1\Driver\AutoInstall\%1.sys type= kernel
   start= auto group= "file system" DisplayName= %1 error= normal
   ```

⇨ 现在可以启动目标机器。

处理授权

请注意，授权不能以这种方式转让。请使用预装授权、批量授权或其他方法提供授权。

3.4.4 对整个机器进行更新

✓ 通过创建（或重新创建）TwinCAT 项目，已为机器平台生成了启动数据。创建（或重新创建）项目时不需要与目标系统连接。

✓ 实际硬件配置与项目配置一致。

✓ 如果要在多台机器上（而非指定机器上）进行机器更新，则需启用以下选项：
路由设置中的 **Use Relative NetIds**（使用相对网络标识）（System > Routes，NetIdManagement）以及所有网络和 USB 设备的 adapter settings（适配器设置）中的
Virtual Device Names
（虚拟设备名称）（例如，I/O > Devices > EtherCAT Master，适配器选项卡）
机器的网络适配器名称必须与配置中的适配器名称相匹配。

1. 从文件夹
..\<Solution name>\<Project name>_Boot\<Platform>\ 中复制 TwinCAT 项目的启动数据，即所有文件和文件夹。
2. 用复制的启动数据替换机器的 TwinCAT 启动目录中的启动数据。
   ⇨ < TC3.1.4026.0: C:\TwinCAT\3.1\Boot
   ⇨ >=TC3.1.4026.0: C:\ProgramData\Beckhoff\TwinCAT\3.1\Boot
3. 如果您使用 C++ 模块，请复制 C++ 驱动程序（请参见执行 C++ 更新 [▶ 187]一章）。
4. 重新启动机器的 TwinCAT 系统。

⇨ 更新 TwinCAT 系统的启动数据，从而更新 TwinCAT 系统本身。

3.4.5 克隆机器

要将 TwinCAT 或 PLC 项目的启动数据从一台机器迁移到另一台机器，请从第一台机器的启动目录中复制启动数据，并替换另一台机器启动目录中的启动数据。

如果要复制启动数据的 TwinCAT 系统处于运行模式，并且 Persistent（保持型）数据也要交换，则应首先将 TwinCAT 系统从运行模式切换到配置模式，以便将 Persistent（保持型）数据存入 .bootdata 文件，并在启动目录中可供复制。（请参见 Port_xxx.bootdata [▶ 183]）

3.5 自动启动程序

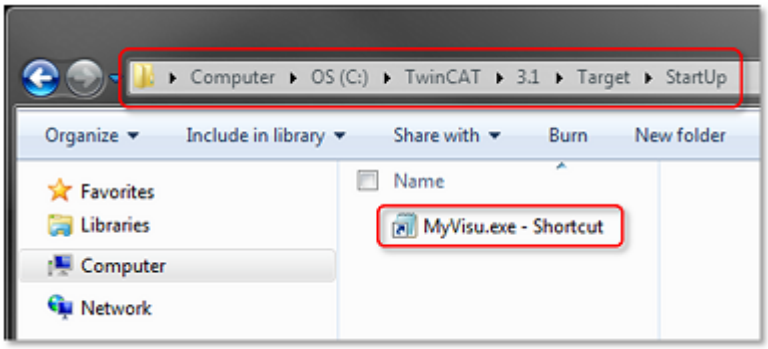
TwinCAT 3 提供选项可在开机后自动启动选定程序。这对于必须在执行前启动 TwinCAT 的程序（如可视化软件）尤其有用。

要在 TwinCAT 启动后自动启动程序，必须在 TwinCAT 目录下的一个特殊启动文件夹中创建该程序的快捷方式。程序本身必须与 TwinCAT 安装在同一台 PC 上。启动 TwinCAT 运行时系统后首次激活运行模式后，将执行启动文件夹中的快捷方式。

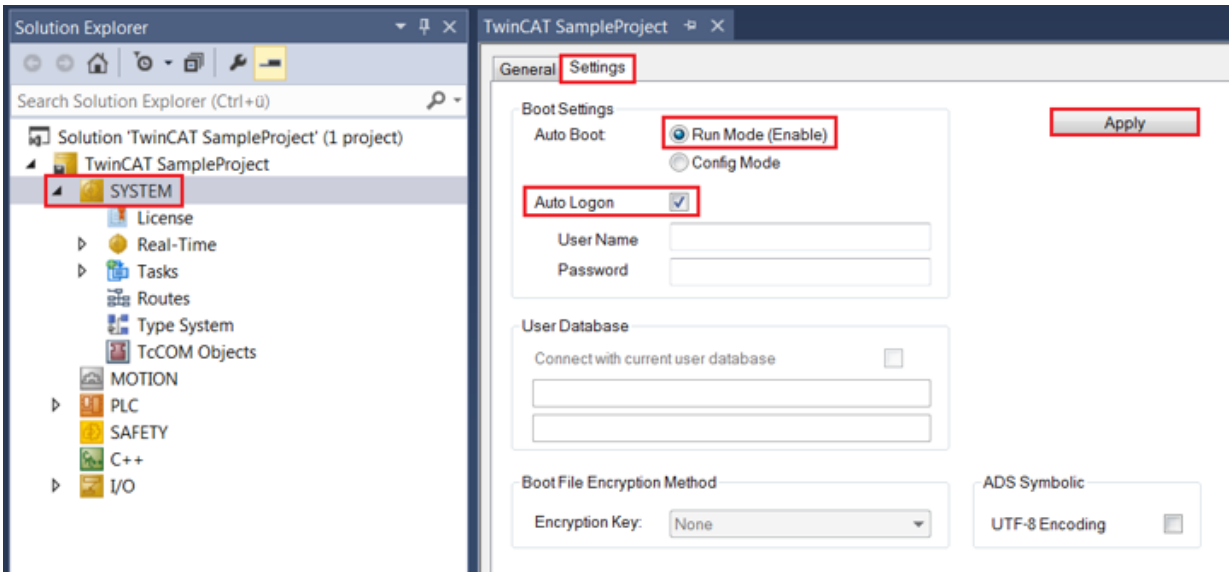
路径 <TwinCAT>\3.x\Target\StartUp 指向启动文件夹。指定结果如下：

<TwinCAT>	TwinCAT 安装文件夹 • < TC3.1.4026.0: C:\TwinCAT\ • >=TC3.1.4026.0: C:\Program Files (x86)\Beckhoff\TwinCAT\
3.x	TwinCAT 版本（所有 TwinCAT 版本都存储在 TwinCAT 安装文件夹中的独立文件夹中）。
x	TwinCAT Build 的占位符，如"3.1"。

1. 将程序的快捷方式保存在 <TwinCAT>\3.x\Target\StartUp 文件夹中。



2. 确保 TwinCAT 以运行模式启动。
3. 在 TwinCAT 项目树中，双击**系统**并选择**设置**选项卡。



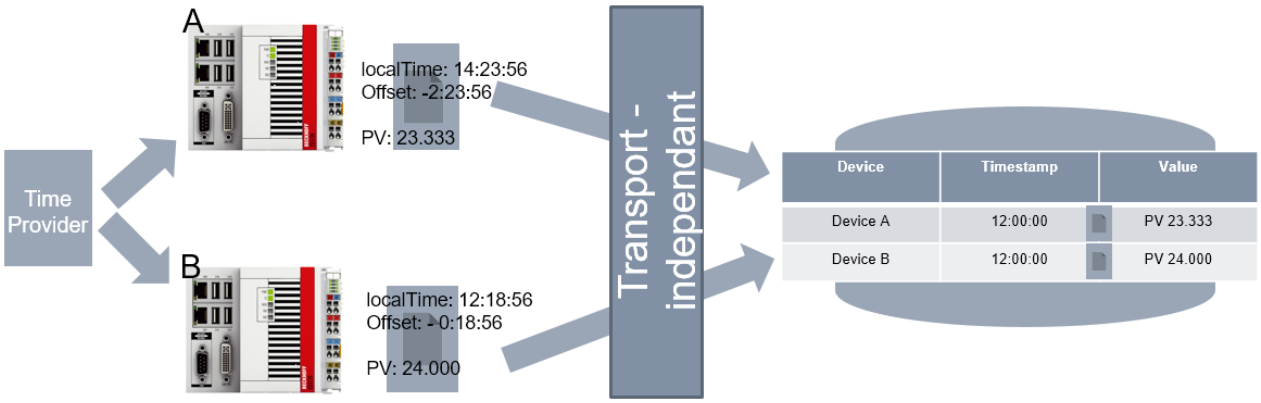
4. 启用**运行模式（启用）**和**自动登录**选项。
5. 单击**应用**。

3.6 校正时间戳

3.6.1 概述

控制器生成的数据要在现代分布式系统中进行收集和链接。
由于控制器一开始是独立设备，因此它们具有独立的时序基准。在普通数据库中，不可能将数据与时间关联。

为了解决这个问题，控制器之间的同步早已成为可能，例如使用网络协议 IEEE1588 或 PTP。
不过，在许多情况下，为数据提供统一的时间戳就足够了。控制器之间可以独立运行，这样一方面可以降低与上述协议相关的硬件成本，另一方面控制器之间也不存在技术依赖。
本章介绍用于调整时间戳以存储时间同步数据的 TwinCAT 组件。



图中说明了基本思路：独立控制器获取本地时间戳，并使用偏移对其进行调整，然后用于存储公共数据。

为此，TwinCAT 实时系统中提供了一个核心组件，即外部时间接口。本组件

- 从配置源（外部时间提供程序）接收校正时间的偏移。
- 根据当前当地时间，为外部时间消耗程序提供校正时间。

校正时间可被实时系统内外的不同组件使用。

源通常是 NTP 服务器或基于 EtherCAT 的 DC 时间信号，例如通过 PTP（IEEE1588）通过 EL6688 进行同步。不过，消耗程序也可以实现源，这样其他时间信号也可以作为源来实现。

因此，除了上述 TwinCAT 实时系统的中央组件外，该概念还包括两类组件：

1. 外部时间提供程序：为调整中央组件的时间戳提供偏移。
   例如，提供程序通过 NTP（网络时间协议，参见 RFC 4330）获取时间戳，从中计算出与本地系统时间的偏移，并使其可用。
2. 外部时间消耗程序：使用从中央组件获得的偏移。因此，可以在组件中使用时间戳，从而在远程设备上获得可比较数据。
   所有使用时间戳的 TwinCAT 组件以及客户应用程序都可以作为时间的使用者。

3.6.2 系统要求

技术数据	要求
操作系统	Windows 10、Windows Embedded Standard 7
目标平台	PC 架构 (x86, x64)
TwinCAT 版本	TwinCAT 3.1 版本 4024.0 或更高版本
所需 TwinCAT 设置级别	TwinCAT 3 XAE, XAR
所需 TwinCAT 授权	任何运行时授权 (PLC, C++)

3.6.3 限制

必须考虑到一些重要限制：

- 此处描述的外部时间接口不会改变 TwinCAT 系统时间
- 外部时间偏移由提供程序提供给消耗程序。由此可见
  - 提供程序必须正确计算偏移。
  - 不能保证时间戳的单调性。
- 外部时间偏移不会被保存供随后检索。这意味着在 TwinCAT 系统中只管理当前偏移。

3.6.4 技术简介

TwinCAT 为外部时间提供程序和外部时间消耗程序提供不同的接口，以便利用校正时间戳的概念。

在外部时间消耗程序端，不同的 TwinCAT 组件都能使用外部时间戳。此外，还有不同的应用程序访问选项。

在外部时间提供程序端，提供的模块可以通过 NTP 计算并提供偏移。此外，还有一个模块可以通过 DC 使用偏移。此外，还为 TwinCAT C++ 提供了用于提供偏移的相应接口，以便客户创建自己的外部时间提供程序。

不同使用情况下的时间戳

需要注意的是，TwinCAT 在这个概念中区分了四种类型的时间戳：

1. 无：本地系统时间，且无校正
2. 软：建议用途，例如用于 NTP
3. 中：建议用途，例如用于 IEEE1588
4. 硬：建议用于硬件同步等不发生漂移的情况

外部时间提供程序提供其中一种可能的偏移；每种类型只能定义一个提供程序。

然后，外部时间消耗程序可以使用任何偏移；所有四种偏移类型都可以根据需要使用。因此，可以在不同的组合或运行模式中使用不同的时间戳。例如，可以用本地系统时间进行本地诊断，同时，来自不同系统的汇总数据可以用偏移类型“软”校正，并存储在一个共同数据库中。

校正时间戳的接口使用长度为 8 字节的数据类型，从 1.1.1601 开始以 100 ns 步长为单位计数。

3.6.4.1 消耗程序

外部时间消耗程序是可以对本地系统时间进行偏移校正的组件。为此，组件必须选择或配置软、中或硬类型的偏移，并进行相应的查询。

3.6.4.1.1 作为偏移消耗程序的 TwinCAT 组件

下列 TwinCAT 组件支持校正时间戳方法 — 相关文档介绍了如何启用该功能：

- TwinCAT 3 事件记录器

该名单还将扩大。

3.6.4.1.2 应用实施

应用程序可在不同组件中使用外部时间偏移：

- 实时 PLC：PLC 可以查询偏移量、或对本地时间戳进行相应校正。
- 实时 — C++：C++ TcCOM 模块能够查询偏移并采取相应行动。
- 用户模式 — ADS 设备通知：可以校正随 ADS 设备通知发送的时间戳。
- 用户模式 — ADS 读取：校正时间戳可通过 ADS 读取进行检索。这可以在 ADS Sum 命令中使用，与数据一起检索时间戳。

这些接口已在相应的 API 章节中作了说明。

3.6.4.2 提供程序

外部时间提供程序是通过外部信息源确定与本地系统时间相关的外部时间偏移并将其提供给 TwinCAT 的组件。这使得外部时间消耗程序可以独立于提供程序而接收校正时间。

TwinCAT 还为提供程序提供：

- NTP 提供程序：一种实现，通过 (S)NTP 从 NTP 服务器查询和提供时间信号。

- DC 提供程序：一种实现，将 DC 时间作为偏移从 EtherCAT 主轴传递到 TwinCAT（例如通过 IEEE1588 或 PTP）
- 此外，消耗程序还可以提供自己的提供程序。

3.6.4.2.1 NTP 提供程序

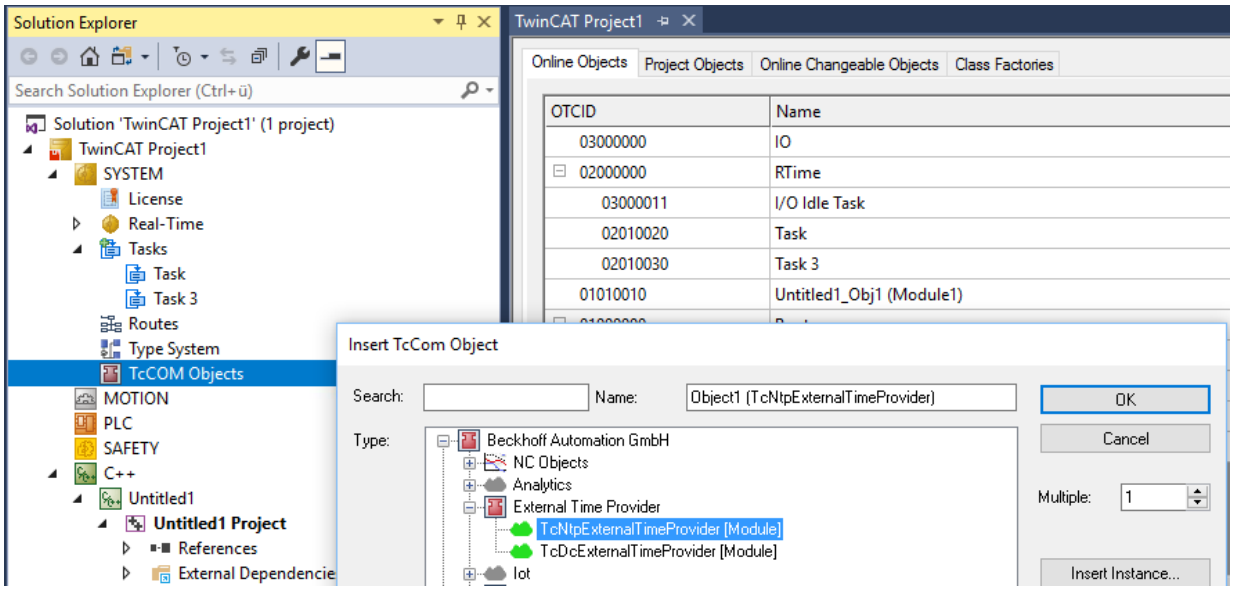
NTP 提供程序是一个 (S)NTP 客户端，循环接收来自 NTP 服务器的时间信号。这样，它就能计算出系统时间与 NTP 服务器时间信号的偏移，并相应地使其可用。

配置

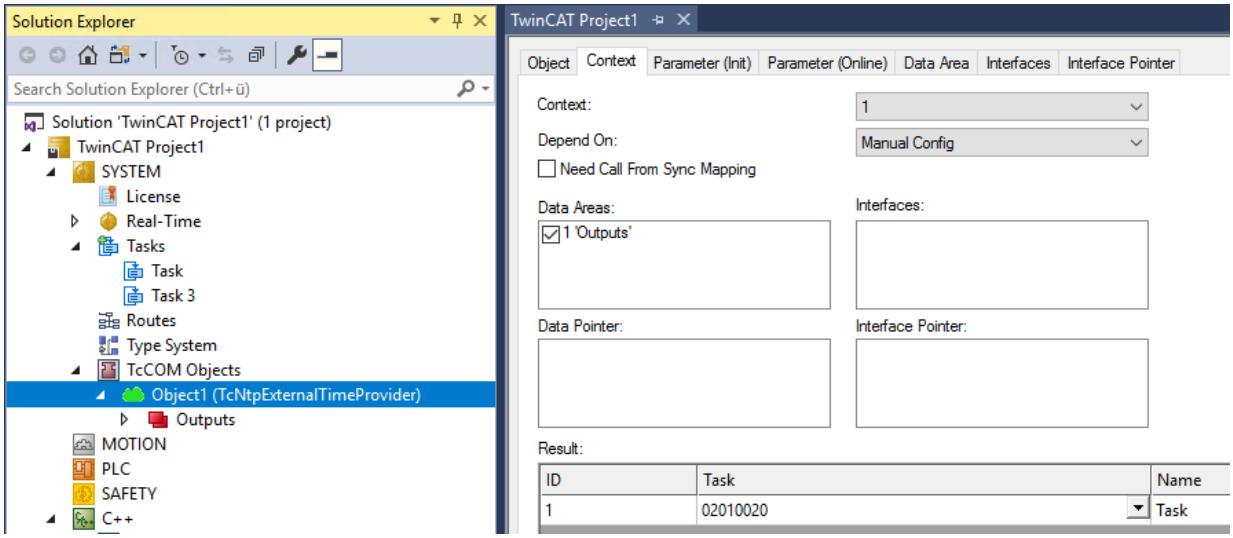
NTP 提供程序通过 TcCOM 模块 TcNtpExternalTimeProvider 实现。该模块作为 TcCOM 模块如下调试：

✓ TwinCAT 项目

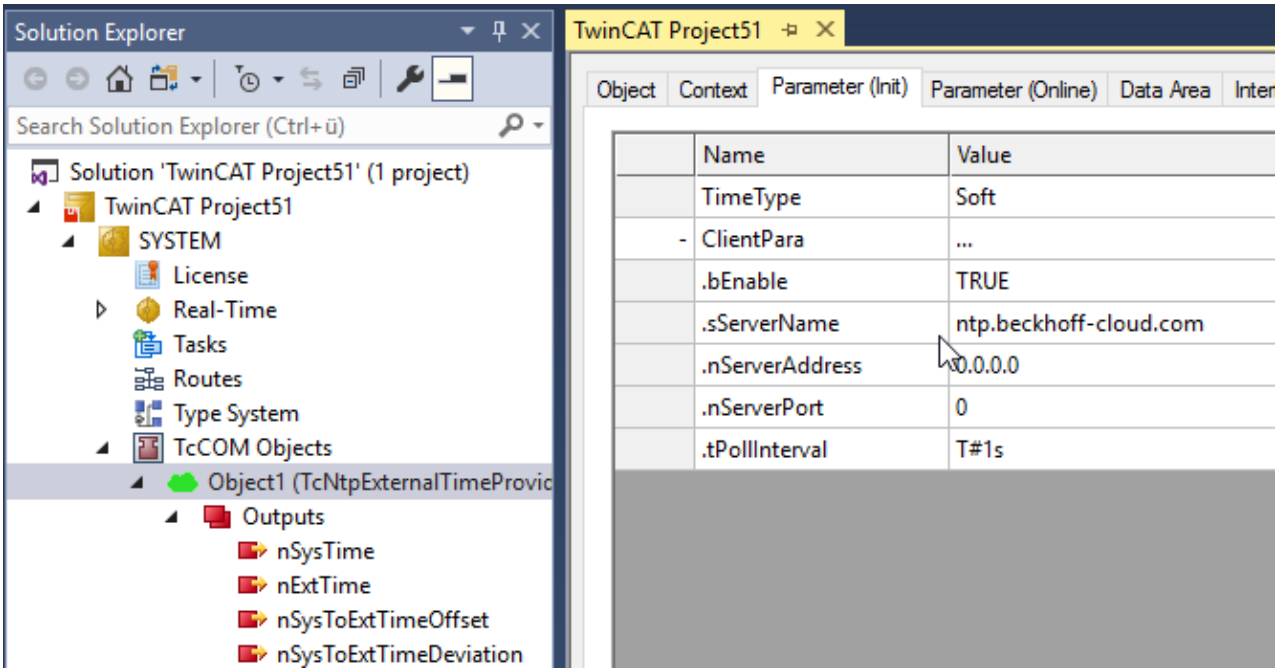
1. 在 System->TcCOM Objects 下插入 TcCOM 模块，并在 External Time Provider 类别中选择 TcNtpExternalTimeProvider 类型。



2. 模块需要一个被调用的任务。这可通过模块的**上下文**选项卡进行参数设置：



⇨ 可对 TcCOM 模块进行参数设置：



配置在**参数 (Init)** 选项卡中进行。参数含义如下：

- **TimeTime**：本模块要确定偏移的类型。

Client Para:

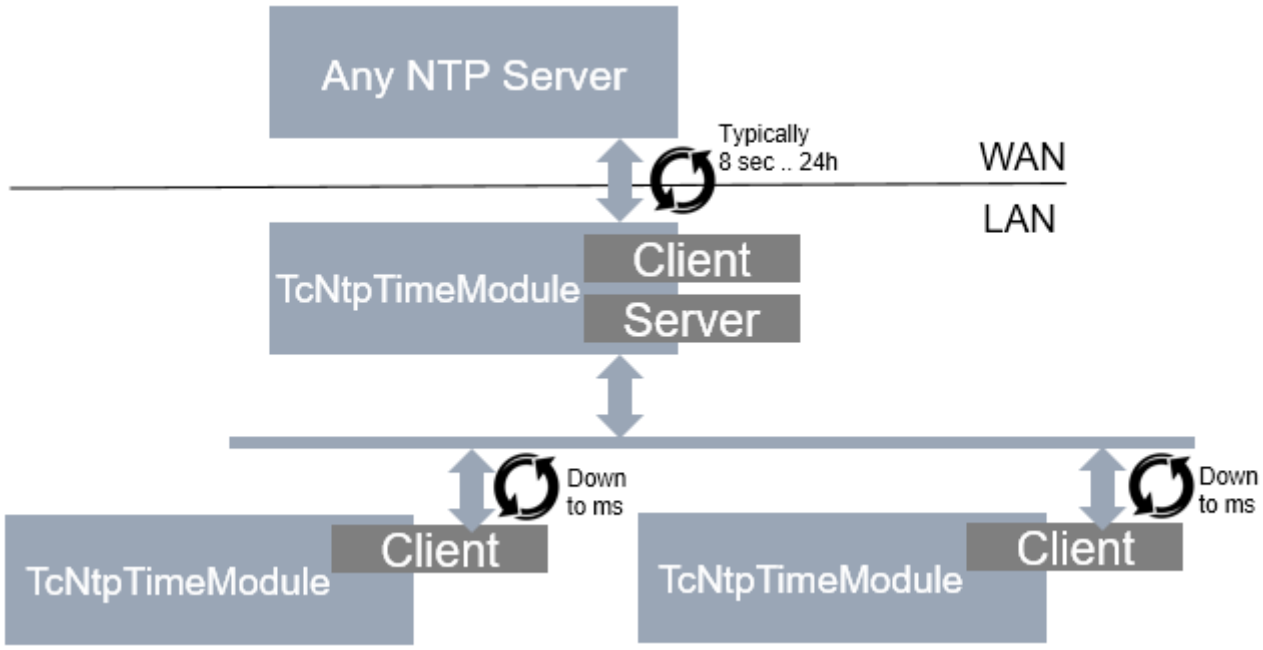
- **bEnable**：可禁用模块以防止 NTP 通信。
- **sServerName**用作源的 NTP 服务器名称。
- **nServerAddress**：NTP 服务器的 IP 地址（在 sServerName 为空时使用）。
- **nServerPort**：要使用的 NTP 服务器的 UDP 端口（默认：123）。
- **tPollIntervall**：开始 NTP 查询的时间间隔。服务器指定的最大值会被考虑在内，这可能会降低请求速度。

该模块通过 ITcSetExternalTime [▶ 199] 接口将确定的偏移传递给 TwinCAT。此外，输出还可用于映射。

NTP 提供程序作为 NTP 服务器

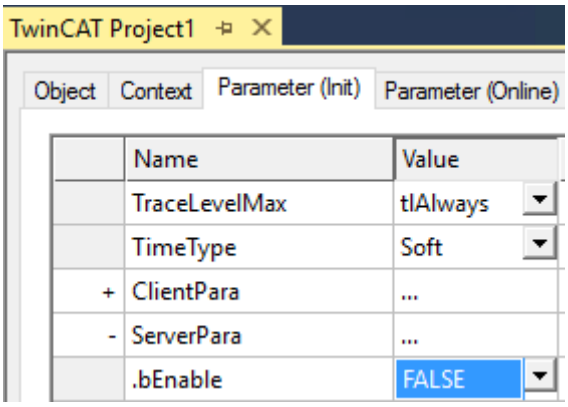
此外，同一模块还可充当 NTP 服务器。因此，可以从外部 NTP 服务器（作为客户端）获取时间信号，同时提供给下级系统。

对于外部服务器，NTP 协议通常要求至少 8 秒或更长的查询时间。而作为 NTP 服务器的 NTP 提供程序则允许更频繁的查询间隔。



服务器功能

服务器功能通常是隐藏的。可通过**显示隐藏参数**进行显示和配置：



- **bEnable**：启用此模块的 NTP 服务器功能。为此，请在 Windows 防火墙中打开 udp/123 端口。
- **nPort**：用于提供服务器的 UDP 端口（默认：123）。

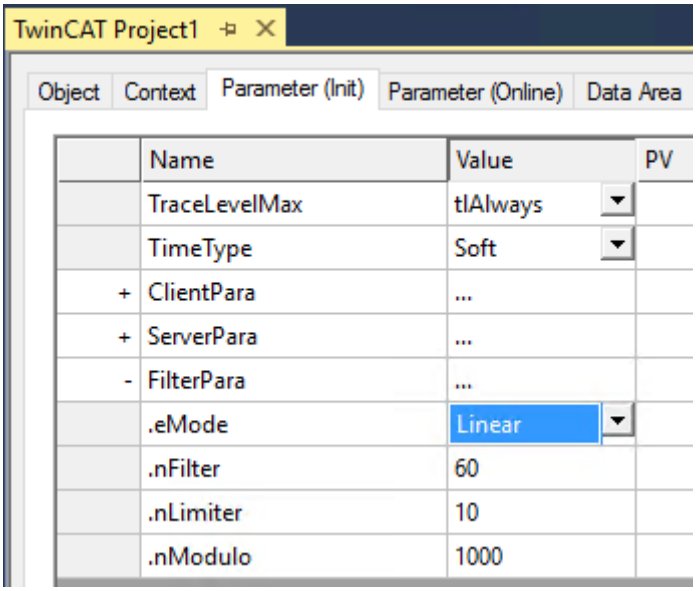
以下参数用于调整所提供的 NTP 信息。默认情况下，参数设置与协议中指定的一致；但可在此处进行重写：

- **nLeap**：手动配置 Leap Indicator。
- **nStratum**：手动配置地层。
- **nRoot**：手动配置根服务器信息，具体根据地层定义。

滤波功能

如果偏移由 NTP 服务器查询确定，模块可独立执行从旧偏移到新偏移的转换。

该功能通常是隐藏的。可通过**显示隐藏参数**进行显示和配置：



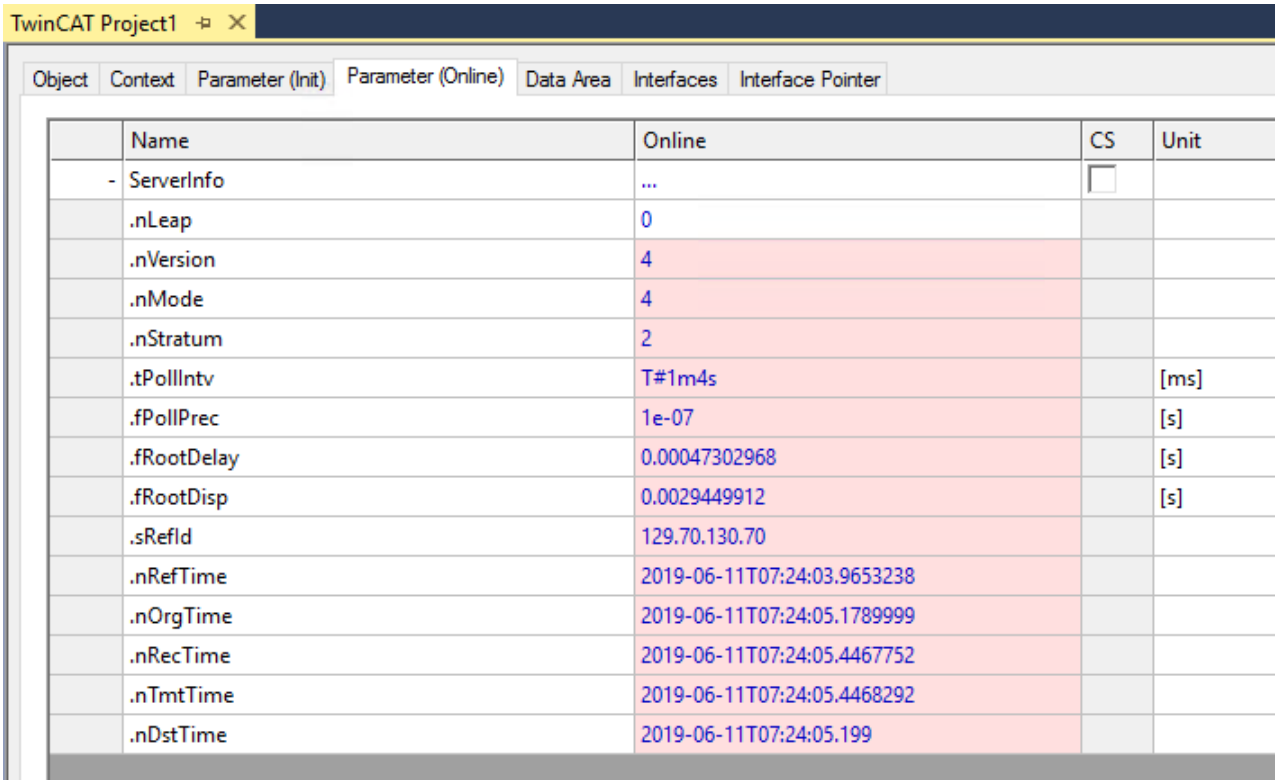
- **eMode**：多种模式可供选择。目前，要么不进行调整，要么进行线性调整（默认）。

如果选择"线性"作为 eMode，则适用以下参数：

- **nFilter**：取平均值的数值个数，即 NTP 响应的数量。如果轮询间隔为 1 秒，则 nFilter = 60 将产生一分钟的滤波效果。（默认：60）。
- **nLimiter**：偏移在每个循环内最多改变这个值。如果本地时钟和外部时钟的差值为 100 ms，循环时间为 1 ms，那么在 nLimiter = 10 的情况下，需要 100,000 个循环或 1.6 分钟，直到偏移解决为止。（默认：1 µs）。
- **nModulo**：偏移的四舍五入。通常应根据循环时间来选择。偏移通过该模除进行调整，这样就不会产生"非整数"时间。DC 时间将返回循环时间的模除；经偏移校正后，时间戳仍为"整数"。偏移/时间戳会随之改变，但如果进行调整，也会出现小幅跳变。如 nLimiter 下所述，在 nModulo = 1000 的情况下，偏移以及相对时间戳将每 100 个循环时间递增 0.1 ms。

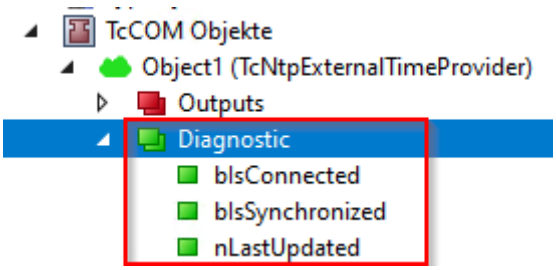
诊断

诊断信息可在**参数（在线）**选项卡下查看。



注释栏中对每一行都有相应的描述。

此外，相应的符号也可用于程序化评估：



3.6.4.2.2 DC 提供程序

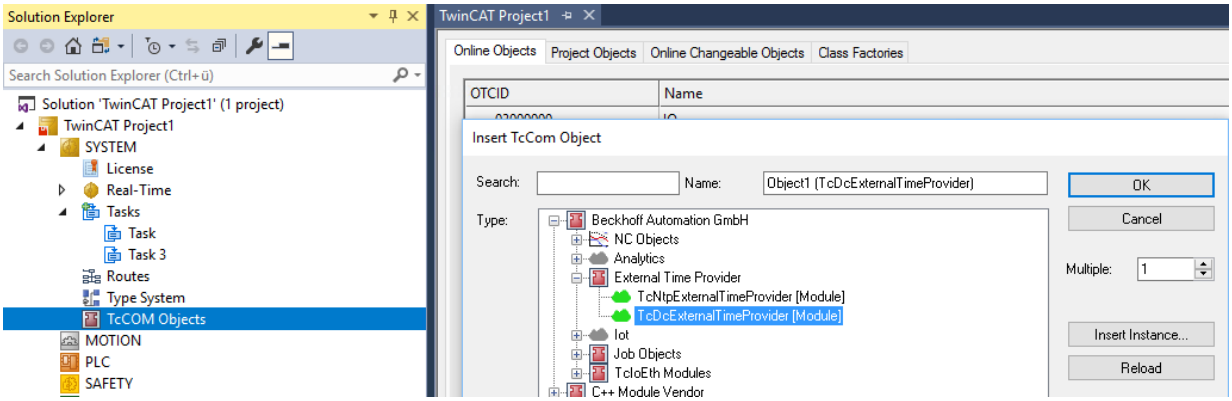
DC 提供程序通过映射从 EtherCAT 主轴获取偏移。它可用于使用 I/O 范围内的时间值作为偏移，如 EtherCAT 主轴（直流时间）或 EL6695 提供的时间值。

配置

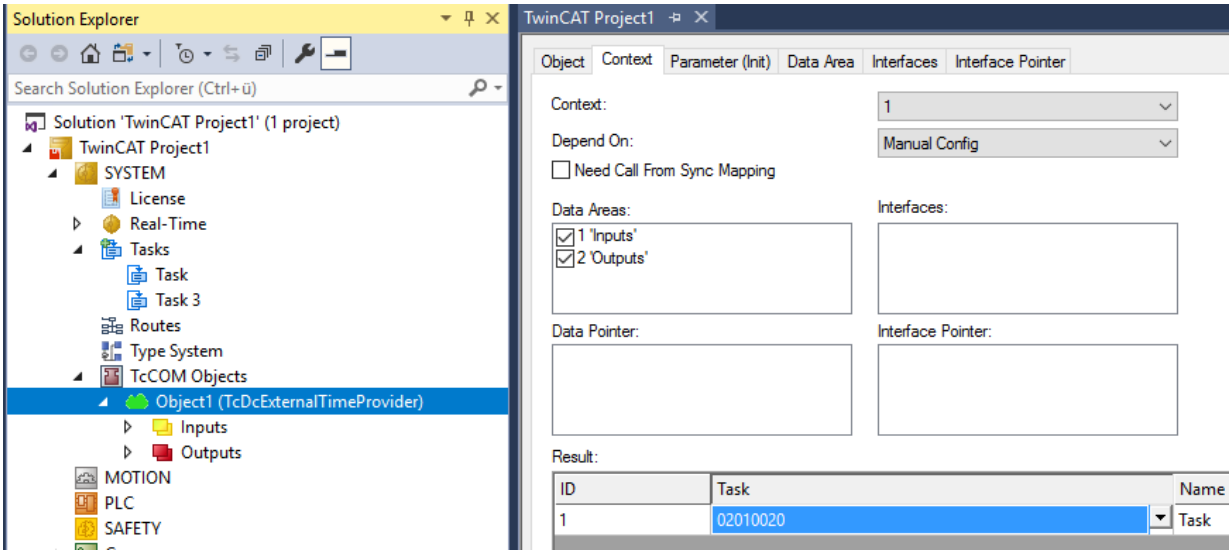
DC 提供程序作为 TcCOM 模块 TcDcExternalTimeProvider 实现。该模块作为 TcCOM 模块如下调试：

✓ TwinCAT 项目

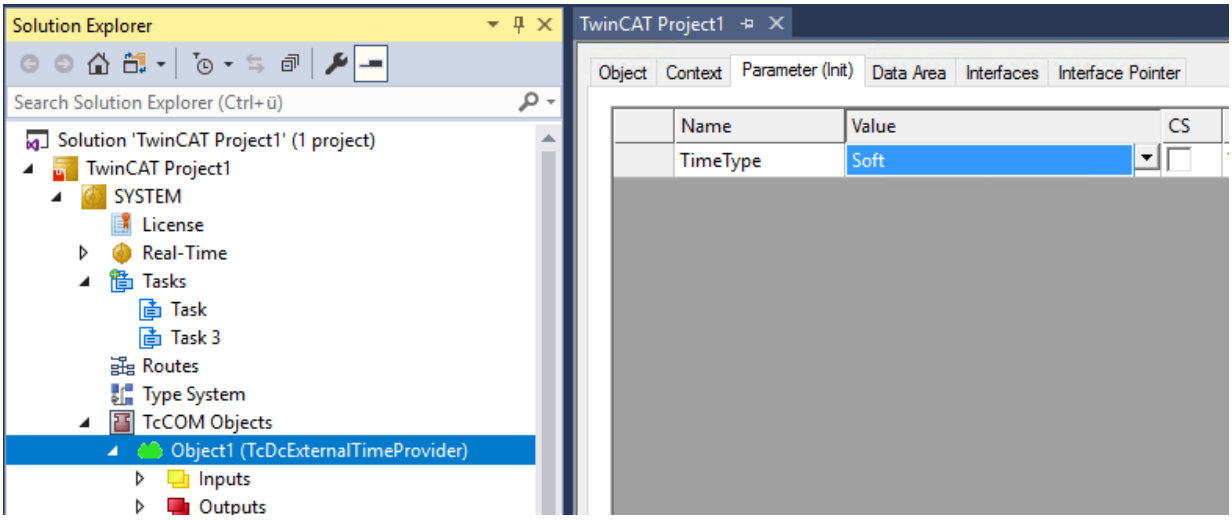
1. 在 System->TcCOM Objects 下插入 TcCOM 模块，并在 External Time Provider 类别中选择 TcDcExternalTimeProvider 类型。



2. 模块需要一个被调用的任务。这可通过模块的上下文（context）选项卡进行参数设置：



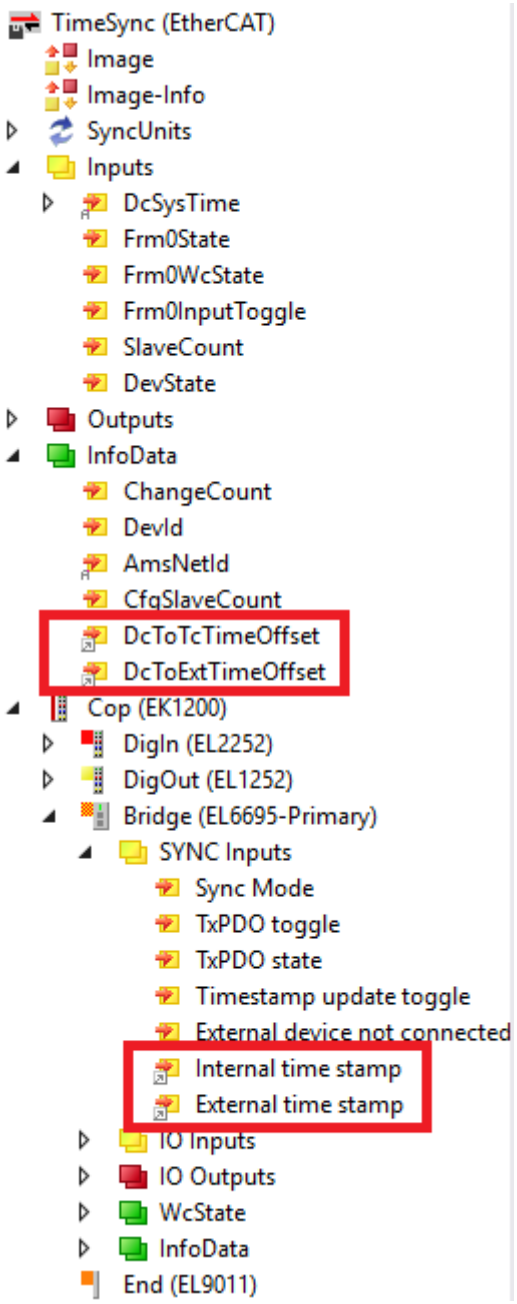
⇨ 可对模块进行参数设置：



参数配置在参数（Init）选项卡中进行。参数含义如下：

- TimeTime：本模块要确定偏移的偏移类型。

- 该模块通过映射获取偏移：



该模块通过 ITcSetExternalTime [▶ 199] 接口将确定的偏移传递给 TwinCAT。此外，输出还可用于映射。

3.6.4.2.3 应用实施

应用程序可以通过使用 TwinCAT C++ 中的 ITcSetExternalTime 接口来提供自己的时间偏移提供程序。

如有必要，该模块可为各自的偏移提供循环值。

顺序

模块实现以下序列

✓ TcCOM 模块已实例化

1. 模块通过 RegisterExternalTimeProvider 将自己注册为某类偏移（软/中/硬）的提供程序
2. 必要时，可使用 SetExternalTimeOffset 循环提供偏移
3. 模块使用 UnregisterExternalTimeProvider 登出

注册可确保一次只能使用一个模块的偏移。

有关 ITcSetExternalTime 接口的更详细说明，请参见 ITcSetExternalTime 接口 [▶ 199]一章。

3.6.5 实时 API

此时，接口和结构被记录下来，以处理来自实时的校正时间戳。

3.6.5.1 结构

3.6.5.1.1 枚举时间类型

TwinCAT 提供四种不同的时间戳。枚举时间类型用于区分它们。

语法

```
enum TimeType {
SystemTime = 0,
ExternalTimeHard = 1,
ExternalTimeMedium = 2,
ExternalTimeSoft = 3, // e.g. NTP
};
```

数值

这三种外部时间戳类型在实际中如何使用取决于应用。下方示例只是建议。

名称	描述
ExternalTimeHard	建议用于无漂移的硬偏移
ExternalTimeMedium	建议用于精确偏移，如 IEE1588
ExternalTimeSoft	建议用于一般偏移，如 NTP

3.6.5.2 接口

此时，将介绍用于校正时间戳的接口。

对于不同的时间格式和表示法，C++ SDK 中有相应的列表。
参见：InfosysC/C++

3.6.5.2.1 ITcSetExternalTime 接口

ITcSetExternalTime 接口由 TcCOM 对象服务器实现。它可用于提供外部确定偏移。

语法

```
TCOM_DECL_INTERFACE("00000067-0000-0000-e000-000000000064", ITcSetExternalTime)
struct __declspec(novtable) ITcSetExternalTime : public ITcExternalTime
```

方法

名称	描述
RegisterExternalTimeProvider [▶ 200]	为与 TimeType 有关的偏移注册提供程序
UnregisterExternalTimeProvider [▶ 200]	为与 TimeType 有关的偏移登出提供程序
SetExternalTimeOffset [▶ 200]	为已注册的 TimeType 提供新偏移

注释

该接口不适用于 PLC。

3.6.5.2.1.1 RegisterExternalTimeProvider 方法

为与 TimeType 有关的偏移注册提供程序

语法

```
HRESULT TCOMAPI RegisterExternalTimeProvider(OTCID oidProvider, TimeType type) = 0;
```

参数

oidProvider: （类型：OTCID）提供程序的对象 ID；通常是调用方的对象 ID

type: （类型：TimeType [▶ 199]）要注册的 TimeOffset 类型。

返回值

类型：HRESULT

通知注册成功

描述

3.6.5.2.1.2 UnregisterExternalTimeProvider 方法

为与 TimeType 有关的偏移登出提供程序

语法

```
HRESULT TCOMAPI UnregisterExternalTimeProvider(OTCID oidProvider, TimeType type) = 0;
```

参数

oidProvider: （类型：OTCID）提供程序的对象 ID；通常是调用方的对象 ID

type: （类型：TimeType [▶ 199]）要登出的 TimeOffset 类型。

返回值

类型：HRESULT

通知取消注册成功

描述

3.6.5.2.1.3 SetExternalTimeOffset 方法

为已注册的 TimeType 提供新偏移

语法

```
HRESULT TCOMAPI SetExternalTimeOffset(OTCID oidProvider, TimeType type, __int64 offset) = 0;
```

参数

oidProvider: （类型：OTCID）提供程序的对象 ID；通常是调用方的对象 ID

type: （类型：TimeType [▶ 199]）TimeOffset 类型

offset: （type: __int64 ）新偏移值。

返回值

类型：HRESULT

通知成功。

描述

它对偏移 ExternalTime = Internal Time + Offset 有效。也就是说，如果 TwinCAT 中的时间是过去时，偏移必须大于 0。

3.6.5.2.2 ITcExternalTime 接口

ITcExternalTime 接口由 TcCOM 对象服务器实现。这可用于检索和使用外部确定的偏移。

语法

```
TCOM_DECL_INTERFACE("00000066-0000-0000-e000-000000000064", ITcExternalTime)
struct __declspec(novtable) ITcExternalTime : public ITcUnknown
```

方法

名称	描述
SystemTimeToExternalTime [▶ 201]	计算与系统时间相关的校正时间戳
ExternalTimeToSystemTime [▶ 201]	根据校正时间戳计算系统时间
GetExternalTimeOffset [▶ 202]	检索与 TimeType 有关的偏移
GetExternalTimeProvider [▶ 202]	查询当前提供程序的对象 ID

3.6.5.2.2.1 SystemTimeToExternalTime 方法

计算与系统时间相关的校正时间戳

语法

```
HRESULT TCOMAPI SystemTimeToExternalTime(TimeType type, __int64& time) = 0;
```

参数

type：（类型：TimeType [▶ 199]）用于计算的 TimeOffset 类型

time：(type: __int64&) 通过偏移校正的时间戳

返回值

类型：HRESULT

通知成功。

描述

3.6.5.2.2.2 ExternalTimeToSystemTime 方法

根据校正时间戳计算系统时间

语法

```
HRESULT TCOMAPI ExternalTimeToSystemTime(TimeType type, __int64& time) = 0;
```

参数

类型：（类型：TimeType [▶ 199]）用于计算的 TimeOffset 类型

time:（type:__int64&）经过偏移调整的校正时间戳。

返回值

类型：HRESULT

通知成功。

描述

调用时有效的偏移用于确定本地系统时间。

3.6.5.2.2.3 GetExternalTimeOffset 方法

检索与 TimeType 有关的偏移

语法

```
HRESULT TCOMAPI GetExternalTimeOffset(TimeType type, __int64& offset) = 0;
```

参数

type：（type：TimeType [▶ 199]）要检索的 TimeOffset 类型

offset：（type: __int64& ）设置为偏移的值。

返回值

类型：HRESULT

通知成功。

描述

3.6.5.2.2.4 GetExternalTimeProvider 方法

查询当前提供程序的对象 ID

语法

```
HRESULT TCOMAPI GetExternalTimeProvider(TimeType type, OTCID& oidProvider) = 0;
```

参数

type：（type：TimeType [▶ 199]）要查询其提供程序的 TimeOffset 类型。

oidProvider： (type: OTCID&) 设置为提供程序对象 ID 的 ObjectID。

返回值

类型：HRESULT

通知成功。

描述

3.6.6 ADS API

还可以通过 ADS 查询 TimeOffsets。为此，有两种方法

1. ADS 通知：ADS 通知包含一个时间戳，其中包含数据更改的时间。
   ADS 客户端在 AddDeviceNotification 之前发送 ADS 命令。这样，目标系统就会登记该 ADS 客户端需要哪种类型的校正时间戳。
2. ADS 读取：可通过 ADS 读取读出校正时间戳。这可用于在 ADS Sum 命令中获取 ADS 命令执行时的校正时间戳。

索引组	索引偏移	访问	数据类型	描述	说明
外部时间 0xF088					
	adsioffs_externaltime_set 0x0000	R	LONG	读取各 ADS 客户端当前配置的偏移类型（AmsNetAddr，包括客户端端口）。	返回值是类型 0 = 无，1 = 硬，2 = 中，3 = 软
	adsioffs_externaltime_set 0x00__	W		设置各 ADS 客户端 ADSDevice 通知的偏移类型（AmsNetAddr，包括客户端端口）。	__ 是类型 0 = 无，1 = 硬，2 = 中，3 = 软
	adsioffs_externaltime_offset 0x01__	R	LONGLONG	读取类型的当前偏移。	__ 是类型：0 = 无，1 = 硬，2 = 中，3 = 软
	adsioffs_externaltime_offset 0x01__	W	LONGLONG	设置类型的当前偏移。	__ 是类型：0 = 无，1 = 硬，2 = 中，3 = 软
	adsioffs_externaltime_absolute 0x02__	R	LONGLONG	读取校正时间戳。	__ 是类型：0 = 无，1 = 硬，2 = 中，3 = 软
	ADSIOFFS_EXTERNALTIME_PROVIDER 0x03__	R	ULONG	从 TimeOffset 提供程序读取对象 ID。	__ 是类型：0 = 无，1 = 硬，2 = 中，3 = 软
	ADSIOFFS_EXTERNALTIME_SETALL 0x0400	R	LONG	读取未设置其他类型时使用的类型。	返回值是类型 0 = 无，1 = 硬，2 = 中，3 = 软
	adsioffs_externaltime_setall 0x04__	W		设置在未设置其他类型时使用的类型。	__ 是类型 0 = 无，1 = 硬，2 = 中，3 = 软

这些定义可以在"Ads.h"文件中找到。

ADS 消耗程序样例 [▶ 204]说明了该应用。

3.6.7 示例

为方便用户使用，现提供各种样例，说明校正时间戳的使用方法：

- PLC 消耗程序 [▶ 204]：PLC 程序访问校正时间戳。
- C++ 消耗程序 [▶ 205]：C++ TcCOM 模块访问校正时间戳。
- ADS 消耗程序 [▶ 204]：用户模式下的 ADS 客户端访问校正时间戳。
- C++ 提供程序 [▶ 205]：C++ TcCOM 模块确定偏移并提供偏移。

校正时间戳也会被 TwinCAT 系统的其他组件使用。所需的配置可在相应组件中找到。

3.6.7.1 ADS 消耗程序

ADS 消耗程序样例会按照 ADS API [▶ 202] 中的描述检索校正时间戳。

下载

您可以在此访问该样例的https://infosys.beckhoff.com/content/1033/tc3_Grundlagen/Resources/7705550603.zip。

✓ 启动与 ADS 消耗程序样例进行通信的 TwinCAT 目标系统。可使用 PLC 消耗程序 [▶ 204]样例。

1. 解压下载的 ZIP 文件。
2. 打开 Visual Studio 中包含的 vcxproj 文件。
3. 调整目标 AmsNetID。（TcExternalTimeAdsClient.cpp 第 119 行）

⇨ 样例已做好操作准备。

描述

样例代码见 CPP 文件 TcExternalTimeAdsClient.cpp

Main() 方法说明了接收校正时间戳的不同用例：

- 从系统服务读取不同偏移的提供程序、偏移和校正时间戳：未校正（0）、软（1）、中（2）、硬（3），加上一个无效值（4），以说明错误行为。
- 从 PLC 程序中读取不同偏移的校正时间戳。
- 读取使用的提供程序和所有提供程序。
- 订阅 PLC 中的任意变量；通过通知提供的时间具有校正时间戳。输出在 AdsNotificationCallback() 方法中进行。

3.6.7.2 PLC 消耗程序

PLC 消耗程序样例从 TwinCAT 系统中获取校正时间戳并使用。

下载

您可以在此访问该样例的https://infosys.beckhoff.com/content/1033/tc3_Grundlagen/Resources/7705583115.zip。

1. 点击**打开项目 ....**，在 TwinCAT 3 中打开其中包含的 tszip 文件
2. 选择目标系统。
3. 在本地机器上构建样例（例如，**Build->Build Solution**）。
4. 点击  激活配置。

⇨ 样例已做好操作准备。

描述

TcNtpExternalTimeProvider 在**系统 > TcCOMObjects** 下配置。
如果无法访问默认的 pool.ntp.org，您可以在**参数（Init）**下对自己的 NTP 服务器进行参数设置。

PLC 程序主要由功能块 FB_TcExternalTime 组成。它提供从 TwinCAT 系统读取校正时间戳的功能。变量 _eTimeType 表示类型（软、中、硬），可以参数设置。

在 MAIN 中，该功能块用于 eTimeType"软"，以确保使用 NTP 设置的校正时间。

3.6.7.3 C++ 消耗程序

C++ 消耗程序样例从 TwinCAT 系统中获取并使用校正时间戳。

下载

您可以在此访问该样例的https://infosys.beckhoff.com/content/1033/tc3_Grundlagen/Resources/7705552907.zip。

1. 单击"**打开项目 .... "**，在 TwinCAT 3 中打开其中包含的 zip 文件
2. 选择目标系统。
3. 在本地机器上构建样例（例如，**Build->Build Solution**）。
4. 点击  激活配置。

⇨ 样例已做好操作准备。

描述

TcNtpExternalTimeProvider 在**系统 > TcCOMObjects** 下配置。

如果无法访问默认的 pool.ntp.org，您可以在**参数（Init）**下对自己的 NTP 服务器进行参数设置。

C++ 模块在 CycleUpdate() 方法中循环确定本地时间戳，并进行校正。可以使用调试器在相应步骤中进行追踪。校正时间戳作为参数提供（在线）。

所需的类型可作为参数"TimeType"在 TcCOM 对象中进行配置。

3.6.7.4 C++ 提供程序

C++ 提供程序样例会确定一个偏移，并将其存储在 TwinCAT 系统中，以便消耗程序使用。

下载

您可以在此访问该样例的https://infosys.beckhoff.com/content/1033/tc3_Grundlagen/Resources/7705555211.zip。

1. 解压下载的 .zip 文件。
2. 点击**打开项目 ....**，在 TwinCAT 3 中打开其中包含的 .zip 文件。
3. 选择目标系统。
4. 在本地机器上构建样例（例如，**Build->Build Solution**）。
5. 点击  激活配置。

⇨ 样例已做好操作准备。

描述

偏移提供程序接收要作为数据区域"ExternalTime.nOffset"提供的偏移。这将作为 TimeType 介质传输到 TwinCAT 系统，也可在运行时在**参数（Init）**下进行配置。

在 CycleUpdate() 方法中，使用 RegisterExternalTimeProvider 为 TimeType 创建相应寄存器后，SetExternalTimeOffset 方法将用于此目的。

3.6.8 常见问题

3.6.8.1 Windows 作为 NTP 客户端

Windows 本身就提供了一个适用于系统时间的 NTP 客户端。此外，还可以使用以下脚本检索 NTP 时间，这对调试非常有用：

```
@echo off
set /p Server=Server:
w32tm /stripchart /computer:%Server% /packetinfo /samples:10
pause
```

3.6.8.2 Windows 作为 NTP 服务器

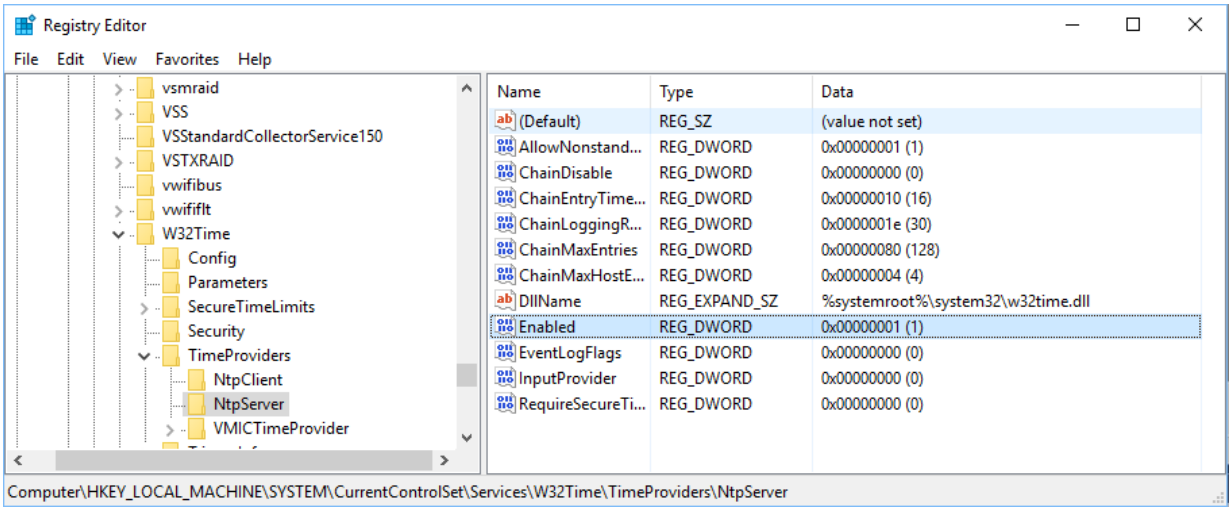
Windows 本身提供了一个 NTP 服务器来提供时间戳。

请注意，只有一个组件可以使用 NTP 端口（udp/123）。这意味着可以使用 TwinCAT NTP 服务器功能 [▶ 192]或 Windows NTP 服务器。

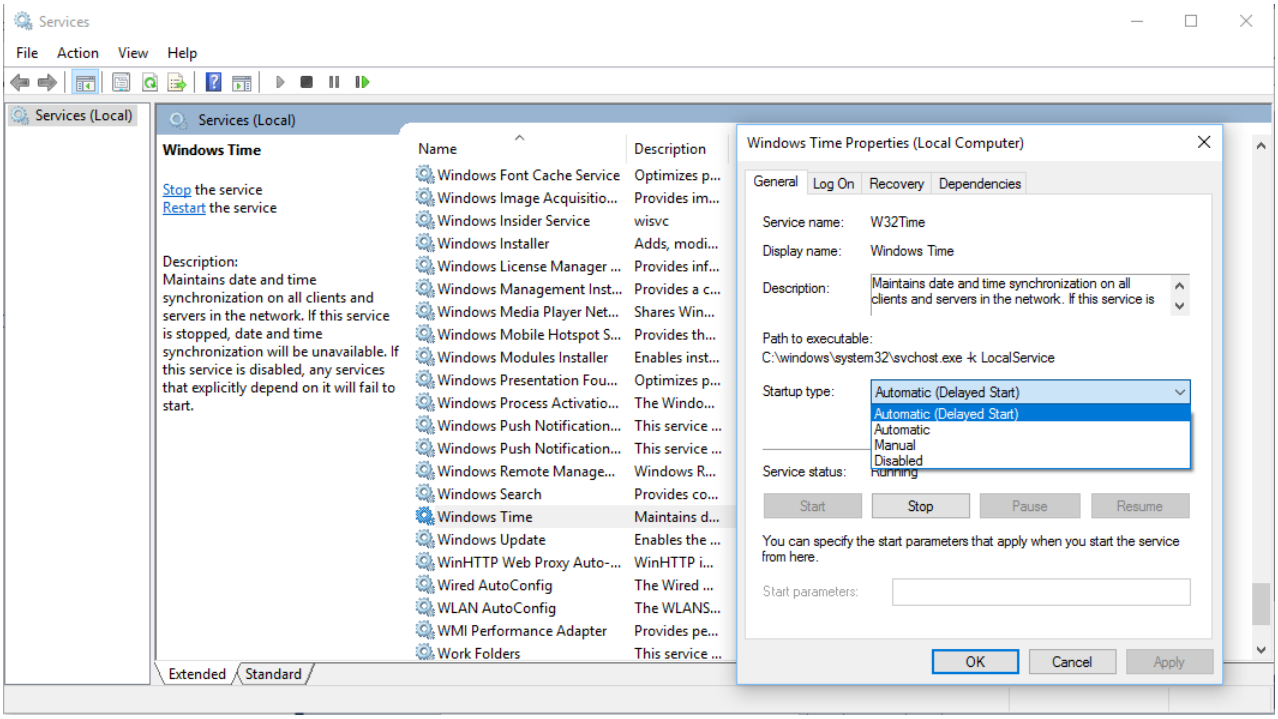
Windows NTP 服务器默认为禁用，可之后激活：

✓ Windows 10

1. 注册表键已设置：
   `HKLM\System\CurrentControlSet\Services\W32Time\TimeProviders\NtpServer`
   `Enabled = 1`



2. 启动 Windows Time 系统服务，并酌情将其设置为自动启动。



3.7 TcRTeInstall

TcRTeInstall 工具可管理控制系统的实时以太网兼容设备。这需要为控制系统的标准以太网控制器连接安装能够处理实时任务的驱动程序。

需要安装 TwinCAT 3

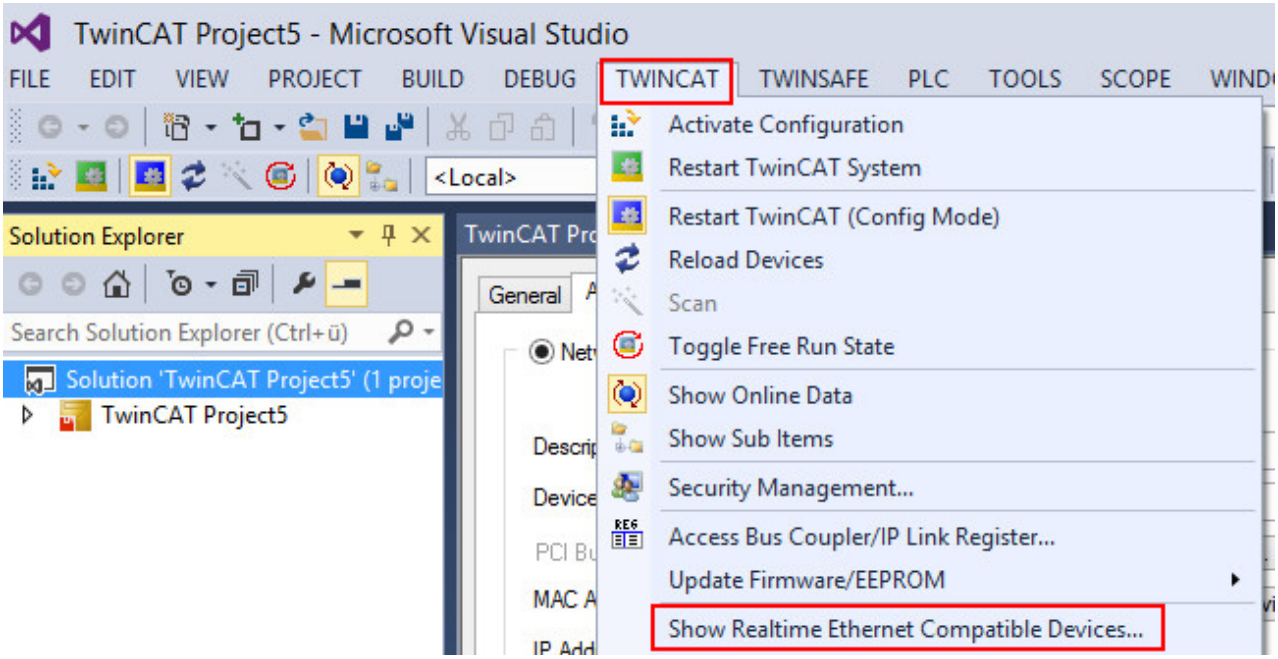
TcRTeInstall 应用程序只能与完整安装的 TwinCAT 3（XAE、运行时环境、XAR）结合使用。

需要管理员权限

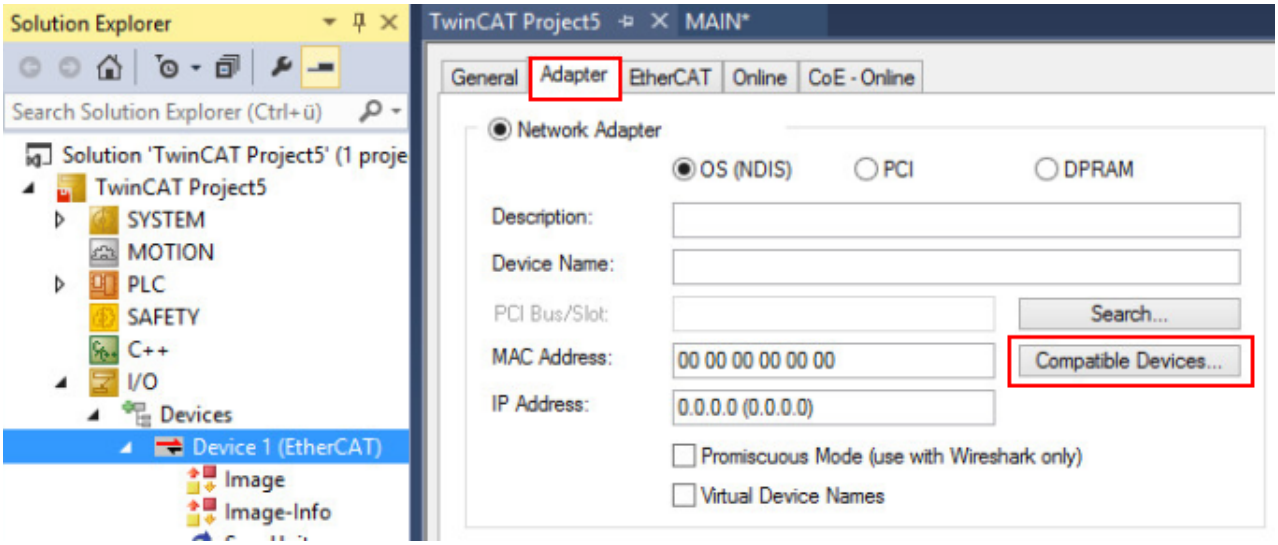
要运行 TcRTeInstall 应用程序，需要控制系统管理员权限。

在 TwinCAT 3 XAE 中调用

通过菜单 **TWINCAT → 显示实时以太网兼容设备……** 调用驱动程序 .



或者，您也可以通过在 I/O 配置中添加具有网络功能的设备（如 EtherCAT 设备）来安装驱动程序。在网络支持设备的适配器对话框中，通过**兼容设备……**按钮调用 TcRTeInstall 应用程序：



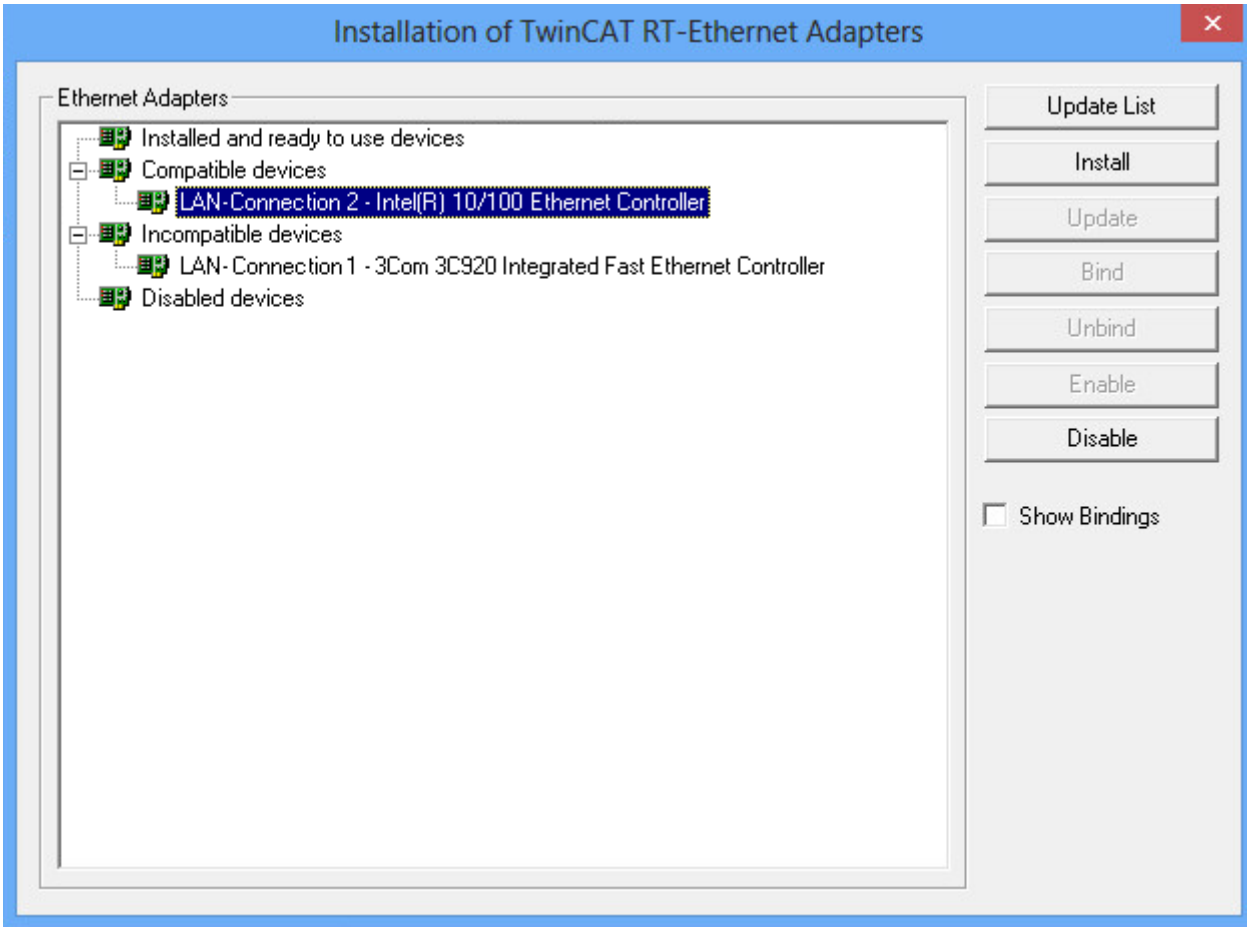
在运行时环境中调用

您可以在 TwinCAT 3 运行时系统上直接调用 TwinCAT RT 以太网适配器的安装应用程序。

位置：c:\TwinCAT3.1\System\TcRteInstall.exe

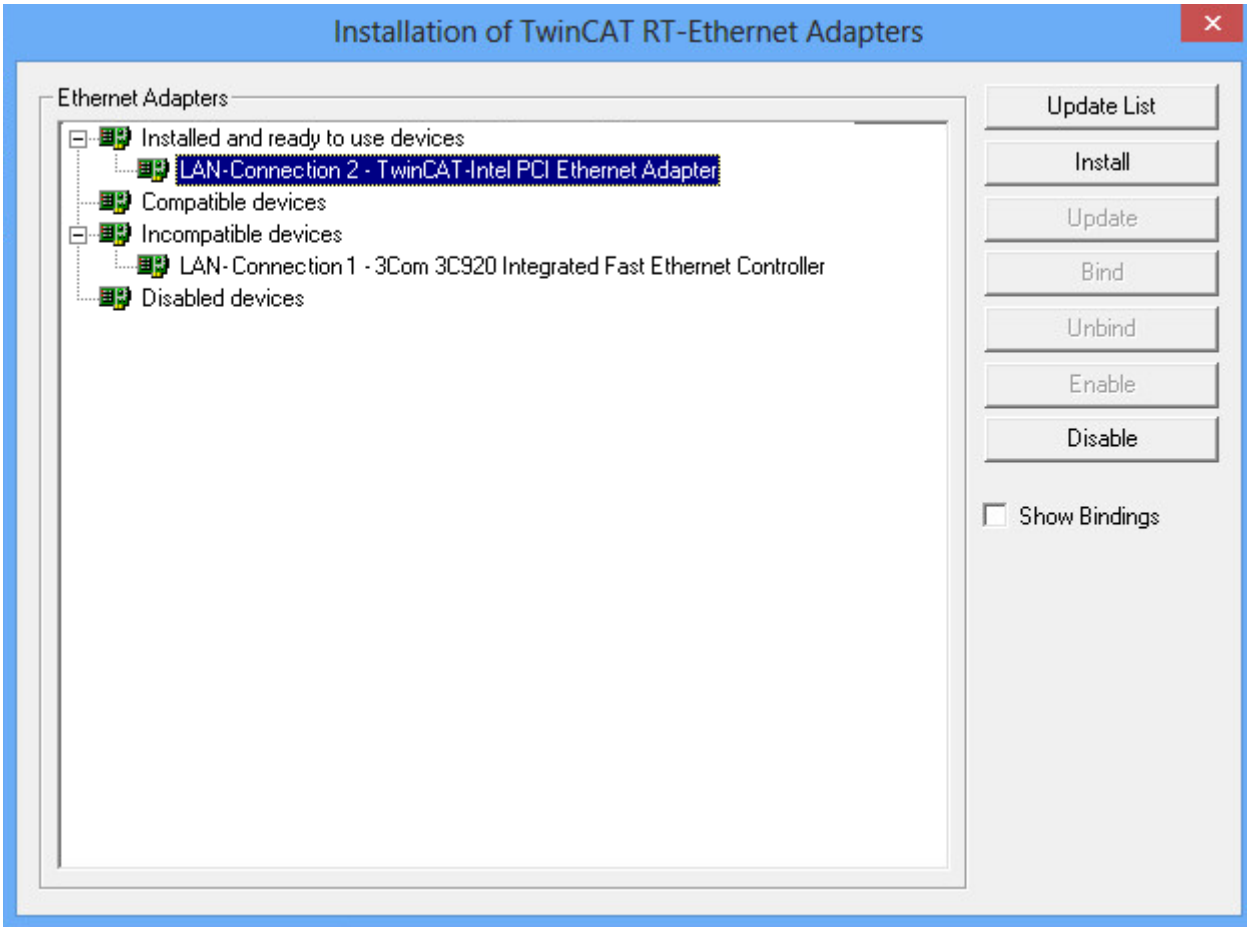
管理网络连接

TcRteInstall 会显示支持实时的（Compatible devices）和不支持实时的（Incompatible devices）网络接口卡。

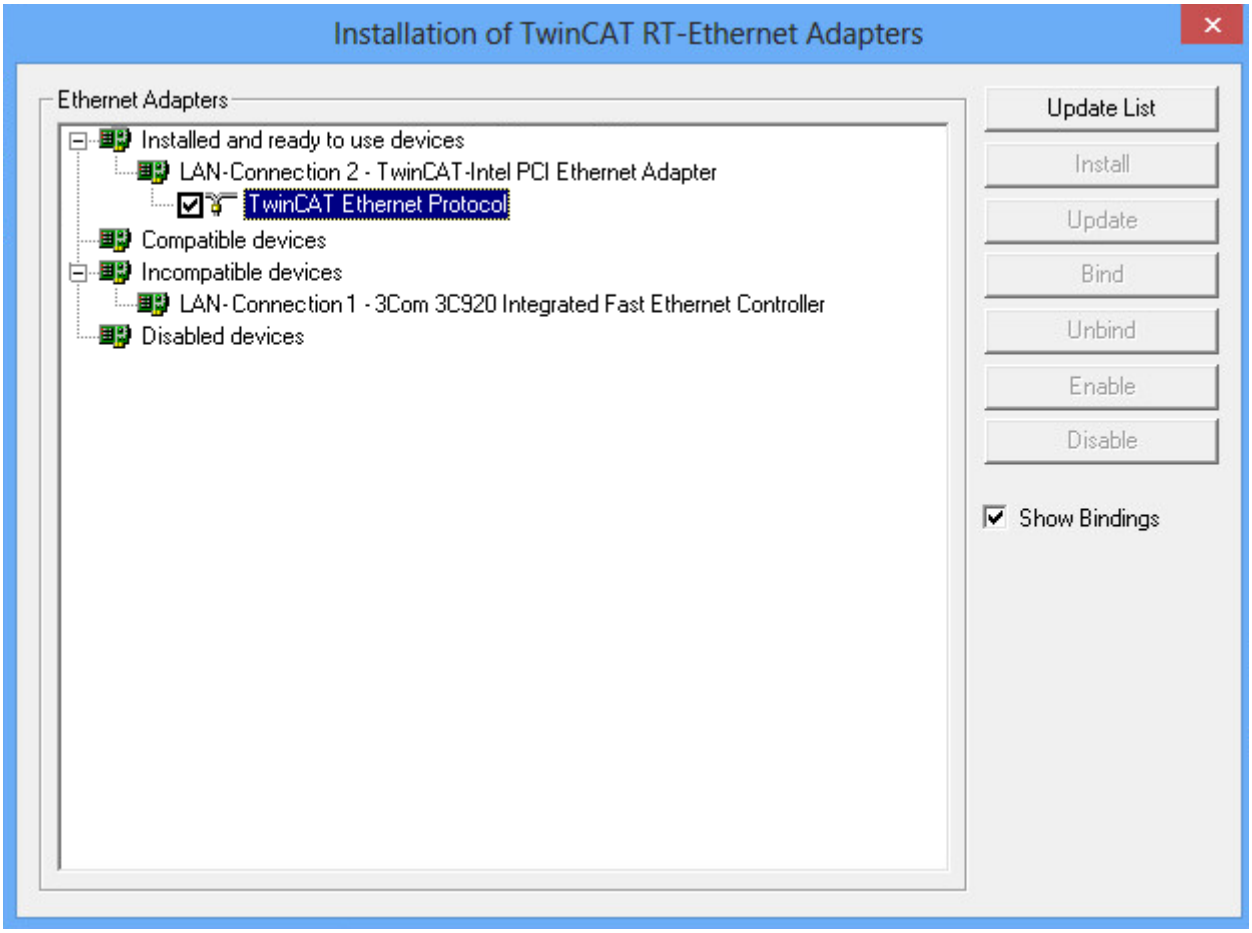


1. 从"兼容设备"列表中选择具有实时功能的网络接口卡。
2. 点击安装按钮。

⇨ 为所选设备安装用于实时以太网的 TwinCAT 驱动程序和 TwinCAT 以太网协议。



选项**显示绑定**显示已安装 RT 以太网设备的连接协议。


Installation of TwinCAT RT-Ethernet Adapters
Ethernet Adapters
Installed and ready to use devices
LAN-Connection 2 - TwinCAT-Intel PCI Ethernet Adapter
TwinCAT Ethernet Protocol
Compatible devices
Incompatible devices
LAN-Connection 1 - 3Com 3C920 Integrated Fast Ethernet Controller
Disabled devices
Update List
Install
Update
Bind
Unbind
Enable
Disable
Show Bindings

4 类型系统

TwinCAT 3 提供了一个用于管理数据类型的类型系统。类型系统由系统基本类型组成，可通过客户项目扩展自定义数据类型。

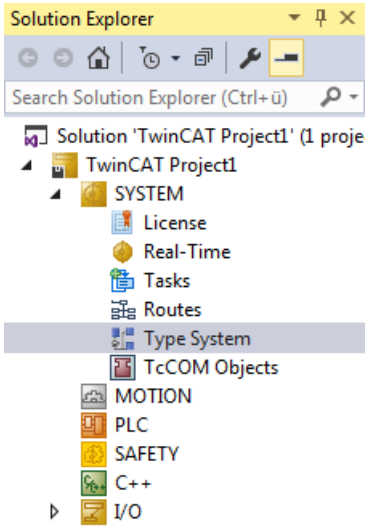
本文档介绍 TwinCAT 3 类型系统和数据类型管理。用于创建和描述数据类型的 TMC 编辑器在名为"C++"的文档中的 TwinCAT 模块类编辑器 (TMC) 部分中有所描述。

4.1 基于项目的类型系统

TwinCAT 3 类型系统是特定于项目的；也就是说，它是 Visual Studio 解决方案中 TwinCAT 3 项目的一个固定组件。

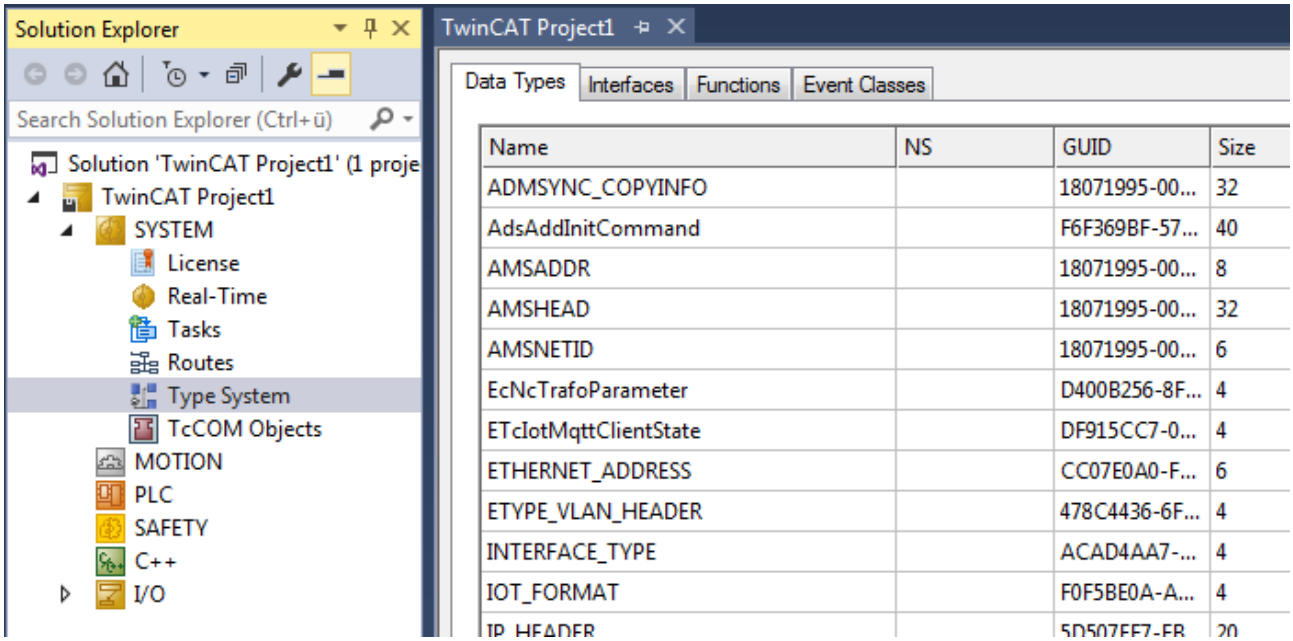
数据类型可以在不同的点上进行定义，并在必要时转移到 TwinCAT 3 类型系统中。因此，TwinCAT 3 类型系统中不存在的本地数据类型也可以存在。

在 TwinCAT 3 项目树中，类型系统是 SYSTEM 子树中的一个对象。



4.2 数据类型

TwinCAT 3 类型系统通过编辑器在四个不同的选项卡上显示数据类型。双击 TwinCAT 3 项目树中的"类型系统"对象可打开编辑器。



数据类型（Data Types）选项卡上显示以下数据类型（TMC 编辑器："规范"）：

- 别名：这些数据类型只是其他数据类型的同义词。例如，可以在特定项目中将时间范围（持续时间）定义为 UINT。
- 结构：这些数据类型是其他数据类型的结构，而其他数据类型也可以是结构。
- 枚举：这些数据类型描述枚举。
- 数组：这些数据类型是数组，具有定义的尺寸和各自的长度。

接口显示在**接口**选项卡上。该数据类型描述了可由功能块或 TcCOM 模块等不同组件提供或使用的接口。接口由具有各自签名的方法组成。

功能（Functions）选项卡显示从 TMC/TML 文件中读取定义的 PLC 功能和 PLC 功能块。

事件类选项卡定义了用于 TwinCAT 3 EventLogger 的事件类。

4.3 数据类型的处理

要通过 TwinCAT 3 类型系统（type system）创建或修改数据类型，请从类型系统编辑器相应选项卡上第一个表列的上下文菜单中选择**添加新项**或**编辑**命令。这两个命令都会打开 TMC 编辑器，您可以在其中编辑数据类型。

PLC 项目的数据类型

可在 PLC 项目中创建和保存数据类型 (DUT)。这些数据类型最初存在于 PLC 项目的本地，从 TwinCAT 3 的角度来看是不可用的。如果这些数据类型被用于输入/输出内存映射，它们就会被导入到 T(%I* / %Q*)winCAT 3 类型系统中，这样它们也可以通过映射进行链接。

通过 PLC 项目树中 DUT 上下文菜单中的**转换为全局类型**命令，可以将 DUT 转换为更高级 TwinCAT 项目的类型系统。此后，数据类型可通过外部类型在 PLC 中使用，并在 TwinCAT 3 类型系统（type system）中进行管理。

要将数据类型从 TwinCAT 3 类型系统转移到 PLC 项目，可以使用"数据类型（Data Types）"对话框中的源代码。

来自 C++ 项目的数据类型

在 C++ 项目中，数据类型与模块同时在 TMC 编辑器中定义。与 PLC 项目中的内部 DUT 一样，这些数据类型是本地的，因此在 TwinCAT 3 类型系统中是不可见的。

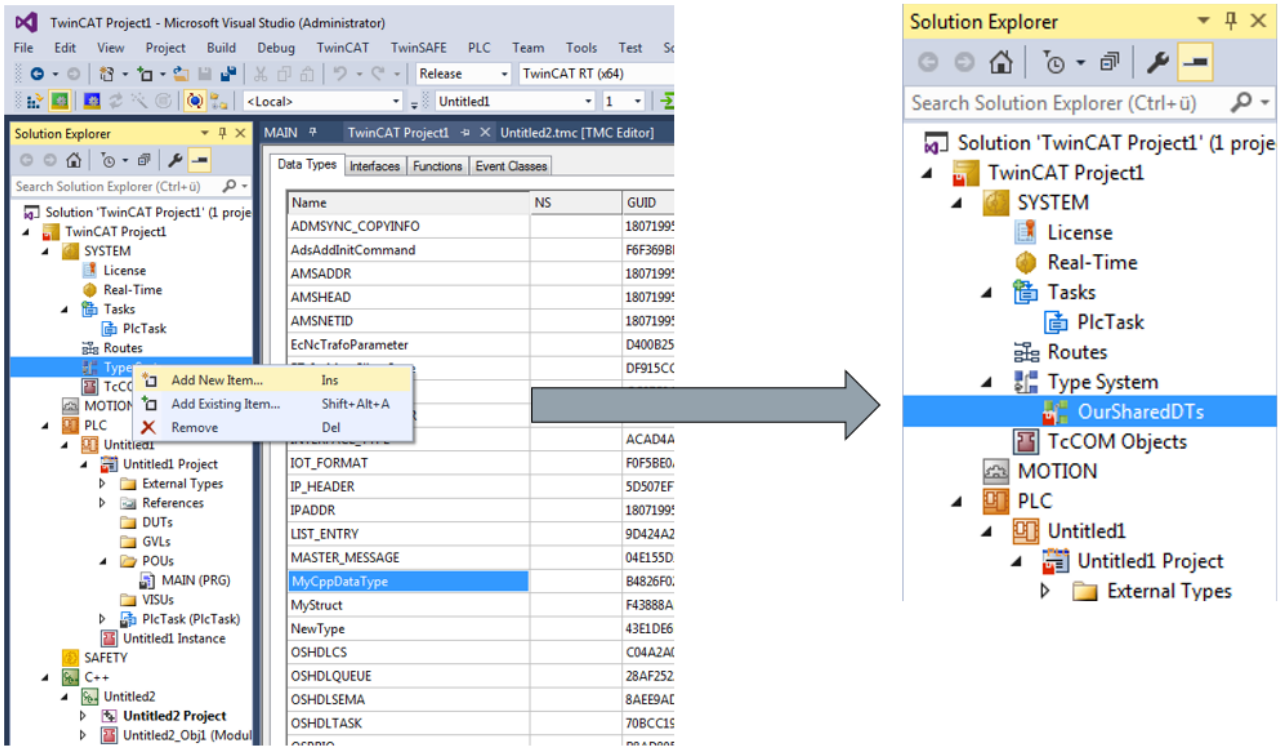
通过在 C++/Matlab 模块中使用数据类型（该模块也已实例化），数据类型被插入到 TwinCAT 3 的类型系统中。

您也可以通过激活**持久（即使未使用）**复选框，将数据类型插入 TwinCAT 3 类型系统，而无需在实例化的 C++ 模块中使用该数据类型。

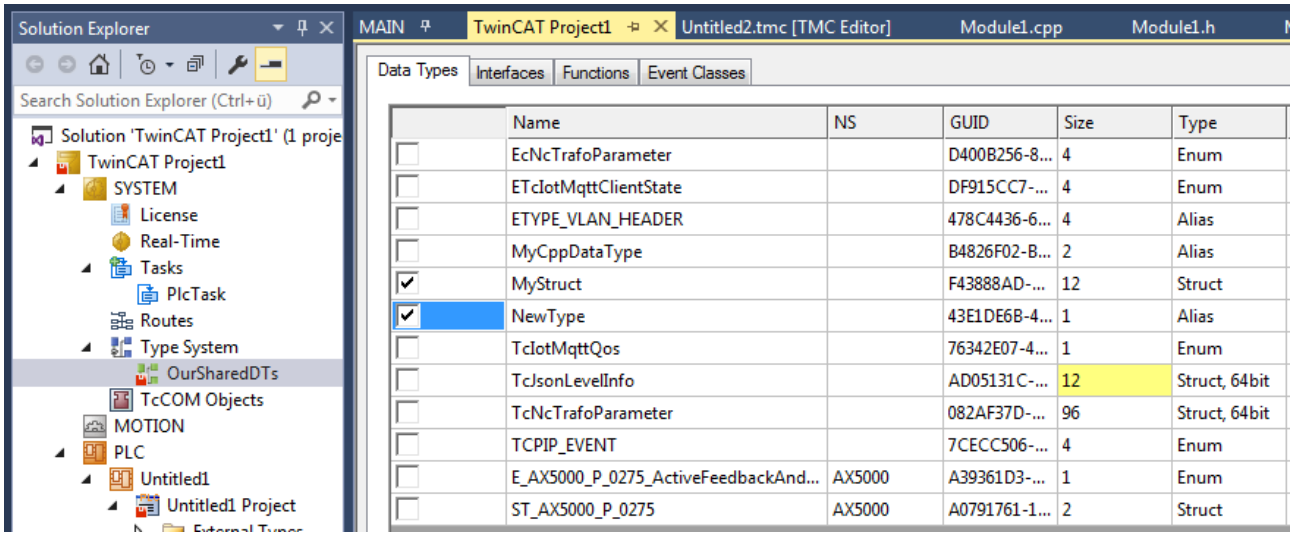
在多个项目中使用数据类型

在某些情况下，在多个项目中使用数据类型可能很有用。特别是对于 EAP/网络变量，在发布方和订阅方使用相同的数据类型可能非常有用。

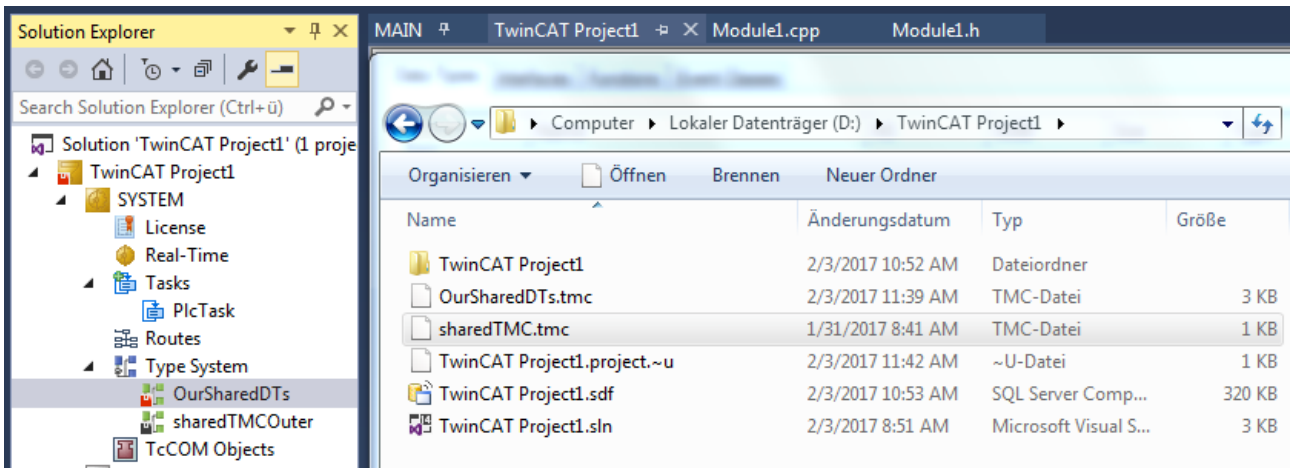
您可以在"类型系统（Type System）"节点下为此创建单独的 TMC 文件。



在 TMC 文件的编辑窗口中，每种数据类型前面都会出现一个复选框。使用复选框可以指定将哪种数据类型存入相应的 TMC 文件。



这些数据类型会另外存入 TMC 文件。您可以在不同的电脑上和不同的项目中使用这些文件，例如通过文件交换或版本控制的方式。
但是，文件本身不得同时被不同的项目使用，因此，这些文件通常存储在项目目录中，然后这个项目在不同的计算机上可以作为副本使用，例如通过版本控制。



由于 GUID 是用来识别数据类型的，因此类型系统会自动识别这种双重沉积。

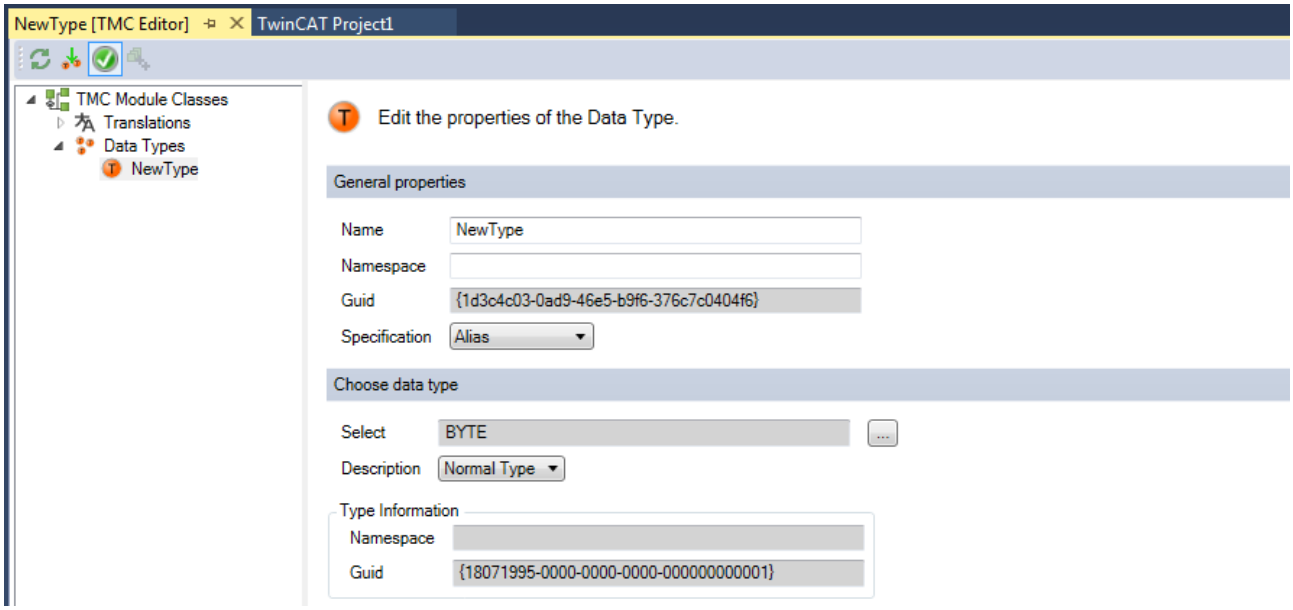
在多个项目中集成数据类型后使用它们时，应确保尽可能只在一个地方对数据类型进行更改。否则，不同的变量就无法再合并为一个共同版本。

另请参见：

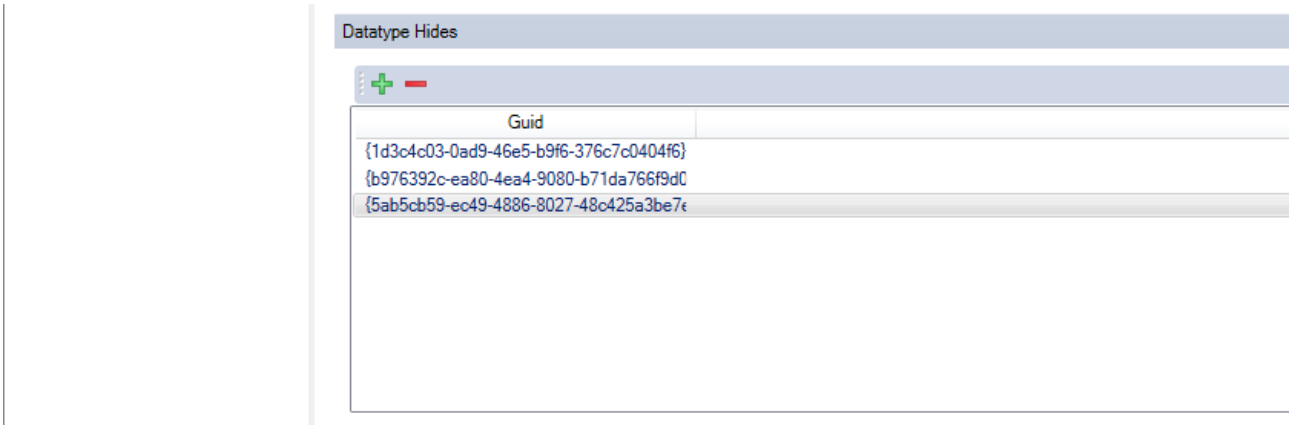
数据类型的管理和识别 [▶ 215]

4.4 数据类型的管理和识别

TwinCAT 3 类型系统中的数据类型基本上是根据其 GUID 来识别的。因此，可以存在多个名称相同的数据类型。这同样适用于数据类型的不同版本。数据类型的每个版本都会获分配一个新 GUID。



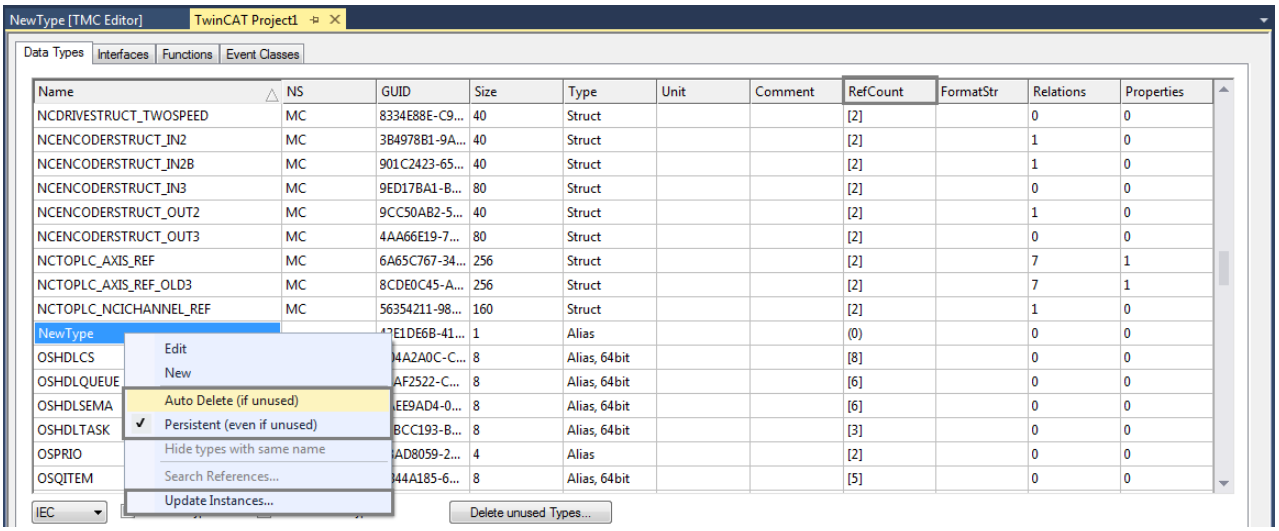
同时，每个数据类型都有一个隐藏数据类型的列表（"数据类型隐藏（Datatype Hides）"）。



这使得在项目中同时使用不同版本的数据类型成为可能。

在类型系统编辑器（**数据类型（Data Types）**选项卡）中，数据类型的上下文菜单中的**更新实例…… （Update Instances...）**命令会为数据类型的选定用途分别使用最新版本。

TwinCAT 对每种数据类型都有一个所谓的参考计数器。这个计数器可以在类型系统编辑器的 **RefCount** 栏中看到。在项目和编辑器等中每使用一次数据类型，计数器就会递增一次。如果计数器为 0，则不再使用该数据类型，并将其丢弃。



如果激活了数据类型上下文菜单中的**持久（即使未使用）（Persistent (even if unused)）**设置，即使在 TwinCAT 项目中未使用该数据类型，数据类型说明也将保存在 TwinCAT 项目文件 (*.tsproj) 中。默认情况下，通过类型系统编辑器直接新建的数据类型会激活该设置。这样可以确保在使用新数据类型之前保存 TwinCAT 项目时，数据类型不会被直接删除。

如果在 TwinCAT 项目树中**类型系统（Type System）**对象的下方使用 SharedTMC，则不应针对该文件中的数据类型启用该设置，因为数据类型既保存在项目中，也保存在 SharedTMC 中。对于通过 SharedTMC 编辑器直接新建的数据类型，该设置默认为停用。

自动删除（Auto Delete）（如果未使用）设置不应手动更改，但为了完整起见还是要显示。激活此设置的数据类型在 PLC 项目中被隐藏，不能在 PLC 项目中使用。例如，该设置不应用于自动清理类型系统。未使用的数据类型不会自动保存在 TwinCAT 项目中，因此在重新加载 TwinCAT 项目后，这些数据类型将不再存在于类型系统中。

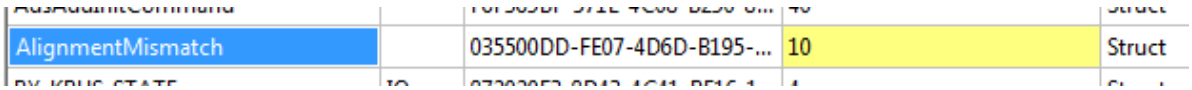
4.5 数据类型对齐

数据类型的内存布局由对齐方式决定。有关对齐的更多信息，请参见名为"PLC"的文档中的"对齐方式"部分。

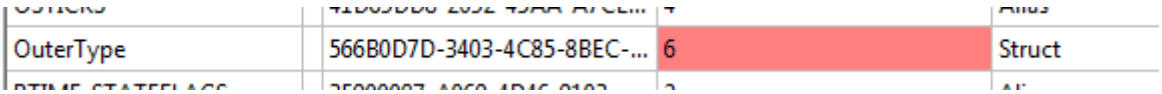
默认对齐方式为 8 字节，可确保在不同平台上访问数据类型时，在运行时和访问方面都能发挥最佳功能。只有在特殊情况下才能违反这一规定。

TwinCAT 3 类型系统用颜色标记数据类型。

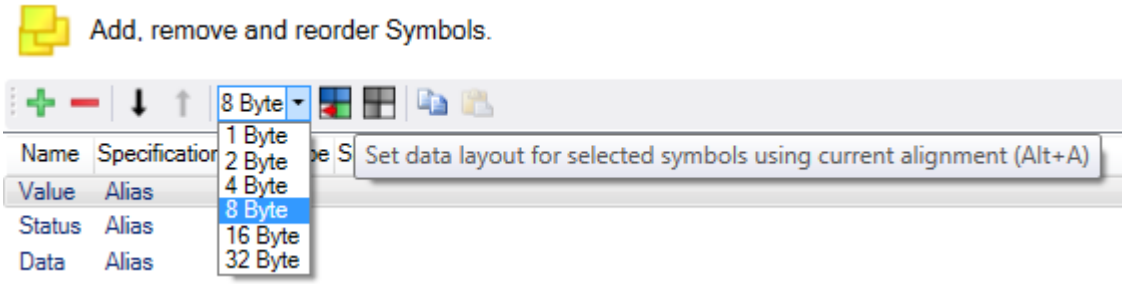
- 如果数据类型的长度不是最大内部字段（最多 8 字节）的倍数，则显示黄色。因此，对于这种数据类型的数组，对齐方式不再符合规则。



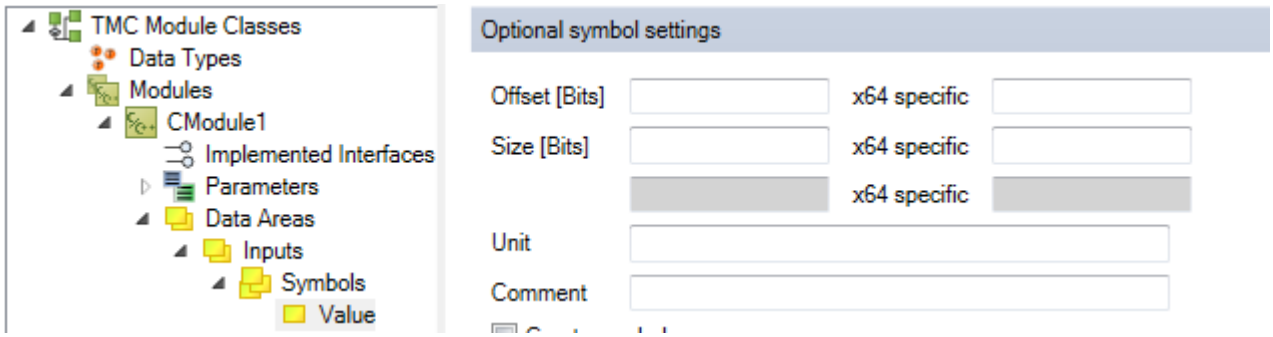
- 如果数据类型中的对齐方式不再符合规则，则显示红色。



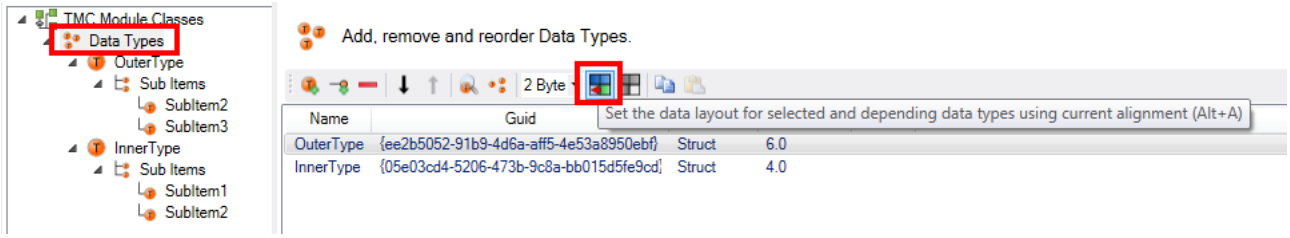
TMC 编辑器可为选定的对齐方式指定数据类型的内存布局。



或者，也可以使用偏移手动指定布局。



如果用于另一种数据类型的数据类型大小发生变化，那么这种数据类型也必须进行调整。为此，TMC 编辑器在数据类型概览级别提供了适当的递归函数。



4.6 与类型系统连接的文件

TwinCAT 3 的类型系统完全用 XML 表示。

根据应用领域的不同，有不同的文件包含数据类型：

- .tsproj 文件 — TwinCAT 项目
  该文件包含整个 TwinCAT 项目，包括完整的 TwinCAT 3 类型系统。
- .tmc 文件 — TwinCAT 模块类文件
  这些文件用于描述 TcCOM 模块本身。其中包括模块类说明和使用的数据类型。同时，如上所述，这些文件用于实现项目间数据类型的交换。

- .tmi 文件 — TwinCAT 模块实例文件
  这些文件描述一个类的实例。它们由 TwinCAT 3 开发环境存储在目标设备，以便描述一个类的实例。此外，还可以使用 .tmi 文件将实例信息从一个项目传输到另一个项目。

5 TcCom modules

TwinCAT 组件对象模型定义了 TwinCAT 3 Runtime modules 的属性和行为，并描述了各自独立开发和编译的不同软件组件如何协同工作。在以下章节中将会阐述 TcCOM 模块的概念和操作方法。

5.1 TwinCAT 组件对象模型 (TcCOM) 概念

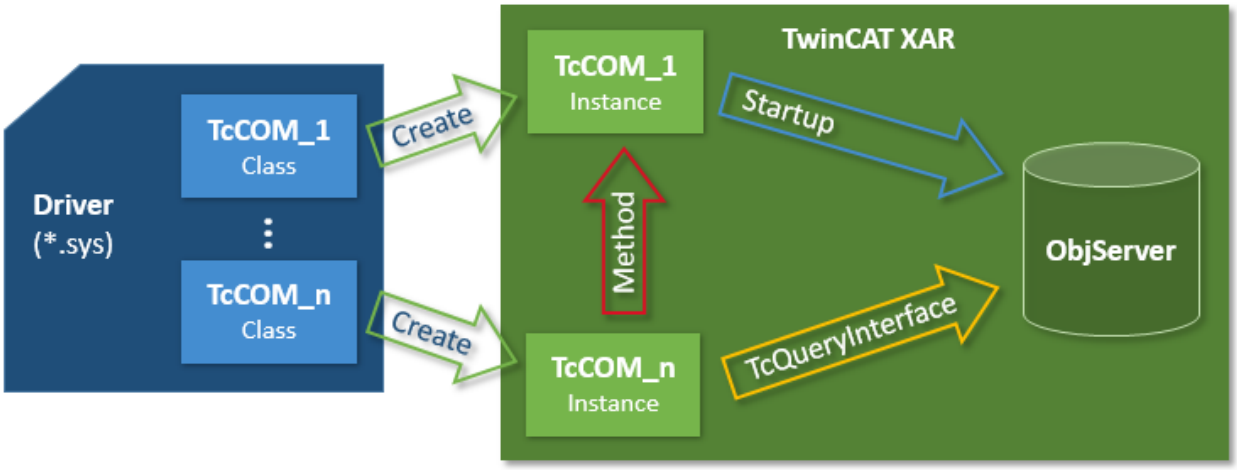
TwinCAT 组件对象模型定义了模块的特性和行为。该模型源自 Microsoft Windows 的“组件对象模型”(COM)，功能说明了各种独立开发和编译的软件组件之间的协作的方式。为了达成这一目标，必须明确界定模块的行为模式并遵循其接口规范，使其能够相互作用。例如，这种接口还可以便捷的实现不同制造商开发模块之间的交互。

在某种程度上，TcCOM 以 COM（Microsoft Windows 体系的组件对象模型）为基础，尽管只使用 COM 的一个子集。然而，与 COM 相比，TcCOM 包含更多超出 COM 的定义，例如状态机模块。

TcCOM 模块概述和应用

本介绍性概述旨在使各个主题更容易理解。

一个或多个 TcCOM 模块集成在一个驱动程序中。该驱动程序由 TwinCAT 工程使用 MSVC 编译器创建。TMC（TwinCAT 模块类）文件对模块和接口进行了说明。如今，驱动程序及其 TMC 文件可以在工程系统之间进行交换和组合。



现在可以使用Twincat工程工具创建这些模块的实例。它们与 TMI 文件相关联。这些实例可进行参数设置，并且可以相互关联，或与其他模块组合形成 I/O系统。相应的配置会传输到目标系统，并在该系统上执行。

相应的模块会被启动，并在 TwinCAT ObjectServer 上注册。TwinCAT XAR 还会提供过程映像。模块可以查询 TwinCAT ObjectServer，以获得与特定接口有关的另一个对象的引用。如果有这样的引用，便可在模块实例上调用接口方法。

以下章节将对各个主题进行论证。

ID 管理

不同类型的 ID 用于模块之间以及模块内部的交互。TcCOM 使用 GUID（128 位）和 32 位长整数。

TcCOM 使用

- GUID 作为：ModulID、ClassID 和 InterfaceID。
- 32 位长整数用于：ParameterID、ObjectID、ContextID、CategoryID。

接口

COM 的一个重要组成部分是接口，TcCOM 也是如此。

接口定义了一组方法，这些方法组合在一起可以执行某项任务。接口通过唯一的 ID (InterfaceID) 进行引用，只要接口不改变，这个 ID 就永远不能修改。该 ID 使模块能够决定其是否能与其他模块协作。同时，如果接口定义明确，那么开发过程可以独立进行。因此，修改接口会导致生成不同的 ID。TcCOM 理念旨在使 InterfaceID 可以叠加其他（旧的）InterfaceID（TMC 功能说明/TMC 编辑器中的“隐藏项”）。这样，两个版本的接口均可使用，而另一方面，同时始终明确最新InterfaceID。同样的概念也适用于数据类型。

TcCOM 已经已定义一系列的接口，这些接口在某些情况下是预先规定的（如 ITComObject），但在大多数情况下是可选的。许多接口只有在某些应用领域才有意义。其他接口则为通用型，通常可以重复使用。系统支持用户自定义接口，例如可实现两个第三方模块之间的相互交互。

- 所有接口都源自基本接口 ItcUnknown，该接口与 COM 的IUnknown类似，接口查询服务 (TcQueryInterface) 和模块生命周期管理（TcAddRef 和 TcRelease）提供基本服务。
- 每个模块都必须实现 ITComObject 接口，该接口包含用于访问模块名称、ObjectID、父模块的 ObjectID、参数和状态机的方法。

许多模块都使用多个通用接口：

- ITcCyclic接口由需周期性调用的模块实现（"CycleUpdate"）。该模块可通过 TwinCAT 任务的 ITcCyclicCaller 接口注册，以获取周期调用。
- ITcADI 接口可用于访问模块的数据区。
- 默认情况下，ITcWatchSource 已实现；它为 ADS DeviceNotification 和其他功能提供了便利。
- ITcTask 接口由实时系统的任务实现，提供有关周期时间、优先级和其他任务信息。
- ITComObjectServer 接口由 ObjectServer 实现，并被所有模块引用。

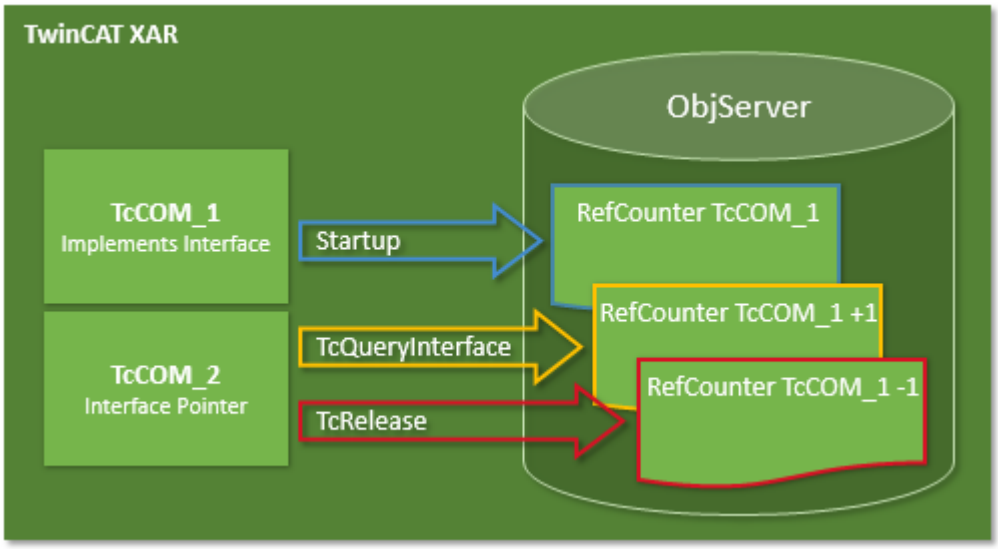
倍福已提供整个系列的通用接口。通用接口的优点是可以支持模块的交换和循环使用。只有在没有适当通用接口的情况下，才应定义自己的接口。

类工厂

“类工厂”用于在 C++ 中创建模块。驱动程序中包含的所有模块都有一个共同的类工厂。类工厂向 ObjectServer 注册一次，并为创建某些模块类提供服务。模块类由模块的唯一 ClassID 标识。当 ObjectServer 请求新模块时（基于配置器的初始化数据或在运行时通过其他模块），模块会根据 ClassID 选择合适的类工厂，并通过其 ITcClassFactory 接口触发模块的创建。

模块生命周期管理

与 COM 类似，模块的生命周期通过引用计数器 (RefCounter) 来确定。每次查询模块接口时，引用计数器都会递增。释放接口时，计数器会递减。当模块登录 ObjectServer（ITComObject 接口）时，也会查询接口，这样引用计数器至少为 1。注销时计数器递减。当计数器为 0 时，模块会自动删除，通常是在注销 ObjectServer 之后。如果另一个模块已维护一个引用（具有一个接口指针），则该模块继续存在，而接口指针保持有效，直至释放该指针。



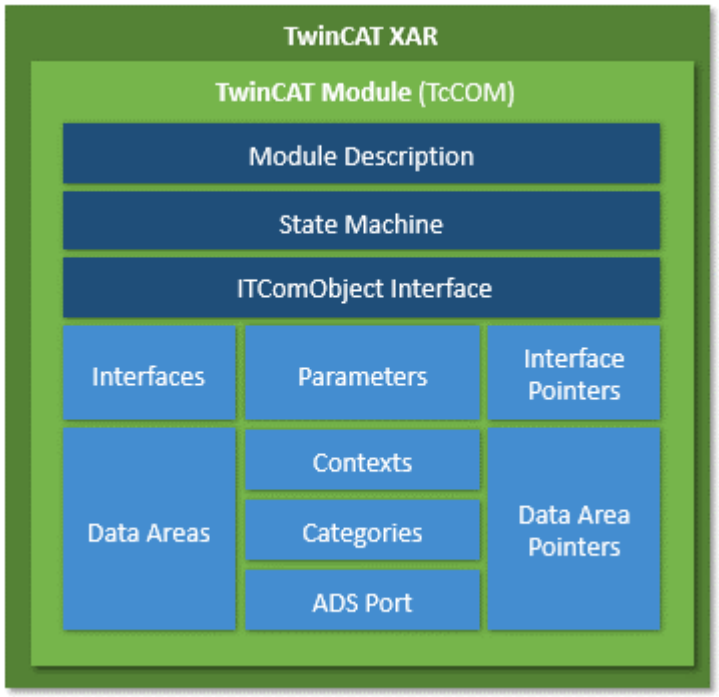
5.1.1 TwinCAT 模块属性

TcCOM 模块包含一组形式化定义的属性，分为强制属性与可选属性两类。这些属性通过标准化定义实现互换应用。每个模块都具有模块说明，用于功能说明模块属性。它们用于配置模块及其相互之间的关系。

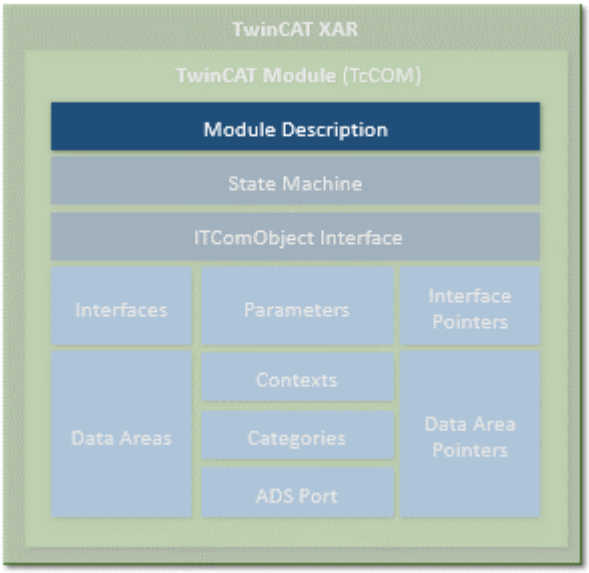
如果模块在 TwinCAT 运行时中实例化，它就会在中央系统实例 (ObjectServer) 中自行注册。这样，其他模块和通用工具便可对其进行访问和参数设置。模块可以独立编译，因此也可以独立开发、测试和更新。模块可以非常简单，例如可能仅包含低通滤波器等基本功能。或者其内部也可以非常复杂，包含机器子组件的整个控制系统。

模块的应用非常广泛；自动化系统的所有任务都可以在模块中指定。因此，对于主要代表自动化系统基本功能（如实时任务、现场总线驱动程序或 PLC 运行时系统）的模块，以及用于控制机器单元的特定用户或应用算法，不作任何区分。

下图显示的是常见 TwinCAT 模块及其主要属性。深蓝色块定义了强制属性，而浅蓝色块则定义了可选属性。



模块说明



每个 TcCOM 模块都有一些一般功能说明参数。其中包括一个 ClassID，可以明确引用模块类。由相应的 ClassFactory 实例化。每个模块实例都有一个 ObjectID，该 ID 在 TwinCAT 运行时中是唯一的。此外，还有一个父级 ObjectID，指的是可能的逻辑父对象。

通过 ITComObject 接口可以获取下文所述模块的信息、状态机和参数（请参见“接口”章节）。

类功能说明文件 (*.tmc)

模块类别信息集路在类功能说明文件（TwinCAT Modole Class；*.tmc）中。

开发人员使用这些文件来功能说明模块属性和接口，以便其他人调用和集成模块。除常规信息（供应商数据、模块类 ID 等）之外，文件中还会功能说明模块的可选属性。

- 支持类别
- 已实现的接口
- 带有相应符号的数据区
- 参数
- 接口指针
- 可以设置的数据指针

系统配置器主要将类功能说明文件用作以下操作的基础：将模块实例整合到配置中，指定参数并配置与其他模块的关联。

同时包括模块中所有数据类型的功能说明，然后由配置器在其通用数据类型系统中采用。这样，系统中存在的 TMC 功能说明的所有接口都可以被所有模块使用。

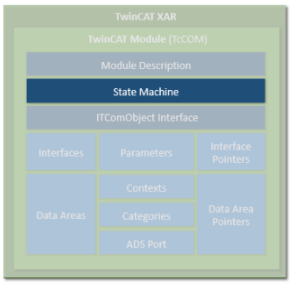
涉及多个模块的较复杂配置也可在类功能说明文件中有所功能说明，这些文件已针对特定应用进行预配置和关联。因此，在开发阶段，内部由多个子模块组成的复杂机器单元的模块，可以作为实体进行定义和预配置。

实例功能说明文件 (*.tmi)

特定模块的实例在实例功能说明文件（TwinCAT Moudule Instance；*.tmi）中有所功能说明。实例功能说明文件采用与类功能说明类似的格式，但与类功能说明文件不同，实例功能说明文件已包含项目中特殊模块实例的参数、接口指针等具体规范。

实例功能说明文件由 TwinCAT 工程 (XAE) 在为特定项目创建类功能说明实例时创建。其主要用于配置涉及的所有工具之间的数据交换。然而，实例功能说明也可以跨项目使用，例如，在新项目中再次使用专门进行参数设置的模块。

状态机

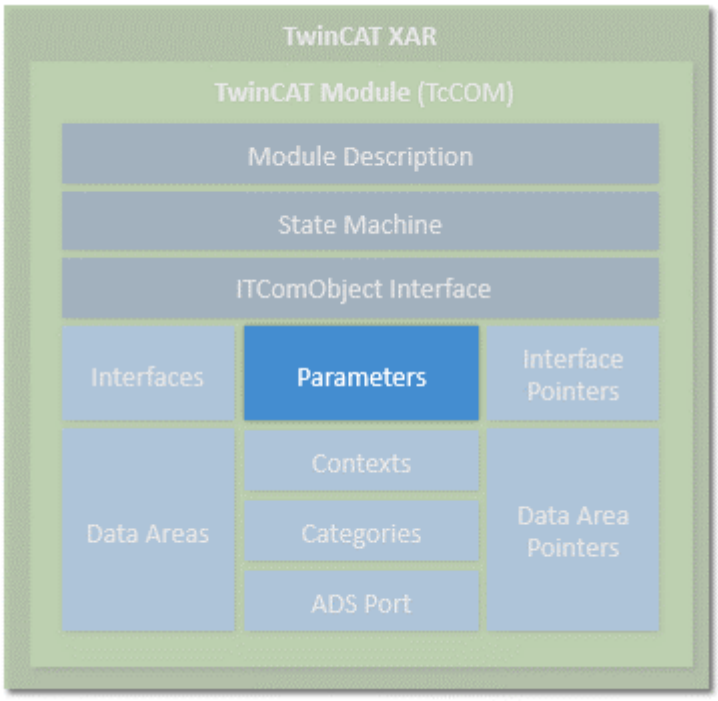


每个模块都包含状态机，该机器功能说明的是模块的初始化状态以及从外部修改该状态的方法。状态机功能说明的是模块启动和停止时的状态。其中涉及模块创建、参数设置以及与其他模块的协同生产。

应用特定状态（如现场总线或驱动程序的状态）可以用各自的状态机来功能说明。TcCOM 模块的状态机定义了 INIT、PREOP、SAFEOP 和 OP 状态。虽然状态名称与 EtherCAT 现场总线的状态名称相同，但实际状态却不同。当 TcCOM 模块为 EtherCAT 执行现场总线驱动程序时，它配有两个状态机（模块和现场总线状态机），且二者需按顺序执行。模块状态机必须达到运行状态 (OP) 后，现场总线状态机才能启动。

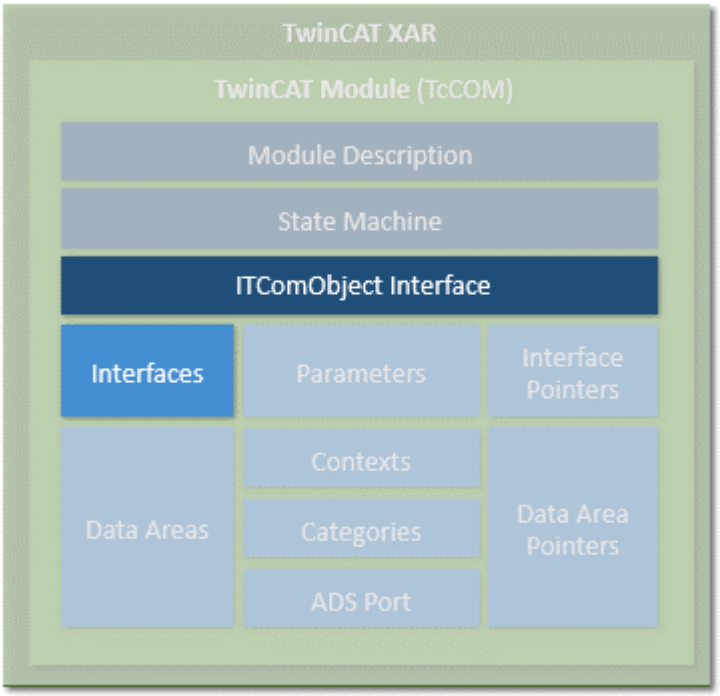
状态机的详细介绍 [▶ 227]将单独阐述。

参数



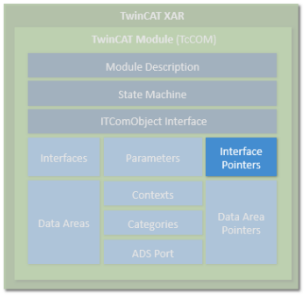
模块可以包含参数，这些参数可以在初始化时或之后的运行时（OP 状态）读取或写入。每个参数都有一个参数 ID。参数 ID 的唯一性可以分为三种类型：全局性、受限全局性或模块特定性。有关更多详情，请参见“ID 管理”章节。除参数 ID 外，参数还包含当前数据；数据类型取决于参数，并根据相应的参数 ID 明确定义。

接口



接口由一组明确方法（函数）组成，这些方法为模块提供交互通道，使其他模块能够与该模块建立通信。如上所述，接口具有唯一 ID。模块必须至少支持 ITComObject 接口，此外还可根据需要包含多个接口。通过调用“TcQueryInterface”方法，并指定相应的接口 ID，即可查询接口引用。

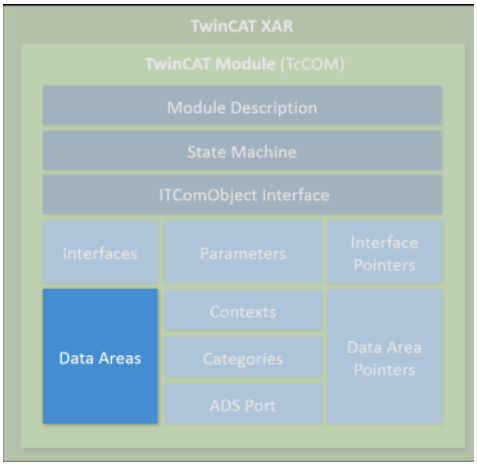
接口指针



接口指针可视为接口的对应交互对象。如果模块要使用另一个模块的接口，则必须具有相应接口类型的接口指针，并确保其指向目标模块。之后方可调用目标模块的方法。

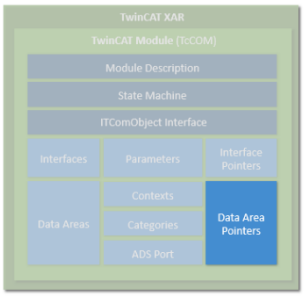
接口指针通常会在状态机启动时设置。在 INIT 向 PREOP (IP) 转换期间，模块通过相应接口接收其他模块的对象 ID；在 PREOP 向 SAFEOP (PS) 或 SAFEOP 向 OP (SO) 转换期间，通过 ObjectServer 搜索其他模块的实例，并通过方法查询接口设置相应接口。在状态逆向切换时，即从 SAFEOP 到 PREOP (SP) 或从 OP 到 SAFEOP (OS) 期间，必须再次启用接口。

数据区



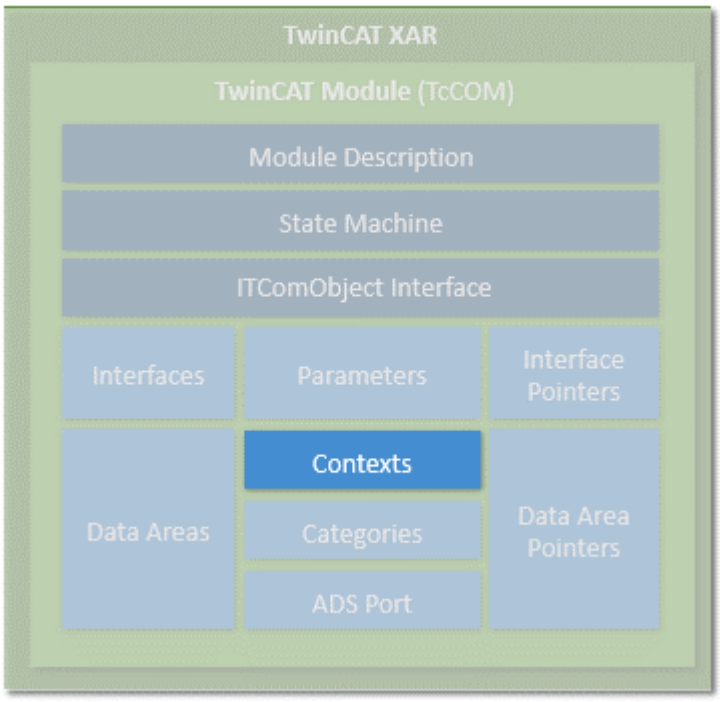
模块可以包含数据区，这些数据区可供环境（如其他模块或 TwinCAT 的 IO 区域）调用。这些数据区可以储存任何数据。它们通常用于过程映像数据（输入和输出）。数据区的结构在模块的设备功能说明中定义。如果模块拥有数据区，并希望其他模块也能访问这些数据区，那么该模块就需要实现 ITcADI 接口，以便能够访问这些数据。数据区可以包含符号信息，这些信息更详细地功能说明了相应数据区的结构。

数据区指针



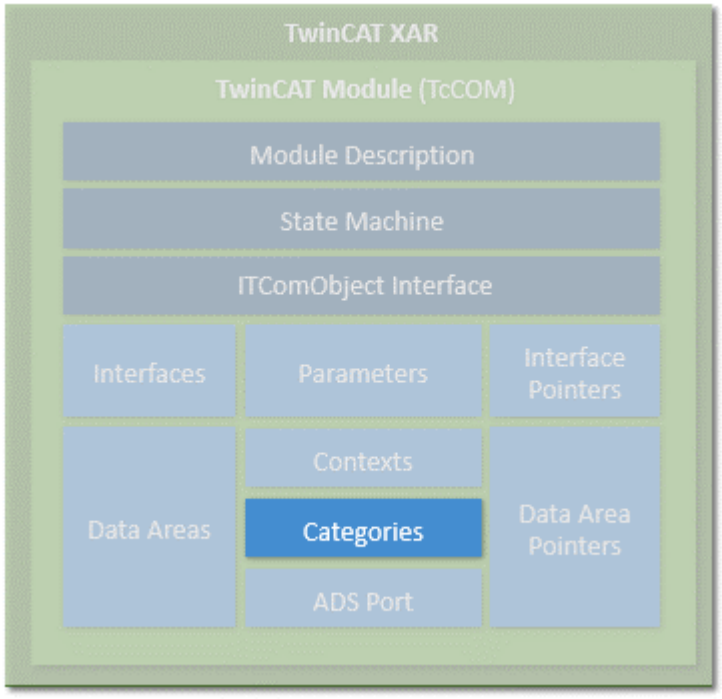
如果模块要访问其他模块的数据区，则可包含数据区指针。这些通常在状态机初始化时指向其他模块的数据区或数据区段。访问权限为直接进入内存区，因此必要时因此必要时需实现相应的保护机制，以应对并发访问冲突。在许多情况下，最好使用相应的接口。

环境



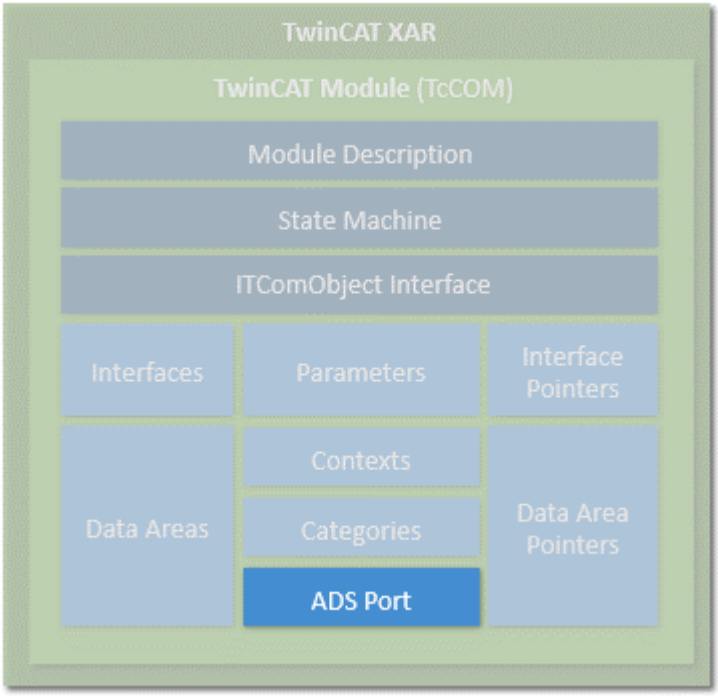
环境应视为实时任务环境。例如，在配置模块时需要环境。简单模块通常在单一时间环境中运行，因此无需详细说明。其他模块可能在多个环境中部分处于活动状态（例如，EtherCAT 主站可以支持多个独立的实时任务，或者控制回路可以在另一个循环时间内处理下一层的控制回路）。如果模块具有多个随时间变化的环境，则必须在模块说明中加以说明。

类别



模块可以通过实现接口 ITComObjectCategory 来提供类别。ObjectServer 会对类别进行列举，可通过 ObjectServer (ITComObjectEnumPtr) 来查询与类别相关联的对象。

ADS



在 ObjectServer 中输入的每个模块都可以通过 ADS 访问。例如，ObjectServer 调用模块的 ITComObject 接口，实现读写参数，或访问状态机。此外，还可以实现专用的 ADS 端口，通过该端口接收专用的 ADS 命令。

系统模块

TwinCAT 运行时还可提供许多系统模块，使其他模块也能使用基本的运行时服务。这些系统模块具有固定不变的 ObjectID，其他模块可以通过它来访问。这种系统模块的其中一个示例为实时系统，通过 ITcRTime 接口提供基本的实时系统服务，即生成实时任务。ADS 路由器同时作为系统模块实现，因此其他模块可以在此注册其 ADS 端口。

创建模块

模块可以通过 C++ 和 IEC 61131-3 两种语言创建。创建过程需使用 TwinCAT PLC的面向对象扩展功能。这两种语言模块可以通过接口进行交互，交互方式与纯 C++ 模块相同。面向对象的扩展功能可提供与 C++ 相同的接口。

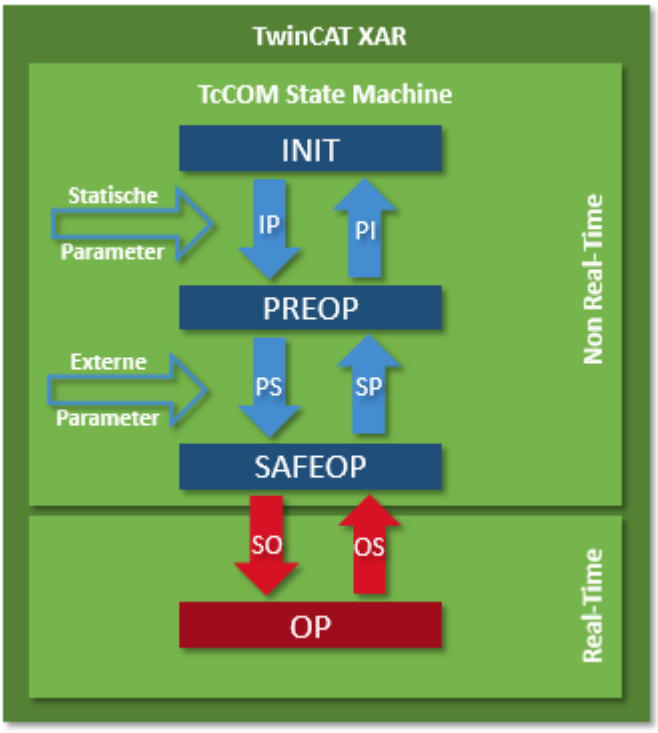
PLC 模块同时通过 ObjectServer 注册，因此可以通过 ObjectServer 访问 PLC 模块。PLC 模块的复杂程度各不相同。无论是只生成小型滤波模块，还是将完整的 PLC 程序打包到一个模块中，都没有区别。由于实现了自动化，每个 PLC 程序都是 TwinCAT 模块意义上的一个模块。所有传统 PLC 程序都会自动封装到一个模块中，并在ObjectServer及一个或多个任务模块中完成注册。对 PLC 模块过程数据的访问（例如与现场总线驱动程序的映射）也可通过定义的数据区和 ITcADI 进行控制。

只要 PLC 程序员决定将 PLC 程序的某些部分明确定义为 TwinCAT 模块，以便可以适当灵活地调用这些模块，那么对于 PLC 程序员来说，这种行为仍然是透明且不可见的。

5.1.2 TwinCAT 模块状态机

除了四种状态（INIT、PREOP、SAFEOP 和 OP）外，还有相应的状态转换，在这些状态转换中，必须执行或可以执行一般或特定模块的操作。状态机的设计简洁。在任何情况下，都只能转换到下一步或上一步。

正向转换：INIT 到 PREOP (IP)、PREOP 到 SAFEOP (PS) 以及 SAFEOP 到 OP (SO)。反向转换，存在以下状态转换：OP 到 SAFEOP (OS)、SAFEOP 到 PREOP (SP) 以及 PREOP 到 INIT (PI)。包括 SAFEOP 状态在内，所有状态和状态转换都在非实时环境中进行。只有从 SAFEOP 到 OP 的转换、OP 状态以及从 OP 到 SAFEOP 的转换是在实时环境中进行的。在分配或启用资源时、模块与其他模块注册或注销注册时具有重要意义。



状态：INIT

INIT 状态只是一种虚拟状态。创建模块后，模块状态立即从 INIT 变为 PREOP，即执行 IP 状态转换。实例化和 IP 状态转换总是同时进行，因此模块绝不会停留在 INIT 状态。只有在移除模块时，模块才会短暂保持 INIT 状态。

转换：INIT 到 PREOP (IP)

在 IP 状态转换期间，模块使用其唯一的 ObjectID 向 ObjectServer 注册。初始化参数也是在创建对象时分配的，并被传输到模块中。在此转换期间，模块无法与其他模块建立连接，因为不清楚其他模块是否已经存在并在 ObjectServer 注册。当模块需要系统资源（如内存）时，可在状态转换期间分配这些资源。在从 PREOP 转换到 INIT (PI) 期间，必须再次释放所有已分配的资源。

状态：PREOP

在 PREOP 状态下，模块创建完成，模块通常已进行完全参数设置，即使在从 PREOP 转换到 SAFEOP 的过程中也可能会进一步添加参数。虽然尚未创建与其他模块的连接，但该模块已在 ObjectServer 中注册。

转换：PREOP 到 SAFEOP (PS)

在这种状态转换中，模块可以与其他模块建立连接。为此，模块通常会收到其他模块的 ObjectID 和初始化数据等，现在通过 ObjectServer 将这些数据转换为与这些模块的实际连接。

转换一般可由系统根据配置器触发，或由其他模块（如父模块）触发。在此状态转换过程中，可以传输更多参数。例如，父模块可以将自己的参数传送给子模块。

状态：SAFEOP

模块仍处于非实时环境中，等待系统或其他模块切换到 OP 状态。

转换：从 SAFEOP 转换到 OP (SO)

从 SafeOP 到 OP 的状态转换、状态 OP 以及从 OP 到 SAFEOP 的转换都是在实时环境中进行的。可能不会再分配系统资源。但其他模块现在可以申请资源，模块也可以向其他模块注册，例如在任务执行期间获得周期性调用。

该转换过程不允许执行长时间运行的任务。例如，文件操作应在 PS 期间执行。

状态：OP

在 OP 状态下，模块开始工作，在 TwinCAT 系统中处于完全激活状态。

转换：OP 到 SAFEOP (OS)

这种状态转换是在实时环境中进行的。SO 转换的所有操作均会撤销，SO 转换期间申请的所有资源都会再次释放。

转换：从 SAFEOP 到 PREOP (SP)

PREOP→SAFEOP （PS ）转换的所有操作均会撤销，PS 转换期间申请的所有资源都会再次释放。

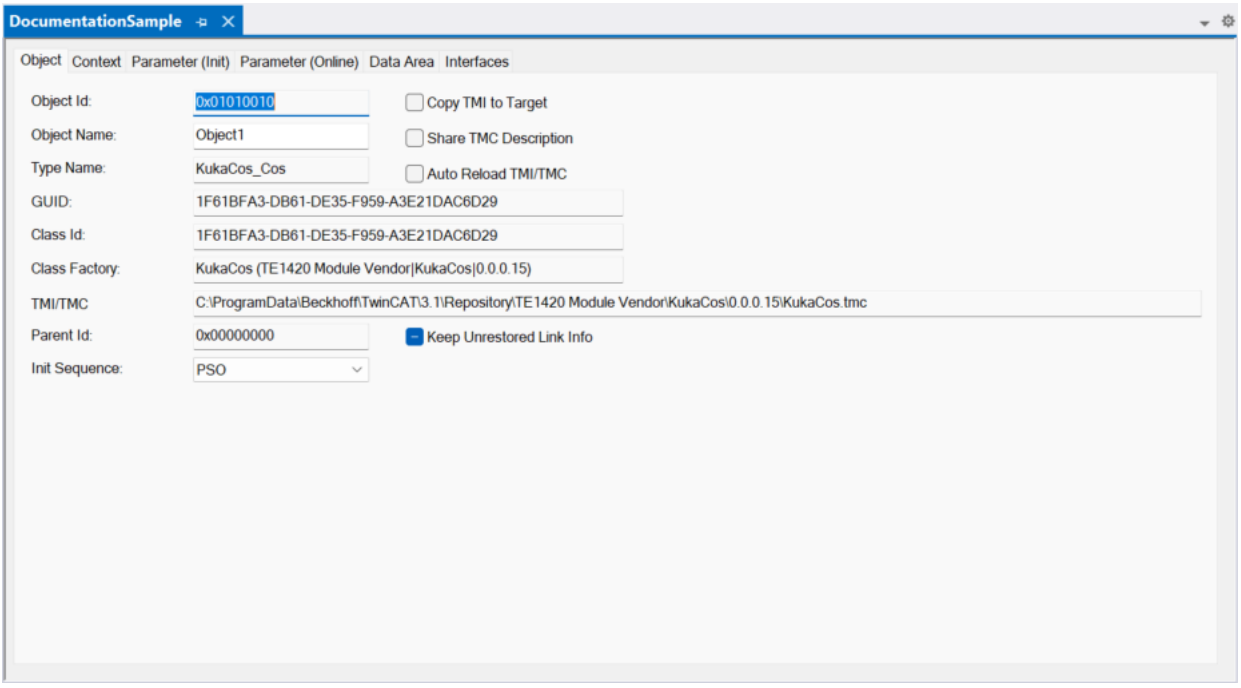
转换：从 PREOP 到 INIT (PI)

IP 转换的所有操作均会撤销，IP 转换期间申请的所有资源都会再次释放。模块从 ObjectServer 注销，通常会触发自销毁流程（请参见“Service life”）。

5.2 TcCom modules 的操作方法

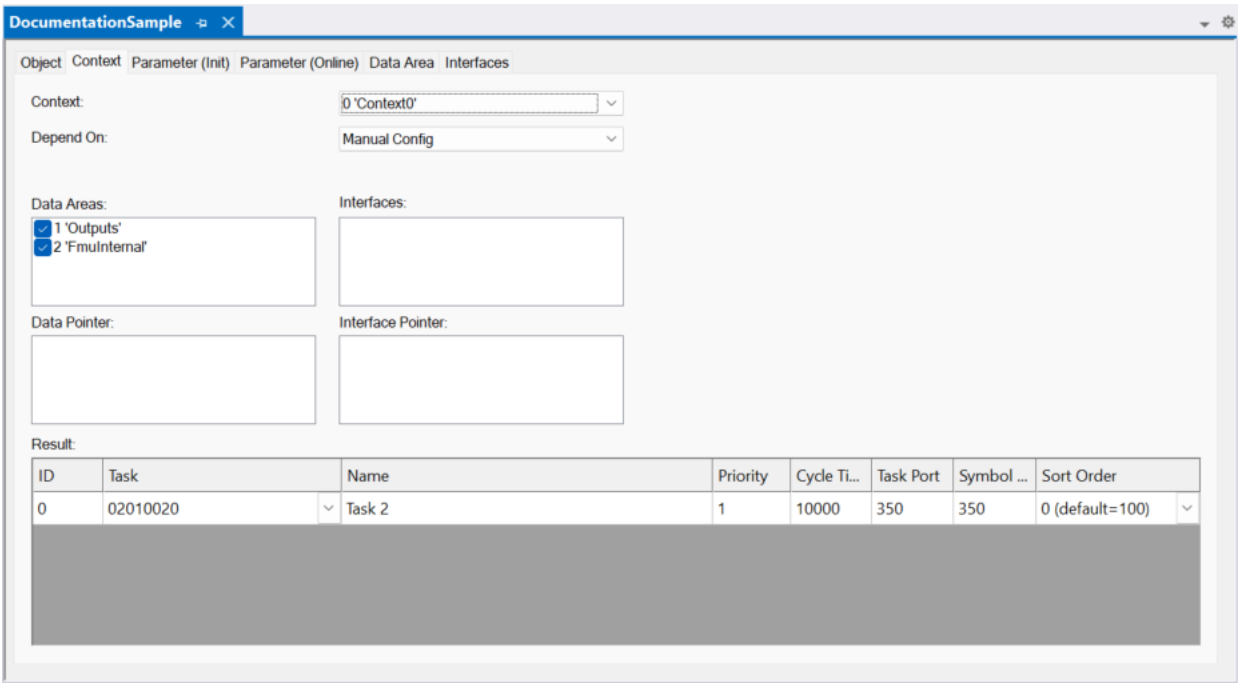
下面的章节将介绍 TcCom 模块的操作方法。为此，该章节将同时说明两个方面的内容：在 TwinCAT 模块属性 [▶ 221] 一章中定义的属性在 TcCom 模块各选项卡中的对应，以及这些模块在 TwinCAT 项目中的存储或使用。

5.2.1 Object（对象）选项卡



Object ID（对象 ID）	TwinCAT 项目中 TcCom 对象实例的内部 ID
Object Name（对象名称）	TcCom 对象实例的名称
Type Name（类型名称）	TcCom 对象类型的名称
GUID	模块类的 GUID
Class Id（类 ID）	实现模块类的类的 GUID。
Class Factory（类工厂）	TwinCAT 驱动程序的名称
TMI/TMC	TMI/TMC 文件的路径
Parent Id（父 ID）	TwinCAT 项目中父对象的内部 ID
Init Sequence（初始化序列）	初始化序列决定了 TcCOM 对象应启动至何种状态。
Copy TMI to Target（复制 TMI 到目标系统）	设置是否应将 TcCom 对象的实例信息也写入目标系统。
Share TMC Description（共享 TMC 说明）	设置同一类的多个模块实例是否共享 TMC 描述。
Auto Reload TMI/TMC（自动重新加载 TMI/TMC）	当链接的 TMI/TMC 文件被重新写入硬盘时，系统将自动重新加载该文件。
Keep Unrestored Link Info（保留未恢复的链接信息）	设置重新加载时无法恢复的模块是否保存其链接信息。

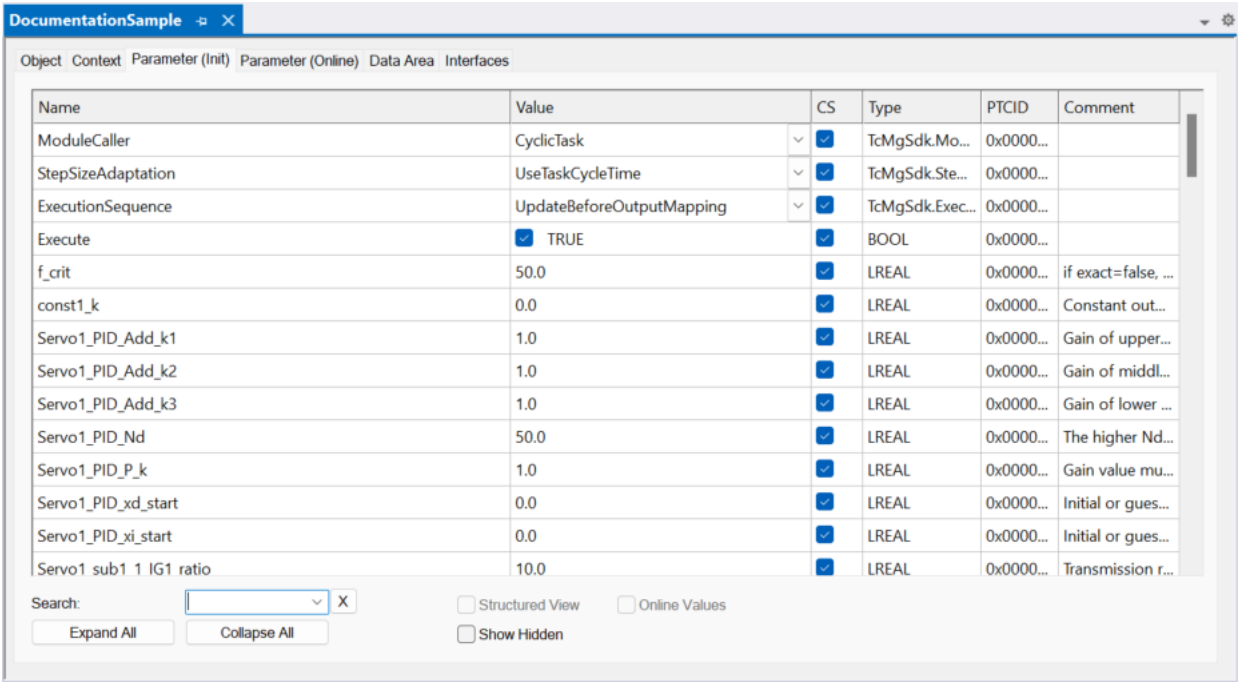
5.2.2 Context（上下文）项卡



Context（上下文）	选择要更改设置的 context （使用下表中的 ID）。
Depend On（依赖项）	设置如何调用这个 Context（上下文）。（手动设置、基于父对象、基于数据区、基于任务设置）
Data Areas（数据区）	选择要从选定的 context（上下文）更新的数据区。
ID	context（上下文）的 ID
任务	分配给 context（上下文）的任务 ID
名称	分配任务的名称
优先级	分配任务的优先级
周期时间	分配任务的周期
Task Port（任务端口）	分配的任务的 Task Port（任务端口）
Symbol Port（符号端口）	选定任务的 ADS 符号端口
Sort Order（排序）	module 在 task 上注册时所采用的排序。（数字越小，在注册模块列表中被调用的可能性越大。）

5.2.3 Parameter（Init）选项卡

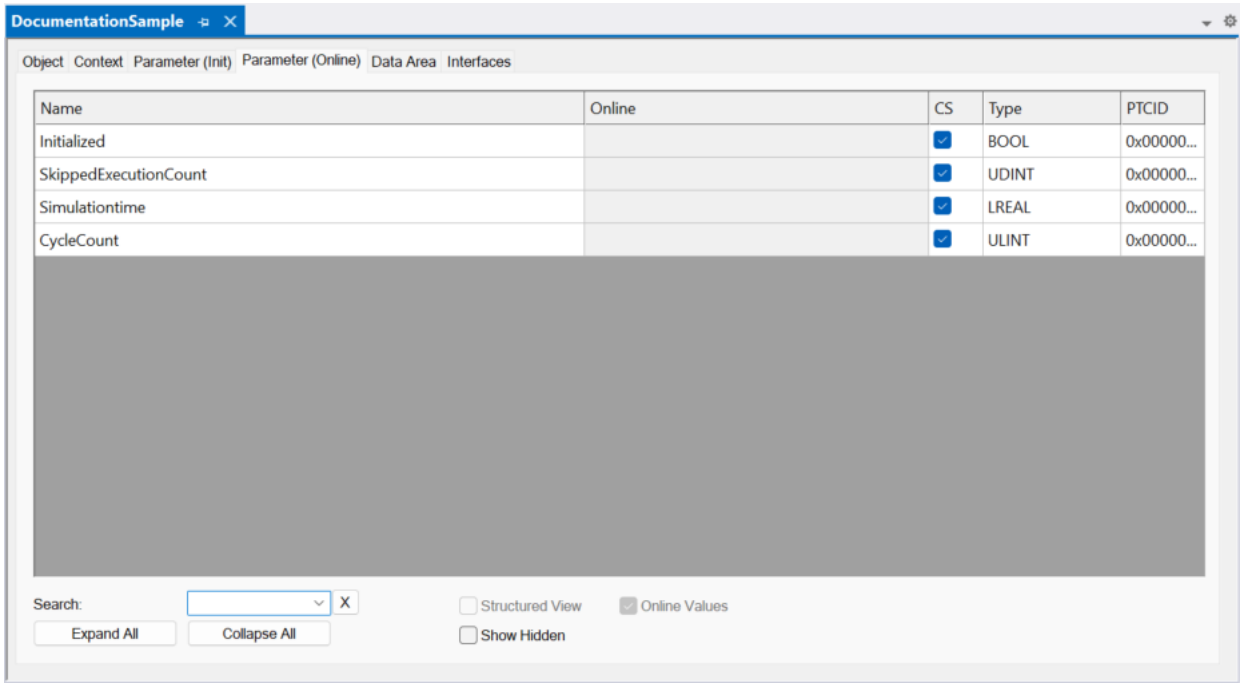
在本控制卡中列出了 TcCom 模块的初始参数。



名称	参数的名称
值	参数的值
CS	如果勾选此选项，则会为相应的参数生成一个 ADS symbol（符号）。
类型	参数的数据类型
PTCID	Parameter ID
注释	注释

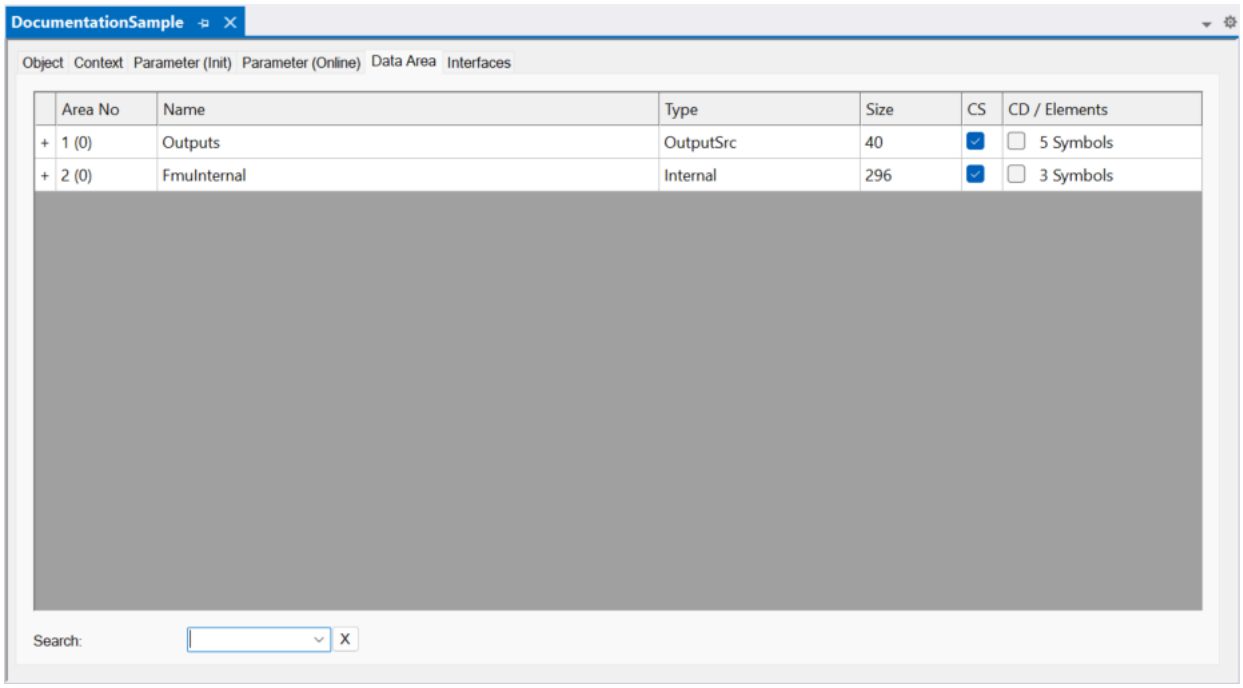
5.2.4 Parameter (Online) 选项卡

在此选项卡中显示当前 TcCom 模块的在线值。



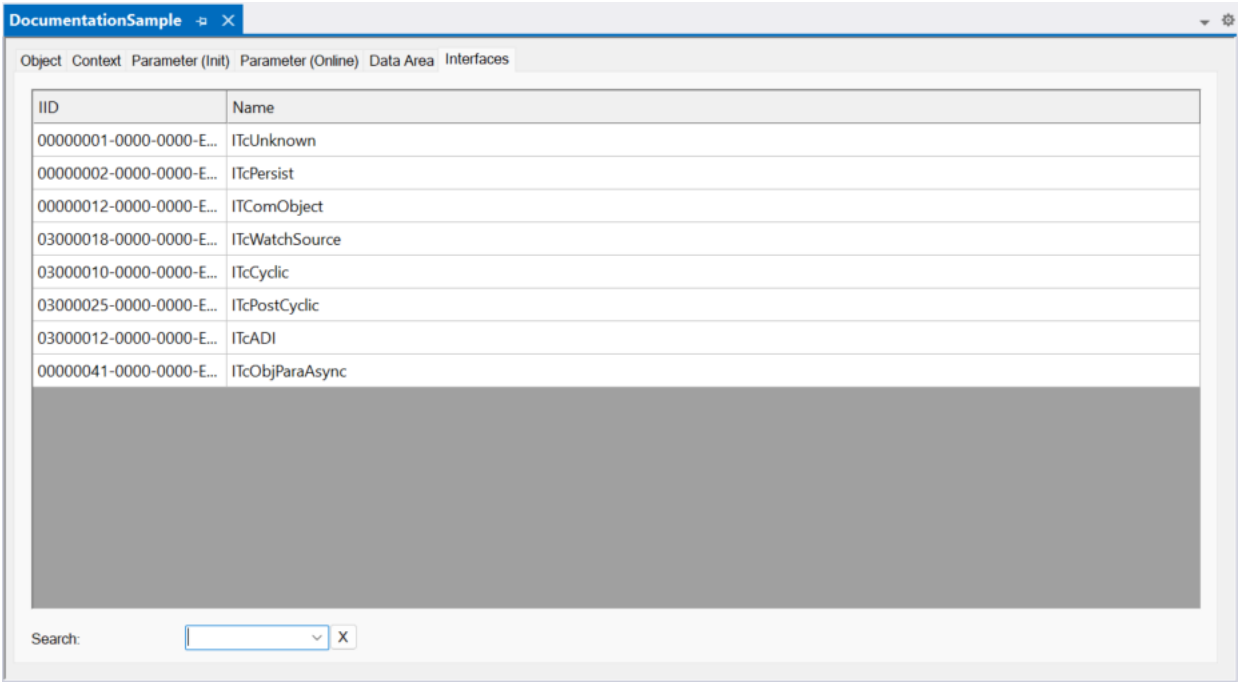
名称	参数的名称
值	参数的值
CS	如果勾选此选项，则会为相应的参数生成一个 ADS symbol（符号）。
类型	参数的数据类型
PTCID	Parameter ID

5.2.5 Data Area（数据区）选项卡



Area No（区域编号）	数据区的 ID
名称	数据区的名称
类型	数据区或变量的类型
大小	数据区或变量的大小
CS	为数据区创建 ADS Symbol（符号）的选项
CD / Elements（元素）	在 runtime 系统上创建数据类型的选项，以便可以通过 symbol（符号）访问它们。

5.2.6 Interfaces（接口）选项卡



IID	接口 ID
名称	接口名称

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

Third-party trademark statements

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

BiSS is a trademark of IC Haus GmbH.

CANopen and CANopen FD are registered trademarks of CAN in AUTOMATION - International Users and Manufacturers Group e.V.

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

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.

Sercos is a registered trademark of Sercos International e.V.

更多信息:
www.beckhoff.com/automation

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