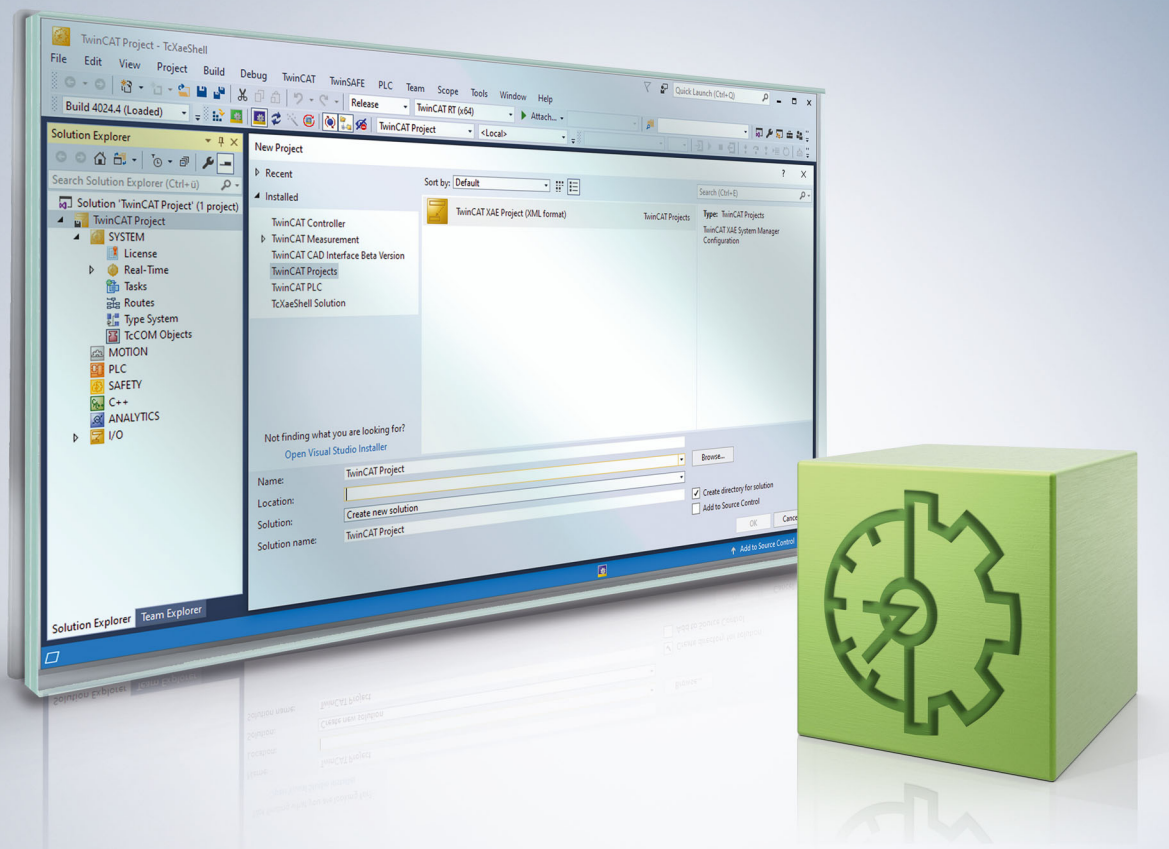
BECKHOFF New Automation Technology

手册 | ZH

TE1010

TwinCAT 3 | Realtime Monitor



2024-11-11 | 版本： 2.0.0

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

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

商标

Beckhoff®、TwinCAT®、TwinCAT/BSD®、TC/BSD®、EtherCAT®、EtherCAT G®、EtherCAT G10®、EtherCAT P®、Safety over EtherCAT®、TwinSAFE®、XFC®、XTS® 和 XPlanar® 是德国倍福自动化有限公司的注册商标并已获得授权。
本文档中所使用的其它名称可能是商标名称，任何第三方为其自身目的而引用，都可能触犯商标所有者的权利。

正在申请的专利

涵盖 EtherCAT 技术，包括但不限于以下专利申请和专利：
EP1590927、EP1789857、EP1456722、EP2137893、DE102015105702
并在多个其他国家进行了相应的专利申请或注册。

EtherCAT® 是注册商标和专利技术，由德国倍福自动化有限公司授权使用。

版权所有

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 概述

TwinCAT 3 Realtime Monitor 可对 TwinCAT 运行时环境中任务的运行时行为进行精确诊断和优化。它以图形化方式显示实时任务及其模块在所有核中的时间处理过程。此外，用户定义的进程及其依赖关系还可以通过控制软件的适当仪器以图形化方式显示。

Realtime Monitor 可以完全直观地呈现目标系统上控制软件的时间行为，并进行全面的时间分析。因此，它既支持故障诊断，也支持配置时间优化，尤其是在多核系统上。

3 安装

系统要求

技术数据	描述
操作系统	Windows 10
目标平台	Windows
最低 TwinCAT 版本	TwinCAT 3.1.4024.0

TwinCAT Package Manager：安装 (TwinCAT 3.1 Build 4026)

有关安装产品的详细说明，请参阅 TwinCAT 3.1 Build 4026 安装说明中的“安装工作负载”章节。

安装以下工作负载才能使用产品：

- TE 1010 | TwinCAT 3 Realtime Monitor

TwinCAT 设置：安装 (TwinCAT 3.1 Build 4024)

使用单独的安装程序进行安装。如要安装 TwinCAT 3 Realtime Monitor，请运行安装向导并按照说明进行操作。

授权

TwinCAT 3 Realtime Monitor (TE1010) 是一款工程产品。因此，仅可在工程系统上进行授权。完整版本的授权指南可参见 TwinCAT 3 授权文档。

本产品不提供 7 天试用授权。

4 技术简介

下一章将介绍关于使用 TwinCAT 3 Realtime Monitor 的技术基础知识。

4.1 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，可在规定的保证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）。此规定时间也称为循环时间。

应用程序本身可以由多个程序块组成，这些程序块反过来又会调用其他程序或功能块等。（另请参见 IEC 61131-3 标准）。可将程序块分配给实时任务，而实时任务反过来又会以规定的循环时间和优先级调用程序块。

TwinCAT 3 Real-Time 是一种实时扩展，可在 Windows 7 的 Microsoft Windows 操作系统下或 TwinCAT/BSD 下用于当前的 TwinCAT 3.1 版本。TwinCAT 3 Real-Time 支持以下功能，以满足工业过程控制的规定要求：

- 实时调度能力
- 并行执行进程
- 直接硬件访问

此外，TwinCAT 3 Real-Time 还提供多核支持，以满足对高性能和灵活/可扩展控制平台日益增长的需求。可用核既可以专用于 TwinCAT，也可以与 Windows 共享。因此，在以下几个部分中，这些核被称为“隔离核”或"共享核"。

实时调度能力

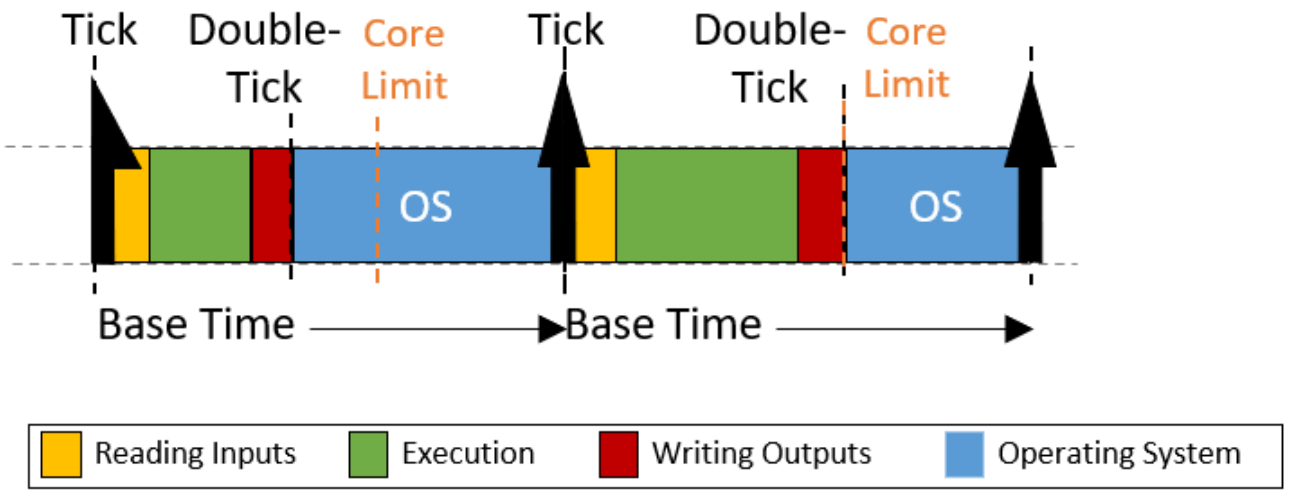
TwinCAT 3 Real-Time 采用双时标法运行。这意味着无论是切换到实时模式还是从实时模式切换回来，都是由一个中断信号触发的。切换到实时模式时触发的中断同时也会启动调度。经过一个可调整时间段后（至少在设置循环时间的 90% 之后），TwinCAT 会以非实时模式下切换回“共享”核，以便客户操作系统有足够的计算时间来满足硬件功能等所需的响应时间。隔离核是例外。

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。

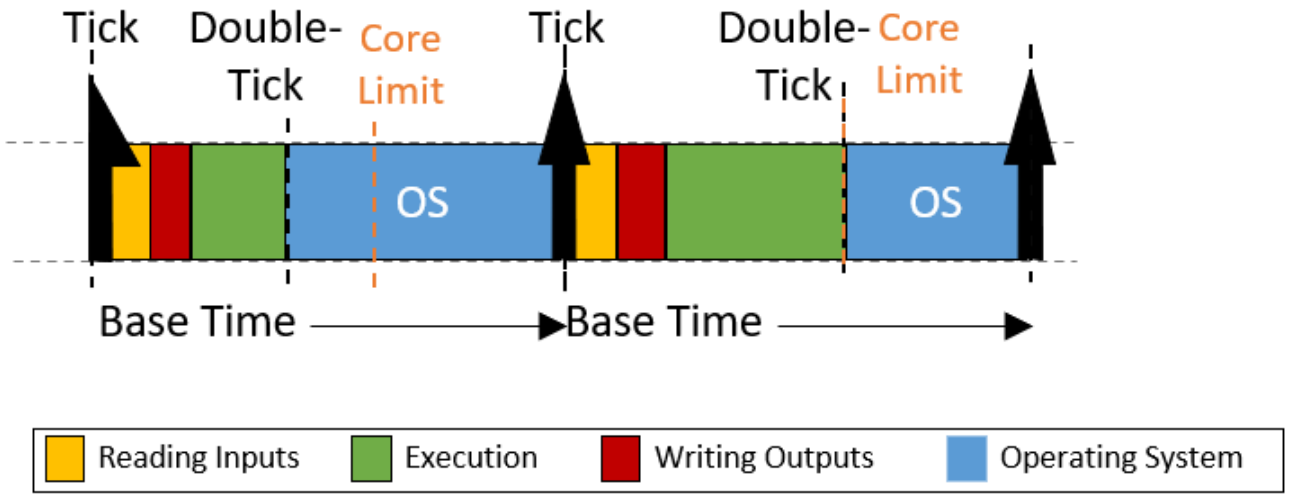
根据循环时间派生任务优先级的调度方式也称为速率单调调度。TwinCAT 3 Real-Time 会自动激活"自动优先级管理"选项。由于该选项并不总是每个应用的最佳解决方案，因此可以手动调整优先级。

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

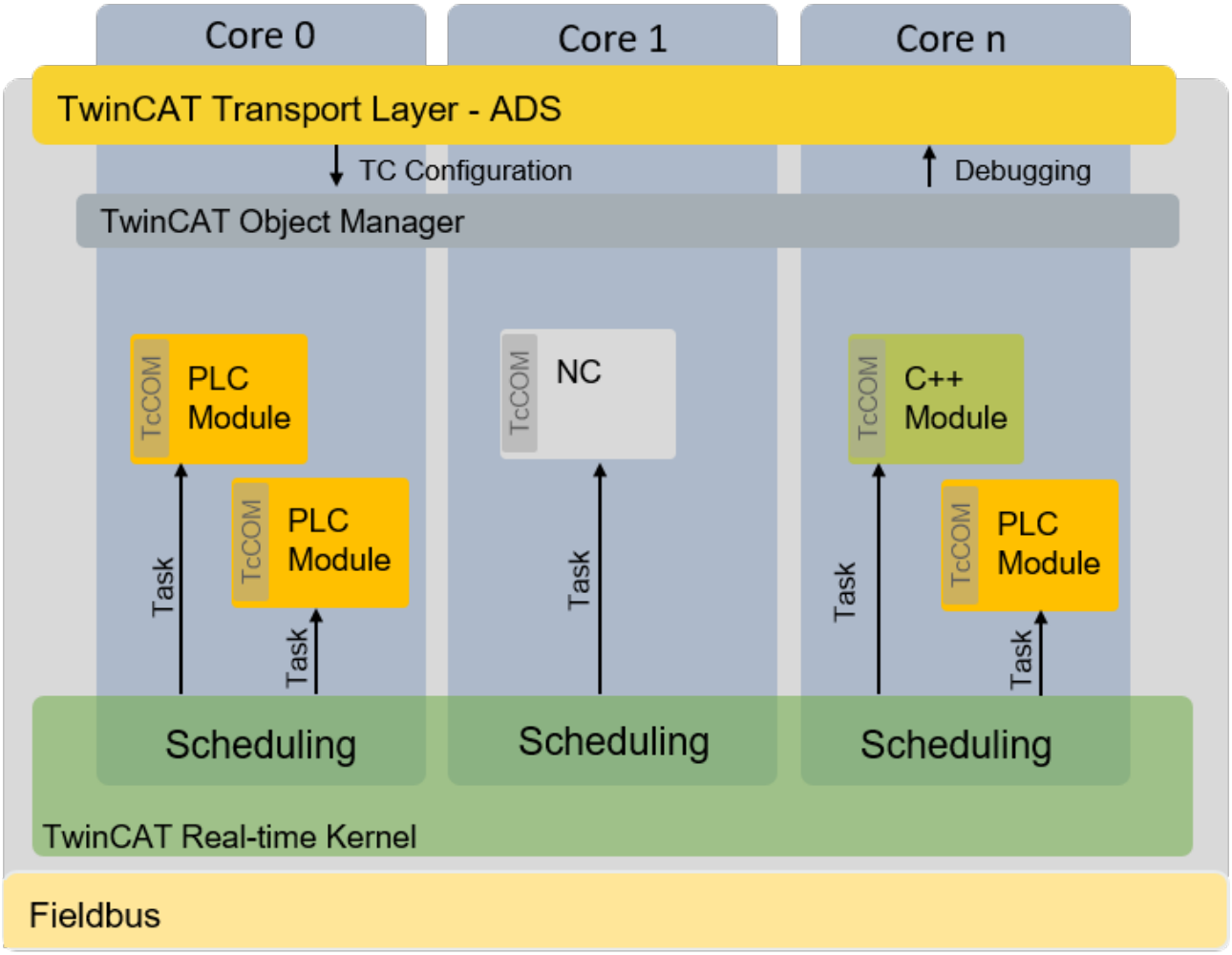
抢占式多任务处理是指在发生中断（如被优先级更高的进程中断）时，系统会保存进程（CPU 和浮点寄存器）的当前状态，并暂停当前的进程。如果出现这种情况，调度程序会根据任务的优先级确定要执行的（新）进程。 被中断的进程完成后，进程上下文就会恢复，并继续执行“旧”流程。

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，必须在 Windows 内核模式下执行 TwinCAT 3 Real-Time。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）。

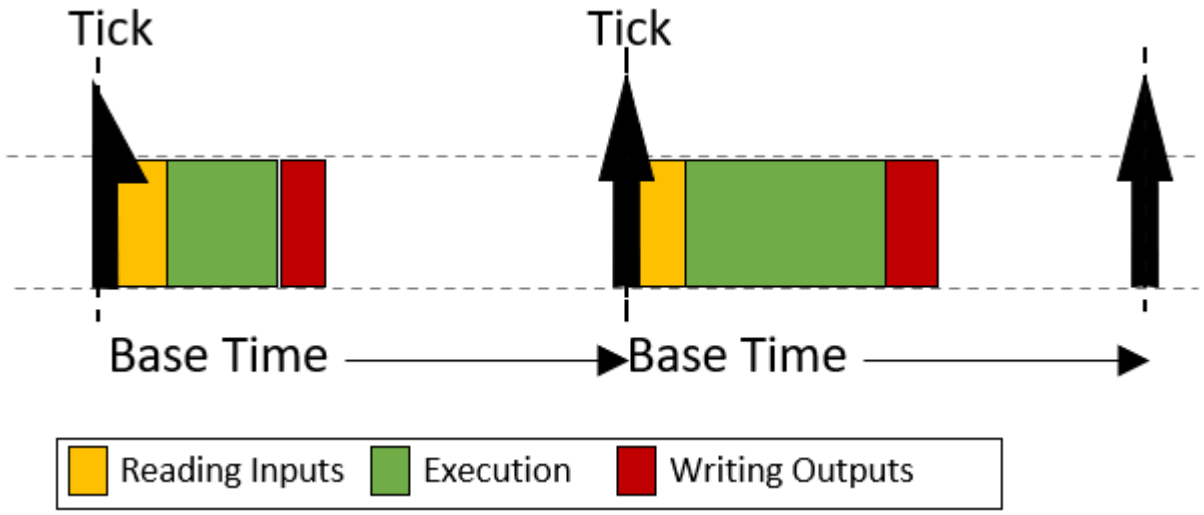
TwinCAT 3 运行时环境示意图

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，它在为 TwinCAT 激活的核上执行，并在那里处理调度。后者在单个核上独立进行。



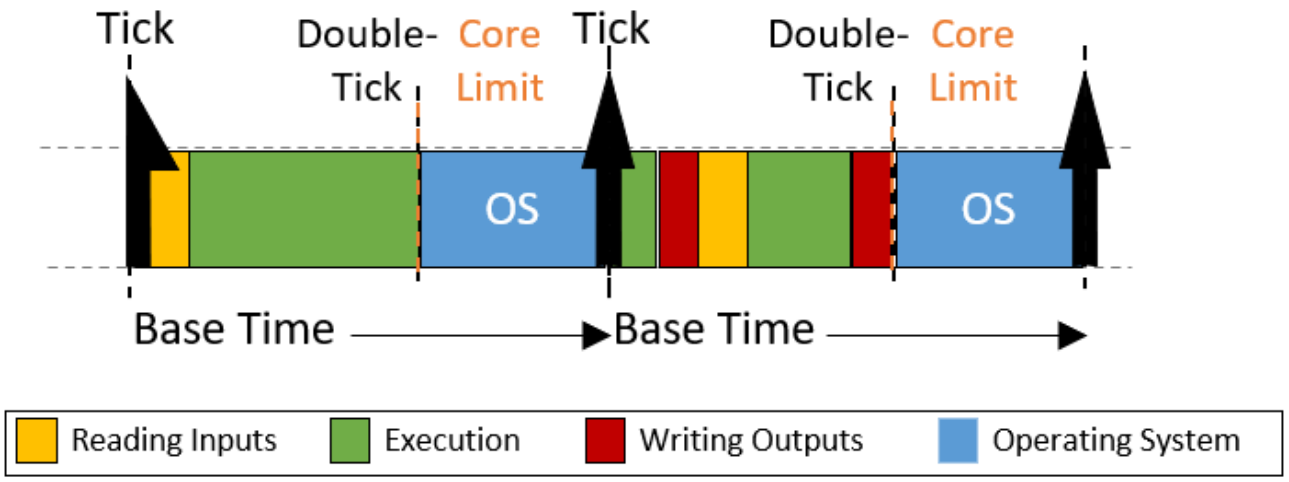
隔离核

如实时调度 [▶ 9]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 [▶ 10]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）。

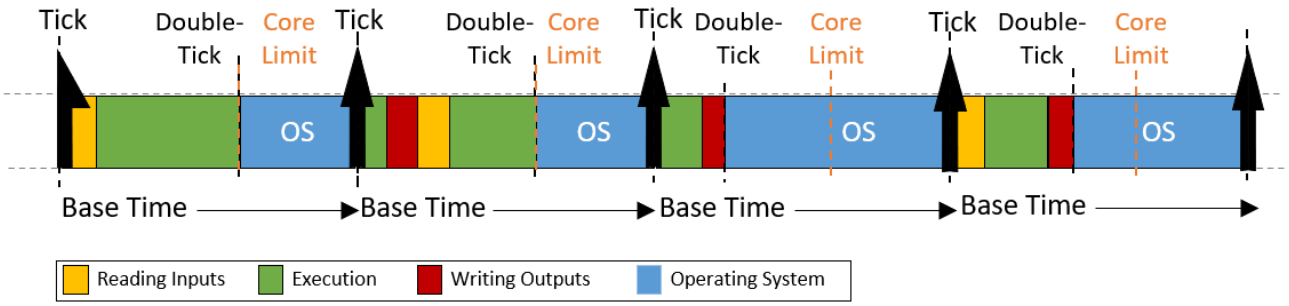


超过循环时间时的行为

如果超过了任务的规定循环时间，则在下一个周期中继续处理"旧"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。

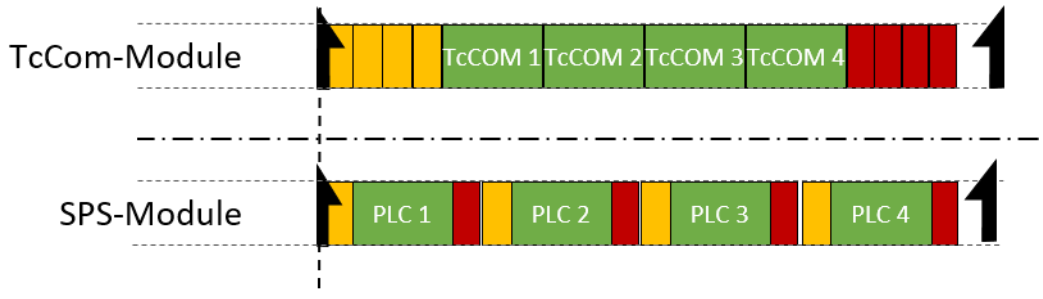


PLC 和"TcCom"运行模块的执行差异

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。

4.2 实时监控器中的显示

简单地说，TwinCAT 3 Realtime Monitor 可用于显示分组事件。为避免与存储在 TwinCAT EventLogger 中的消息或警报相混淆，我们将 TwinCAT 3 Realtime Monitor 处理的数据称为（时间）标记。

这些标记可用来表示任务或用户进程的时间行为。为此，我们会向这些标记分配 ID、标记类型、上下文和时间戳。此外，如果需要，还可以提供 UINT 格式的用户定义日期，以便在 Realtime Monitor 的显示中包含附加信息（如错误编号、状态机的状态等）。

标记 ID：

标记 ID 用于识别显示的任务/进程。换句话说，与同一任务/进程相关的所有标记都应使用相同的标记 ID。

标记类型：

TwinCAT 3 Realtime Monitor 可显示事件或进程/操作的时间变化。对于进程/操作的表示，会将其标记为一个序列。序列可以分为一个或多个间隔。可以键入标记来定义序列或间隔的开始或结束。此外，还可以随时间显示应用程序中的事件。因此，对以下几种标记进行了区分：

1. 标记：
   标记可用于记录事件，如警报时间或状态变化等。
2. 序列开始：
   序列开始表示允许任务/进程开始的时间（优先级较高的任务/进程可能会导致延迟）。
3. 间隔开始：
   间隔开始指定了任务/进程实际开始的时间。由于中断等原因，一个序列可能包含多个间隔开始。
4. 间隔停止：
   间隔停止指定了任务/进程不再执行的时间。例如，由于较高优先级的任务或未实现的依赖关系导致出现中断，可能会发生这种情况。
5. 序列停止：
   序列停止表示任务不再运行或进程终止的时间点。

上下文：

上下文描述了标记或标记组的概要。

对于系统任务，在一个核上处理的所有任务均会合并到一个上下文中（如核 0）。因此，这种（实时）上下文映射了实时核中的调度情况。对于这些实时上下文，分配给上下文的任务在任何时候都只有一个处于活动状态。此限制不适用于用户特定标记组。

使用简单标记时（通过使用 FB_RTMon_LogMark [▶ 48]），用户特定标记组会根据其应用端口自动进行分组。例如，在端口为 851 的 PLC 项目中存储的所有标记均会被分配到上下文 ID 为 851（十六进制 0x353）的上下文中。

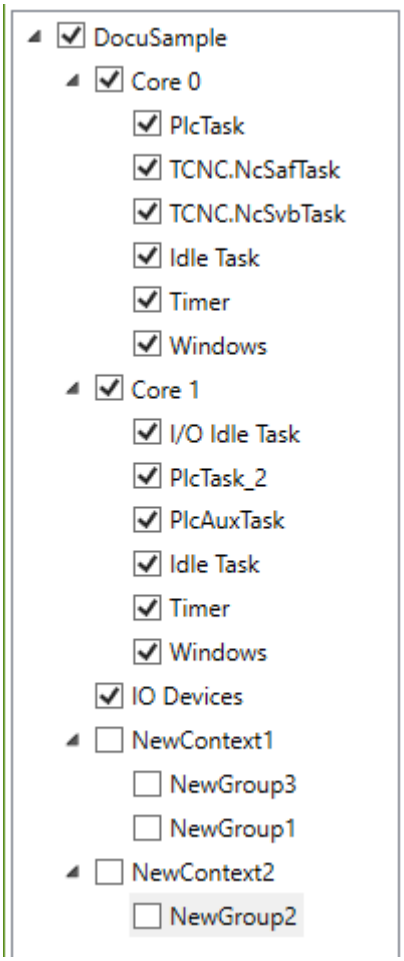
在使用更通用的标记时（基于功能块 FB_RTMon_LogMarkBase [▶ 51]），可以独立定义上下文（即相关性）。例如，可以按进程类型或机器模块（功能单元）进行分组。

树状视图中的表示法：

如上文所述，所有描述相同任务或进程的标记均使用相同的标记 ID。这些标记组合成一个标记组，并在 TwinCAT 3 Realtime Monitor 的树状视图中分配一个条目。

系统任务树中会自动创建一个带有相应任务名称的条目。

例如，对于描述进程的用户相关标记组，必须手动完成上述操作。对于每个已检测到的用户特定标记组，树状视图中会自动出现一个条目 **NewGroup**，可通过标记 ID（与该组的属性窗口中的组 ID 相对应）进行识别。可根据需要对该组进行重新命名（参见 上下文节点 [▶ 44]）。



如 实时监控器中的显示 [▶ 13] 项下所述，各标记组会组合成不同的上下文。在系统任务和使用简单标记时会自动进行此操作。使用扩展标记时，在应用程序代码中传输的上下文 ID 将用于此目的。

标记 ID（组 ID）和上下文的名称均可导出，以供日后重复使用，以及后续进行重新导入。

在图表表示法中，标记组（即任务/进程的所有标记）显示在同一行。有关更多信息，请参见 实时监控器中的显示 [▶ 14]。

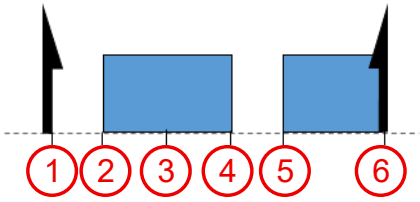
图表中的图例：

图表图例中的符号：

	序列开始
	序列停止
	表示间隔开始或停止。
	标记

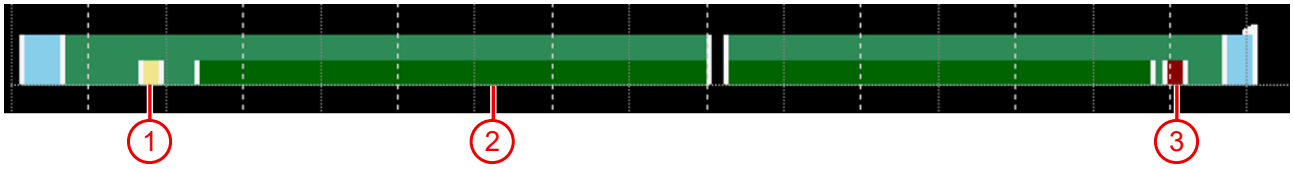
表示法示例：

下图显示了一个任务的可能时间行为的示例。在时间 (1)，任务根据设定的周期时间接收运行“许可”。由于缺少依赖关系，或者由于优先级更高的任务仍处于运行状态，开始时间可能会推迟到时间 (2)。在时间 (3) 传输了一个标记。它可以是“系统事件”标记或用户定义标记。与标记相关的工具提示提供了详细信息。标记本身对任务/进程的时间行为没有任何影响。在时间 (4)，任务中断（例如，再次由于联锁或更高优先级的任务而中断）。在时间 (5)，任务继续运行。在时间 (6)，任务完成。



PLC 运行时模块的处理示意图：

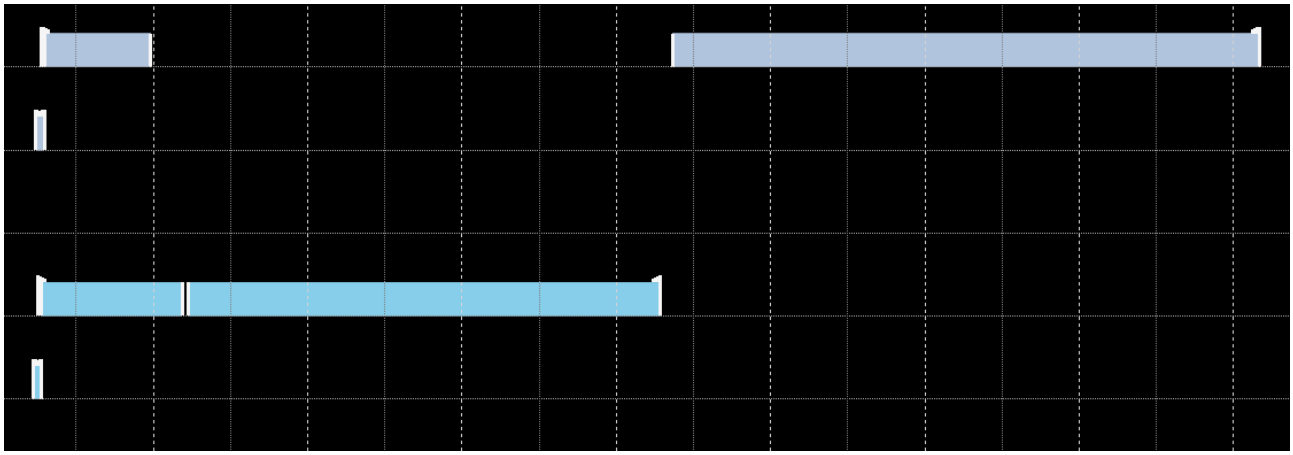
如实时调度 [▶ 9]章节所述，每个 PLC 运行时模块都会自行调用输入和输出更新。在调用 PLC 任务进行周期更新时，会对该 PLC 进行全方位处理。因此，如果激活了详细日志记录，会将 PLC 的处理映射到任务的周期更新中。下图通过举例方式说明了这一点。时间 (1) 表示执行 PLC 运行时模块的输入更新。PLC 代码的周期处理在区域 (2) 进行，在本示例中被另一项任务中断。处理完成后，在时间 (3) 进行 PLC 运行时模块的输出更新。在本示例中，任务本身不执行输入或输出更新。



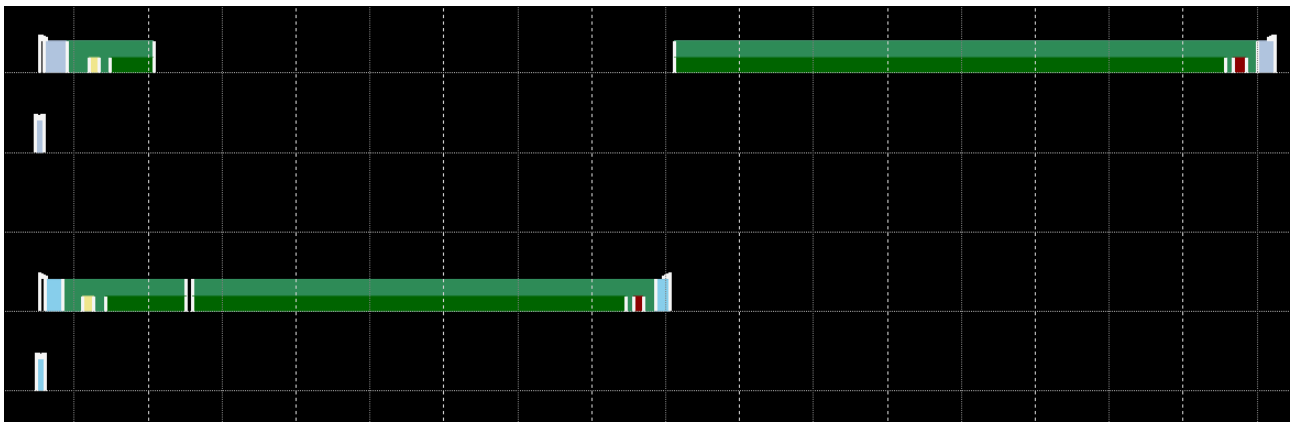
详细日志记录：

选项“detailedLogging”（请参见项目 项目节点 [▶ 44] 或 上下文节点 [▶ 44]）详细阐述了实时任务（以及某一项任务）的执行情况。以下两图阐明了两者的区别。

已启用标准日志记录：



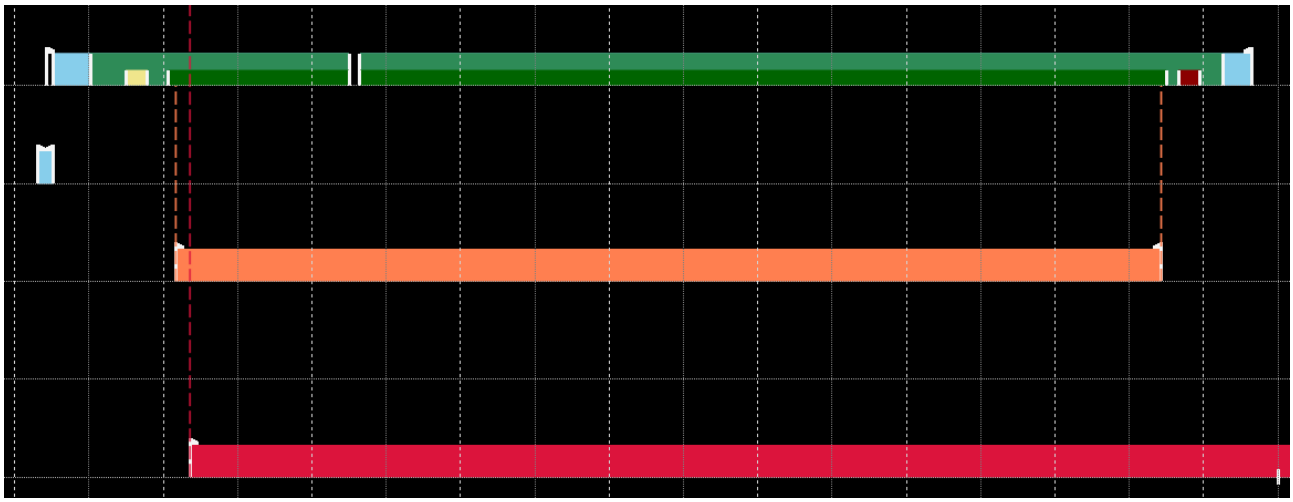
已启用详细日志记录：



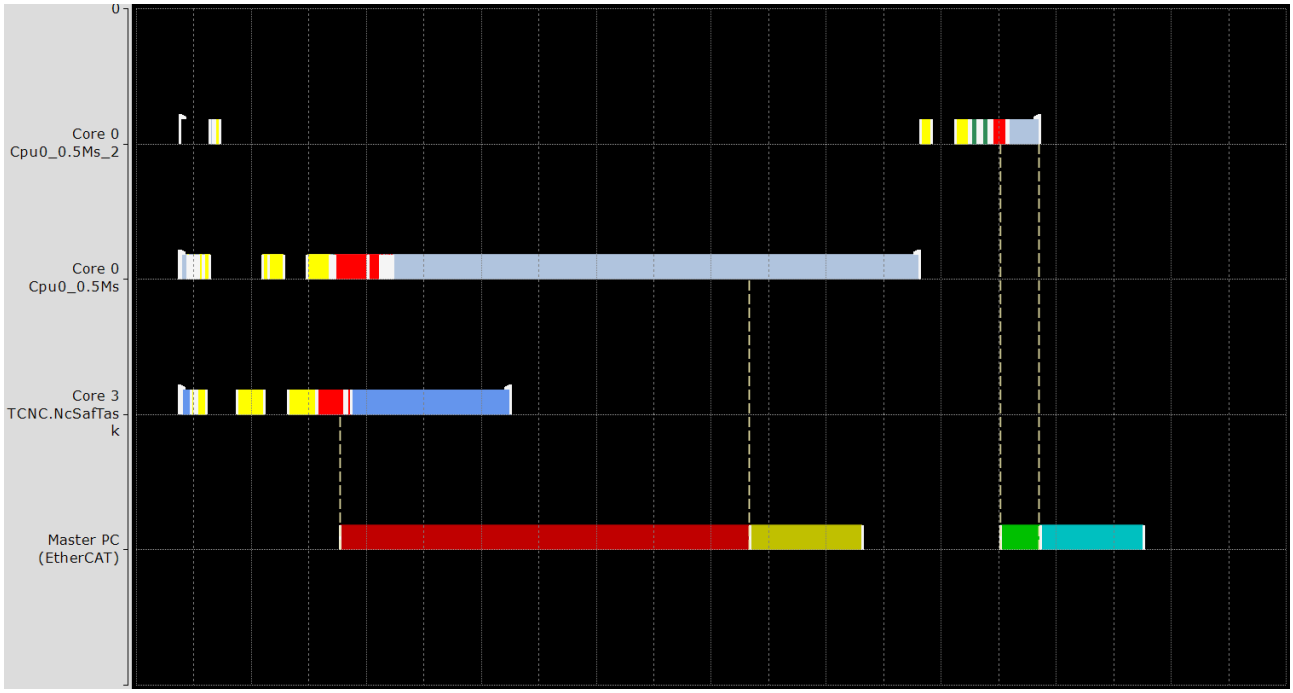
任务引用表示法：

任务引用可用于查看在 TwinCAT 3 Realtime Monitor 中将标记分配给单个实时控制任务的情况。这样既可以分配单个 EtherCAT 帧的发送，也可以将用户上下文分配给实时控制任务。任务引用以虚线表示。

“Show Task Reference（显示任务引用）”选项（请参见标记组元素 [▶ 45]）可针对用户上下文（如用户进程）激活此选项。下图显示了将用户进程（以橙色显示）分配至给 PLC 任务的过程。



EtherCAT 帧会自动启用该选项。下图举例说明了这一点。框架的颜色表示法与 TwinCAT 中相同。



显示窗口中的上下文顺序

根据要检查的目标系统上激活的 TwinCAT 项目，可以显示分布在不同核上的大量上下文。这可能会导致难查看单个上下文的相关性。那么，我们可以根据需要在显示窗口中排列上下文。

1. 禁用项目设置中的“KeepGraphOrder”选项（请参见项目节点 [▶ 44]）。
2. 取消选择项目条目树中的所有上下文。
3. 按显示窗口中的显示顺序选择上下文。

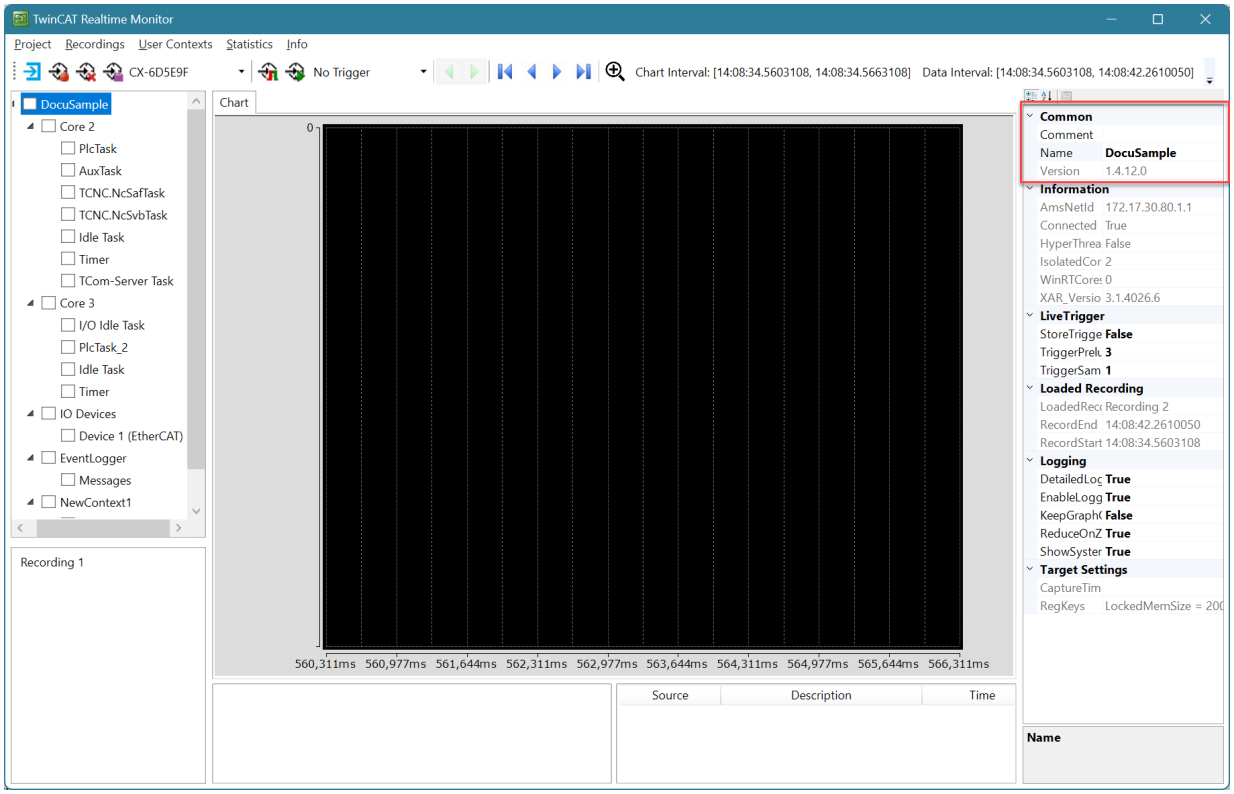
示例：在上图中，所有上下文均被隐藏，并按顺序重新选中，这样即可轻松看到每个任务所发送的 EtherCAT 帧，而不会有任何重叠。

5 快速入门

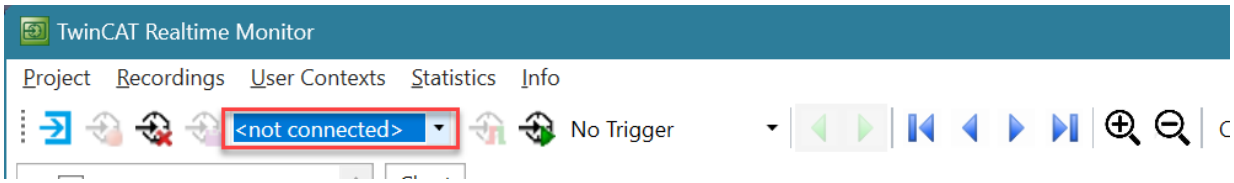
下一章旨在介绍如何使用 TwinCAT 3 Realtime Monitor。

✓ 首先，来看一个在 TwinCAT 3.1 Runtime 版本 3.1.4024.0 或更高版本上运行的项目。

1. 打开 Realtime Monitor。
2. 点击 TwinCAT 3 Realtime Monitor 中的”**Project（项目）**“菜单，选择“**New Project（新建项目）**”，建立新的项目。可通过项目属性更改项目名称。



3. 使用 Realtime Monitor 工具栏选择要分析的目标系统。

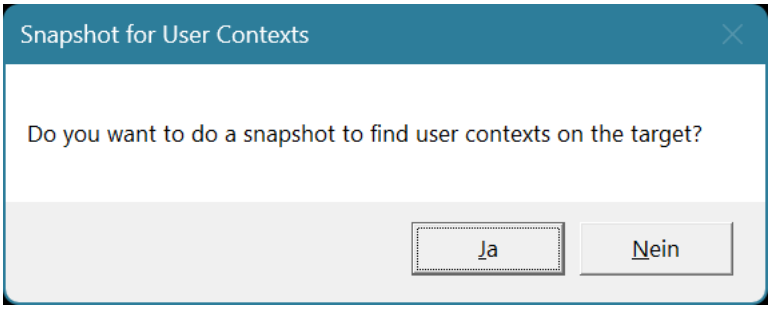


⇨ 将出现一个提示，询问是否要从目标系统中加载配置。

4. 点击“**Yes（是）**”进行确认。

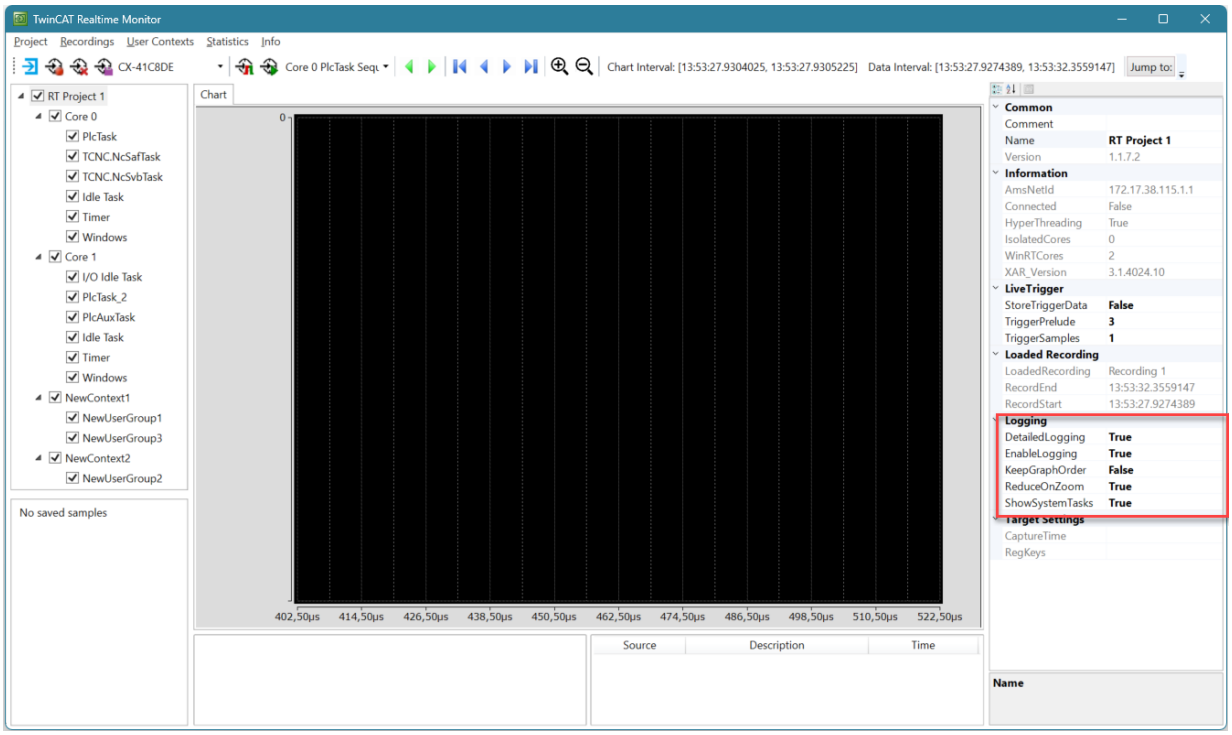
⇨ 目标系统的活动配置已加载，上下文以树状结构分层显示。

⇨ 此时会出现一个对话框，询问 Realtime Monitor 是否应拍摄目标系统的简短快照，以便自动检测和创建用户上下文。



5. 点击“**Yes（是）**”进行确认。

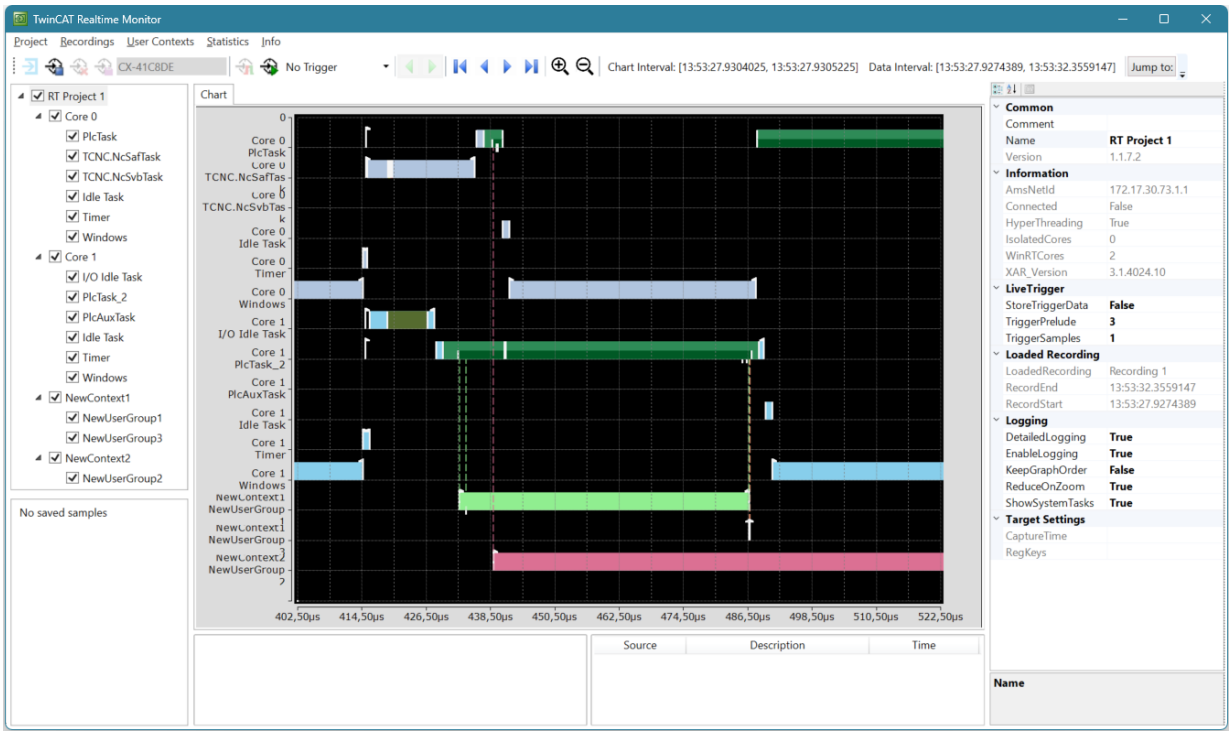
6. 在树状结构中，选择希望在 Realtime Monitor 中显示的上下文/任务，并将其选中。



7. 在树状视图中选择项目，将属性窗口中的“**Detailed Logging（详细日志记录）**”选项设为“True（真）”。（请参见上一步图像中的红色标记或标记组元素 [▶ 45] 的说明）。

8. 点击 Realtime Monitor 工具栏上的”**Start Log data（启动日志记录数据）**“  按钮。

⇨ 开始记录实时行为。



停止记录：

按照如下步骤停止正在进行的记录并保存数据：

✓ 实时监控记录正在运行。

1. 点击 Realtime Monitor 工具栏中的”**Stop data collection（停止数据收集）**“按钮  。
2. 选择菜单项“Recordings（记录）”> **“Save data as Recording（将数据另存为记录）”。**

⇨ Realtime Monitor 的记录此时停止，数据将作为记录保存在项目中。

还请参阅有关此

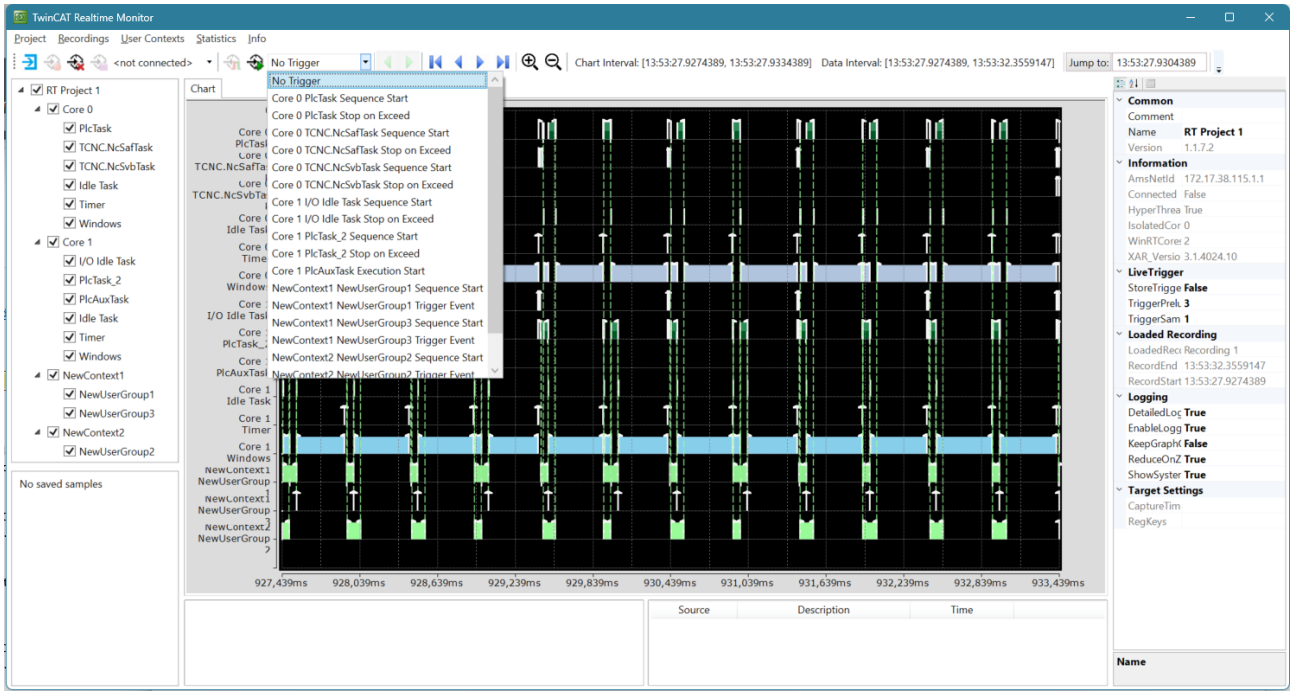
项目节点 [▶ 44]

6 触发器的配置

触发器可用于查找 Realtime Monitor 记录数据流中的重要事件。

以下触发事件会自动输入到每个上下文（任务、用户上下文）的选择列表中：

- 序列开始（用于任务和用户上下文）
- 超出时停止（针对一项任务）
- 触发事件（针对用户上下文中的用户定义标记）。



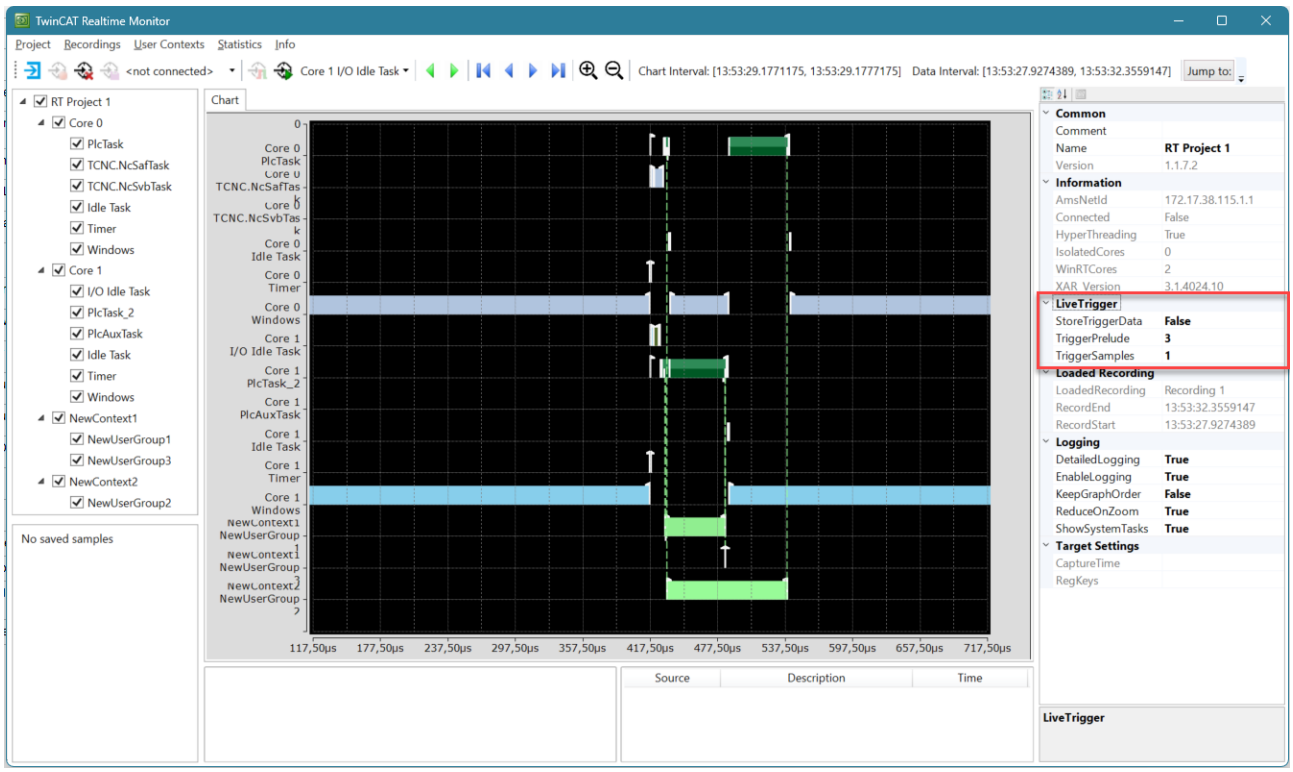
您可以使用以下选项，具体取决于您处理的是记录的数据或是实时数据：

触发实时数据：

如果控制器处于运行模式，且 Realtime Monitor 项目与当前配置相对应，则可使用  按钮根据实时数据启动触发。如果实时数据中出现选择框中选择的事件，Realtime Monitor 会自动停止记录。

以下用于触发实时数据的设置可保存在项目节点设置的“Live trigger（实时触发）”部分：

StoreTriggerData:	定义是否将触发数据保存为记录。
TriggerPrelude:	定义事件发生前保存的时间（以秒计）。
TriggerSamples:	保存的事件记录数量。



记录的数据评估：

在选择框中选择触发事件后，您可使用绿色箭头按钮在事件发生的各个时间之间进行浏览。（请参见章节工具栏 - 实时监控器工具栏 [▶ 40]）

播放选定的触发事件：

通过选择  按钮，可以“快进”方式播放所选触发事件的各个时间点。这样即可快速了解这一事件的发生时间，以及在某些情况下发生时间的变化程度。

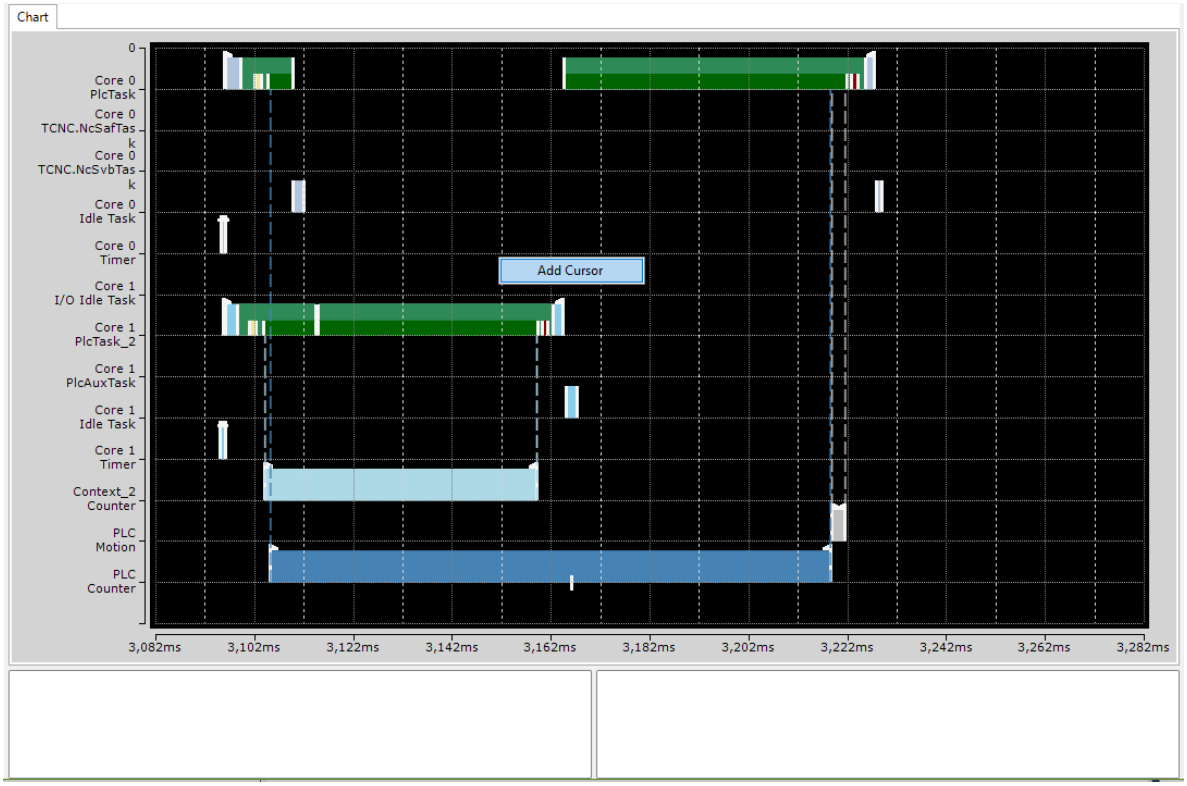
7 光标的使用

为了测量时间或显示在特定时间发生的所有（系统）事件，可以在 TwinCAT 3 实时监控器中使用光标。

添加光标

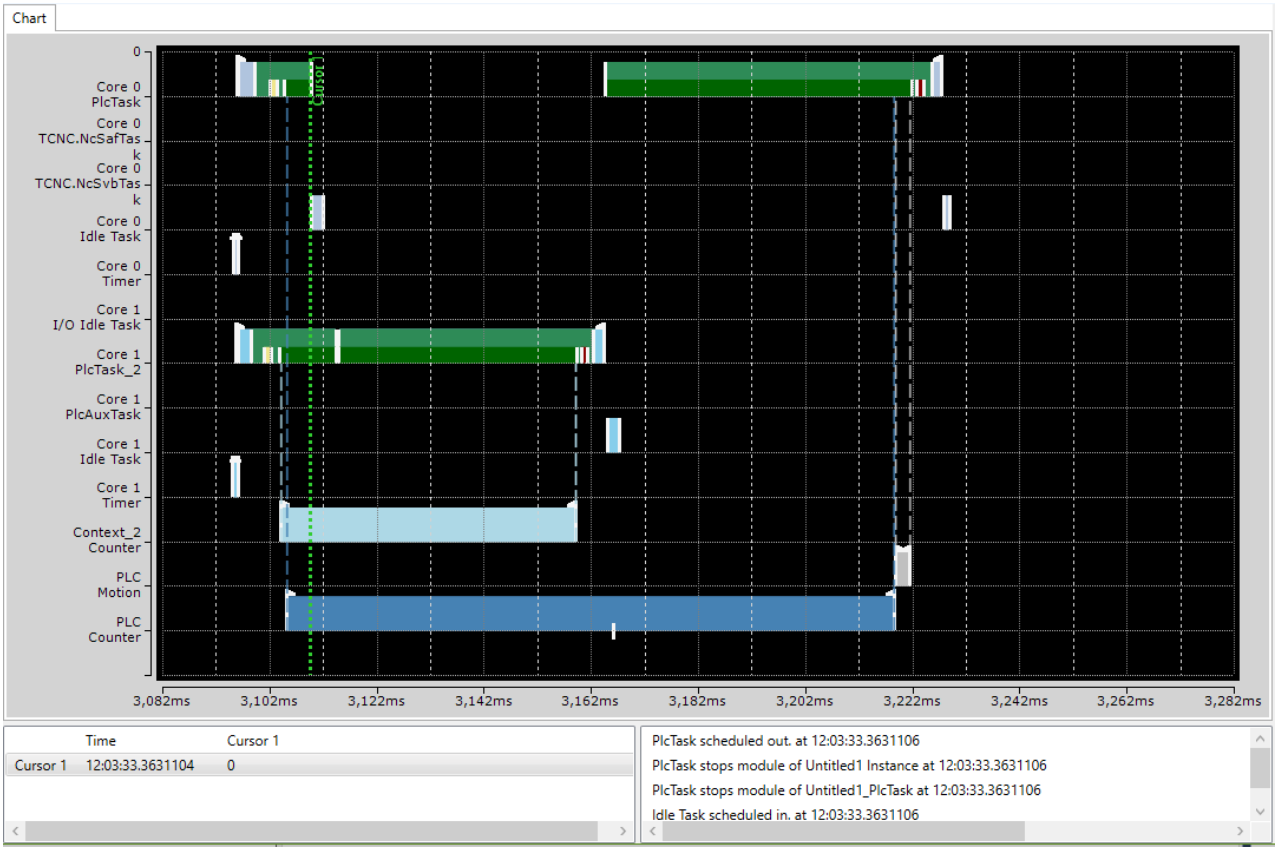
若要添加光标，请按以下步骤操作：

1. 在图表显示区域内点击鼠标右键。
   ⇨ 会打开一个上下文菜单，其中包含 **Add Cursor （添加光标）**和删除所有现有光标的命令。



⇨

2. 点击 **Add Cursor （添加光标）**。
   ⇨ 一个新光标在图表中央创建完成。

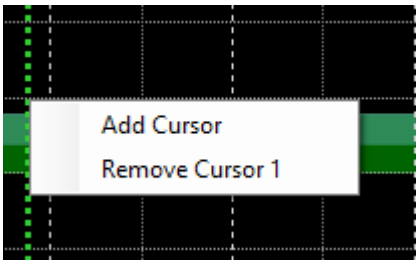


删除光标

删除光标有两种方法：

图表内部

1. 在图表显示区域内点击鼠标右键
   ⇨ 会打开一个上下文菜单，其中包含删除所有现有光标的命令。



2. 使用要删除的光标的 **Remove Cursor （删除光标）**命令。

⇨ 该光标被删除。

光标窗口内部

1. 右键点击要删除的光标。
   ⇨ 会打开一个上下文菜单，其中包含删除光标的命令。
2. 使用 **Remove Cursor （删除光标）**命令删除该光标。

⇨ 该光标被删除。

使用光标导航

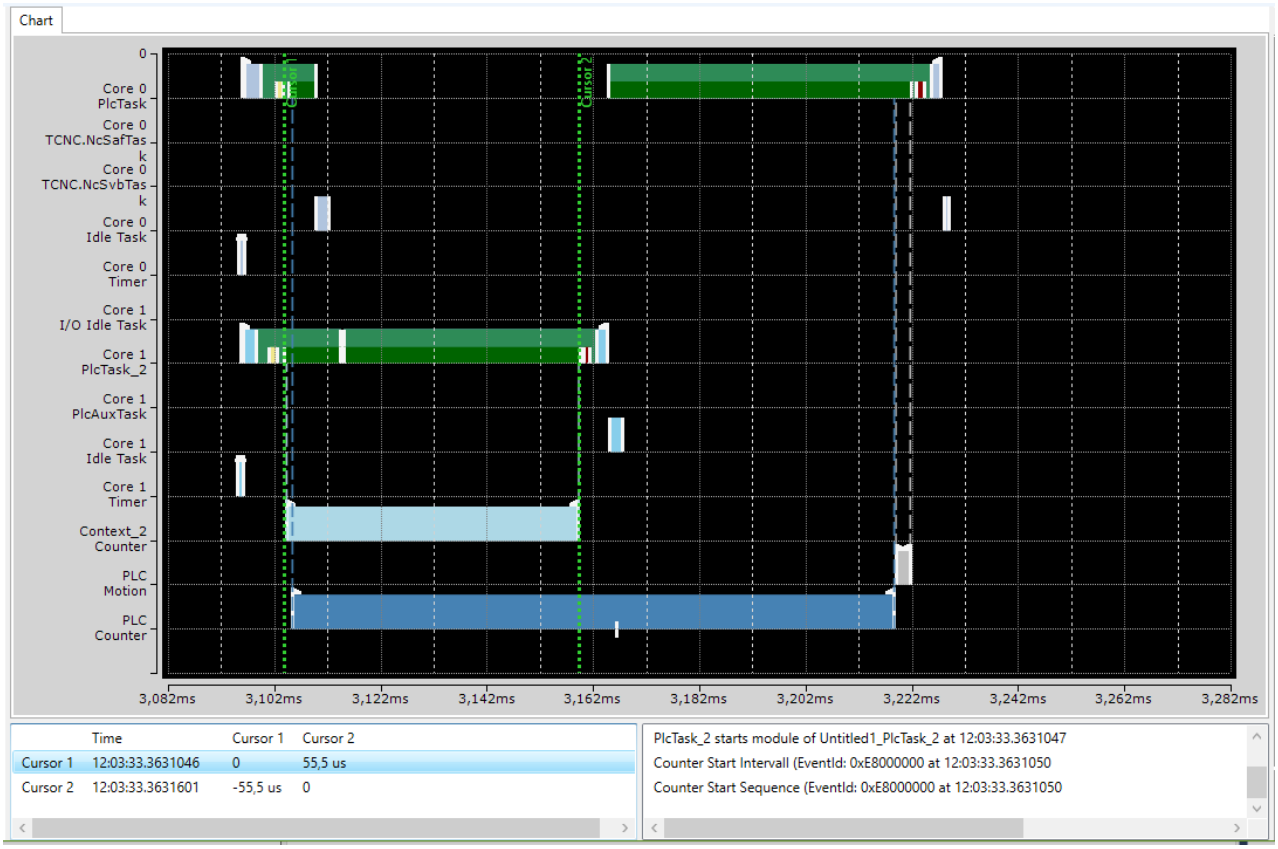
所有创建的光标都会显示在光标窗口中。

	Time	Cursor 1	Cursor 2	Cursor 3
Cursor 1	12:03:33.3630374	0	122,6 us	67,2 us
Cursor 2	12:03:33.3631600	-122,6 us	0	-55,4 us
Cursor 3	12:03:33.3631046	-67,2 us	55,4 us	0

双击光标会使图表中的显示跳转至光标所在的确切位置。光标位于显示区域的中心。

测量时间

光标可用于精确确定进程的执行时间或用户事件的发生时间。为此，请将光标移至显示屏中的相关时间处。光标会自动"锁定"事件，并在光标窗口中显示该活动光标在该时间发生的事件。在下图中，这是光标 1 的应用程序事件“**计数器开始间隔**”。



若要测量"Context_2_Counter"进程的持续时间，请按以下步骤操作：

1. 为进程开始创建一个光标或使用现有光标。
2. 将光标移至进程"Context_2_Counter"的序列/间隔开始标记处，以便在事件窗口中显示该事件（参见上图）。
3. 使用另一个光标进行相应操作，并将其移至进程"Context_2_Counter"的序列/间隔停止标记处。
   ⇨ 现在，所有现有光标之间的差异都会以表格形式显示在光标窗口中。
4. 直接从表中读取正在使用的光标的值。在本示例中，执行时间为 55.5 µs。

光标属性

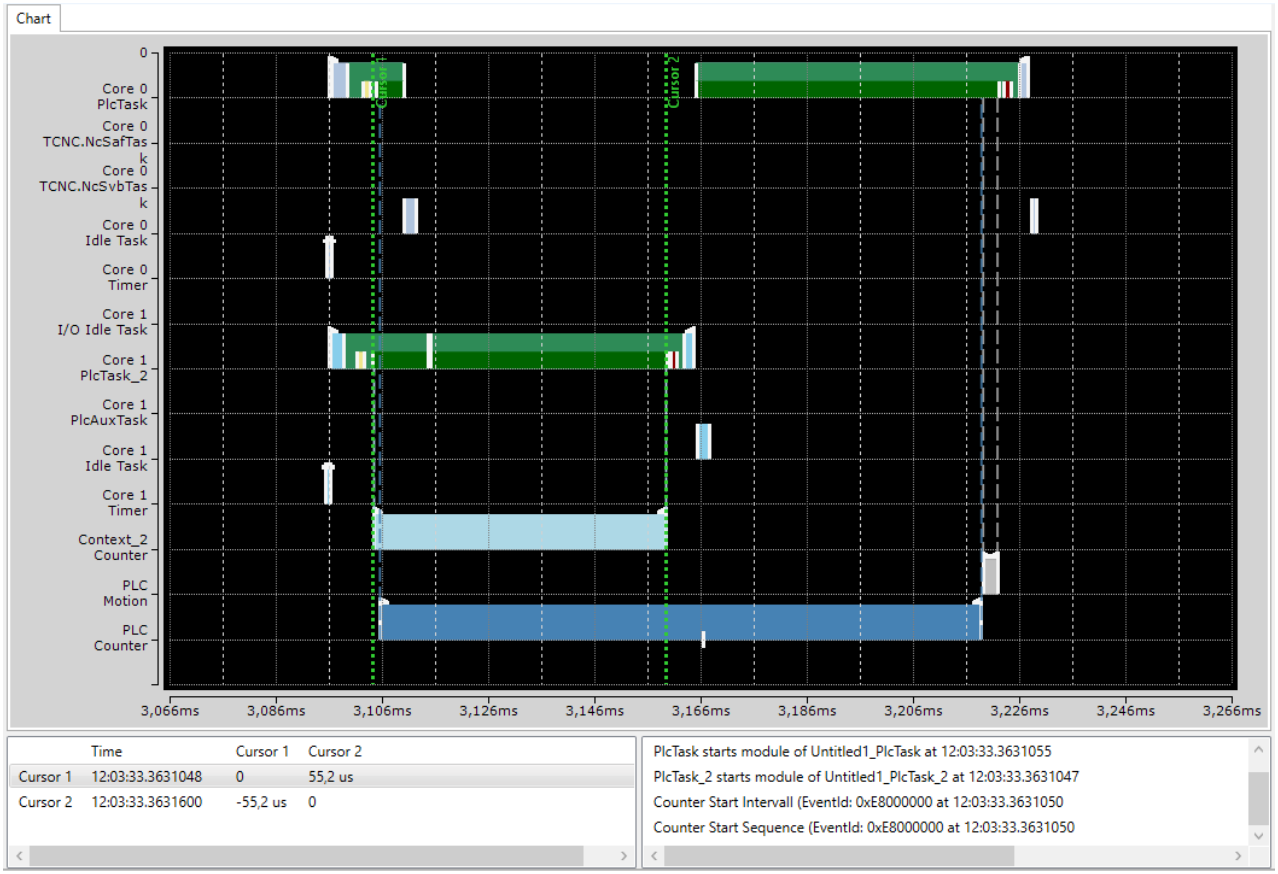
光标所具备的属性如下。

属性	含义
光标信息	
颜色	允许更改活动光标的颜色。
文本	显示光标处显示的文本。
信息	
触发光标	激活 TriggerCursor（触发光标）属性，该属性会使触发模式下的光标保持在图表窗口中的同一位置，而不是锁定在某个时间点上（这样会导致光标从显示区域消失）。

事件窗口

事件窗口显示了活动光标此时发生的所有事件。在下图中，光标 1 发生了以下事件：

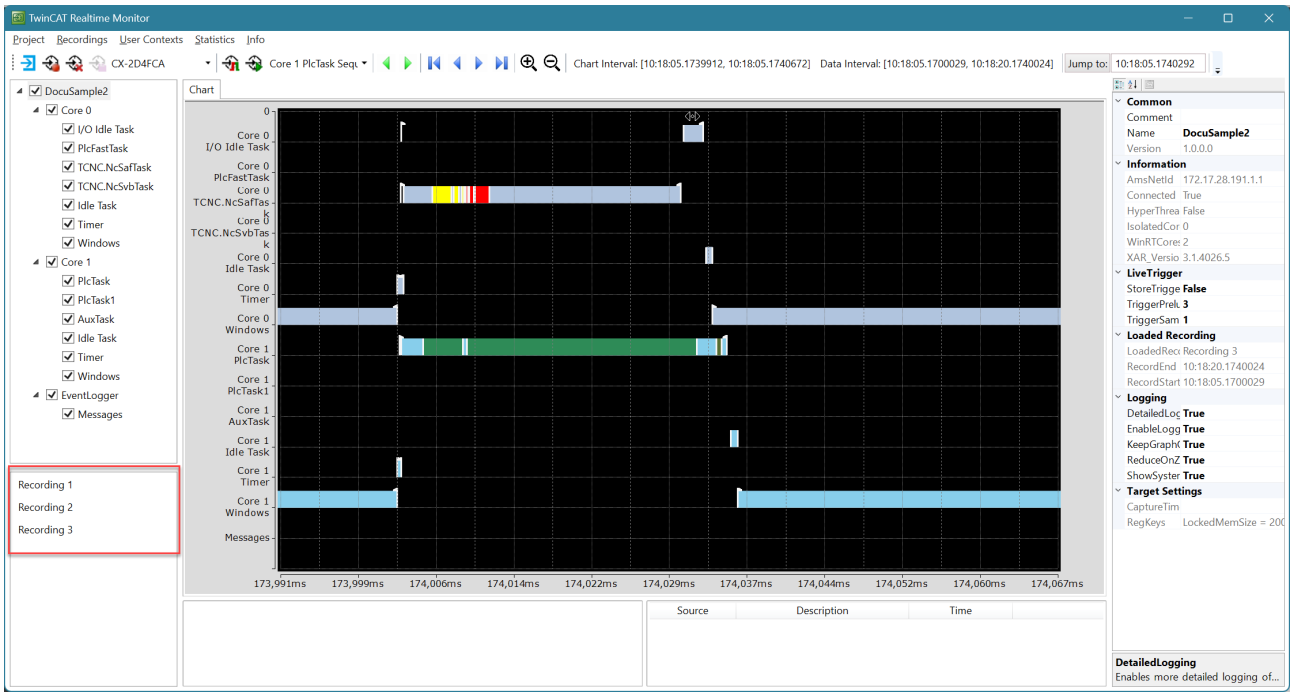
- PlcTask 开始处理运行时模块 Untitled1。
- PlcTask_2 也开始处理运行模块 Untitled2。
- 计数器应用进程同时启动序列和间隔。



8 记录

Realtime Monitor 1.2 以及之后版本具有记录功能。

在一个 Realtime Monitor 项目中，可使用相同的配置创建多个记录并存储在项目中。这些记录会显示在记录列表中。（请参见下图红色框）



您可通过上下文菜单和“Recordings（记录）”菜单管理这些记录。

保存记录

在使用 Realtime Monitor 完成记录后，您可使用工具栏 [▶ 40]中的  按钮（Save collected data within project（在项目中保存收集的数据））或使用“Recordings（记录）”菜单中的“Save data as Recording（另存为记录）”项，在项目中将数据另存为记录。

添加记录

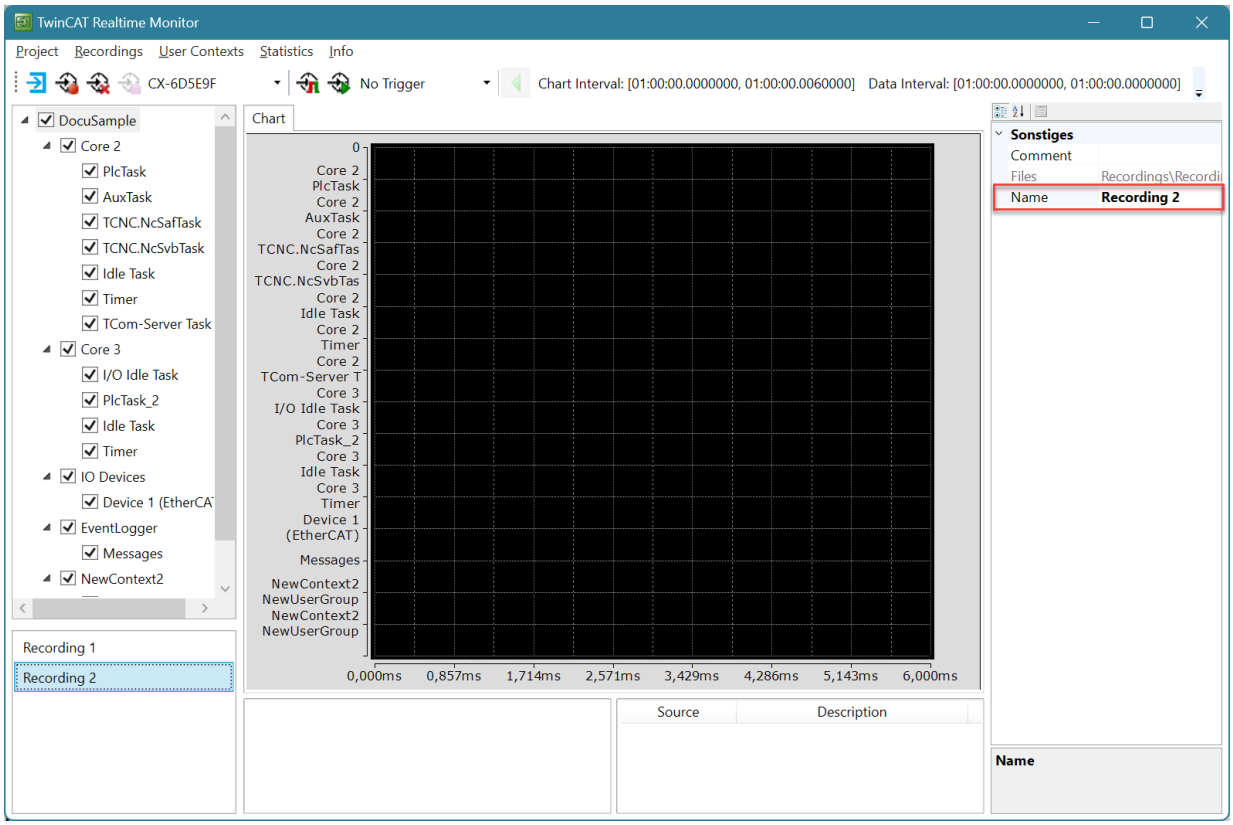
若要将记录的数据添加到 Realtime Monitor 项目中，请使用“Recordings（记录）”菜单中的“Import Recording（导入记录）”菜单项。然后系统会将该记录添加到列表末尾，并按升序进行相应编号。

Realtime Monitor 的数据文件只包含项目名称。数据文件本身没有名称，只有时间戳。因此，如果从项目中移除记录，之后再重新添加，显示的名称（例如显示的记录编号）将会有所不同。所加载记录的开始和结束时间将显示在项目属性中。（请参见项目节点 [▶ 44]）

重新命名记录：

1. 如要重新命名记录，请选择该记录并切换到属性窗口。
2. 点击记录名称字段，可进行更改。

3. 按“Return（返回）”按钮确认更改。



记录的所有数据文件均保存在一个文件夹中。重命名记录会更改存储文件夹的名称，而不是文件的名称。如果将此记录加载到另一个 Realtime Monitor 项目中，将不会保存名称更改。

选择记录

如要在显示窗口中显示记录，请选择该记录并使用上下文菜单中的**“Load Recording to Chart （将记录加载到图表）”**命令，或双击要显示的j记录。

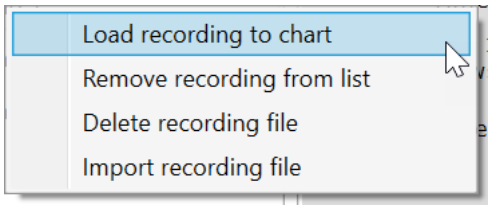
移除记录

如果要从项目中移除记录，但不从硬盘中删除该记录，请在记录列表中选择此记录，使其突出显示。然后使用上下文菜单项**“Remove Recording from List（从列表中移除记录）”**。

删除记录

如果要从项目和硬盘中删除记录，请在记录列表中选择此记录，并进行标记。然后使用上下文菜单项**“Delete Recording File（删除记录文件）”**。

记录列表中的上下文菜单项



下表显示了所有上下文菜单项：

命令	含义
将记录加载至图表	将所选记录加载至显示窗口。
从列表中移除记录	将记录从列表中移除。记录文件保留在硬盘上。
删除记录文件	从列表中删除记录，并从硬盘中删除相应文件。
导入记录文件	将记录导入到 Realtime Monitor 项目。

还请参阅有关此

记录列表 [▶ 42]

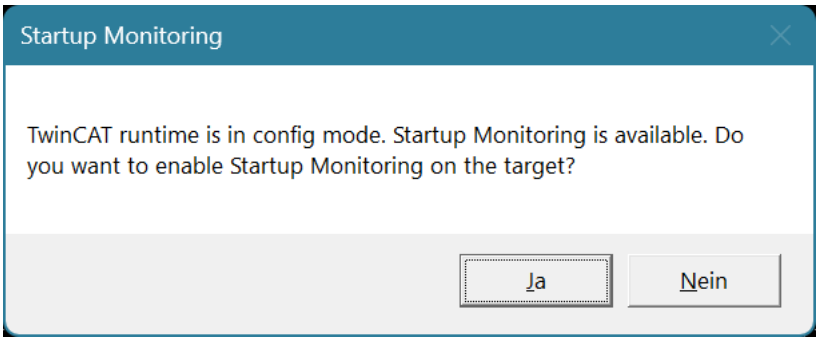
9 记录启动行为

利用 TwinCAT 3.1 版本 3.1.4026 或更高版本，可记录 TwinCAT 3.1 实时系统的启动行为，并通过 TwinCAT 3 Realtime Monitor 进行显示。

记录启动行为的程序：

✓ 执行快速入门 [▶ 18]章节中的第 1-7 步，这样即可获得配置完好的 Realtime Monitor 项目。

1. 将 TwinCAT 3.1 Runtime 切换到配置模式。
2. 按下 Realtime Monitor 中的记录按钮。
   ⇨ 将出现以下信息：



3. 点击“**Yes（是）**”进行确认。
4. 将 TwinCAT 3.1 Runtime 切换至运行模式。

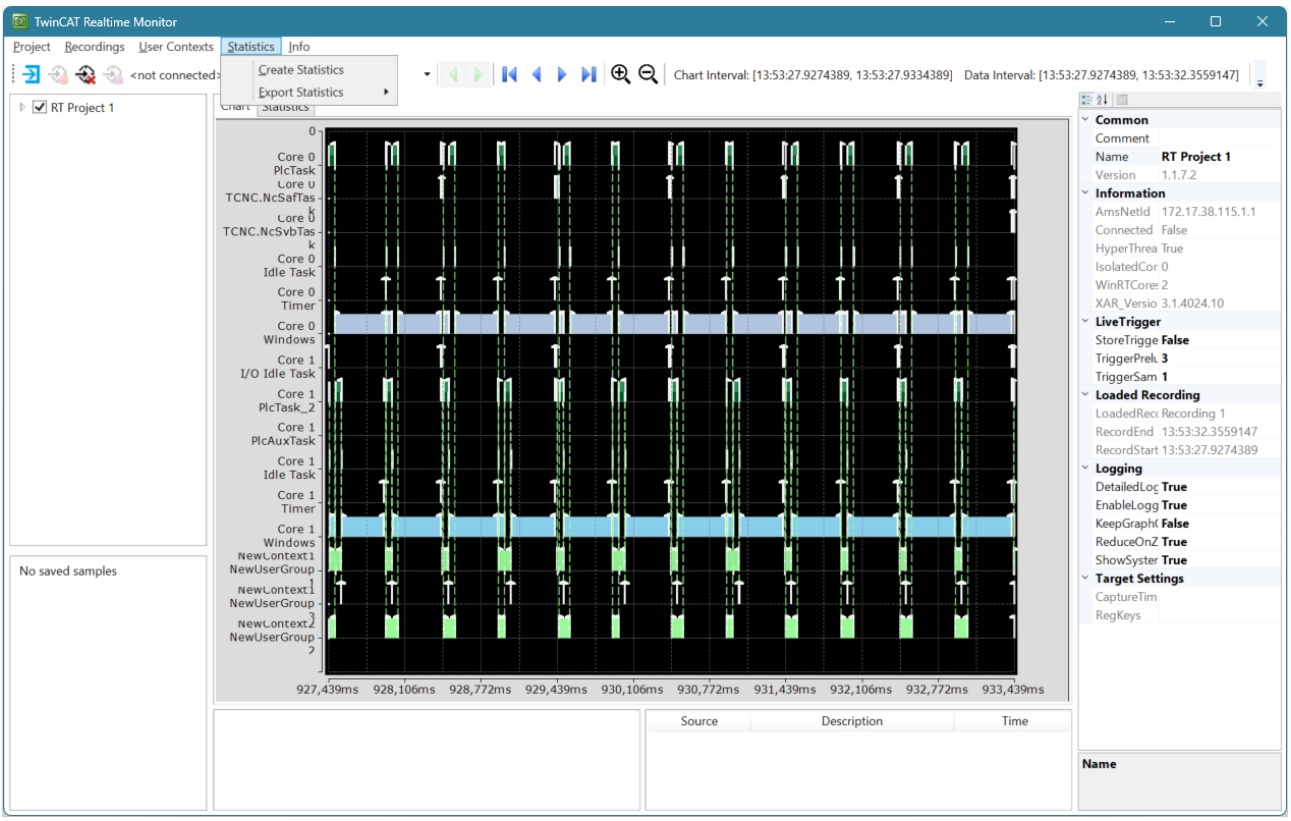
⇨ Realtime Monitor 会自动开始记录。

10 统计信息

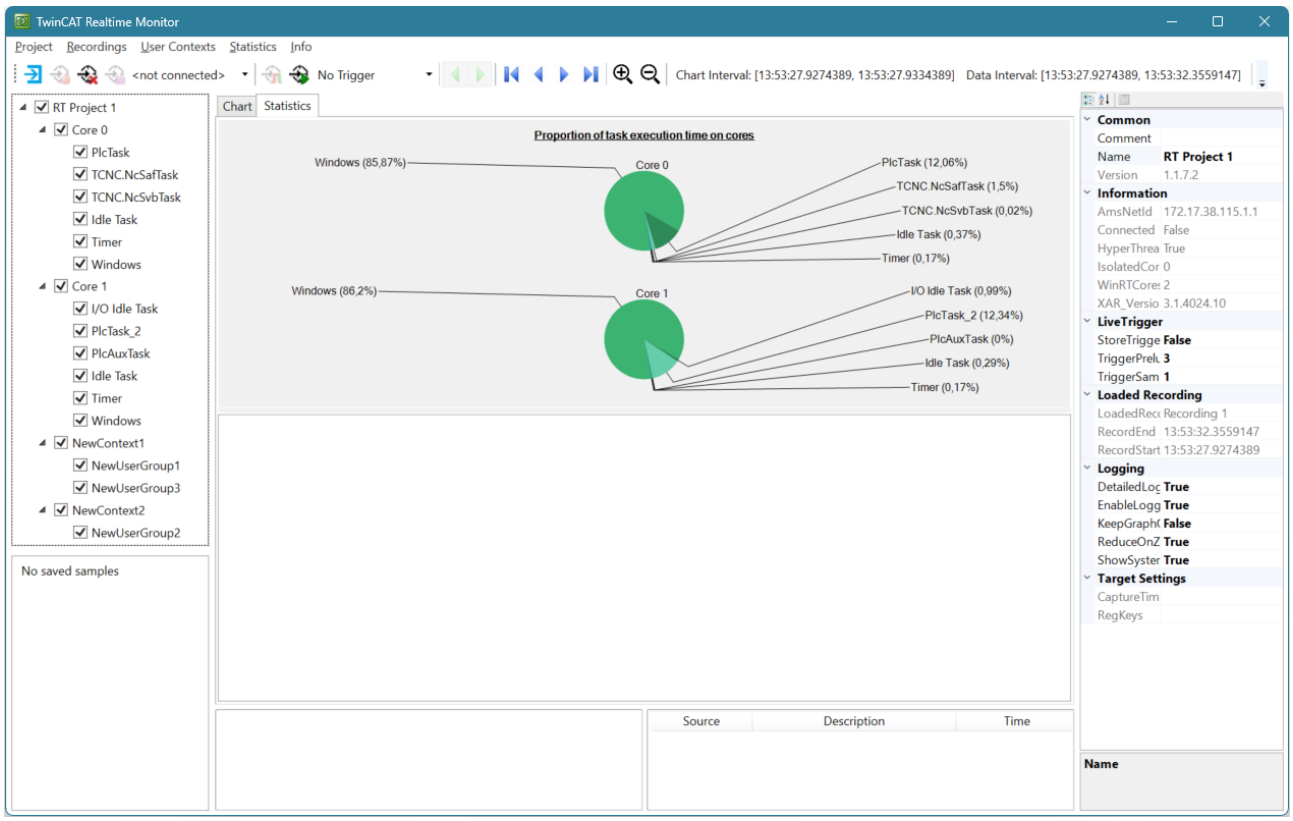
为能够对实时系统的配置进行说明，可以借助 TwinCAT 3 Realtime Monitor 生成统计信息（请参见“参考”>“用户界面”>“统计信息 [▶ 38]”章节）。

创建和显示统计信息

1. 如要针对加载的项目创建统计信息，请在**“Statistics（统计信息）”**菜单中选择**“Create Statistics（创建统计信息）”**。



⇨ 创建统计信息并打开以下视图：



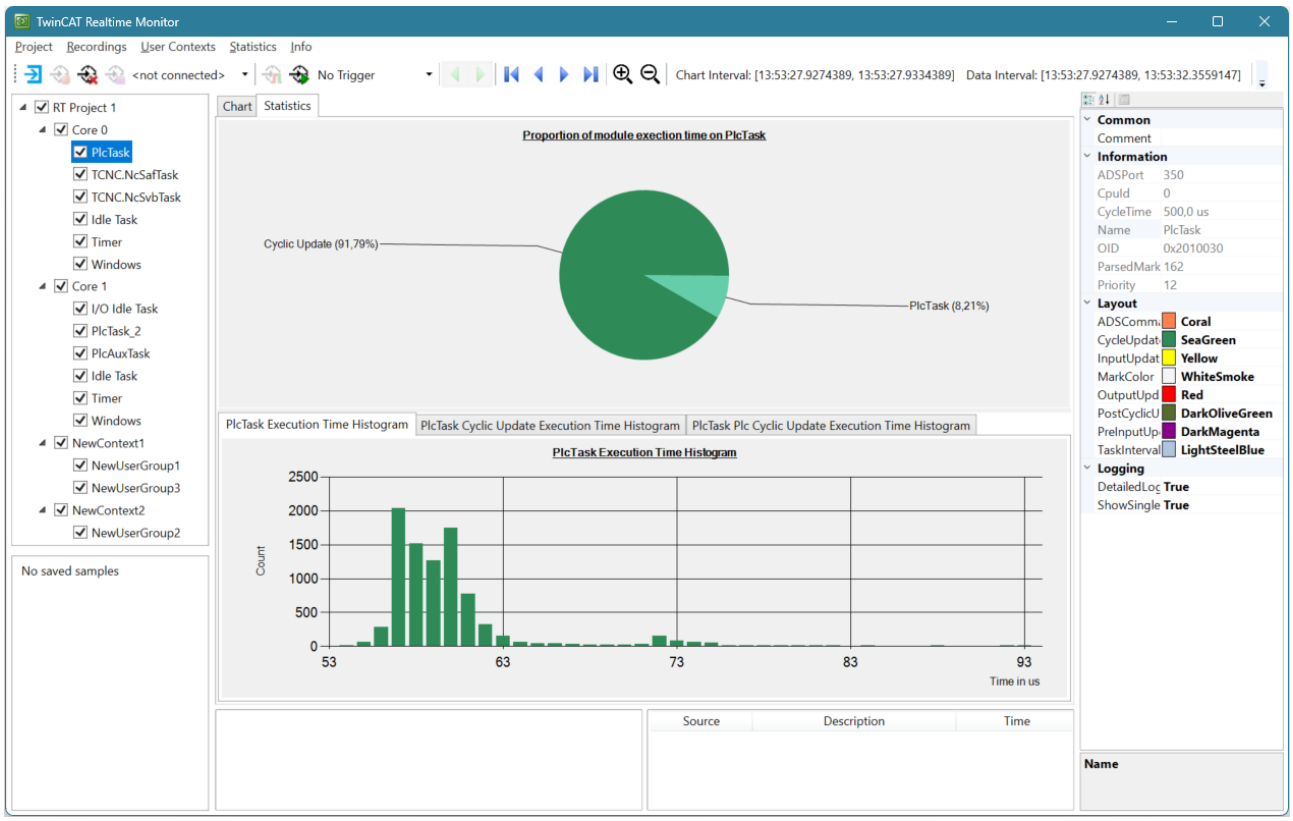
每个实时核的执行时间分布以饼图形式显示。在实时核上执行所有任务的时间总和，包括空闲任务和可能的操作系统上下文（隔离核除外），始终为 100%。

如果选择项目树中使用的某个实时核，便会看到单个实时核的类似显示。

在饼图下方的区域，您可查看执行时间随时间的变化情况，即查看分布在多个选项卡（每个 TwinCAT 任务一个）中的直方图。

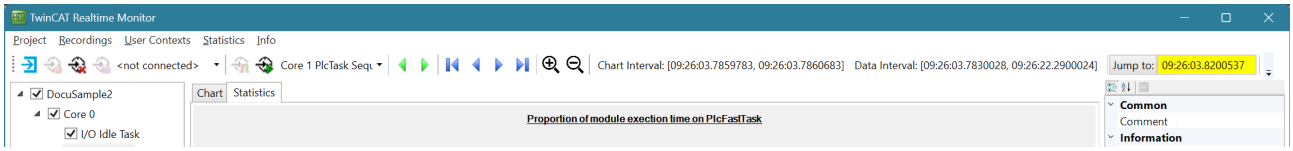
如果直接在项目树中选择一个实时任务，即可对时间行为进行更精确的分析。对于 PLC 任务，您可以获得循环更新视图，即在单独的选项卡中显示执行应用程序代码的区域。

条形图显示了记录期间执行时间的分布情况。如要查看所有执行时间，请使用鼠标滚轮滚动条形图。这样即可放大或缩小显示。



如要查看第一次出现相应循环时间的上下文，请按以下步骤操作：

1. 用鼠标左键双击分布图中相应的条形图。
   ⇨ 在“Jump to（跳转到）”字段之后输入时间戳，并以黄色突出显示。

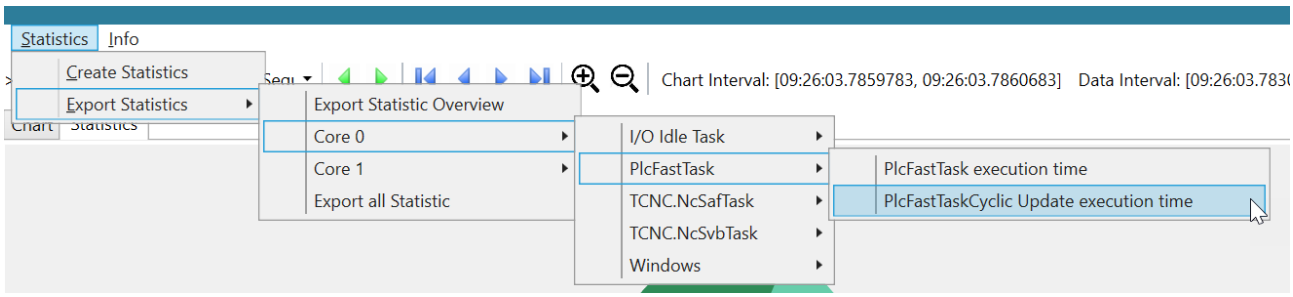


2. 点击“Jump to（跳转到）”按钮，切换到图表视图。
   ⇨ 相应的时间戳会显示在图表中央。
3. 如要取消“Jump to（跳转到）”窗口中的选择，请点击图表视图中的任意区域。

导出统计信息：

如要在您的工具中执行进一步分析，可以导出各项任务的时间戳。导出的文件格式为 *.csv 文件。

4. 在“Statistics（统计信息） -> Export Statistics（导出统计信息）”菜单中，选择您要导出的时间上下文。



⇨ 您将收到一个 *.csv 文件，其中包含相应的时间戳。

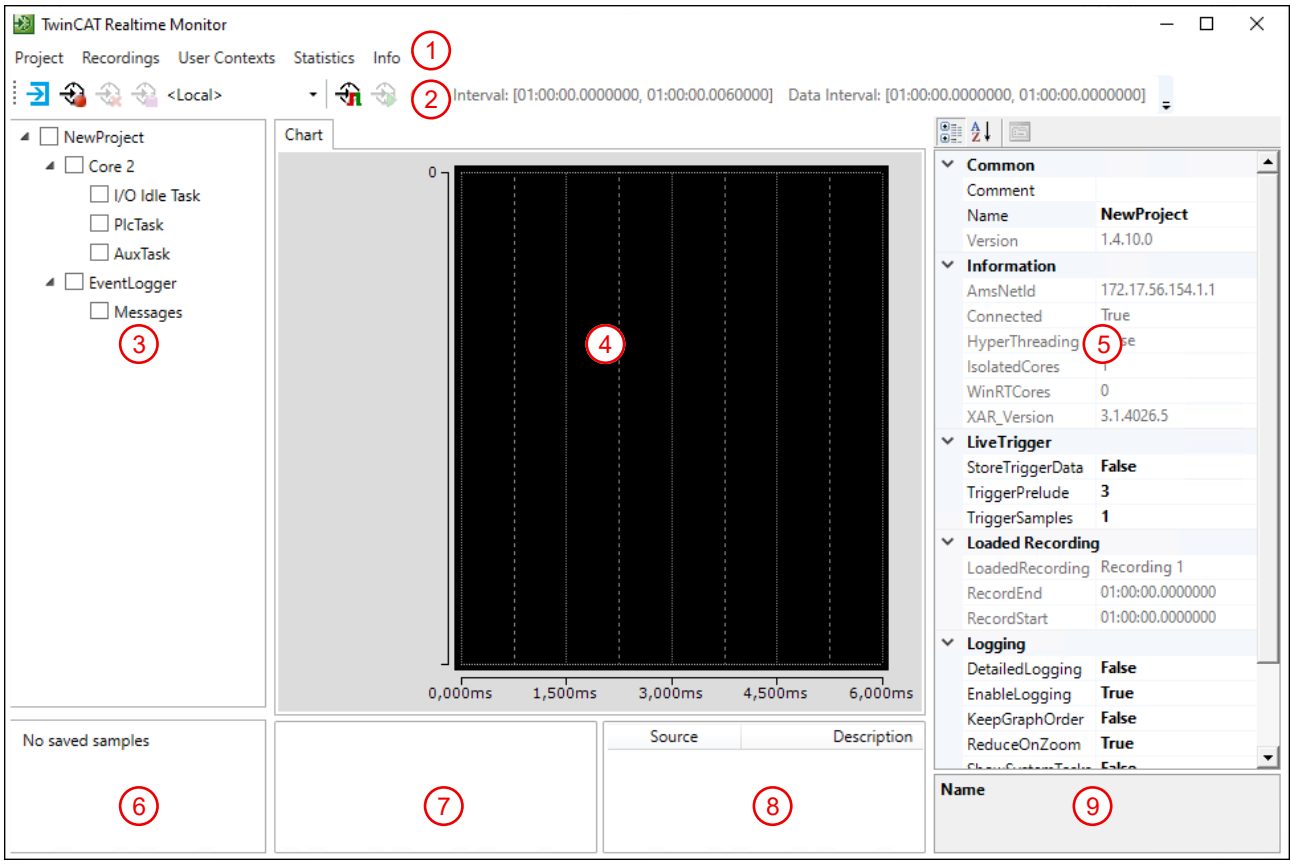
同时，还可导出全部统计信息。

1. 在“Statistics（统计信息）”菜单中，选择“Export all Statistic（导出所有统计信息）”选项。

⇨ 您将收到一个 ZIP 压缩包，其中包含单独的 *.csv 文件，用于项目中的每个时间上下文。

11 用户界面参照

TwinCAT 3 Realtime Monitor 的用户界面由以下几个部分组成：

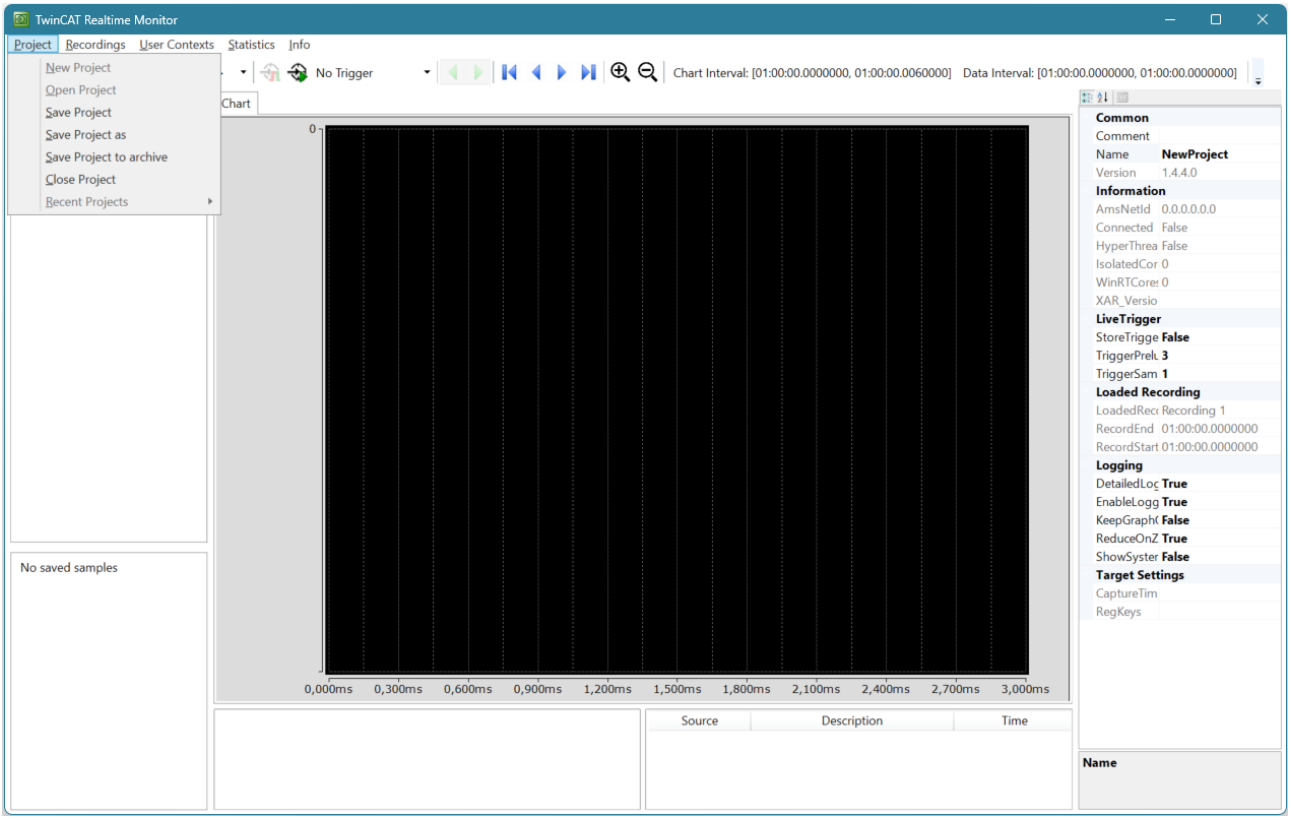


1	菜单栏
2	工具栏
3	项目树
4	显示窗口
5	属性窗口
6	记录
7	光标窗口
8	事件窗口
9	所选属性的描述显示

11.1 菜单栏

TwinCAT 3 Realtime Monitor 的菜单栏提供以下命令。

11.1.1 项目



新建项目

功能：该命令用于创建一个新的 Realtime Monitor 项目。

调用： Project（项目） > New Project（新建项目）

打开项目

功能：该命令用于打开现有 Realtime Monitor 项目。

调用： Project（项目） > Open Project（打开项目）

保存项目

功能：该命令用于保存现有 Realtime Monitor 项目。

调用： Project（项目） > Save Project（保存项目）

将项目另存为

功能：该命令用于将现有 Realtime Monitor 项目保存为待定义的名称。

调用：Project（项目） > Save Project as（将项目另存为）

将项目存档

功能：该命令用于将现有 Realtime Monitor 项目（包括其中包含的记录）保存为 zip 文件。

调用：Project（项目） > Save Project to archive（将项目存档）

关闭项目

功能：该命令用于关闭现有 Realtime Monitor 项目。

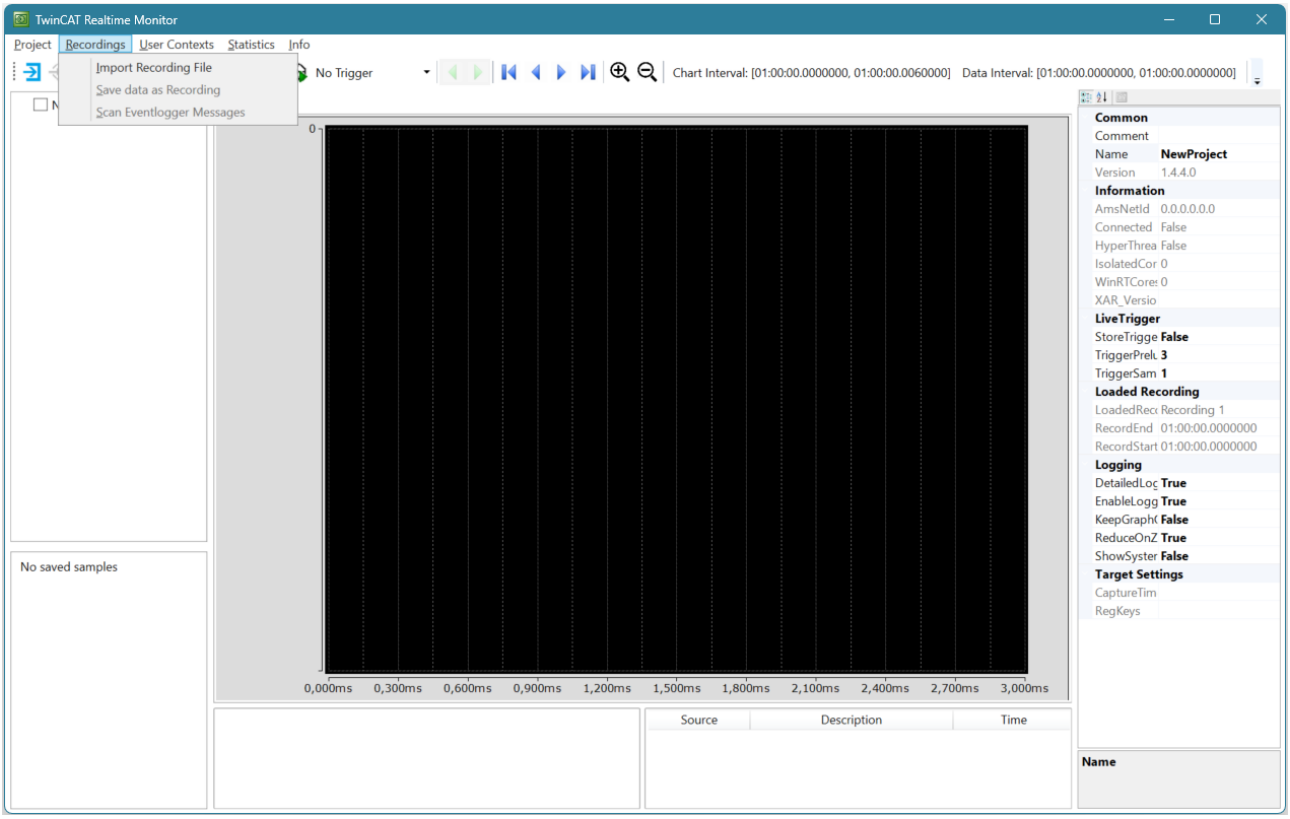
调用：Project（项目） > Close Project（关闭项目）

最近项目

功能： 该命令用于显示最近打开的 Realtime Monitor 项目列表，您可在此处重新打开项目。

调用： Project（项目） > Recent Projects（最近项目）

11.1.2 记录



导入记录文件

功能： 该命令用于导入记录文件。

调用： Recordings（记录） > Import Recording File（导入记录文件）

将数据另存为记录

功能： 该命令用于将记录的数据另存为记录。

调用： Recordings（记录） > Save data as Recording（将数据另存为记录）

保存 Eventlogger 信息

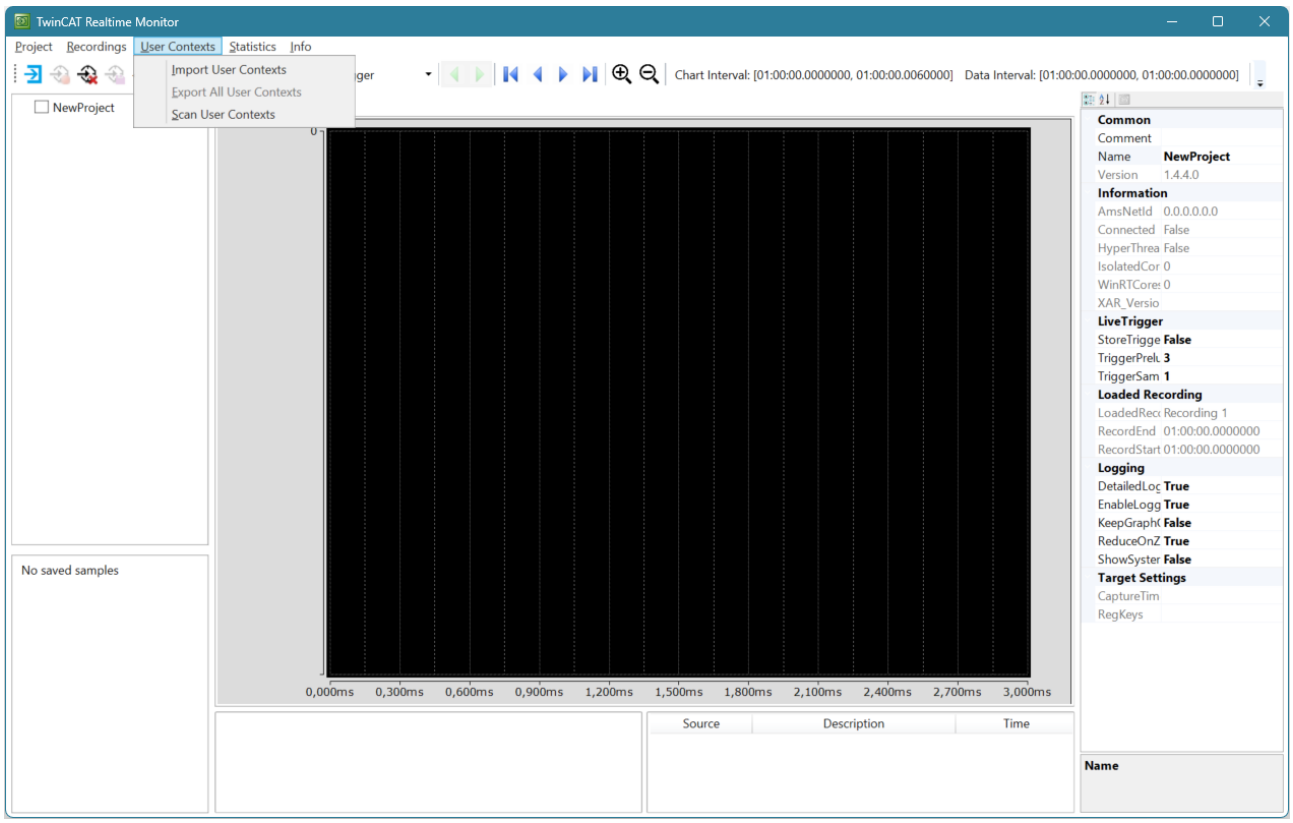
功能： 该命令用于扫描 EventLogger 信息，并将其作为事件添加到 Realtime Monitor 数据中。

调用： Recordings（记录） > Save Eventlogger Messages（保存 Eventlogger 信息）

还请参阅有关此

记录 [▶ 27]

11.1.3 用户上下文



导入用户上下文

功能：该命令用于将现有用户上下文导入 Realtime Monitor 项目中。如果项目包含以前（自动）找到的具有相同事件组和事件 ID 的上下文，系统会询问用户是否要用所保存的名称和设置替换该上下文。

调用：User Contexts（用户上下文） > Import User Contexts（导入用户上下文）

导出所有用户上下文

功能：该命令用于从打开的 Realtime Monitor 项目中导出现有用户上下文。

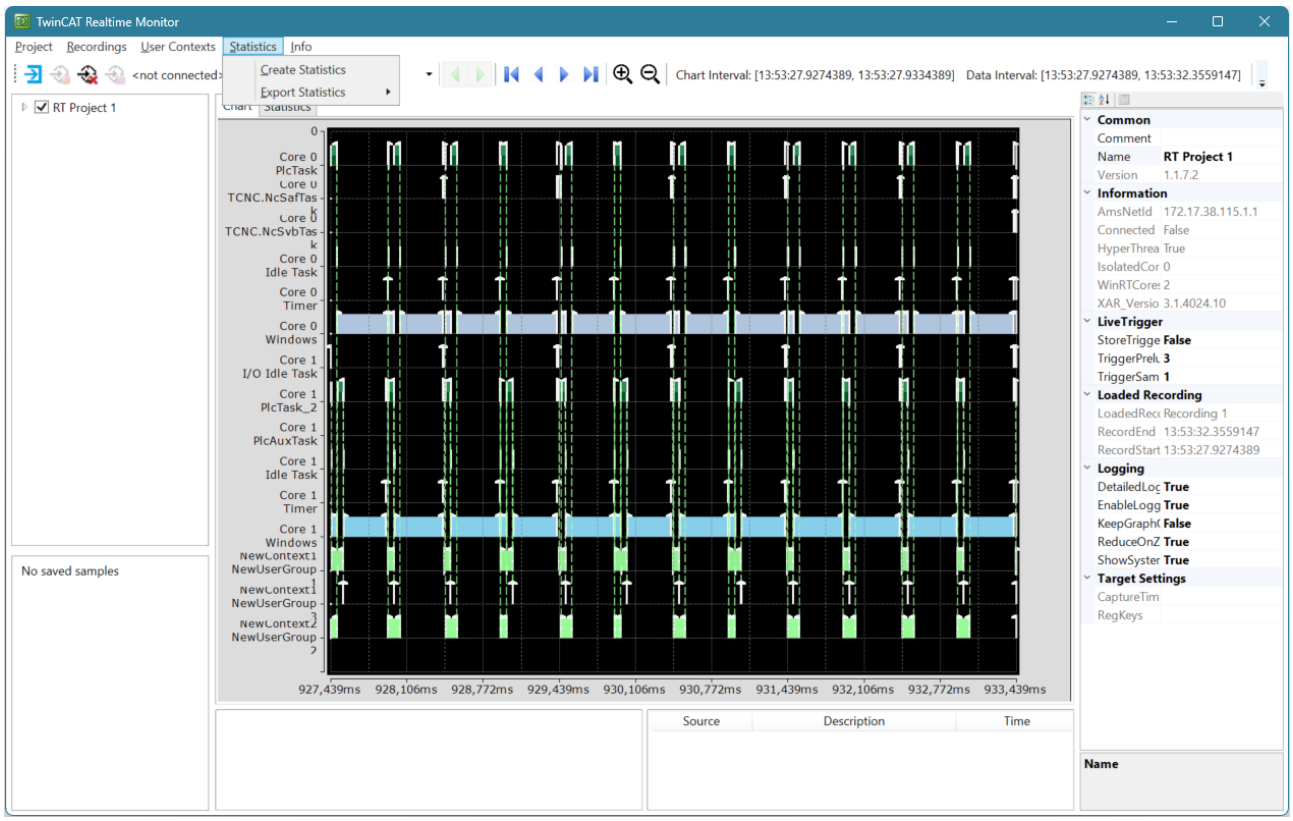
调用：User Contexts（用户上下文） > Export All User Contexts（导出所有用户上下文）

扫描用户上下文

功能：该命令用于扫描现有用户上下文并将其插入 Realtime Monitor 项目中。

调用：User Contexts（用户上下文） > Scan User Contexts（扫描用户上下文）

11.1.4 统计信息

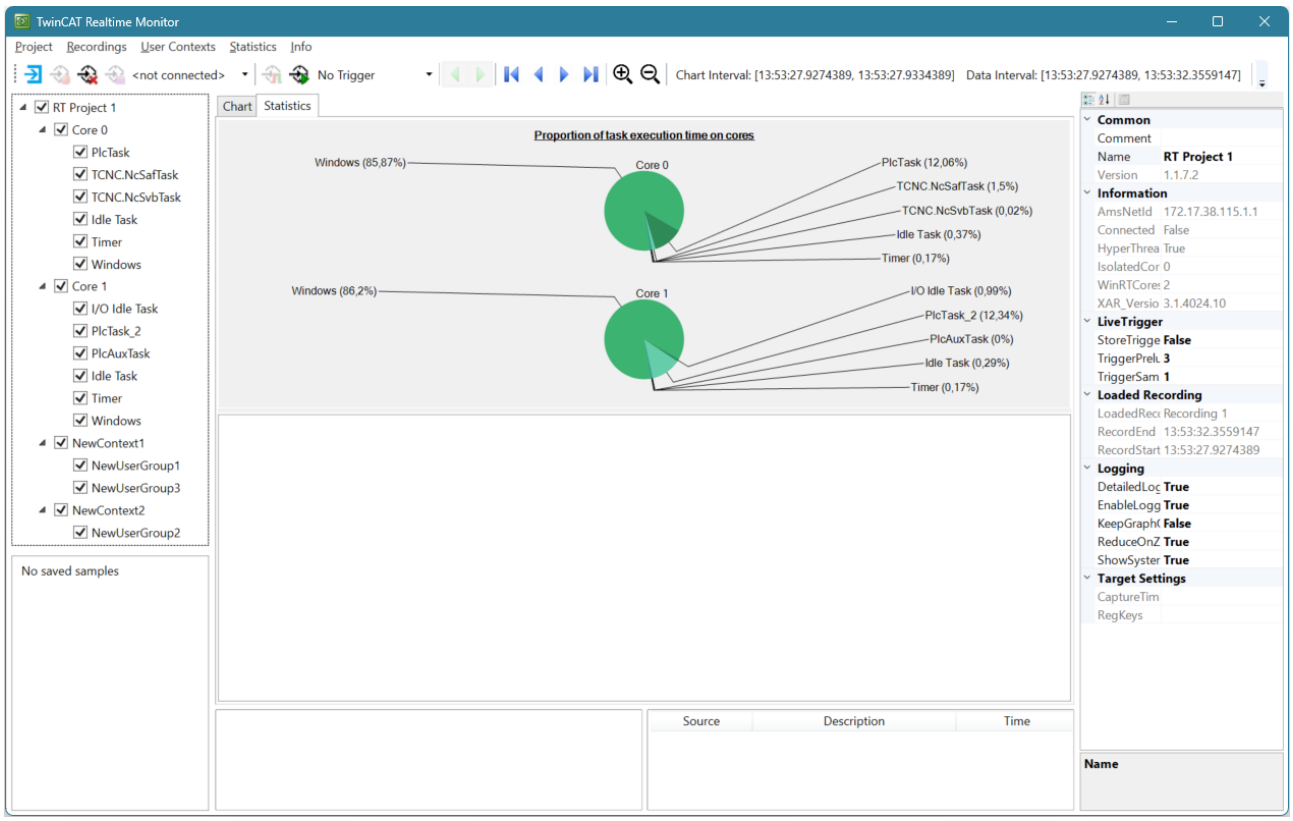


创建统计信息

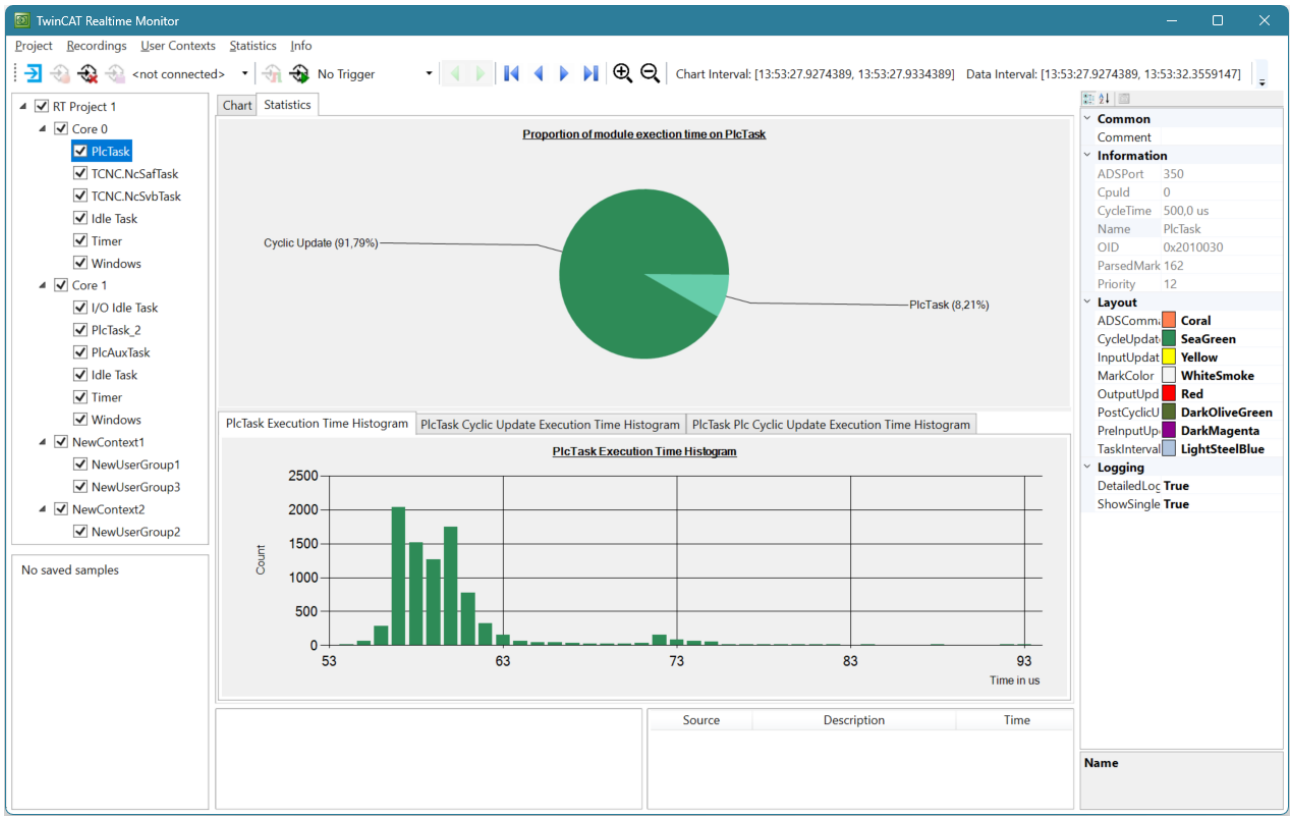
功能：该命令用于评估以 TwinCAT 3 Realtime Monitor 记录的标记并生成统计信息。该信息会显示在 **Statistics（统计信息）**选项卡中。

调用：Tools（工具） > Create Statistics（创建统计信息）

生成统计信息的示例：



选择特定任务后：

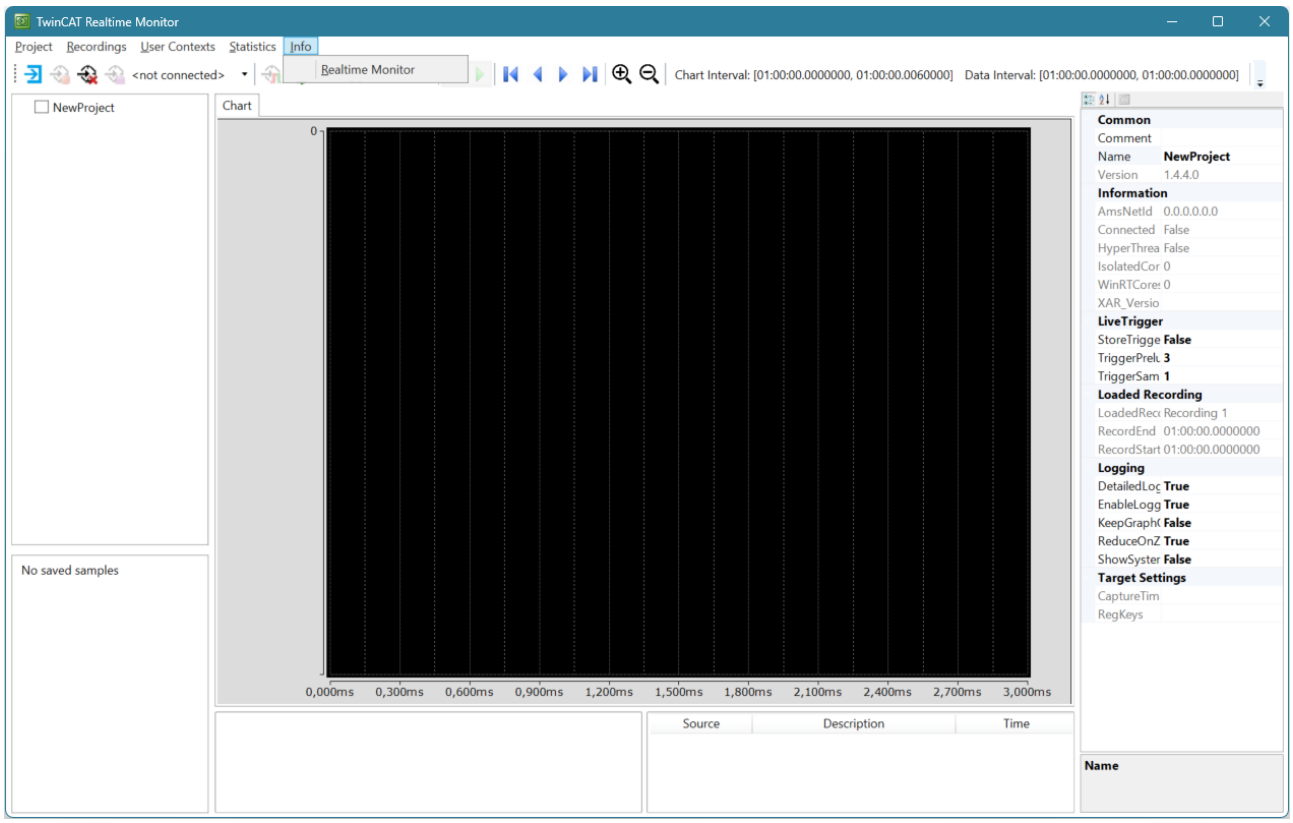


导出统计信息

功能：该命令可用于将选定的统计信息导出至 CSV 文件中。如果导出了所有统计信息，单个 CSV 文件将打包成一个压缩包。

调用：Tools（工具） > Export Statistics（导出统计信息）

11.1.5 信息



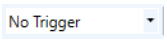
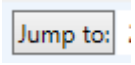
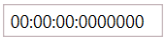
Realtime Monitor

功能：该命令可用于打开一个对话窗口，其中会显示所安装的 TwinCAT 3 Realtime Monitor 版本号。

调用：Info（信息）> Realtime Monitor

11.2 工具栏 - 实时监控器工具栏

TwinCAT 3 Realtime Monitor 工具栏提供以下几种命令。

	从设定的目标系统加载项目配置。
	开始记录。
	停止记录。
	删除显示的数据和记录的数据。
	在项目中保存记录的数据。
	选择目标系统
	开始触发实时数据。
	开始触发记录的数据。
	选择触发器
	手动跳转至下一个触发事件
	手动跳转至上一个触发事件
	跳转至显示开始位置。
	将显示向左移动。
	将显示向右移动。
	跳转至显示结束位置。
	放大
	缩小
图表间隔	当前区域显示的时间间隔
数据间隔	记录数据的时间间隔
	跳转到下一个输入字段中指定的时间。
	用于输入时间的输入字段

11.3 项目树

项目树分层显示所有标记组及其分配到上下文的情况。系统任务树中会自动创建一个带有相应任务名称的条目。在将系统任务分配至核后，会将其分组到相应的上下文中。

同时，还会在项目树中为用户相关的标记组创建一个条目。根据所使用的调用，用户程序中的上下文分配（请参见 FB_RTMon_LogMark [▶ 48] 或 FB_RTMon_LogMarkBase [▶ 51]）或与用户程序的 ADS 端口相关，或基于用户定义的上下文 ID。

根据用户相关节点属性页面对其进行命名（请参见 上下文节点 [▶ 44] 或 标记组元素 [▶ 45]）。

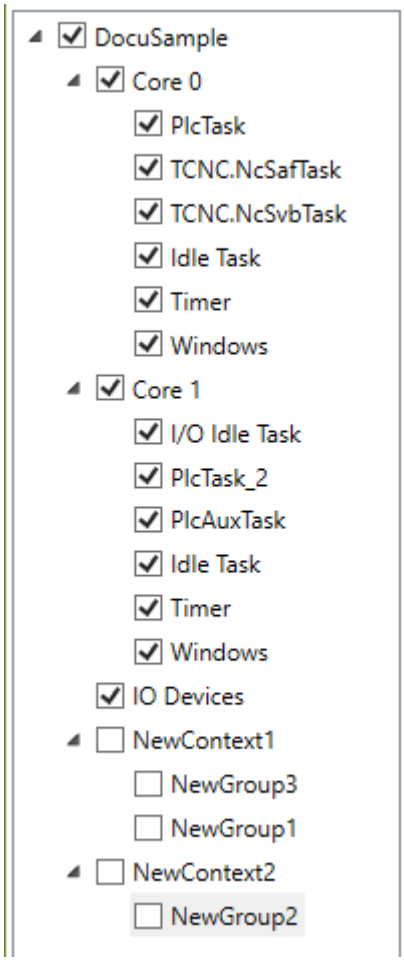
项目树中的上下文菜单项

下表显示了项目树中的所有上下文菜单项（以及可以使用的节点类型）。

命令	节点类型	含义
添加新用户上下文	项目节点	添加用户上下文。
导入用户上下文	项目节点	导入用户相关上下文，包括所有子元素。
添加新用户组	用户上下文节点	添加用户相关标记组。
移除用户上下文	用户上下文节点	删除用户上下文。
导出用户上下文	用户上下文节点	导出用户相关上下文，包括所有子元素。
移除用户组	用户相关标记组节点	删除用户相关标记组。

示例：

下图显示的是项目树表示法，它在记录开始后自动生成。点击“Yes（是）”，确认关于是否搜索用户上下文的查询。除了分布在核 0 和核 1 上的系统任务外，还生成了三个用户相关标记组，在此尚未命名。



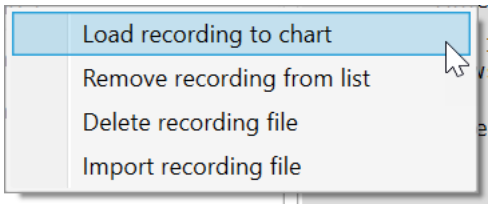
还请参阅有关此

- 项目树 [▶ 42]
- 项目节点 [▶ 44]
- 用户上下文 [▶ 37]

11.4 记录列表

可在一个 Realtime Monitor 项目中管理多个记录。这些记录会显示在记录列表中。有关记录主题的更多信息，请参阅“记录 [▶ 27]”章节。

记录列表中的上下文菜单项



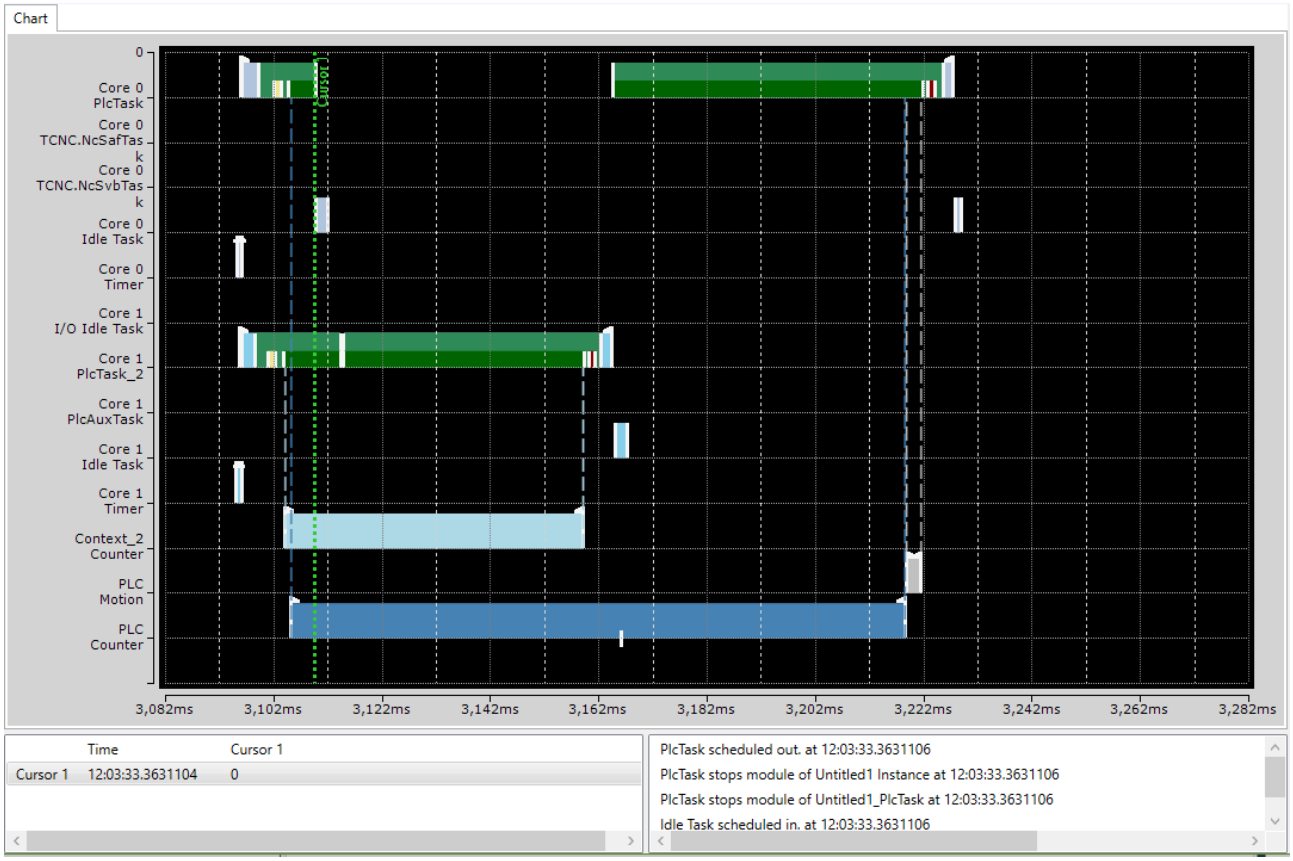
下表显示了所有上下文菜单项：

命令	含义
将记录加载至图表	将所选记录加载至显示窗口。
从列表中移除记录	将记录从列表中移除。记录文件保留在硬盘上。
删除记录文件	从列表中删除记录，并从硬盘中删除相应文件。
导入记录文件	将记录导入到 Realtime Monitor 项目。

11.5 显示窗口

在显示窗口（图表）中，（时间）标记随时间显示，并按照各标记组进行排序。

您可使用工具栏功能（请参见工具栏 - 实时监控器工具栏 [▶ 40]）以及鼠标来导航和缩放显示窗口。



如要进行时间测量或分析（请参见光标的使用 [▶ 23]），可使用上下文菜单设置、删除或移动光标。

11.6 属性窗口

属性窗口中显示了项目树中当前活动（选定）元素的属性。

Logging（日志记录）区域中所列的属性始终适用于树形图中的所有子元素。**Different Settings（不同设置）**值表示子元素的值不同。更改值后，子元素的值也会随之更改。

11.6.1 项目节点

可在 TwinCAT 3 Realtime Monitor 的项目节点上进行以下设置：

属性	含义
通用	
Comment	项目注释
Name	项目名称
Version	项目版本
Information	
AmsNetId	目标系统的 AmsNetId
Connected	与目标系统的连接状态
HyperThreading	表示超线程是否激活。
IsolatedCores	显示项目中使用的隔离核的数量。
WinRTCores	显示项目中使用的 Windows 实时核的数量
XARVersion	显示目标系统的 TwinCAT 版本。
Live Trigger	
StoreTriggerData	将触发数据存储为记录。
TriggerPrelude	以秒为单位定义触发事件前的时间（触发提前量）。
TriggerSamples	定义要存储的触发器样例数。
Loaded Recording	
LoadedRecording	当前加载的记录的名称
RecordingEnd	记录的结束时间
RecordingStart	记录的开始时间
Logging	
DetailedLogging	启用详细日志记录。
EnableLogging	启用日志记录。
KeepGraphOrder	在图形显示中保持上下文的顺序，即使这些上下文被重新设为不可见和可见。
ReduceOnZoom	缩放时减少显示深度（直接相邻的标记合并为一个条形），以提高性能。
ShowSystemTasks	同样显示系统任务。
TargetSettings/ Capture Time	在目标系统上进行记录时的时间戳。目标系统的时间便用于此目的。
RegKeys	显示哪些注册表项的设置会影响实时性。

Logging（日志记录）区域中所列的属性始终适用于所有子元素。因此，这些属性适用于项目级别的整个 Realtime Monitor 项目。如果 **Logging（日志记录）**区域中某个属性后的值为“Different Settings（不同设置）”，则表示各子节点中的值不同。通过更改项目级别的值，可设置所有子元素的值。

11.6.2 上下文节点

可在 TwinCAT 3 Realtime Monitor 的上下文节点上进行以下设置。这些设置因实时上下文（此处的上下文对应计算机核）和应用程序上下文而异。

实时上下文：

属性	含义
通用	
Comment	可选注释
信息	
BaseTime	核的基准时间
DefaultCore	表示该核是否为默认核。
Id	显示核的 ID。
Name	显示核的名称。
RT_Percentage	显示设定的最大实时负载。
类型	显示核的类型（WindowsRT/隔离核）。
Logging	
DetailedLogging	启用详细日志记录。
EnableLogging	启用/禁用日志记录。
ShowSingleMarker	显示单个标记。

EventLogger

属性	含义
注释	
Comment	Comment
Information	
Id	
Name	EventLogger
Logging	
EnableLogging	启用 EventLogger 事件日志记录。

应用程序上下文：

属性	含义
通用	
Comment	Comment
Information	
上下文 ID	在标记处传输的上下文 ID
Name	上下文的名称
Logging	
ShowSingleMarker	显示单个标记。
ShowTaskReference	显示任务引用。

如果使用功能块 FB_RTMon_LogMark [▶ 48]，PLC 运行时模块的端口号会自动设置为上下文 ID。

11.6.3 标记组元素

在 TwinCAT 3 Realtime Monitor 的标记组/进程节点上可以进行以下设置。这些设置因实时任务和应用程序进程/标记而异。

实时任务：

属性	含义
Common	
Comment	Comment
Information	
ADSPort	任务的 ADS 端口
CpuId	执行任务的 CPU ID。
CycleTime	任务周期时间
EventCount	执行的数量（在记录时间内）
Name	任务名称
OID	任务对象 ID
ParsedMasks	找到的标记数
优先级	设置优先级
Layout	
ADSCommandColor	处理 ADS 命令区域的颜色（默认：珊瑚色）
CycleUdateColor	任务周期更新的颜色（默认：绿色）
InputUpdateColor	任务输入更新的颜色（默认：黄色）
MarkColor	标记颜色（默认：白色）
OutputUpdateColor	任务输出更新的颜色（默认：红色）
PostCyclicUpdateColor	任务发布周期更新的颜色（默认：深绿色）
PreInputUpdateColor	任务预输入更新的颜色（默认：金色）
TaskIntervallColor	任务间隔标记的颜色（默认：浅蓝色）
Logging	
DetailedLogging	启用详细日志记录。
ShowSingleMarker	启用显示单个标记。

用户进程：

属性	含义
Common	
Comment	Comment
Information	
GroupId	标记组/进程的 ID
Name	待显示进程的名称
ParsedMarks	找到的标记数
Layout	
EventIntervallColor	进程间隔/主动执行的颜色（默认：蓝色）
MarkColor	标记的颜色（默认：白色）
Logging	
ShowSingleMarker	启用显示单个标记。
ShowTaskReference	启用任务引用的显示。（用虚线将进程标记分配至实时任务）

11.7 光标窗口

所有创建的光标都会显示在光标窗口中。

	Time	Cursor 1	Cursor 2	Cursor 3
Cursor 1	12:03:33.3630374	0	122,6 us	67,2 us
Cursor 2	12:03:33.3631600	-122,6 us	0	-55,4 us
Cursor 3	12:03:33.3631046	-67,2 us	55,4 us	0

双击光标会使图表中的显示跳转至光标所在的确切位置。光标位于显示区域的中心。

上下文菜单项“**Remove Cursor（移除光标）**”可用于删除选定的光标。

关于光标的使用详见光标使用 [▶ 23]部分。

11.8 事件窗口

事件窗口显示了活动光标此时发生的所有事件。在下图中，光标 1 发生了以下事件：

- NC-SAF 任务终止。
- NC-SAF 任务完成。
- 调度程序启动 PlcTask。
- PlcTask 开始处理运行时模块 Untitled1。

12 PLC API

12.1 功能块

12.1.1 FB_RTMon_LogMark

FB_RTMon_LogMark 是一个扩展功能块，可用于设置“简单”（时间）标记。

```
VAR_INPUT
    nContextId        : UINT := TwinCAT_SystemInfoVarList._AppInfo.AdsPort;
    bLogCallingTask   : BOOL := TRUE; // specifies whether a reference to the calling task should be
set with each mark
END_VAR
```

nContextId: 用于设置此标记的上下文 ID。初始值为 *TwinCAT_SystemInfoVarList._AppInfo.AdsPort*。如不更改，ADS 端口号将自动用作上下文 ID。

bLogCallingTask: 布尔参数，用于定义执行的任务是否也保存为标记。初始值为 TRUE（真）。

```
VAR_OUTPUT
    bError            : BOOL;         // TRUE if an error occurred
    hrErrorCode       : HRESULT;     // outputs the error code which occurred
END_VAR
```

bError: 布尔值，用于指示功能块是否处于错误状态。

hrErrorCode: HResult-错误代码。如果输出 *bError* 为 true（真），则可在此输出端读取错误代码。

描述:

FB_RTMon_LogMark 扩展了通用功能块FB_RTMon_LogMarkBase [▶ 51]。调用任务的 ADS 端口会自动初始化上下文 ID 信息，因此在调用过程中无需再考虑该信息。方法调用中也包含了各种标记选项（请参见TcMark 选项 [▶ 53]），因此功能块无需事先创建标记。用户只需传输标记 ID（标记组）。它用于标识要显示的进程。

也可选择使用事件 ID，这样用户便可以传输用户记录（如状态机的状态、错误消息.....）。

12.1.1.1 日志间隔开始


```
// Starts logging interval
METHOD LogIntervalStart : HRESULT
VAR_INPUT
    nGroupId    : UINT; // Defines the group to which the interval belongs
    nEventId    : UINT; // Set to distinguish different events inside the group
END_VAR
```

描述

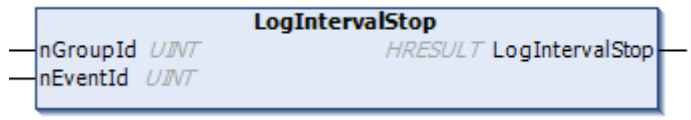
该方法会为已传输的标记 ID 创建一个带间隔开始的标记。

参数：

nGroupId：要写入标记的标记 ID（标记组）。

nEventId：可选事件 ID。

12.1.1.2 日志间隔停止




```
METHOD LogIntervalStop : HRESULT
VAR_INPUT
    nGroupId    : UINT; // Defines the group to which the interval belongs
    nEventId    : UINT; // Set to distinguish different events inside the group
END_VAR
```

描述

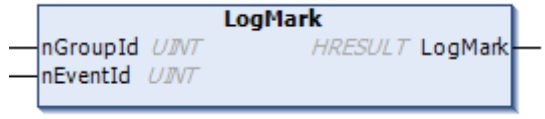
该方法会为已传输的标记 ID 创建一个带间隔停止的标记。

参数：

nGroupId：要写入标记的标记 ID（标记组）。

nEventId：可选事件 ID。

12.1.1.3 日志标记




```
// Logs a mark without start/stop
METHOD LogMark : HRESULT
VAR_INPUT
    nGroupId    : UINT; // Defines the group to which the mark belongs
    nEventId    : UINT; // Set to distinguish different events inside the group
END_VAR
```

描述

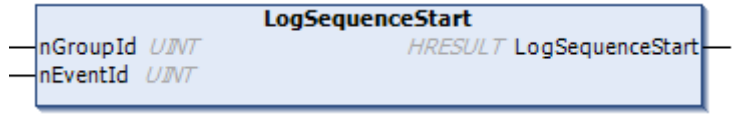
该方法会为所传输的标记 ID 创建一个标记。可以选择使用事件 ID 来区分不同的用户事件，或在 TwinCAT 3 实时监控器显示屏中显示附加数据（以 UINT 的格式）。

参数：

nGroupId：要写入标记的标记 ID（标记组）。

nEventId：可选事件 ID。

12.1.1.4 日志序列开始




```
// Starts logging sequence
METHOD LogSequenceStart : HRESULT
VAR_INPUT
```

```
    nGroupId    : UINT; // Defines the group to which the sequence belongs
    nEventId    : UINT; // Set to distinguish different events inside the group
END_VAR
```

描述

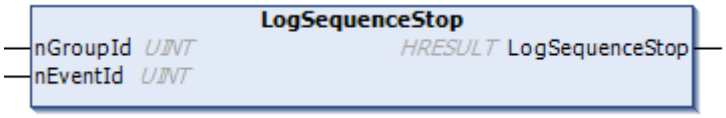
该方法会为已传输的标记 ID 创建一个带序列开始的标记。

参数：

nGroupId：要写入标记的标记 ID（标记组）。

nEventId：可选事件 ID。

12.1.1.5 日志序列停止




```
// Stops logging sequence
METHOD LogSequenceStop : HRESULT
VAR_INPUT
    nGroupId    : UINT; // Defines the group to which the sequence belongs
    nEventId    : UINT; // Set to distinguish different events inside the group
END_VAR
```

描述

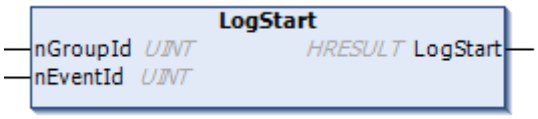
该方法会为已传输的标记 ID 创建一个带序列停止的标记。

参数：

nGroupId：要写入标记的标记 ID（标记组）。

nEventId：可选事件 ID。

12.1.1.6 日志开始




```
// Starts logging sequence and interval
METHOD LogStart : HRESULT
VAR_INPUT
    nGroupId    : UINT; // Defines the group to which the sequence and intervall belong
    nEventId    : UINT; // Set to distinguish different events inside the group
END_VAR
```

描述

该方法会为已传输的标记 ID 创建一个带序列和间隔开始的标记。

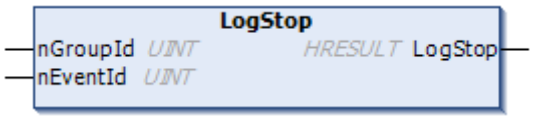
因此，该标记表示进程激活/启动的当下时间。

参数：

nGroupId：要写入标记的标记 ID（标记组）。

nEventId：可选事件 ID。

12.1.1.7 日志停止




```
// Stops logging sequence and interval
METHOD LogStop : HRESULT
VAR_INPUT
    nGroupId    : UINT; // Defines the group to which the sequence and intervall belong
    nEventId    : UINT; // Set to distinguish different events inside the group
END_VAR
```

描述

该方法会为已传输的标记 ID 创建一个带序列和间隔停止的标记。

因此，该标记表示进程终止的直接时间点。

参数：

nGroupId：要写入标记的标记 ID（标记组）。

nEventId：可选事件 ID。

12.1.2 FB_RTMon_LogMarkBase


```
FUNCTION_BLOCK FB_RTMon_LogMarkBase
VAR_INPUT
END_VAR
VAR_OUTPUT
    bError        : BOOL;        // TRUE if an error occurred
    hrErrorCode   : HRESULT;     // outputs the error code which occurred
END_VAR
```

描述：

FB_RTMon_LogMarkBase 是一个基本功能块，可用于设置（时间）标记。

与功能块 FB_RTMon_LogMark [▶ 48] 不同，必须在这里传输上下文 ID。这样就可以对要显示的进程进行分组（例如，按进程类型或功能单元进行分组）。

也可以选择使用事件 ID，这样用户便可以传输用户记录（如状态机的状态、错误消息）。

12.1.2.1 日志标记


```
METHOD LogMark : HRESULT
VAR_INPUT
    nContextId : UINT;     // defines the context
    nGroupId   : UINT;     // defines the group inside the context
    nEventId   : UINT;     // defines the specific event inside the group
    nMarkCtrl  : UDINT;    // mask for mark options (listed in TcMarkOption)
END_VAR
```

描述

该方法会为已传输的标记组 ID 创建一个标记。使用参数 nMarkCtrl（参见 TcMark 选项 [▶ 53]）传输标记类型。

可以选择使用事件 ID 来区分不同的用户事件，或在 TwinCAT 3 实时监控器显示屏中显示附加数据（以 UINT 的格式）。

参数：

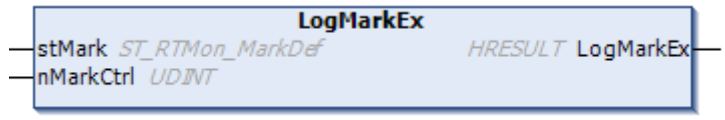
nContextId： 定义标记在 TwinCAT 3 实时监控器中分组的上下文 ID。

nGroupId： 要写入标记的标记 ID（标记组）。

nEventId： 可选事件 ID。

nMarkCtrl： 定义标记类型。

12.1.2.2 LogMarkEx




```
METHOD LogMarkEx : HRESULT
VAR_INPUT
    stMark       : ST_RTMon_MarkDef;
    nMarkCtrl    : UDINT; // mask for mark options (listed in TcMarkOption)
END_VAR
```

描述

该方法会创建一个标记。使用数据类型 ST_RTMon_MarkDef [▶ 52] 对标记进行定义。使用参数 nMarkCtrl（参见 TcMark 选项 [▶ 53]）传输标记类型。

参数：

stMark： 要写入的已定义标记的传输参数。

nMarkCtrl： 定义标记类型。

12.2 数据类型

12.2.1 ST_RTMon_MarkDef

表示标记的数据类型。

```
// defines a mark
TYPE ST_RTMon_MarkDef:
STRUCT
    nContextId : UINT;      // defines the context
    nGroupId : UINT;       // defines the group inside the context
    nEventId : UINT;       // defines the specific event inside the group
END_STRUCT
END_TYPE
```

描述

使用该数据类型可以定义通用标记（无类型）。然后，除标记类型外，会采用功能块 FB_RTMon_LogMarkBase [▶ 51] 的方法 LogMarkEx [▶ 52] 对其进行传输。

nContextId： 使用 ContextId 可以对标记组（即要显示的进程）进行分组（如按进程类型或功能单元进行分组）。

nGroupId： 定义要显示的进程/进程事件。

nEventId：可选用户记录。例如，它可以用于在 TwinCAT 3 实时监控器中显示状态机的状态或错误代码。

在 TwinCAT 3 实时监控器中，ContextId 和 GroupId 都可以命名。可以使用 用户上下文 [▶ 37] 或 用户上下文 [▶ 37] 功能导出或导入这些内容，以便对它们进行进一步记录。

12.3 全局常量

12.3.1 TcMark 选项

此全局变量列表中的常量定义了可能的标记类型（参见 实时监控器中的显示 [▶ 13]）。

```
VAR_GLOBAL CONSTANT
    Start             : UDINT := 16#E0000000;
    Stop              : UDINT := 16#C0000000;
    SequenceStart     : UDINT := 16#A0000000;
    SequenceStop      : UDINT := 16#80000000;
    IntervalStart     : UDINT := 16#60000000;
    IntervalStop      : UDINT := 16#40000000;
    RefToCaller       : UDINT := 16#08000000; // reference to caller
END_VAR
```

除标记类型外，还定义了 RefToCaller 选项，可在 TwinCAT 3 实时监控器中显示任务参照。如果激活该选项，则必须 将其 与所需的标记类型进行 OR 运算。

样例：

```
fbLogMark.LogMarkEx(markCounter, TcMarkOption.Start OR TcMarkOption.RefToCaller);
```

该样例显示了标记"markCounter"的设置，标记类型为"Start"，选项为"RefToCaller"。

如果要在 TwinCAT 3 实时监控器中显示任务参照，必须激活 **Show Task Reference（显示任务参照）**选项（参见 标记组元素 [▶ 45]）。

13 C++ API

13.1 数据类型

13.1.1 TcMark16

表示标记的数据类型。

```
typedef struct {
USHORT ContextId;
USHORT GroupId;
USHORT EventId;
} TcMark16;
```

描述：

使用该数据类型可以定义通用标记（无类型）。

ContextId： 使用 ContextId 可以对标记组（即要显示的进程）进行分组（如按进程类型或功能单元进行分组）。

GroupId： 定义要显示的进程/进程事件。

事件 ID： 可选用户记录。例如，它可以用于在 TwinCAT 3 实时监控器中显示状态机的状态或错误代码。

在 TwinCAT 3 实时监控器中，ContextId 和 GroupId 都可以命名。可以使用 用户上下文 [▶ 37] 或 用户上下文 [▶ 37] 功能导出或导入这些内容，以便对它们进行进一步记录。

13.2 分类

13.2.1 CTcLogMark

```
CTcLogMark(USHORT nContextId, ITComObjectServer* ipSrv = NULL);
```

描述：

CTcLogMark 类是一个 C++ 类，可通过 C++ 应用程序代码设置（时间）标记，以便可以通过 TwinCAT 3 Realtime Monitor 进行显示。

13.2.1.1 日志间隔开始

```
virtual HRESULT LogIntervalStart(USHORT GroupId, USHORT EventId);
```

描述：

该方法会为已传输的标记 ID 创建一个带间隔开始的标记。

参数：

GroupId： 要写入标记的标记 ID（标记组）。

EventId： 可选的事件 ID。

13.2.1.2 日志间隔停止

```
virtual HRESULT LogIntervalStop(USHORT GroupId, USHORT EventId);
```

描述：

该方法会针对已传输的标记 ID 创建一个带间隔停止的标记。

参数：

GroupId：要写入标记的标记 ID（标记组）。

EventId：可选的事件 ID

13.2.1.3 日志标记

```
virtual HRESULT LogMark(USHORT GroupId, USHORT EventId, ULONG CtrlId);
```

描述：

该方法会为已传输的标记组 ID 创建一个标记。标记类型由 TcLogMark.h 中的常量决定（参见 常量 [▶ 56]）。

可以选择使用事件 ID 来区分不同的用户事件，或在 TwinCAT 3 实时监控器显示屏中显示附加数据（以 USHORT 格式）。

13.2.1.4 LogMarkEx

```
virtual HRESULT LogMarkEx(TcMark16* pMark, ULONG CtrlId);
```

描述

该方法会创建一个标记。使用数据类型 TcMark16 [▶ 54] 对标记进行定义。标记类型由 TcLogMark.h 中的常量决定（参见 常量 [▶ 56]）。

13.2.1.5 日志序列开始

```
virtual HRESULT LogSequenceStart(USHORT GroupId, USHORT EventId);
```

描述：

该方法会为已传输的标记 ID 创建一个带序列开始的标记。

13.2.1.6 日志序列停止

```
virtual HRESULT LogSequenceStop(USHORT GroupId, USHORT EventId);
```

描述：

该方法会为已传输的标记 ID 创建一个带序列停止的标记。

13.2.1.7 日志开始

```
virtual HRESULT LogStart(USHORT GroupId, USHORT EventId);
```

描述：

该方法会为已传输的标记 ID 创建一个带序列和间隔开始的标记。

因此，该标记表示进程激活/启动的当下时间。

13.2.1.8 日志停止

```
virtual HRESULT LogStop(USHORT GroupId, USHORT EventId);
```

描述

该方法会为已传输的标记 ID 创建一个带序列和间隔停止的标记。

因此，该标记表示进程终止的直接时间点。

13.2.1.9 设置上下文 ID

```
virtual void SetContextId(USHORT nContextId);
```

描述：

该方法可设置使用的上下文 ID。

13.2.1.10 InitLogMark

```
virtual HRESULT InitLogMark(ITComObjectServer* ipSrv);
```

描述：

初始化 CTcLog 标记类实例。

参数：

ipSrv：指向 TcObjectServer 的接口指针。

13.2.1.11 发布日志标记

```
virtual HRESULT ReleaseLogMark();
```

描述：

发布 CTcLogMark 类实例的资源。

13.3 常量

这些在 TcLogMark.h 中定义的常量定义了可能的标记类型 [▶ 13]。

```
#define TCMARK_START 0xE0000000
#define TCMARK_STOP 0xC0000000
#define TCMARK_SEQ_START 0xA0000000
#define TCMARK_SEQ_STOP 0x80000000
#define TCMARK_IVAL_START 0x60000000
#define TCMARK_IVAL_STOP 0x40000000
#define TCMARK_REF_CALLER 0x08000000
```

14 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

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