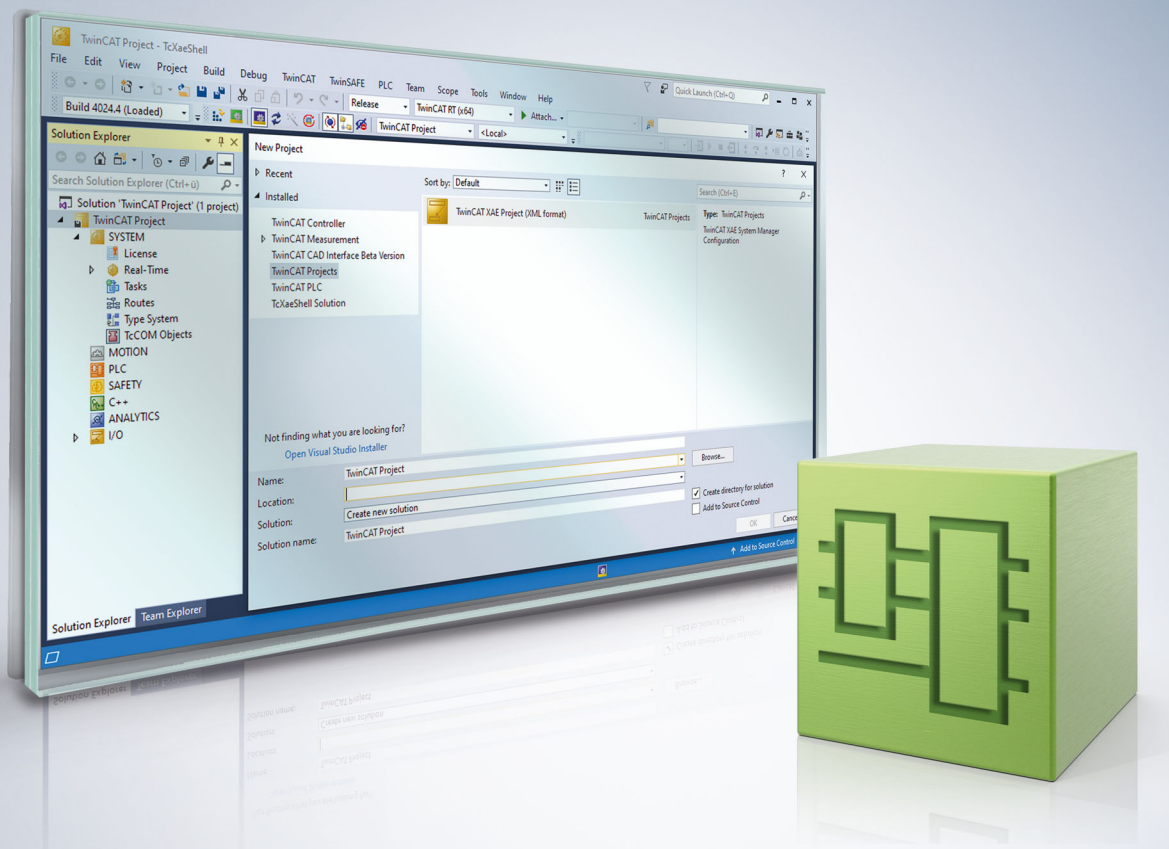
BECKHOFF New Automation Technology

手册 | ZH

TE1000

TwinCAT 3 | PLC Library: Tc2_System



2026-01-14 | 版本： 1.17.0

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

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

© Beckhoff Automation GmbH。
未经明确授权，不得复制、分发、使用和传播本文档内容。
违者将被追究赔偿责任。Beckhoff Automation GmbH 保留所有发明、实用新型和外观设计专利权。

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 概述

并非所有 PLC 应用中常用的功能块和函数都已在 IEC61131-3 中实现标准化。Tc2_System 库包含适用于 TwinCAT 系统的函数和功能块，它们不属于 IEC61131-3 的标准范围，为制造商的专用内容。

通用功能块

名称	描述
DRAND [▶ 13]	随机数生成器
FB_IecCriticalSection [▶ 13]	实现临界区互斥
FB_SetLedColor_BAPI [▶ 16]	将用户 LED 切换到支持 BIOS API 的 PC 和嵌入式控制器
GETCURTASKINDEX [▶ 18]	确定当前任务的索引

ADS 功能块

名称	描述
ADSREAD [▶ 35]	通过 ADS 读取数据
ADSREADEX [▶ 37]	通过 ADS 读取数据并返回已读取数据的字节数
ADSWRITE [▶ 38]	通过 ADS 写入数据
ADSRDWRT [▶ 39]	通过 ADS 读取并写入数据
ADSRDWRTEX [▶ 41]	通过 ADS 写入数据并返回已读取的数据字节数
ADSRDSTATE [▶ 20]	通过 ADS 读取设备状态
ADSWRTCTL [▶ 21]	通过 ADS 向设备写入控制字
ADSRDDEVINFO [▶ 23]	通过 ADS 读取设备信息

ADS 扩展功能块

名称	描述
ADSREADIND [▶ 25]	ADSREAD 指示
ADSWRITEIND [▶ 28]	ADSWRITE 指示
ADSRDWRTIND [▶ 30]	ADSRDWRT 指示
ADSREADRES [▶ 32]	ADSREAD 响应
ADSWRITERES [▶ 33]	ADSWRITE 响应
ADSRDWRTRES [▶ 34]	ADSRDWRT 响应

数据访问功能块

功能块可用于在 PC 本地处理来自 PLC 的文件。TwinCAT 系统通过其 AMS 作为网络识别地址。得益于该机制，可以在网络中的其他 TwinCAT 系统上存储或编辑文件。访问文件由 3 个步骤组成：

1. 打开文件。
2. 读或写已打开的文件。
3. 关闭文件。

打开文件的目的是在仅有文件名称的前提下在外部文件和运行的程序之间建立临时连接。关闭文件的目的是表示处理结束并将其置于确定的输出状态，以供其他程序处理。

名称	描述
FB_EOF [▶ 43]	检查文件结尾
FB_FileOpen [▶ 44]	打开文件
FB_FileClose [▶ 47]	关闭文件
FB_FileGets [▶ 49]	从文件中获取字符串
FB_FilePuts [▶ 51]	将字符串放入文件
FB_FileRead [▶ 52]	从文件中读取
FB_FileWrite [▶ 53]	写入文件
FB_FileSeek [▶ 55]	移动文件指针
FB_FileTell [▶ 56]	获取文件指针位置
FB_FileDelete [▶ 58]	删除文件
FB_FileRename [▶ 59]	重命名文件
FB_CreateDir [▶ 60]	创建新目录
FB_RemoveDir [▶ 62]	删除目录

EventLogger 功能块

TwinCAT EventLogger 的任务是管理在 TwinCAT 系统中出现的所有消息（事件）；用于消息转发，并在必要时将这些消息写入到 TwinCAT 日志文件中。

名称	描述
ADSLOGEVENT [▶ 63]	发送并确认 TwinCAT EventLogger 消息
ADSCLEAREVENTS [▶ 66]	发送并确认 TwinCAT EventLogger 消息
FB_SimpleAdsLogEvent [▶ 67]	发送并确认 TwinCAT EventLogger 消息

TwinCAT EventLogger 与 TwinCAT 3 EventLogger

TwinCAT EventLogger 已被其后续版本 TwinCAT 3 EventLogger 取代。TwinCAT 3 最高支持 3.1.4024 版本的旧版 TwinCAT EventLogger 。较新版本的 TwinCAT（>= 3.1.4026.0）仅支持较新的 TwinCAT 3 EventLogger。相关 PLC 功能块可在 PLC 库 Tc3_EventLogger 中找到。

IEC 步骤/SFC 标志位功能块

如果在 SFC 程序/项目中使用 IEC 步骤或 SFC 标志位，则需要以下函数/功能块。

名称	描述
AnalyzeExpression [▶ 69]	如果使用 SFC 标志位，则此项是必需的
AnalyzeEspressionTable [▶ 71]	如果使用 SFC 标志位，则此项是必需的
AnalyzeExpressionCombined [▶ 74]	如果使用 SFC 标志位，则此项是必需的
AppendErrorString [▶ 74]	如果使用 SFC 标志位格式化错误描述字符串，则此项是必需的
SFCActionControl [▶ 74]	启用 IEC 步骤

看门狗功能块

名称	描述
FB_PcWatchdog [▶ 75]	激活或停用 PC 看门狗 仅适用于配备以下主板的 IPC：IP-4GVI63、CB1050、CB2050、CB3050、CB1051、CB2051、CB3051
FB_PcWatchdog_BAPI [▶ 76]	激活或停用 PC 看门狗 仅适用于配备以下主板的 IPC：CBxx63，且 BIOS 版本 >=0.44

时间功能块

名称	描述
GETCPUCOUNTER [▶ 78]	读取 CPU 周期计数器
GETCPUACCOUNT [▶ 77]	读取 PLC 任务周期计数器

通用函数

名称	描述
F_CheckMemoryArea [▶ 79]	返回有关所请求的指定大小的变量所在内存区域的信息
F_CmpLibVersion [▶ 79]	将现有库与所需版本进行比较
F_CreateIPv4Addr [▶ 80]	将单个 IPv4 地址字节转换为字符串
F_ScanIPv4AddrIds [▶ 81]	将一个 IPv4 地址字符串转换为单个地址字节
F_GetMappingPartner [▶ 83]	返回映射对应方的对象 ID
F_GetMappingStatus [▶ 83]	返回 PLC 变量的当前映射状态
F_GetStructMemberAlignment [▶ 83]	读取有关所用内存对齐方式的信息
F_SplitPathName [▶ 88]	将路径名拆分成四个独立的组成部分
SETBIT32 [▶ 89]	设置 DWORD 中的一个位
CSETBIT32 [▶ 90]	设置/重置 DWORD 中的一个位
GETBIT32 [▶ 91]	读取 DWORD 中的一个位
CLEARBIT32 [▶ 91]	清除 DWORD 中的一个位
GETCURTASTINDEXEX [▶ 92]	确定任务索引
LPTSIGNAL [▶ 93]	在一个并行端口引脚上输出信号
TestAndSet [▶ 93]	设置并检查一个标志位，且该过程不可被中断

ADS 函数

以下函数借助 ADS 接口，可通过 PLC 调用实现 Windows-NT 操作系统的部分功能（例如消息框输出）。

名称	描述
ADSLOGDINT [▶ 95]	将 DINT 变量记录到 NT Eventlog 和/或 Messagebox 中
ADSLOGLREAL [▶ 96]	将 (L)REAL 变量记录到 NT Eventlog 和/或 Messagebox 中
ADSLOGSTR [▶ 97]	将 STRING 变量记录到 NT Eventlog 和/或 Messagebox 中
F_CreateAmsNetId [▶ 99]	创建 AmsNetId 字符串
F_ScanAmsNetIds [▶ 100]	将 AmsNetId 字符串转换为地址字节数组

字符函数

名称	描述
F_ToASC [▶ 101]	将字符串字符转换为 ASCII 码
F_ToCHR [▶ 100]	将 ASCII 码转换为字符串字符

I/O 端口访问

名称	描述
F_IOPortRead [▶ 101]	从 I/O 端口读取
F_IOPortWrite [▶ 102]	写入 I/O 端口

内存函数

可直接访问 PLC Runtime 系统内存区域的函数数量。

注意

系统崩溃或访问到禁止访问的内存区域

由于允许这些函数直接访问物理内存，因此在使用时需要特别小心！不正确的参数值可能会导致系统崩溃或访问到禁止访问的内存区域。

名称	描述
MEMCMP [▶ 104]	比较两个内存区域中的变量值
MEMCPY [▶ 105]	将变量值从一个内存区域复制到另一个内存区域
MEMMOVE [▶ 106]	复制重叠内存区域中的值
MEMSET [▶ 107]	将内存区域中的变量设置为特定值

时间函数

名称	描述
F_GetSystemTime [▶ 108]	读取操作系统时间戳
F_GetTaskTime [▶ 108]	读取任务的目标开始时间

3 功能块

3.1 通用功能块

3.1.1 DRAND

该功能块允许生成 LREAL 类型的（伪）随机数。

输入

```
VAR_INPUT
    Seed : INT;
END_VAR
```

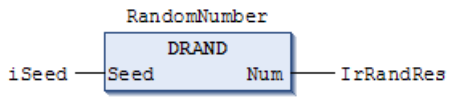
名称	类型	描述
Seed	INT	用于指定随机数序列的初始值。

输出

```
VAR_OUTPUT
    Num : LREAL;
END_VAR
```

名称	类型	描述
Num	LREAL	该输出可返回一个伪随机数，范围为 0.0 ... 1.0，双精度。此处的生成器可创建一个数字序列，每个周期有 1075 个随机值。

FBD 中的功能块示例：



在此示例中，可生成并返回 LREAL 值 0.643412。输入参数 Seed 会影响序列的初始值。例如，如果需要在不同的会话中使用确定可复现的随机数序列，则必须使用相同的 Seed 值。

前提条件

开发环境	目标平台	待集成的 PLC 库（类别组）
TwinCAT v3.1.0	PC 或 CX（x86、x64、Arm®）	Tc2_System（系统）

3.1.2 FB_IecCriticalSection

该功能块可用于使临界区相互排斥。临界区的特点是修改会影响 1 个或多个变量（通常情况下），而这些变量在修改期间具有不一致的状态。因此，每次必须仅由 1 个任务执行此类修改。为此，功能块可提供方法 Enter() 和 Leave()。成功调用 Enter() 可访问临界区，然后该区被视为已被占用。修改完成后，必须通过 Leave() 退出临界区。

由于任务停止而导致周期超时

如果其他任务尝试通过 Enter() 调用来访问已占用的临界区，则会被 TwinCAT 调度程序阻塞。在再次启用该部分之前，任务阻塞会一直存在！一旦启用，将会继续处理程序代码，并进入临界区。

- 确保临界区保持简短，以避免等待任务的周期超限。如果多个任务正在等待进入临界区，则应根据其优先级授予访问权限。

如果任务因尝试进入已占用的临界区而被 TwinCAT 调度程序阻塞，则可在没有“忙等待”的情况下实现这一点。因此，低优先级任务可以在这段时间内使用 CPU 容量。

Windows CE

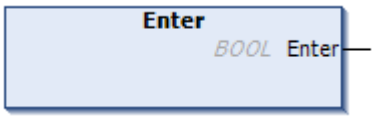
对于 TwinCAT v3.1.4022.29 及以上版本，在 Windows CE 操作系统下支持该功能。（在旧版 TwinCAT 中，这些方法会返回 FALSE。）

替代方案

通过函数 TestAndSet() [▶ 93] 也可以实现临界区。该函数可用于选择和检查临界区的内容。但是，该功能不具有阻塞效果，可能会出现在 1 个周期内无法处理该部分的情况。

通常情况下，应该保持尽可能少的临界区数量和长度。

Enter() 方法



该方法标志着临界区的开始。

可能的返回值：

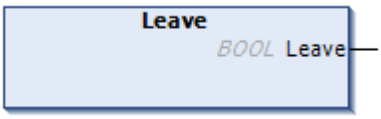
TRUE：

- 可以进入临界区。

FALSE：

- 不可以进入临界区。
- 运行时尚不支持该功能块。
- 临界区被另一个 PLC 任务占用。该任务在断点处停止。返回值 FALSE 可以避免任务的永久性阻塞，并确保 I/O 的更新。

Leave() 方法



该方法标志着临界区的结束。务必始终在临界区完成时调用它。

可能的返回值：

TRUE：

- 已成功退出该部分。

FALSE：

- 运行时不支持该功能块。
- 临界区没有被 Enter 占用。

功能块的应用示例：

功能块 FB_IecCriticalSection 可实现对要保护的共享文件的访问。在全局范围内创建功能块实例和要保护的数据。

```
VAR_GLOBAL
    fbCrititcalSection : FB_IecCriticalSection;
END_VAR

IF fbCrititcalSection.Enter() THEN
    (* start of critical section *)

    (* end of critical section *)
    fbCrititcalSection.Leave();
END_IF
```

要求

开发环境	目标平台	要集成的 PLC 库（类别组）
TwinCAT v3.1.4020	PC 或 CX（x86、x64） WES、WES7、Win7、Win10	Tc2_System（系统）
TwinCAT v3.1.4022.29	PC 或 CX（x86、ARM） WinCE	Tc2_System（系统）

3.1.3 FB_ReadTaskExceedCounter

该功能块用于读取超时计数器。每当所选任务超过设定的任务时间，系统将递增超时计数器。这意味着无法在周期内保持实时性。

导致超实时运行的原因可能有很多，但通常是由于 PLC Runtime 以及该 Runtime 内的应用程序造成的。例如，诸如 FOR、WHILE、REPEAT 等编程循环结构，因为这些循环结构总是在单个周期内处理。

输入

```
VAR_INPUT
    bExecute         : BOOL;
    nTaskAdsPort     : UINT;
END_VAR
```

名称	类型	描述
bExecute	BOOL	上升沿激活功能块。
nTaskAdsPort	UINT	所选任务的 ADS 端口 可能的分配示例： TwinCAT_SystemInfoVarList._TaskInfo[GETCURTASKINDEXEX()].AdsPort

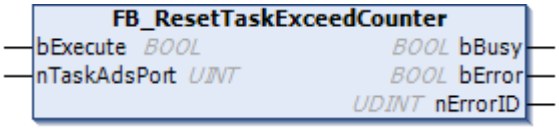
输出

```
VAR_OUTPUT
    bBusy          : BOOL;
    bError         : BOOL;
    nErrorID       : UDINT;
    nExceedCounter : UDINT;
END_VAR
```

名称	类型	描述
bBusy	BOOL	功能块处于活动状态并开始工作
bError	BOOL	在功能块中检测到错误
nErrorID	UDINT	ADS 错误代码
nExceedCounter	UDINT	读取超时计数器的值

开发环境	目标平台	待集成的 PLC 库
TwinCAT v3.1.4024.22	PC 或 CX（x86、x64、ARM）	Tc2_System（系统） >= 3.4.25.0

3.1.4 FB_ResetTaskExceedCounter



该功能块可重置超时计数器。每当所选任务超过设定的任务时间，超时计数器将会递增。这意味着无法在周期内保持实时性。

导致超实时运行的原因可能有很多，但通常是由于 PLC Runtime 以及该 Runtime 内的应用程序造成的。例如，诸如 FOR、WHILE、REPEAT 等编程循环结构，因为这些循环结构总是在单个周期内处理。

输入

```
VAR_INPUT
    bExecute          : BOOL;
    nTaskAdsPort      : UINT;
END_VAR
```

名称	类型	描述
bExecute	BOOL	上升沿激活功能块。
nTaskAdsPort	UINT	所选任务的 ADS 端口 可能的分配示例： TwinCAT_SystemInfoVarList._TaskInfo[GETCURTASKINDEXEX()].AdsPort

输出

```
VAR_OUTPUT
    bBusy             : BOOL;
    bError            : BOOL;
    nErrorID          : UDINT;
END_VAR
```

名称	类型	描述
bBusy	BOOL	功能块处于活动状态并开始工作
bError	BOOL	在功能块中检测到错误
nErrorID	UDINT	ADS 错误代码

开发环境	目标平台	待集成的 PLC 库
TwinCAT v3.1.4024.22	PC 或 CX（x86、x64、ARM）	Tc2_System（系统） >= 3.4.25.0

3.1.5 FB_SetLedColor_BAPI

该功能仅适用于带有 Usr-LED 且 BIOS 版本支持 BIOS-API 的 IPC 和嵌入式控制器。

功能块 FB_SetLedColor_BAPI 可用于在支持 BIOS API 的 PC 和嵌入式控制器上切换用户 LED。通过 bExecute 和 eNewColor 参数的上升沿切换 LED 颜色。LED 可被关闭（eNewColor = eUsrLED_Off）或设置为红色（eNewColor = eUsrLED_Red）、蓝色（eNewColor = eUsrLED_Blue）或绿色（eNewColor = eUsrLED_Green）。

输入

```
VAR_INPUT
    sNetID          : T_AmsNetID;
    eNewColor       : E_UsrLED_Color;
    bExecute        : BOOL;
    tTimeout        : TIME;
END_VAR
```

名称	类型	描述
sNetID	T_AmsNetID	设备的 AMS 网络 ID（空字符串或本地网络 ID）（类型T_AmsNetID [▶ 115]）
eNewColor	E_UsrLED_Color	新的 LED 颜色（类型E_UsrLED_Color [▶ 114]）
bExecute	BOOL	上升沿触发命令执行。一旦功能块不再处于活动状态，必须立即复位输入端 (bBusy=FALSE)。
tTimeout	TIME	ADS 内部通信终止前的时间

输出

```
VAR_OUTPUT
    bBusy     : BOOL;
    bError    : BOOL;
    nErrID    : UDINT;
END_VAR
```

名称	类型	描述
bBusy	BOOL	TRUE，只要该功能块处于活动状态
bError	BOOL	如果命令执行过程中发生错误，则为 TRUE
nErrID	UDINT	包含 ADS 错误代码或最后执行的命令的特定错误代码。在输入端执行命令后重置为 0。

要求

开发环境	目标平台	待集成的 PLC 库（类别组）
TwinCAT v3.1.0	PC 或 CX（x86、x64、ARM）	Tc2_System（系统）v3.4.14

3.1.6 FB_SetLedColorEx_BAPI

该功能仅适用于带有 Usr-LED 且 BIOS 版本支持 BIOS-API 的 IPC 和嵌入式控制器。

功能块 FB_SetLedColorEx_BAPI 可用于在支持 BIOS API 的 PC 和嵌入式控制器上切换用户 LED（USR、U1 或 U2）。通过 bExecute 和 eNewColor 参数的上升沿切换 LED 颜色。LED 可被关闭（eNewColor = eUsrLED_Off）或设置为红色（eNewColor = eUsrLED_Red）、蓝色（eNewColor = eUsrLED_Blue）或绿色（eNewColor = eUsrLED_Green）。

输入

```
VAR_INPUT
    sNetID          : T_AmsNetID;
    nLedID          : USINT
    eNewColor       : E_UsrLED_Color;
    bExecute        : BOOL;
    tTimeout        : TIME;
END_VAR
```

名称	类型	描述
sNetID	T_AmsNetID	设备的 AMS 网络 ID（空字符串或本地网络 ID）（类型T_AmsNetID [▶ 115]）
nLedID	USINT	用于选择用户 LED 的 ID：对于仅带有一个用户 LED 的设备，可通过 nLedID = 0（默认值为 0）选择用户 LED。对于带有多个用户 LED 的设备，可通过 nLedID = 1 选择 U1 LED，或通过 nLedID = 2 选择 U2 LED。
eNewColor	E_UsrLED_Color	新的 LED 颜色（类型E_UsrLED_Color [▶ 114]）
bExecute	BOOL	上升沿触发命令执行。一旦功能块不再处于活动状态，必须立即复位输入端 (bBusy=FALSE)。
tTimeout	TIME	ADS 内部通信终止前的时间

输出

```
VAR_OUTPUT
    bBusy    : BOOL;
    bError   : BOOL;
    nErrID   : UDINT;
END_VAR
```

名称	类型	描述
bBusy	BOOL	TRUE，只要该功能块处于活动状态
bError	BOOL	如果命令执行过程中发生错误，则为 TRUE
nErrID	UDINT	包含 ADS 错误代码或最后执行的命令的特定错误代码。在输入端执行命令后重置为 0。

要求

开发环境	目标平台	待集成的 PLC 库（类别组）
TwinCAT v3.1.0	PC 或 CX（x86、x64、ARM）	Tc2_System（系统）v3.6.1

3.1.7 GETCURTASKINDEX

过时的功能块

该功能块已过时。使用函数 GETCURTASKINDEXEX() [▶ 92] 代替。

功能块 GETCURTASKINDEX 可确定当前调用它的任务的任务索引。

要区分当前调用是发生在实时上下文中还是来自 PLC 周期任务，请参见函数 GETCURTASKINDEXEX [▶ 92] 的文档。例如，初始化期间 FB_init 方法的自动调用并非来自 PLC 周期任务。

输入

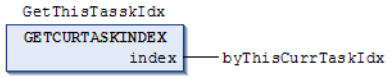
```
VAR_INPUT
(*none*)
END_VAR
```

输出

```
VAR_OUTPUT
    index : BYTE;
END_VAR
```

名称	类型	描述
index	BYTE	返回调用任务的当前任务索引（1..4）。

在 FBD 中调用功能块的示例：



前提条件

开发环境	目标平台	待集成的 PLC 库（类别组）
TwinCAT v3.1.0	PC 或 CX（x86、x64、Arm®）	Tc2_System（系统）

3.1.8 FB_CreateGUID

该功能块可生成一个新的 GUID。如果在输入端将一个 GUID 类型的数组指定为缓冲区，则通过一次调用即可获取不同的新 GUID 的列表。

输入

```
VAR_INPUT
    bExecute        : BOOL;
    sNetId          : T_AmsNetId;
    tTimeout        : TIME := DEFAULT_ADS_TIMEOUT;
    pGuidBuffer     : POINTER TO GUID;
    nGuidBufferSize : UDINT;
END_VAR
```

名称	类型	描述
bExecute	BOOL	功能块由该输入端的上升沿激活。
sNetId	T_AmsNetId	字符串，包含 ADS 命令所指向的目标设备的 AMS NetId（类型：T_AmsNetId [▶ 115]）。
tTimeout	TIME	规定执行 ADS 命令时不得超过的超时长度。
pGuidBuffer	POINTER TO GUID	表示所生成的 GUID 缓冲区的地址。在 ARRAY OF GUID 上可以指定地址。
nGuidBufferSize	UDINT	表示指定缓冲区的字节大小。

输出

```
VAR_OUTPUT
    bBusy    : BOOL;
    bError   : BOOL;
    nErrorId : UDINT;
END_VAR
```

名称	类型	描述
bBusy	BOOL	激活功能块后，将该输出设置为 TRUE，并保持设置状态，直至收到反馈。只要 bBusy 为 TRUE，就无法执行新命令。
bError	BOOL	如果在命令执行过程中出现错误，则在 bBusy 输出复位后设置该输出。
nErrorId	UDINT	如果设置了 bError 输出，则返回 ADS 错误代码 [▶ 136]。

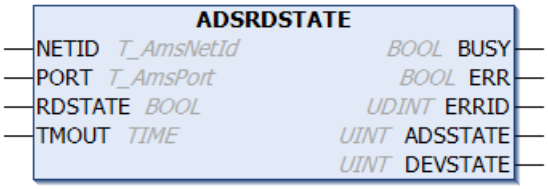
要求

开发环境	目标平台	待集成的 PLC 库（类别组）
TwinCAT v3.1.4022	PC 或 CX（x86、x64、ARM）	Tc2_System（系统） >= 3.4.18.0

3.2 ADS 功能块

3.2.1 控制 + 状态

3.2.1.1 ADSRDSTATE



该功能块请求 ADS 设备的状态。

输入

```
VAR_INPUT
    NETID    : T_AmsNetId;
    PORT     : T_AmsPort;
    RDSTATE  : BOOL;
    TMOUT    : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

名称	类型	描述
NETID	T_AmsNetId	字符串，包含 ADS 命令所指向的目标设备的 AMS NetId（类型：T_AmsNetId [▶ 115]）。
PORT	T_AmsPort	ADS 设备的端口号（类型：T_AmsPort [▶ 116]）。
RDSTATE	BOOL	ADS 命令由该输入端的上升沿触发。
TMOUT	TIME	表示函数被取消前的时间。

输出

```
VAR_OUTPUT
    BUSY       : BOOL;
    ERR        : BOOL;
    ERRID      : UDINT;
    ADSSTATE   : UINT;
    DEVSTATE   : UINT;
END_VAR
```

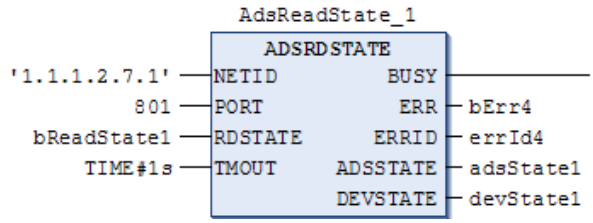
名称	类型	描述
BUSY	BOOL	在功能块执行命令之前，该输出保持为 TRUE，但最长持续时间为 Timeout 输入所提供的时间。当 BUSY = TRUE 时，输入端将不接受任何新命令。请注意，监控的不是执行服务的时间，而是接受服务的时间。
ERR	BOOL	一旦在执行命令过程中出错，该输出将切换为 TRUE。特定命令的错误代码包含在 ERRID 中。如果功能块出现超时错误，ERR 为 TRUE，ERRID 为 1861（十六进制 0x745）。在输入端执行命令后重置为 FALSE。
ERRID	UDINT	ADS 错误代码 [▶ 136]或最后执行命令的特定命令错误代码。在输入端执行命令后重置为 0。
ADSSTATE	UDINT	包含 ADS 目标设备的状态识别码。
DEVSTATE	UDINT	包含 ADS 目标设备的特定状态识别码。在这里返回的代码是 ADS 设备特有的补充信息。

ADS 目标设备的状态识别码

此处返回的代码适用于所有 ADS 服务器：

- ADSSTATE_INVALID = 0;
- ADSSTATE_IDLE = 1;
- ADSSTATE_RESET = 2;
- ADSSTATE_INIT = 3;
- ADSSTATE_START = 4;
- ADSSTATE_RUN = 5;
- ADSSTATE_STOP = 6;
- ADSSTATE_SAVECFG = 7;
- ADSSTATE_LOADCFG = 8;
- ADSSTATE_POWERFAILURE = 9;
- ADSSTATE_POWERGOOD = 10;
- ADSSTATE_ERROR = 11;
- ADSSTATE_SHUTDOWN = 12;
- ADSSTATE_SUSPEND = 13;
- ADSSTATE_RESUME = 14;
- ADSSTATE_CONFIG = 15;
- ADSSTATE_RECONFIG = 16;
- ADSSTATE_STOPPING = 17;
- ADSSTATE_INCOMPATIBLE = 18;
- ADSSTATE_EXCEPTION = 19;

在 FBD 中调用功能块的示例：

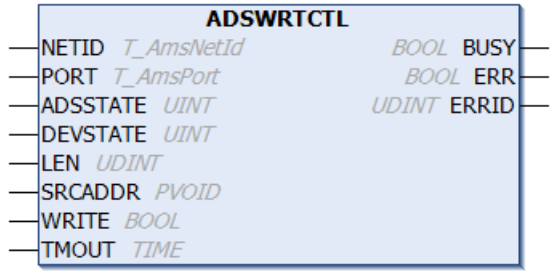


在此示例中，询问计算机上网络地址为 1.1.1.2.7.1 的 PLC Runtime 系统 1（端口号 801）的状态。回答为 adsState = 1 (IDLE)，无补充代码 devState=0。

前提条件

开发环境	目标平台	待集成的 PLC 库（类别组）
TwinCAT v3.1.0	PC 或 CX（x86、x64、Arm®）	Tc2_System（系统）

3.2.1.2 ADSWRTCTL



该功能块允许执行 ADS 控制命令，以改变 ADS 设备的状态，例如启动、停止或重置设备。

输入

```
VAR_INPUT
    NETID     : T_AmsNetId;
    PORT      : T_AmsPort;
    ADSSTATE  : UINT;
    DEVSTATE  : UINT;
    LEN       : UDINT;
    SRCADDR   : PVOID;
    WRITE     : BOOL;
    TMOUT     : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

名称	类型	描述
NETID	T_AmsNetId	字符串，包含 ADS 命令所指向的目标设备的 AMS NetId（类型：T_AmsNetId [▶ 115]）。
PORT	T_AmsPort	ADS 设备的端口号（类型：T_AmsPort [▶ 116]）。
ADSSTATE	UINT	ADS 目标设备的状态识别码。
DEVSTATE	UINT	包含 ADS 目标设备的特定状态识别码。在这里给出的代码是 ADS 设备特有的补充信息。
LEN	UDINT	要写入数据的字节数。
SRCADDR	PVOID	用于获取要写入数据的缓冲区的地址。程序员自己负责确定缓冲区的大小，以便可以从中获取“LEN”字节的大小。缓冲区可以是单个变量、数组或结构体，其地址可以通过 ADR 运算符找到。
WRITE	BOOL	ADS 命令由该输入端的上升沿触发。
TMOUT	TIME	表示函数被取消前的时间。

ADS 目标设备的状态识别码

此处显示的代码适用于所有 ADS 服务器：

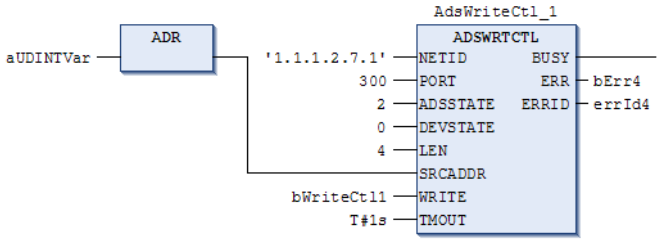
- ADSSTATE_IDLE = 1;
- ADSSTATE_RESET = 2;
- ADSSTATE_INIT = 3;
- ADSSTATE_START = 4;
- ADSSTATE_RUN= 5;
- ADSSTATE_STOP = 6;
- ADSSTATE_SAVECFG = 7;
- ADSSTATE_LOADCFG = 8;

输出

```
VAR_OUTPUT
    BUSY    : BOOL;
    ERR     : BOOL;
    ERRID   : UDINT;
END_VAR
```

名称	类型	描述
BUSY	BOOL	在功能块执行命令之前，该输出保持为 TRUE，但最长持续时间为 Timeout 输入所提供的时间。只要 BUSY = TRUE，输入端就不接受任何新命令。请注意，监控的不是执行服务的时间，而是接受服务的时间。
ERR	BOOL	一旦在执行命令过程中出错，该输出将切换为 TRUE。特定命令的错误代码包含在 ERRID 中。如果功能块出现超时错误，ERR 为 TRUE，ERRID 为 1861（十六进制 0x745）。在输入端执行命令后重置为 FALSE。
ERRID	UDINT	ADS 错误代码 [▶ 136]或最后执行命令的特定命令错误代码。在输入端执行命令后重置为 0。

在 FBD 中调用功能块的示例：



在此示例中，向 I/O 服务器（端口 300）发送了一个复位命令（ADSSTATE=2），并附带补充数据 hex.AFFE。因此，I/O 服务器执行总线复位。

前提条件

开发环境	目标平台	待集成的 PLC 库（类别组）
TwinCAT v3.1.0	PC 或 CX（x86、x64、Arm®）	Tc2_System（系统）

3.2.1.3 ADSRDDEVINFO

该功能块用于读取一般设备信息。

输入

```
VAR_INPUT
    NETID   : T_AmsNetId;
    PORT    : T_AmsPort;
    RDINFO  : BOOL;
    TMOUT   : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

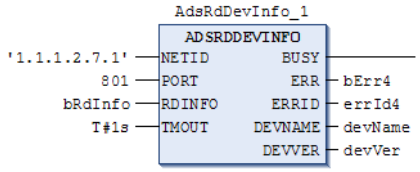
名称	类型	描述
NETID	T_AmsNetId	字符串，包含 ADS 命令所指向的目标设备的 AMS NetId（类型：T_AmsNetId [▶ 115]）。
PORT	T_AmsPort	ADS 设备的端口号（类型：T_AmsPort [▶ 116]）。
RDINFO	BOOL	ADS 命令由该输入端的上升沿触发。
TMOUT	TIME	表示函数被取消前的时间。

输出

```
VAR_OUTPUT
    BUSY    : BOOL;
    ERR     : BOOL;
    ERRID   : UDINT;
    DEVNAME : STRING(19);
    DEVVER  : UDINT;
END_VAR
```

名称	类型	描述
BUSY	BOOL	在功能块执行命令之前，该输出保持为 TRUE，但最长持续时间为 Timeout 输入所提供的时间。当 BUSY = TRUE 时，输入端将不接受任何新命令。请注意，监控的不是执行服务的时间，而是接受服务的时间。
ERR	BOOL	一旦在执行命令过程中出错，该输出将切换为 TRUE。特定命令的错误代码包含在 ERRID 中。如果功能块出现超时错误，ERR 为 TRUE，ERRID 为 1861（十六进制 0x745）。在输入端执行命令后重置为 FALSE。
ERRID	UDINT	ADS 错误代码 [▶ 136]或最后执行命令的特定命令错误代码。在输入端执行命令后重置为 0。
DEV NAME	STRING	ADS 设备的名称
DEVVER	UDINT	ADS 设备的版本号

在 FBD 中调用功能块的示例：



在此示例中，读取计算机 1.1.1.2.7.1 上第一个 PLC Runtime 系统（端口 801）的设备信息。因此，接收到名称“PLC Server”和版本号 02.00.7。

前提条件

开发环境	目标平台	待集成的 PLC 库（类别组）
TwinCAT v3.1.0	PC 或 CX（x86、x64、Arm®）	Tc2_System（系统）

3.2.2 指示 + 响应

3.2.2.1 概述

通过扩展的 ADS 功能块，在 ADS 设备和 PLC 任务之间可以创建客户端—服务器通信。例如，ADS 设备可以是 Windows 应用程序（使用 AdsDLL/AdsOcx）或其他 PLC Runtime 系统。ADS 设备与 PLC 任务之间的通信使用以下服务原语进行处理：

- 请求
- 指示
- 响应
- 确认

ADS 设备与 PLC 任务之间的通信顺序如下：ADS 设备向目标设备（PLC 任务）发送请求。该请求通过指示在目标设备中完成注册。然后，目标设备（PLC 任务）执行相应的服务。待执行的服务通过索引组/偏移参数进行加密。接着，PLC 任务向 ADS 设备发送响应。 ADS 源设备将该相应注册为确认。

ADS 设备通过端口地址和网络地址（NETID）进行寻址。（PLC 任务的端口地址 = _TaskInfo.AdsPort）

为了将请求路由到 PLC 任务，在提出请求时必须在索引组参数中输入最高有效位（例如 0x80000001）。

通过 IndexGroup 0x80000000 - 0x80FFFFFF 进行通信

每个 PLC 任务只能使用一个指示和响应功能块实例（ADSREADIND、ADSREADRES、ADSWRITEIND、ADSWRITERES、ADSRDWRTIND 和 ADSRDWRTRES 中的一个实例）。与可用的 ADS 服务相对应：READ、WRITE 以及 READ & WRITE，每项服务都有相应的指示或响应功能块。

服务	名称	描述
READ	ADSREADIND [▶ 25]	ADSREAD 指示
	ADSREADRES [▶ 32]	ADSREAD 响应
WRITE	ADSWRITEIND [▶ 28]	ADSWRITE 指示
	ADSWRITERES [▶ 33]	ADSWRITE 响应
READ & WRITE	ADSRDWRTIND [▶ 30]	ADS-READ & WRITE 指示
	ADSRDWRTRES [▶ 34]	ADS-READ & WRITE 响应

FIFO

每个 PLC 任务都有 3 个 FIFO，传入的请求（指示）会先存储在其中。也就是说包含一个 ADSREADIND FIFO、一个 ADSWRITEIND FIFO 和一个 ADSRDWRTIND FIFO。
每个 FIFO 中最多可存储 10 个指示，直至这些指示被处理完毕（直至发送响应）。例如，如果同时向一个 PLC 任务发送 12 个 ADSREAD 请求，其中 10 个请求将作为指示存储在 FIFO 中，另外两个请求将通过 ADS 错误消息 1814（0x716）进行确认（丢弃）。在这种情况下，应分析错误代码，并在必要时重复两次失败的 ADSREAD 请求。通过调用 ADSxxxxxxIND 实例，可以从相关的 FIFO 中单独获取指示。只有这样，新指示才能成功存储到 FIFO 中。

通过 IndexGroup 0x8n000000 - 0x8nFFFFFF 进行通信

要在每个 PLC 任务中实现多个客户端—服务器通信，需要使用下列指示功能块。此外，还可以指定 IndexGroup 的期望范围。

通过这种方式可以过滤请求，仅对指定区域作出响应。

共有 16 个可自由选择的范围：
0x80000000 - 0x80FFFFFF
0x81000000 - 0x81FFFFFF

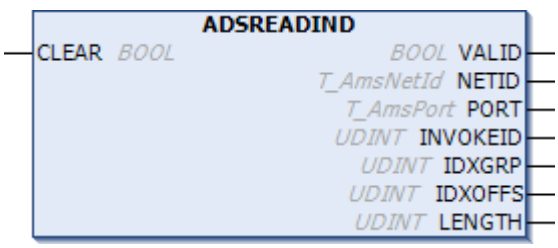
…

0x8E000000 – 0x8EFFFFFF
0x8F000000 – 0x8FFFFFFF

要在指示功能块中指定此类索引组范围，需在输入端 MINIDXGRP 指定所选范围的起始索引组值。
示例：指定 MINIDXGRP:=16#85000000 时，将过滤所有请求，仅将索引组范围在 0x85000000 - 0x85FFFFFF 之间的请求注册为指示。

服务	名称	描述
READ	ADSREADINDEX [▶ 26]	ADSREAD 指示，带有索引组指示
	ADSREADRES [▶ 32]	ADSREAD 响应
WRITE	ADSWRITEINDEX [▶ 29]	ADSWRITE 指示，带有索引组指示
	ADSWRITERES [▶ 33]	ADSWRITE 响应
READ & WRITE	ADSRDWRTINDEX [▶ 31]	ADS-READ & WRITE 指示，带有索引组指示
	ADSRDWRTRES [▶ 34]	ADS-READ & WRITE 响应

3.2.2.2 ADSREADIND



该功能块将 ADSREAD 请求注册为 PLC 任务的指示，并允许对其进行处理。通过上升沿在 VALID 输出端口报告指示的排队情况。通过 CLEAR 输入端的上升沿，指示被报告为已处理。通过 CLEAR 输入端的下降沿释放功能块，以处理更多的指示。指示处理完毕后，必须通过 ADSREADRES [▶ 32] 功能块向源设备发送响应。为此，PORT 和 NETID 参数可用于对源设备进行寻址。INVOKEID 参数通过源设备将响应分配给请求，同时也会作为参数发送回源设备。

输入

```
VAR_INPUT
    CLEAR   : BOOL;
END_VAR
```

名称	类型	描述
CLEAR	BOOL	通过该输入端的上升沿，报告指示已被处理，并重置 ADSREADIND 功能块的输出。通过下降沿释放功能块，以处理更多指示。

输出

```
VAR_OUTPUT
    VALID    : BOOL;
    NETID    : T_AmsNetId;
    PORT     : T_AmsPort;
    INVOKEID : UDINT;
    IDXGRP   : UDINT;
    IDXOFFS  : UDINT;
    LENGTH   : UDINT;
END_VAR
```

名称	类型	描述
VALID	BOOL	如果由功能块注册指示，则输出置位，并保持置位状态，直至通过 CLEAR 输入端的上升沿，报告该指示已被处理。
NETID	T_AmsNetId	包含发送 ADS 命令的源设备的 AMS NetId 的字符串（类型：T_AmsNetId [▶ 115]）。
PORT	T_AmsPort	包含发送 ADS 命令的 ADS 源设备的端口号（类型：T_AmsPort [▶ 116]）。
INVOKEID	UDINT	已发送命令的句柄。InvokeID 由源设备指定，用于识别命令。
IDXGRP	UDINT	所请求的 ADS 服务的索引组号（32 位，无符号）。
IDXOFFS	UDINT	请求的 ADS 服务的索引偏移编号（32 位，无符号）。
LENGTH	UDINT	要读取数据的字节数。

使用 ST 语言调用功能块的示例：

- 使用 AdsReadInd/AdsReadRes 功能块的示例 [▶ 126]

前提条件

开发环境	目标平台	待集成的 PLC 库（类别组）
TwinCAT v3.1.0	PC 或 CX（x86、x64、Arm®）	Tc2_System（系统）

3.2.2.3 ADSREADINDEX

该功能块将 ADSREAD 请求注册为 PLC 任务的指示，并允许对其进行处理。通过上升沿在 VALID 输出端口报告指示的排队情况。通过 CLEAR 输入端的上升沿，指示被报告为已处理。通过 CLEAR 输入端的下降沿释放功能块，以处理更多的指示。指示处理完毕后，必须通过 ADSREADRES [▶ 32] 功能块向源设备发送响应。为此，PORT 和 NETID 参数可用于对源设备进行寻址。INVOKEID 参数通过源设备将响应分配给请求，同时也会作为参数发送回源设备。

与功能块 ADSREADIND [▶ 25] 相比，此处还可以通过额外的输入指定 IndexGroup 的期望范围。

通过这种方式可以过滤请求，仅对指定区域作出响应。

共有 16 个可自由选择的范围：
0x80000000 - 0x80FFFFFF
0x81000000 - 0x81FFFFFF

…

0x8E000000 – 0x8EFFFFFF
0x8F000000 – 0x8FFFFFFF

要在指示功能块中指定此类索引组范围，需在输入端 MINIDXGRP 指定所选范围的起始索引组值。
示例：指定 MINIDXGRP:=16#85000000 时，将过滤所有请求，仅将索引组范围在 0x85000000 - 0x85FFFFFF 之间的请求注册为指示。

输入

```
VAR_INPUT
    CLEAR      : BOOL;
    MINIDXGRP  : UDINT;
END_VAR
```

名称	类型	描述
CLEAR	BOOL	通过该输入端的上升沿，报告指示已被处理，并重置 ADSREADIND 功能块的输出。通过下降沿释放功能块，以处理更多指示。
MINIDXGRP	UDINT	此输入可根据 IndexGroup 范围对请求进行过滤。指定所选范围的起始 IndexGroup 值。

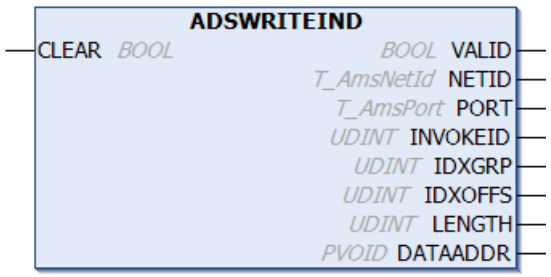
输出

```
VAR_OUTPUT
    VALID    : BOOL;
    NETID    : T_AmsNetId;
    PORT     : T_AmsPort;
    INVOKEID : UDINT;
    IDXGRP   : UDINT;
    IDXOFFS  : UDINT;
    LENGTH   : UDINT;
END_VAR
```

名称	类型	描述
VALID	BOOL	如果由功能块注册指示，则输出置位，并保持置位状态，直至通过 CLEAR 输入端的上升沿，报告该指示已被处理。
NETID	T_AmsNetId	包含发送 ADS 命令的源设备的 AMS NetId 的字符串（类型：T_AmsNetId [▶ 115]）。
PORT	T_AmsPort	包含发送 ADS 命令的 ADS 源设备的端口号（类型：T_AmsPort [▶ 116]）。
INVOKEID	UDINT	已发送命令的句柄。InvokeID 由源设备指定，用于识别命令。
IDXGRP	UDINT	所请求的 ADS 服务的索引组号（32 位，无符号）。
IDXOFFS	UDINT	请求的 ADS 服务的索引偏移编号（32 位，无符号）。
LENGTH	UDINT	要读取数据的字节数。

开发环境	目标平台	待集成的 PLC 库
TwinCAT v3.1.4024.35	PC 或 CX（x86、x64、ARM）	Tc2_System（系统） >= 3.4.26.0

3.2.2.4 ADSWRITEIND



该功能块将 ADSWRITE 请求注册为 PLC 任务的指示，并允许对其进行处理。通过上升沿在 VALID 输出端口报告指示的排队情况。通过 CLEAR 输入端的上升沿，指示被报告为已处理。通过下降沿释放功能块，以处理更多指示。指示处理完毕后，必须通过 ADSWRITERES [▶ 33] 功能块向源设备发送响应。为此，PORT 和 NETID 参数可用于对源设备进行寻址。INVOKEID 参数通过源设备将响应分配给请求，同时也会作为参数发送回源设备。

输入

```
VAR_INPUT
    CLEAR : BOOL;
END_VAR
```

名称	类型	描述
CLEAR	BOOL	通过该输入端的上升沿，指示被报告为已处理，并重置 ADSREADIND 功能块的输出（DATAADDR = 0，LENGTH = 0！）。通过下降沿释放功能块，以处理更多指示。

输出

```
VAR_OUTPUT
    VALID     : BOOL;
    NETID     : T_AmsNetId;
    PORT      : T_AmsPort;
    INVOKEID  : UDINT;
    IDXGRP    : UDINT;
    IDXOFFS   : UDINT;
    LENGTH    : UDINT;
    DATAADDR  : PVOID;
END_VAR
```

名称	类型	描述
VALID	BOOL	如果由功能块注册指示，则输出置位，并保持置位状态，直至通过 CLEAR 输入端的上升沿，报告该指示已被处理。
NETID	T_AmsNetId	包含发送 ADS 命令的源设备的 AMS NetId 的字符串（类型：T_AmsNetId [▶ 115]）。
PORT	T_AmsPort	包含发送 ADS 命令的 ADS 源设备的端口号（类型：T_AmsPort [▶ 116]）。
INVOKEID	UDINT	已发送命令的句柄。InvokeID 由源设备指定，用于识别命令。
IDXGRP	UDINT	所请求的 ADS 服务的索引组号（32 位，无符号）。
IDXOFFS	UDINT	请求的 ADS 服务的索引偏移编号（32 位，无符号）。
LENGTH	UDINT	写入数据的长度（以字节为单位）。
DATAADDR	PVOID	写入数据所在数据缓冲区的地址。

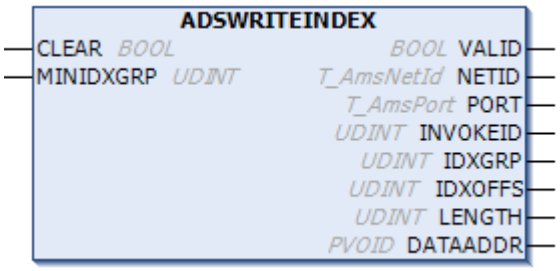
使用 ST 语言调用功能块的示例：

- 使用 AdsWriteInd/AdsWriteRes 功能块的示例 [▶ 127]

前提条件

开发环境	目标平台	待集成的 PLC 库（类别组）
TwinCAT v3.1.0	PC 或 CX（x86、x64、Arm®）	Tc2_System（系统）

3.2.2.5 ADSWRITEINDEX



该功能块将 ADSWRITE 请求注册为 PLC 任务的指示，并允许对其进行处理。通过上升沿在 VALID 输出端口报告指示的排队情况。通过 CLEAR 输入端的上升沿，指示被报告为已处理。通过下降沿释放功能块，以处理更多指示。指示处理完毕后，必须通过 ADSWRITERES [▶ 33] 功能块向源设备发送响应。为此，PORT 和 NETID 参数可用于对源设备进行寻址。INVOKEID 参数通过源设备将响应分配给请求，同时也会作为参数发送回源设备。

与 ADSWRITEIND [▶ 28] 功能块相比，还可以通过附加输入指定 IndexGroup 的期望范围。

通过这种方式可以过滤请求，仅对指定区域作出响应。

共有 16 个可自由选择的范围：
0x80000000 - 0x80FFFFFF
0x81000000 - 0x81FFFFFF

…

0x8E000000 – 0x8EFFFFFF
0x8F000000 – 0x8FFFFFFF

要在指示功能块中指定此类索引组范围，需在输入端 MINIDXGRP 指定所选范围的起始索引组值。
示例：指定 MINIDXGRP:=16#85000000 时，将过滤所有请求，仅将索引组范围在 0x85000000 - 0x85FFFFFF 之间的请求注册为指示。

输入

```
VAR_INPUT
    CLEAR      : BOOL;
    MINIDXGRP  : UDINT;
END_VAR
```

名称	类型	描述
CLEAR	BOOL	通过该输入端的上升沿，指示被报告为已处理，并重置 ADSREADIND 功能块的输出（DATAADDR = 0，LENGTH = 0！）。通过下降沿释放功能块，以处理更多指示。
MINIDXGRP	UDINT	此输入可根据 IndexGroup 范围对请求进行过滤。指定所选范围的起始 IndexGroup 值。

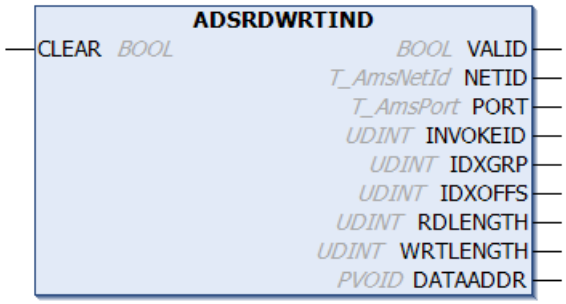
输出

```
VAR_OUTPUT
    VALID      : BOOL;
    NETID      : T_AmsNetId;
    PORT       : T_AmsPort;
    INVOKEID   : UDINT;
    IDXGRP     : UDINT;
    IDXOFFS    : UDINT;
    LENGTH     : UDINT;
    DATAADDR   : PVOID;
END_VAR
```

名称	类型	描述
VALID	BOOL	如果由功能块注册指示，则输出置位，并保持置位状态，直至通过 CLEAR 输入端的上升沿，报告该指示已被处理。
NETID	T_AmsNetId	包含发送 ADS 命令的源设备的 AMS NetId 的字符串（类型：T_AmsNetId [▶ 115]）。
PORT	T_AmsPort	包含发送 ADS 命令的 ADS 源设备的端口号（类型：T_AmsPort [▶ 116]）。
INVOKEID	UDINT	已发送命令的句柄。InvokeID 由源设备指定，用于识别命令。
IDXGRP	UDINT	所请求的 ADS 服务的索引组号（32 位，无符号）。
IDXOFFS	UDINT	请求的 ADS 服务的索引偏移编号（32 位，无符号）。
LENGTH	UDINT	写入数据的长度（以字节为单位）。
DATAADDR	PVOID	写入数据所在数据缓冲区的地址。

开发环境	目标平台	待集成的 PLC 库
TwinCAT v3.1.4024.35	PC 或 CX（x86、x64、ARM）	Tc2_System（系统） >= 3.4.26.0

3.2.2.6 ADSRDWRTIND



该功能块将 ADSRDWRT 请求注册为 PLC 任务的指示，并允许对其进行处理。通过上升沿在 VALID 输出端口报告指示的排队情况。通过 CLEAR 输入端的上升沿，指示被报告为已处理。通过下降沿释放功能块，以处理更多指示。指示处理完毕后，必须通过 ADSRDWRTRES [▶ 34] 功能块向源设备发送响应。为此，PORT 和 NETID 参数可用于对源设备进行寻址。INVOKEID 参数通过源设备将响应分配给请求，同时也会作为参数发送回源设备。

输入

```
VAR_INPUT
    CLEAR : BOOL;
END_VAR
```

名称	类型	描述
CLEAR	BOOL	通过该输入端的上升沿，指示被报告为已处理，并重置 ADSRDWRTIND 功能块的输出。通过下降沿释放功能块，以处理更多指示。

输出

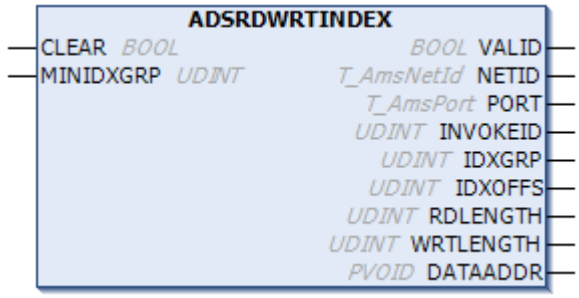
```
VAR_OUTPUT
    VALID     : BOOL;
    NETID     : T_AmsNetId;
    PORT      : T_AmsPort;
    INVOKEID  : UDINT;
    IDXGRP    : UDINT;
    IDXOFFS   : UDINT;
    RDLENGTH  : UDINT;
    WRTLENGTH : UDINT;
    DATAADDR  : PVOID;
END_VAR
```

名称	类型	描述
VALID	BOOL	如果由功能块注册指示，则输出置位，并保持置位状态，直至通过 CLEAR 输入端的上升沿，报告该指示已被处理。
NETID	T_AmsNetId	包含发送 ADS 命令的源设备的 AMS NetId 的字符串（类型：T_AmsNetId [▶ 115]）。
PORT	T_AmsPort	包含发送 ADS 命令的 ADS 源设备的端口号（类型：T_AmsPort [▶ 116]）。
INVOKEID	UDINT	已发送命令的句柄。InvokeId 由源设备指定，用于识别命令。
IDXGRP	UDINT	所请求的 ADS 服务的索引组号（32 位，无符号）。
IDXOFFS	UDINT	请求的 ADS 服务的索引偏移编号（32 位，无符号）。
RDLENGTH	UDINT	要读取的数据的长度（以字节为单位）。
WRTLENGTH	UDINT	写入数据的长度（以字节为单位）。
DATAADDR	PVOID	写入数据所在数据缓冲区的地址。

前提条件

开发环境	目标平台	待集成的 PLC 库（类别组）
TwinCAT v3.1.0	PC 或 CX（x86、x64、Arm®）	Tc2_System（系统）

3.2.2.7 ADSRDWRTINDEX



该功能块将 ADSRDWRT 请求注册为 PLC 任务的指示，并允许对其进行处理。通过上升沿在 VALID 输出端口报告指示的排队情况。通过 CLEAR 输入端的上升沿，指示被报告为已处理。通过下降沿释放功能块，以处理更多指示。指示处理完毕后，必须通过 ADSRDWRTRES [▶ 34] 功能块向源设备发送响应。为此，PORT 和 NETID 参数可用于对源设备进行寻址。INVOKEID 参数通过源设备将响应分配给请求，同时也会作为参数发送回源设备。

与 ADSRDWRTIND [▶ 30] 功能块相比，还可以通过附加输入指定 IndexGroup 的期望范围。

通过这种方式可以过滤请求，仅对指定区域作出响应。

共有 16 个可自由选择的范围：
0x80000000 - 0x80FFFFFF
0x81000000 - 0x81FFFFFF

…

0x8E000000 – 0x8EFFFFFF
0x8F000000 – 0x8FFFFFFF

要在指示功能块中指定此类索引组范围，需在输入端 MINIDXGRP 指定所选范围的起始索引组值。
示例：指定 MINIDXGRP:=16#85000000 时，将过滤所有请求，仅将索引组范围在 0x85000000 - 0x85FFFFFF 之间的请求注册为指示。

输入

```
VAR_INPUT
    CLEAR      : BOOL;
    MINIDXGRP  : UDINT;
END_VAR
```

名称	类型	描述
CLEAR	BOOL	通过该输入端的上升沿，指示被报告为已处理，并重置 ADSRDWRTIND 功能块的输出。通过下降沿释放功能块，以处理更多指示。
MINIDXGRP	UDINT	此输入可根据 IndexGroup 范围对请求进行过滤。指定所选范围的起始 IndexGroup 值。

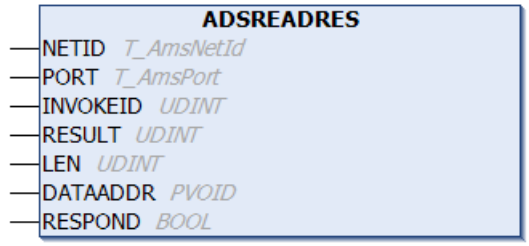
输出

```
VAR_OUTPUT
    VALID      : BOOL;
    NETID      : T_AmsNetId;
    PORT       : T_AmsPort;
    INVOKEID   : UDINT;
    IDXGRP     : UDINT;
    IDXOFFS    : UDINT;
    RDLENGTH   : UDINT;
    WRTLENGTH  : UDINT;
    DATAADDR   : PVOID;
END_VAR
```

名称	类型	描述
VALID	BOOL	如果由功能块注册指示，则输出置位，并保持置位状态，直至通过 CLEAR 输入端的上升沿，报告该指示已被处理。
NETID	T_AmsNetId	包含发送 ADS 命令的源设备的 AMS NetId 的字符串（类型：T_AmsNetId [▶ 115]）。
PORT	T_AmsPort	包含发送 ADS 命令的 ADS 源设备的端口号（类型：T_AmsPort [▶ 116]）。
INVOKEID	UDINT	已发送命令的句柄。InvokeId 由源设备指定，用于识别命令。
IDXGRP	UDINT	所请求的 ADS 服务的索引组号（32 位，无符号）。
IDXOFFS	UDINT	请求的 ADS 服务的索引偏移编号（32 位，无符号）。
RDLENGTH	UDINT	要读取的数据的长度（以字节为单位）。
WRTLENGTH	UDINT	写入数据的长度（以字节为单位）。
DATAADDR	PVOID	写入数据所在数据缓冲区的地址。

开发环境	目标平台	待集成的 PLC 库
TwinCAT v3.1.4024.35	PC 或 CX（x86、x64、ARM）	Tc2_System（系统） >= 3.4.26.0

3.2.2.8 ADSREADRES



ADSREADRES 功能块用于确认 PLC 任务的指示。通过 RESPOND 输入端的上升沿，向 ADS 源设备发送响应。通过 PORT 和 NETID 参数，对源设备进行寻址。源设备通过 INVOKEID 参数将响应分配给请求，该参数也用于 ADSREADIND [▶ 25] 功能块的输出。通过 RESULT 参数可以向 ADS 源设备返回错误代码。

输入

```
VAR_INPUT
    NETID      : T_AmsNetId;
    PORT       : T_AmsPort;
    INVOKEID   : UDINT;
    RESULT     : UDINT;
    LEN        : UDINT;
    DATAADDR   : PVOID;
    RESPOND    : BOOL;
END_VAR
```

名称	类型	描述
NETID	T_AmsNetId	包含要向其发送 ADS 命令的源设备的 AMS NetId 的字符串（类型：T_AmsNetId [▶ 115]）。
PORT	T_AmsPort	应向其发送响应的 ADS 源设备的端口号（类型：T_AmsPort [▶ 116]）。
INVOKEID	UDINT	已发送命令的句柄。InvokeID 由源设备指定，用于识别命令。
RESULT	UDINT	要发送到源设备的 ADS 错误代码 [▶ 136]或特定命令错误代码。
LEN	UDINT	要读取数据的字节数。
DATAADDR	PVOID	应读取的数据缓冲区的地址。
RESPOND	BOOL	功能块由该输入端的上升沿触发激活。

输出

```
VAR_OUTPUT
(*none*)
END_VAR
```

使用 ST 语言调用功能块的示例：

- 使用 AdsReadInd/AdsReadRes 功能块的示例 [▶ 126]

前提条件

开发环境	目标平台	待集成的 PLC 库（类别组）
TwinCAT v3.1.0	PC 或 CX（x86、x64、Arm®）	Tc2_System（系统）

3.2.2.9 ADSWRITERES

ADSWRITERES 功能块用于确认 PLC 任务的指示。通过 RESPOND 输入端的上升沿，向 ADS 源设备发送响应。通过 PORT 和 NETID 参数，对源设备进行寻址。INVOKEID 参数对源设备的请求响应进行排序，并被 ADSWRITEIND [▶ 25] 功能块的输出所采用。通过 RESULT 参数可以向 ADS 源设备返回错误代码。

输入

```
VAR_INPUT
    NETID      : T_AmsNetId;
    PORT       : T_AmsPort;
    INVOKEID   : UDINT;
    RESULT     : UDINT;
    RESPOND    : BOOL;
END_VAR
```

名称	类型	描述
NETID	T_AmsNetId	包含要向其发送 ADS 命令的源设备的 AMS NetId 的字符串（类型：T_AmsNetId [▶ 115]）。
PORT	T_AmsPort	要向其发送 ADS 命令的 ADS 源设备的端口号（类型：T_AmsPort [▶ 116]）。
INVOKEID	UDINT	已发送命令的句柄。InvokeId 由源设备指定，用于识别命令。
RESULT	UDINT	要发送到源设备的 ADS 错误代码 [▶ 136]或特定命令错误代码。
RESPOND	BOOL	功能块由该输入端的上升沿触发激活。

输出

```
VAR_OUTPUT
(*none*)
END_VAR
```

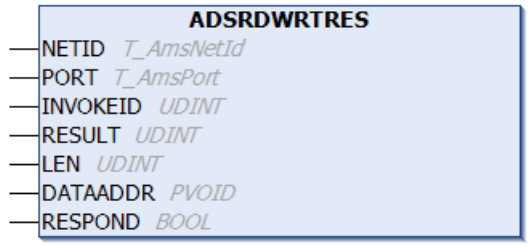
使用 ST 语言调用功能块的示例：

- 使用 AdsWriteInd/AdsWriteRes 功能块的示例 [▶ 127]

前提条件

开发环境	目标平台	待集成的 PLC 库（类别组）
TwinCAT v3.1.0	PC 或 CX（x86、x64、Arm®）	Tc2_System（系统）

3.2.2.10 ADSRDWRTRES



ADSRDWRTRES 功能块用于确认 PLC 任务的指示。通过 RESPOND 输入端的上升沿，向 ADS 源设备发送响应。通过 PORT 和 NETID 参数，对源设备进行寻址。INVOKEID 参数对源设备的请求响应进行排序，并被 ADSRDWRTIND [▶ 30] 功能块的输出所采用。通过 RESULT 参数可以向 ADS 源设备返回错误代码。

输入

```
VAR_INPUT
    NETID    : T_AmsNetId;
    PORT     : T_AmsPort;
    INVOKEID : UDINT;
    RESULT   : UDINT;
    LEN      : UDINT;
    DATAADDR : PVOID;
    RESPOND  : BOOL;
END_VAR
```

名称	类型	描述
NETID	T_AmsNetId	包含要向其发送 ADS 命令的源设备的 AMS NetId 的字符串（类型：T_AmsNetId [▶ 115]）。
PORT	T_AmsPort	要向其发送 ADS 命令的 ADS 源设备的端口号（类型：T_AmsPort [▶ 116]）。
INVOKEID	UDINT	已发送命令的句柄。InvokeID 由源设备指定，用于识别命令。
RESULT	UDINT	要发送到源设备的 ADS 错误代码 [▶ 136]或特定命令错误代码。
LEN	UDINT	读取数据的长度（以字节为单位）。
DATAADDR	PVOID	读取数据所在数据缓冲区的地址。
RESPOND	BOOL	功能块由该输入端的上升沿触发激活。

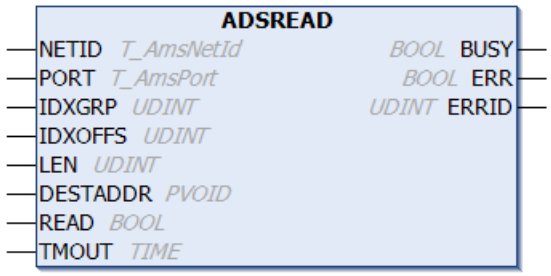
输出

```
VAR_OUTPUT
(*none*)
END_VAR
```

前提条件

开发环境	目标平台	待集成的 PLC 库（类别组）
TwinCAT v3.1.0	PC 或 CX（x86、x64、Arm®）	Tc2_System（系统）

3.2.3 ADSREAD



该功能块执行 ADS 读取命令，以向 ADS 设备请求数据。

过时的响应数据

断开连接后，在重新连接时会输出旧的响应数据。
为了避免出现这种情况，请注意不要将同一个 ADS-Read 实例用于多个目标。

输入

```
VAR_INPUT
    NETID      : T_AmsNetId;
    PORT       : T_AmsPort;
    IDXGRP     : UDINT;
    IDXOFFS    : UDINT;
    LEN        : UDINT;
    DESTADDR   : PVOID;
    READ       : BOOL;
    TMOUT      : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

名称	类型	描述
NETID	T_AmsNetId	字符串，包含 ADS 命令所指向的目标设备的 AMS NetId（类型：T_AmsNetId [▶ 115]）。
PORT	T_AmsPort	ADS 设备的端口号（类型：T_AmsPort [▶ 116]）
IDXGRP	UDINT	所请求的 ADS 服务的索引组号（32 位，无符号）。在寻址设备的 ADS 表中可以找到该值。
IDXOFFS	UDINT	请求的 ADS 服务的索引偏移编号（32 位，无符号）。在寻址设备的 ADS 表中可以找到该值。
LEN	UDINT	要读取数据的字节数
DESTADDR	PVOID	要接收读取数据的缓冲区的地址。程序员负责确定缓冲区的大小，以便其可以容纳 LEN 字节。缓冲区可以是单个变量、数组或结构体，其地址可以通过 ADR 运算符找到。
READ	BOOL	ADS 命令由该输入端的上升沿触发。
TMOUT	TIME	表示函数被取消前的时间。

输出

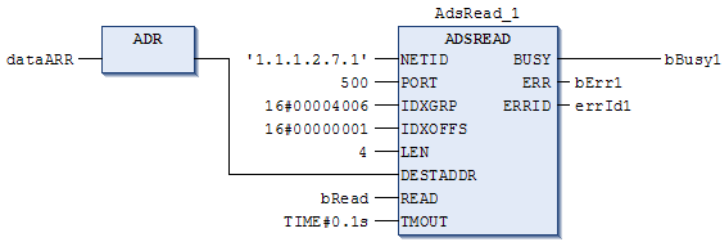
```
VAR_OUTPUT
    BUSY    : BOOL;
    ERR     : BOOL;
    ERRID   : UDINT;
END_VAR
```

名称	类型	描述
BUSY	BOOL	在功能块执行命令之前，该输出保持为 TRUE，但最长持续时间为 Timeout 输入所提供的时间。只要 BUSY = TRUE，输入端就不接受任何新命令。请注意，监控的不是执行服务的时间，而是接受服务的时间。
ERR	BOOL	一旦在执行命令过程中出错，该输出将切换为 TRUE。特定命令的错误代码包含在 ERRID 中。如果功能块出现超时错误，ERR 为 TRUE，ERRID 为 1861（十六进制 0x745）。在输入端执行命令后重置为 FALSE。
ERRID	UDINT	ADS 错误代码 [▶ 136]或最后执行命令的特定命令错误代码。在输入端执行命令后重置为 0。

使用 ST 语言调用功能块的示例：

- 使用 AdsRead 功能块的示例 [▶ 129]

在 FBD 中调用功能块的示例：

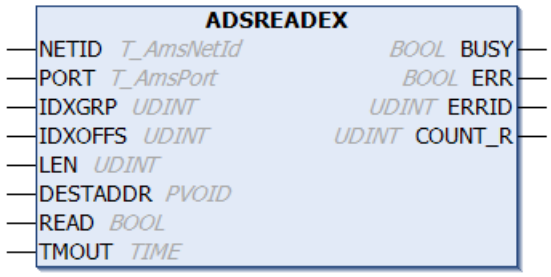


6 号轴的错误状态作为一个 4 字节大小的元素将被查询并写入 dataArr 缓冲区。在 NC-ADS 文档中可以找到 IDXGRP 00004006（十六进制）和 IDXOFFS 00000001（十六进制）。

前提条件

开发环境	目标平台	待集成的 PLC 库（类别组）
TwinCAT v3.1.0	PC 或 CX（x86、x64、Arm®）	Tc2_System（系统）

3.2.4 ADSREADEX



该功能块执行 ADS 读取命令，以向 ADS 设备请求数据。该功能块的功能与 ADSREAD 功能块相同；此外，它还将实际读取的数据字节数作为参数返回。

输入

```
VAR_INPUT
    NETID    : T_AmsNetId;
    PORT     : T_AmsPort;
    IDXGRP   : UDINT;
    IDXOFFS  : UDINT;
    LEN      : UDINT;
    DESTADDR : PVOID;
    READ     : BOOL;
    TMOUT    : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

名称	类型	描述
NETID	T_AmsNetId	字符串，包含 ADS 命令所指向的目标设备的 AMS NetId（类型：T_AmsNetId [▶ 115]）。
PORT	T_AmsPort	ADS 设备的端口号（类型：T_AmsPort [▶ 116]）
IDXGRP	UDINT	所请求的 ADS 服务的索引组号（32 位，无符号）。在寻址设备的 ADS 表中可以找到该值。
IDXOFFS	UDINT	请求的 ADS 服务的索引偏移编号（32 位，无符号）。在寻址设备的 ADS 表中可以找到该值。
LEN	UDINT	要读取数据的字节数
DESTADDR	PVOID	要接收读取数据的缓冲区的地址。程序员负责确定缓冲区的大小，以便其可以容纳 LEN 字节。缓冲区可以是单个变量、数组或结构体，其地址可以通过 ADR 运算符找到。
READ	BOOL	ADS 命令由该输入端的上升沿触发。
TMOUT	TIME	表示函数被取消前的时间。

输出

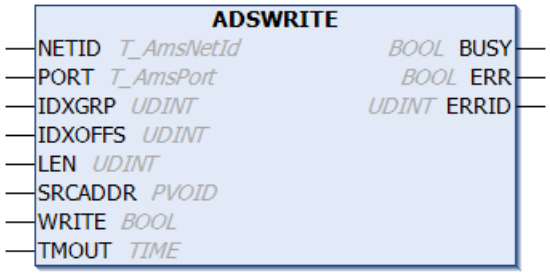
```
VAR_OUTPUT
    BUSY    : BOOL;
    ERR     : BOOL;
    ERRID   : UDINT;
    COUNT_R : UDINT;
END_VAR
```

名称	类型	描述
BUSY	BOOL	在功能块执行命令之前，该输出保持为 TRUE，但最长持续时间为 Timeout 输入所提供的时间。当 BUSY = TRUE 时，输入端将不接受任何新命令。请注意，监控的不是执行服务的时间，而是接受服务的时间。
ERR	BOOL	一旦在执行命令过程中出错，该输出将切换为 TRUE。特定命令的错误代码包含在 ERRID 中。如果功能块出现超时错误，ERR 为 TRUE，ERRID 为 1861（十六进制 0x745）。在输入端执行命令后重置为 FALSE。
ERRID	UDINT	ADS 错误代码 [▶ 136]或最后执行命令的特定命令错误代码。在输入端执行命令后重置为 0。
COUNT_R	UDINT	成功读取的数据字节数

前提条件

开发环境	目标平台	待集成的 PLC 库（类别组）
TwinCAT v3.1.0	PC 或 CX（x86、x64、Arm®）	Tc2_System（系统）

3.2.5 ADSWRITE



用于执行向 ADS 设备传输数据的 ADS 写入命令的功能块。

输入

```
VAR_INPUT
    NETID   : T_AmsNetId;
    PORT    : T_AmsPort;
    IDXGRP  : UDINT;
    IDXOFFS : UDINT;
    LEN     : UDINT;
    SRCADDR : PVOID;
    WRITE   : BOOL;
    TMOUT   : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

名称	类型	描述
NETID	T_AmsNetId	字符串，包含 ADS 命令所指向的目标设备的 AMS NetId（类型：T_AmsNetId [▶ 115]）。
PORT	T_AmsPort	ADS 设备的端口号（类型：T_AmsPort [▶ 116]）
IDXGRP	UDINT	所请求的 ADS 服务的索引组号（32 位，无符号）。在寻址设备的 ADS 表中可以找到该值。
IDXOFFS	UDINT	请求的 ADS 服务的索引偏移编号（32 位，无符号）。在寻址设备的 ADS 表中可以找到该值。
LEN	UDINT	要读取数据的字节数
SRCADDR	PVOID	用于获取要写入数据的缓冲区的地址。程序员负责确定缓冲区的大小，以便可以从中获取 LEN 字节。缓冲区可以是单个变量、数组或结构体，其地址可以通过 ADR 运算符找到。
WRITE	BOOL	ADS 命令由该输入端的上升沿触发。
TMOUT	TIME	表示函数被取消前的时间。

输出

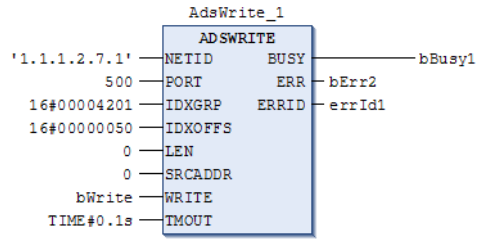
```
VAR_OUTPUT
    BUSY    : BOOL;
    ERR     : BOOL;
    ERRID   : UDINT;
END_VAR
```

名称	类型	描述
BUSY	BOOL	在功能块执行命令之前，该输出保持为 TRUE，但最长持续时间为 Timeout 输入所提供的时间。只要 BUSY = TRUE，输入端就不接受任何新命令。请注意，监控的不是执行服务的时间，而是接受服务的时间。
ERR	BOOL	一旦在执行命令过程中出错，该输出将切换为 TRUE。特定命令的错误代码包含在 ERRID 中。如果功能块出现超时错误，ERR 为 TRUE，ERRID 为 1861（十六进制 0x745）。在输入端执行命令后重置为 FALSE。
ERRID	UDINT	ADS 错误代码 [▶ 136]或最后执行命令的特定命令错误代码。在输入端执行命令后重置为 0。

使用 ST 语言调用功能块的示例：

- 使用 AdsWrite 功能块的示例 [▶ 129]

在 FBD 中调用功能块的示例

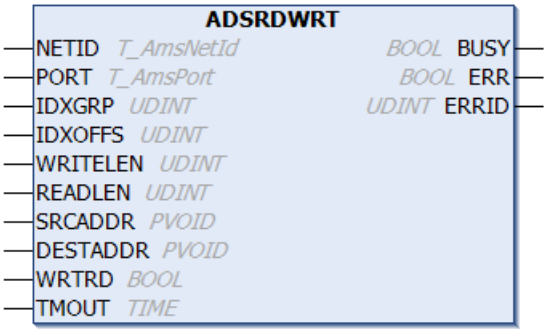


此处通过 IDXGRP 00004201（十六进制）和 IDXOFFS 00000050（十六进制）的写入指令停用 1 号 NC 轴。要激活该轴，必须发送另一个 IDXOFFS 00000051（十六进制）的写入指令。由于该写入指令不需要任何参数，因此输入 LEN 和 SRCADDR 没有任何意义，但必须将其设置为零。

前提条件

开发环境	目标平台	待集成的 PLC 库（类别组）
TwinCAT v3.1.0	PC 或 CX（x86、x64、Arm®）	Tc2_System（系统）

3.2.6 ADSRDWRT



该功能块执行 ADS 读写组合命令。向 ADS 设备传输数据（写入），并通过一次调用读取其响应数据。

输入

```
VAR_INPUT
    NETID     : T_AmsNetId;
    PORT      : T_AmsPort;
```

```
    IDXGRP    : UDINT;
    IDXOFFS   : UDINT;
    WRITELEN  : UDINT;
    READLEN   : UDINT;
    SRCADDR   : PVOID;
    DESTADDR  : PVOID;
    WRTRD     : BOOL;
    TMOUT     : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

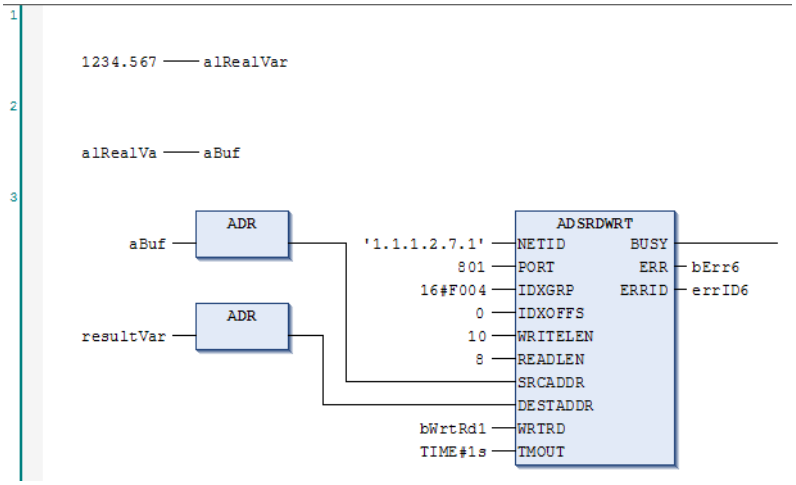
名称	类型	描述
NETID	T_AmsNetId	字符串，包含 ADS 命令所指向的目标设备的 AMS NetId（类型：T_AmsNetId [▶ 115]）。
PORT	T_AmsPort	ADS 设备的端口号（类型：T_AmsPort [▶ 116]）
IDXGRP	UDINT	所请求的 ADS 服务的索引组号（32 位，无符号）。在寻址设备的 ADS 表中可以找到该值。
IDXOFFS	UDINT	请求的 ADS 服务的索引偏移编号（32 位，无符号）。在寻址设备的 ADS 表中可以找到该值。
WRITELEN	UDINT	要写入数据的字节数
READLEN	UDINT	要读取数据的字节数
SRCADDR	PVOID	用于获取要写入数据的缓冲区的地址。程序员负责确定缓冲区的大小，以便可以从中获取 WRITELEN 字节。缓冲区可以是单个变量、数组或结构体，其地址可以通过 ADR 运算符找到。
DESTADDR	PVOID	要接收读取数据的缓冲区的地址。程序员负责确定缓冲区的大小，以便其可以容纳 READLEN 字节。缓冲区可以是单个变量、数组或结构体，其地址可以通过 ADR 运算符找到。
WRTRD	BOOL	ADS 命令由该输入端的上升沿触发。
TMOUT	TIME	表示函数被取消前的时间。

输出

```
VAR_OUTPUT
    BUSY    : BOOL;
    ERR     : BOOL;
    ERRID   : UDINT;
END_VAR
```

名称	类型	描述
BUSY	BOOL	在功能块执行命令之前，该输出保持为 TRUE，但最长持续时间为 Timeout 输入所提供的时间。只要 BUSY = TRUE，输入端就不接受任何新命令。请注意，监控的不是执行服务的时间，而是接受服务的时间。
ERR	BOOL	一旦在执行命令过程中出错，该输出将切换为 TRUE。特定命令的错误代码包含在 ERRID 中。如果功能块出现超时错误，ERR 为 TRUE，ERRID 为 1861（十六进制 0x745）。在输入端执行命令后重置为 FALSE。
ERRID	UDINT	ADS 错误代码 [▶ 136]或最后执行命令的特定命令错误代码。在输入端执行命令后重置为 0。

在 FBD 中调用功能块的示例：

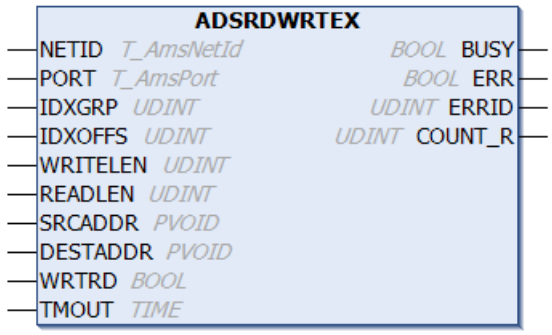


此处从在 NetId 为 1.1.1.2.7.1 的计算机上运行的 PLC 中读取名为“aLRealVar”的变量的值。为此，需提供上述计算机地址、PLC 的第一个 Runtime 系统的端口号、索引组以及按名称读取变量的索引偏移（F004 十六进制，0）。变量的名称将提供给 PLC 服务器；为此，将其存放于缓冲区中。由于该变量为全局变量，因此前面有一个引导点。这使得要写入的数据长度为 10 个字符（1 个点和 9 个字母）。由于要读取的变量为 LREAL 类型，因此要读取的字节数为 8。名称缓冲区的地址作为要写入数据的地址提供，而在接收数据时，则会提供一个 LREAL 变量（resultVar）的地址。图中显示了执行 WriteRead 指令后功能块在流程控制中的状态：之前包含在 aLRealVar 中的值 1234.567 现在也包含在 resultVar 中。

前提条件

开发环境	目标平台	待集成的 PLC 库（类别组）
TwinCAT v3.1.0	PC 或 CX（x86、x64、Arm®）	Tc2_System（系统）

3.2.7 ADSRDWRTEX



该功能块允许执行 ADS 写入/读取组合指令。向 ADS 设备传输数据（写入），并通过一次调用读取其响应数据。与 ADSRDWRT 功能块相反，ADSRDWRTEX 将实际读取的数据字节数作为参数提供。

输入

```
VAR_INPUT
    NETID    : T_AmsNetId;
    PORT     : T_AmsPort;
    IDXGRP   : UDINT;
    IDXOFFS  : UDINT;
    WRITELEN : UDINT;
    READLEN  : UDINT;
    SRCADDR  : PVOID;
    DESTADDR : PVOID;
    WRTRD    : BOOL;
    TMOUT    : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

名称	类型	描述
NETID	T_AmsNetId	字符串，包含 ADS 命令所指向的目标设备的 AMS NetId（类型：T_AmsNetId [▶ 115]）。
PORT	T_AmsPort	ADS 设备的端口号（类型：T_AmsPort [▶ 116]）
IDXGRP	UDINT	所请求的 ADS 服务的索引组号（32 位，无符号）。在寻址设备的 ADS 表中可以找到该值。
IDXOFFS	UDINT	请求的 ADS 服务的索引偏移编号（32 位，无符号）。在寻址设备的 ADS 表中可以找到该值。
WRITELEN	UDINT	要写入数据的字节数
READLEN	UDINT	要读取数据的字节数
SRCADDR	PVOID	用于获取要写入数据的缓冲区的地址。程序员负责确定缓冲区的大小，以便可以从中获取 WRITELEN 字节。缓冲区可以是单个变量、数组或结构体，其地址可以通过 ADR 运算符找到。
DESTADDR	PVOID	要接收读取数据的缓冲区的地址。程序员负责确定缓冲区的大小，以便其可以容纳 READLEN 字节。缓冲区可以是单个变量、数组或结构体，其地址可以通过 ADR 运算符找到。
WRTRD	BOOL	ADS 命令由该输入端的上升沿触发。
TMOUT	TIME	表示函数被取消前的时间。

输出

```
VAR_OUTPUT
    BUSY    : BOOL;
    ERR     : BOOL;
    ERRID   : UDINT;
    COUNT_R : UDINT;
END_VAR
```

名称	类型	描述
BUSY	BOOL	在功能块执行命令之前，该输出保持为 TRUE，但最长持续时间为 Timeout 输入所提供的时间。当 BUSY = TRUE 时，输入端将不接受任何新命令。请注意，监控的不是执行服务的时间，而是接受服务的时间。
ERR	BOOL	一旦在执行命令过程中出错，该输出将切换为 TRUE。特定命令的错误代码包含在 ERRID 中。如果功能块出现超时错误，ERR 为 TRUE，ERRID 为 1861（十六进制 0x745）。在输入端执行命令后重置为 FALSE。
ERRID	UDINT	ADS 错误代码 [▶ 136]或最后执行命令的特定命令错误代码。在输入端执行命令后重置为 0。
COUNT_R	UDINT	成功读取的数据字节数

前提条件

开发环境	目标平台	待集成的 PLC 库（类别组）
TwinCAT v3.1.0	PC 或 CX（x86、x64、Arm®）	Tc2_System（系统）

3.3 文件功能块

下列功能块仅适用于有限范围内的实时记录。
为了获得良好的性能，我们建议使用 TF3500 TwinCAT Analytics Logger 产品，该产品为付费产品。

3.3.1 FB_EOF

功能块 FB_EOF 可以检查是否已到达文件末尾。

该功能块仅适用于有限的范围内的实时记录。
为了获得良好的性能，我们建议使用 TF3500 TwinCAT Analytics Logger 产品，该产品需要付费。

输入

```
VAR_INPUT
    sNetId    : T_AmsNetId;
    hFile     : UINT;
    bExecute  : BOOL;
    tTimeout  : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

名称	类型	描述
sNetId	T_AmsNetId	字符串，包含 ADS 命令所指向的目标设备的 AMS NetId（类型：T_AmsNetId [▶ 115]）。
hFile	UINT	文件句柄，在创建功能块 FB_FileOpen 时生成。
bExecute	BOOL	功能块由该输入端的上升沿激活。
tTimeout	TIME	规定执行 ADS 命令时不得超过的超时长度。

输出

```
VAR_OUTPUT
    bBusy  : BOOL;
    bError : BOOL;
    nErrId : UDINT;
    bEOF   : BOOL;
END_VAR
```

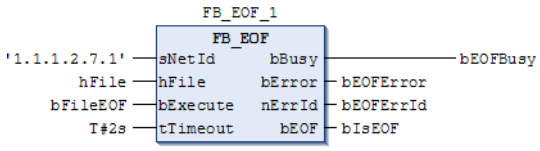
名称	类型	描述
bBusy	BOOL	激活功能块后，将该输出设置为 TRUE，并保持设置状态，直至收到反馈。只要 bBusy 为 TRUE，就无法执行新命令。
bError	BOOL	如果在命令执行过程中出现错误，则在 bBusy 输出复位后设置该输出。
nErrId	UDINT	在设置 bError 输出时，返回 ADS 错误代码 [▶ 136]或特定命令的错误代码。
bEOF	BOOL	当到达文件末尾时，将会设置该输出。

特定命令的错误代码	可能的原因
0x703	无效或未知的文件句柄。
0x70E	文件以错误的方法打开（例如，使用过时的 FILEOPEN 功能块）。

在 FBD 中调用功能块的示例：

```
PROGRAM Test
VAR
    fbEOF       : FB_EOF;
    hFile       : UINT;
    bFileEOF    : BOOL;
    bEOFBusy    : BOOL;
    bEOFError   : BOOL;
```

```
    nEOFErrorId : UDINT;
    bIsEOF      : BOOL;
END_VAR
```



前提条件

开发环境	目标平台	待集成的 PLC 库（类别组）
TwinCAT v3.1.0	PC 或 CX（x86、x64、Arm®）	Tc2_System（系统）

3.3.2 FB_FileOpen

功能块 FB_FileOpen 可创建一个新的文件或打开一个现有的文件进行编辑。

该功能块仅适用于有限的范围内的实时记录。
为了获得良好的性能，我们建议使用 TF3500 TwinCAT Analytics Logger 产品，该产品需要付费。

输入

```
VAR_INPUT
    sNetId      : T_AmsNetId;
    sPathName   : T_MaxString;
    nMode       : DWORD;
    ePath       : E_OpenPath := PATH_GENERIC;
    bExecute    : BOOL;
    tTimeout    : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

名称	类型	描述
sNetId	T_AmsNetId	字符串，包含 ADS 命令所指向的目标设备的 AMS NetId（类型：T_AmsNetId [▶ 115]）。
sPathName	T_MaxString	要打开的文件的存储路径和文件名。路径只能指向计算机的本地文件系统。此处不能指定网络路径（类型：T_MaxString [▶ 118]）。
nMode	DWORD	打开文件的模式。
ePath	E_OpenPath	该输入可以用于选择目标设备上的 TwinCAT 系统路径，以打开文件（类型：E_OpenPath [▶ 112]）。
bExecute	BOOL	功能块由该输入端的上升沿激活。
tTimeout	TIME	规定执行 ADS 命令时不得超过的超时长度。

nMode 的预定义打开模式

打开文件的模式。下面列出的代码为各种打开模式，这些模式在库中已预定义为常量，可由此以符号形式传递至功能块。打开模式可以使用逻辑或。打开模式可以组合，与 C 或 C++ 中 fopen 函数的打开模式类似。

模式	描述
FOPEN_MODEREAD	r：打开一个文件进行读取。如果找不到文件或文件不存在，则会返回错误。
FOPEN_MODEWRITE	w：打开一个空文件进行写入。如果文件已经存在，则会将其覆盖。
FOPEN_MODEAPPEND	a：打开一个文件，在文件末尾进行写入（添加）。如果文件不存在，则会创建一个新文件。
FOPEN_MODEREAD OR FOPEN_MODEPLUS	r+：打开一个文件进行读取和写入。文件必须存在。
FOPEN_MODEWRITE OR FOPEN_MODEPLUS	w+：打开一个空文件进行读取和写入。如果文件已经存在，则会将其覆盖。
FOPEN_MODEAPPEND OR FOPEN_MODEPLUS	a+：打开一个文件，在文件末尾进行读取和写入（附加）。如果文件不存在，则会创建一个新文件。为此，必须知道内存路径，否则会出现错误 1804。如果以模式 a 或 a+ 打开文件，则始终在文件末尾执行所有写入操作。通过 FB_FileSeek 可以移动文件指针，但在写入时，文件指针默认会移动到文件末尾，即不会覆盖现有数据。
FOPEN_MODEBINARY	b：以二进制模式打开文件
FOPEN_MODETEXT	t：以文本模式打开文件

输出

```
VAR_OUTPUT
    bBusy  : BOOL;
    bError : BOOL;
    nErrId : UDINT;
    hFile  : UINT;(* file handle *)
END_VAR
```

名称	类型	描述
bBusy	BOOL	激活功能块后，将该输出设置为 TRUE，并保持设置状态，直至收到反馈。只要 bBusy 为 TRUE，就无法执行新命令。
bError	BOOL	如果在命令执行过程中出现错误，则在 bBusy 输出复位后设置该输出。
nErrId	UDINT	在设置 bError 输出时，返回 ADS 错误代码 [▶ 136]或特定命令的错误代码。
hFile	UINT	包含在成功打开文件后为其创建的文件句柄。 然后，该句柄将作为要编辑的文件的 ID 传输给其他文件功能块。

hFile 的错误代码

特定命令的错误代码	可能的原因
0x703	未知或无效的 nMode 或 ePath 参数。
0x70C	找不到文件。无效的文件名或路径。
0x716	没有更多可用的文件句柄。

信息：

在打开模式下，最多可对 3 个参数进行逻辑或处理：

模式 = [参数1] 或 [参数2] 或 [参数3]
参数1 可能只有一个从属值：

- FOPEN_MODEREAD
- FOPEN_MODEWRITE

- FOPEN_MODEAPPEND

参数2 可能只有一个从属值：

- FOPEN_MODEPLUS

参数3 可能只有一个从属值：

- FOPEN_MODEBINARY
- FOPEN_MODETEXT

如果没有指定二进制或文本模式，则文件将以操作系统变量定义的模式打开。在大多数情况下，文件将以文本模式打开。不过，我们无法做出明确的说明。建议始终指定文本或二进制模式。该系统变量不能在 PLC 中更改！
因此，允许的组合如下：

文本文件打开模式	二进制文件打开模式
FOPEN_MODEREAD OR FOPEN_MODETEXT	FOPEN_MODEREAD OR FOPEN_MODEBINARY
FOPEN_MODEWRITE OR FOPEN_MODETEXT	FOPEN_MODEWRITE OR FOPEN_MODEBINARY
FOPEN_MODEAPPEND OR FOPEN_MODETEXT	FOPEN_MODEAPPEND OR FOPEN_MODEBINARY
FOPEN_MODEREAD OR FOPEN_MODEPLUS OR FOPEN_MODETEXT	FOPEN_MODEREAD OR FOPEN_MODEPLUS OR FOPEN_MODEBINARY
FOPEN_MODEWRITE OR FOPEN_MODEPLUS OR FOPEN_MODETEXT	FOPEN_MODEWRITE OR FOPEN_MODEPLUS OR FOPEN_MODEBINARY
FOPEN_MODEAPPEND OR FOPEN_MODEPLUS OR FOPEN_MODETEXT	FOPEN_MODEAPPEND OR FOPEN_MODEPLUS OR FOPEN_MODEBINARY

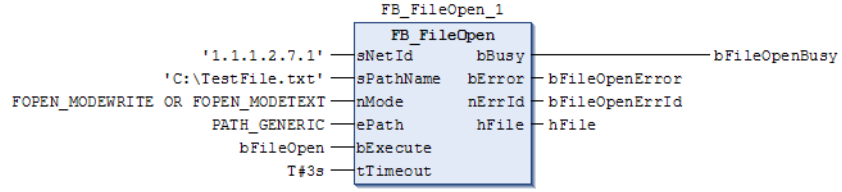
所有其他组合都是错误的。无效的打开模式示例：
FOPEN_MODEBINARY OR FOPEN_MODETEXT
FOPEN_MODEWRITE OR FOPEN_MODEAPPEND

使用 ST 语言调用功能块的示例：

- 从 PLC 访问文件 [▶ 131]

在 FBD 中调用功能块的示例：

```
PROGRAM Test
VAR
    fbFileOpen      : FB_FileOpen;
    bFileOpen       : BOOL;
    bFileOpenBusy   : BOOL;
    bFileOpenError  : BOOL;
    nFileOpenErrId  : UDINT;
    hFile           : UINT;
END_VAR
```



这应在 C 盘根目录下以文本模式创建（或覆盖）文件 TestFile2.txt。

前提条件

开发环境	目标平台	待集成的 PLC 库（类别组）
TwinCAT v3.1.0	PC 或 CX（x86、x64、Arm®）	Tc2_System（系统）

3.3.3 FB_FileClose

功能块 FB_FileClose 可关闭文件，从而将文件置于定义的状态，以便其他程序进一步处理。

该功能块仅适用于有限的范围内的实时记录。
为了获得良好的性能，我们建议使用 TF3500 TwinCAT Analytics Logger 产品，该产品需要付费。

输入

```
VAR_INPUT
    sNetId    : T_AmsNetId;
    hFile     : UINT;
    bExecute  : BOOL;
    tTimeout  : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

名称	类型	描述
sNetId	T_AmsNetId	字符串，包含 ADS 命令所指向的目标设备的 AMS NetId（类型：T_AmsNetId [▶ 115]）。
hFile	UINT	要关闭的文件的句柄。
bExecute	BOOL	功能块由该输入端的上升沿激活。
tTimeout	TIME	规定执行 ADS 命令时不得超过的超时长度。

输出

```
VAR_OUTPUT
    bBusy   : BOOL;
    bError  : BOOL;
    nErrId  : UDINT;
END_VAR
```

名称	类型	描述
bBusy	BOOL	激活功能块后，将该输出设置为 TRUE，并保持设置状态，直至收到反馈。只要 bBusy 为 TRUE，就无法执行新命令。
bError	BOOL	如果在命令执行过程中出现错误，则在 bBusy 输出复位后设置该输出。
nErrId	UDINT	在设置 bError 输出时，返回 ADS 错误代码 [▶ 136]或特定命令的错误代码。

nErrId 的特定命令错误代码

在设置 bError 输出时，返回 ADS 错误代码 [▶ 136]或特定命令的错误代码。

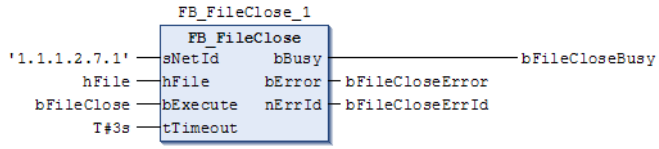
特定命令的错误代码	可能的原因
0x703	无效或未知的文件句柄。
0x70E	文件以错误的方法打开（例如，使用过时的 FILEOPEN 功能块）。

使用 ST 语言调用功能块的示例：

- 从 PLC 访问文件 [▶ 131]

在 FBD 中调用功能块的示例：

```
PROGRAM Test
VAR
    fbFileClose       : FB_FileClose;
    hFile             : UINT;
    bFileClose        : BOOL;
    bFileCloseBusy    : BOOL;
    bFileCloseError   : BOOL;
    nFileCloseErrorId : UDINT;
END_VAR
```



此处将再次关闭与文件句柄（该句柄本身由 FB_FileOpen 生成）相关联的文件。

前提条件

开发环境	目标平台	待集成的 PLC 库（类别组）
TwinCAT v3.1.0	PC 或 CX（x86、x64、Arm®）	Tc2_System（系统）

3.3.4 FB_FileLoad

通过功能块 FB_FileLoad 可以读出文件的内容。以二进制模式隐式打开文件，读出文件内容，然后再次关闭文件。

该功能块仅适用于有限的范围内的实时记录。
为了获得良好的性能，我们建议使用 TF3500 TwinCAT Analytics Logger 产品，该产品需要付费。

输入

```
VAR_INPUT
    sNetId      : T_AmsNetId;
    sPathName   : T_MaxString;
    pReadBuff   : PVOID;
    cbReadLen   : UDINT;
    bExecute    : BOOL;
    tTimeout    : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

名称	类型	描述
sNetId	T_AmsNetId	字符串，包含 ADS 命令所指向的目标设备的 AMS NetId（类型：T_AmsNetId [▶ 115]）。
sPathName	T_MaxString	要打开的文件的存储路径和文件名。路径只能指向计算机的本地文件系统。此处不能指定网络路径（类型：T_MaxString [▶ 118]）
pReadBuff	PVOID	要读取数据的缓冲区的地址。缓冲区可以是单个变量、数组或结构体，其地址可以通过 ADR 运算符找到。
cbReadLen	UDINT	要读取的字节数。
bExecute	BOOL	功能块由该输入端的上升沿激活。
tTimeout	TIME	规定执行内部 ADS 命令时不得超过的超时长度。

输出

```
VAR_OUTPUT
    bBusy  : BOOL;
    bError : BOOL;
    nErrId : UDINT;
    cbRead : UDINT;
END_VAR
```

名称	类型	描述
bBusy	BOOL	激活功能块后，将该输出设置为 TRUE，并保持设置状态，直至收到反馈。只要 bBusy 为 TRUE，就无法执行新命令。
bError	BOOL	如果在命令执行过程中出现错误，则在 bBusy 输出复位后设置该输出。
nErrId	UDINT	在设置 bError 输出时，返回 ADS 错误代码 [▶ 136]或特定命令的错误代码。
cbRead	UDINT	当前读取的字节数

要求

开发环境	目标平台	待集成的 PLC 库（类别组）
TwinCAT v3.1.4022.0	PC 或 CX（x86、x64、ARM）	Tc2_System（系统）不低于 v3.4.22.0

3.3.5 FB_FileGets

功能块 FB_FileGest 从文件中读取字符串。字符串将一直读取至遇到换行字符（包括该字符），或直至文件末尾，或达到 sLine 允许的最大长度。空终止符将自动添加。文件必须以文本模式打开。

该功能块仅适用于有限的范围内的实时记录。
为了获得良好的性能，我们建议使用 TF3500 TwinCAT Analytics Logger 产品，该产品需要付费。

输入

```
VAR_INPUT
    sNetId    : T_AmsNetId;
    hFile     : UINT;
```

```
    bExecute  : BOOL;
    tTimeout  : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

名称	类型	描述
sNetId	T_AmsNetId	字符串，包含 ADS 命令所指向的目标设备的 AMS NetId（类型：T_AmsNetId [▶ 115]）。
hFile	UINT	文件句柄，在创建功能块 FB_FileOpen 时生成。
bExecute	BOOL	功能块由该输入端的上升沿激活。
tTimeout	TIME	规定执行 ADS 命令时不得超过的超时长度。

输出

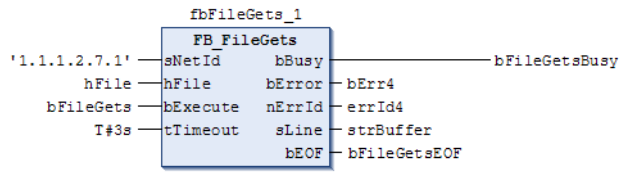
```
VAR_OUTPUT
    bBusy   : BOOL;
    bError  : BOOL;
    nErrId  : UDINT;
    sLine   : T_MaxString;
    bEOF    : BOOL;
END_VAR
```

名称	类型	描述
bBusy	BOOL	激活功能块后，将该输出设置为 TRUE，并保持设置状态，直至收到反馈。只要 bBusy 为 TRUE，就无法执行新命令。
bError	BOOL	如果在命令执行过程中出现错误，则在 bBusy 输出复位后设置该输出。
nErrId	UDINT	在设置 bError 输出时，返回 ADS 错误代码 [▶ 136]或特定命令的错误代码。
sLine	T_MaxString	读取的字符串（类型：T_MaxString [▶ 118]）。
bEOF	BOOL	如果已到达文件末尾，且无法读取更多数据字节（cbRead=0），则会设置该输出。如果可以读取更多数据字节（cbRead>0），则不会设置该输出。

特定命令的错误代码	可能的原因
0x703	无效或未知的文件句柄。
0x70A	读取缓冲区无内存。
0x70E	文件以错误的方法打开（例如，使用过时的 FILEOPEN 功能块）。

在 FBD 中调用功能块的示例：

```
PROGRAM Test
VAR
    fbFileGets         : FB_FileGets;
    hFile              : UINT;
    bFileGets          : BOOL;
    bFileGetsBusy      : BOOL;
    bFileGetsError     : BOOL;
    nFileGetsErrorId   : UDINT;
    strBuffer          : STRING;
    bFileGetsEOF       : BOOL;
END_VAR
```



前提条件

开发环境	目标平台	待集成的 PLC 库（类别组）
TwinCAT v3.1.0	PC 或 CX（x86、x64、Arm®）	Tc2_System（系统）

3.3.6 FB_FilePuts

功能块 FB_FilePuts 将字符串写入文件。字符串将被写入文件至空终止符前（不含该空字符）。文件必须以文本模式打开。

该功能块仅适用于有限的范围内的实时记录。
为了获得良好的性能，我们建议使用 TF3500 TwinCAT Analytics Logger 产品，该产品需要付费。

输入

```
VAR_INPUT
    sNetId    : T_AmsNetId;
    hFile     : UINT;
    sLine     : T_MaxString;(* string to write *)
    bExecute  : BOOL;
    tTimeout  : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

名称	类型	描述
sNetId	T_AmsNetId	字符串，包含 ADS 命令所指向的目标设备的 AMS NetId（类型：T_AmsNetId [▶ 115]）。
hFile	UINT	文件句柄，在创建功能块 FB_FileOpen 时生成。
sLine	T_MaxString	要写入文件的字符串（类型：T_MaxString [▶ 118]）。
bExecute	BOOL	功能块由该输入端的上升沿激活。
tTimeout	TIME	规定执行 ADS 命令时不得超过的超时长度。

输出

```
VAR_OUTPUT
    bBusy  : BOOL;
    bError : BOOL;
    nErrId : UDINT;
END_VAR
```

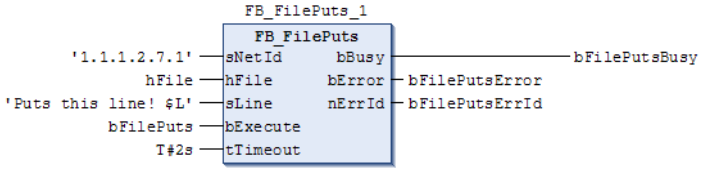
名称	类型	描述
bBusy	BOOL	激活功能块后，将该输出设置为 TRUE，并保持设置状态，直至收到反馈。只要 bBusy 为 TRUE，就无法执行新命令。
bError	BOOL	如果在命令执行过程中出现错误，则在 bBusy 输出复位后设置该输出。
nErrId	UDINT	在设置 bError 输出时，返回 ADS 错误代码 [▶ 136]或特定命令的错误代码。

特定命令的错误代码	可能的原因
0x703	无效或未知的文件句柄。
0x70E	文件以错误的方法打开（例如，使用过时的 FILEOPEN 功能块）。

在 FBD 中调用功能块的示例：

```
PROGRAM Test
VAR
    fbFilePuts       : FB_FilePuts;
    hFile            : UINT;
    bFilePuts        : BOOL;
    bFilePutsBusy    : BOOL;
```

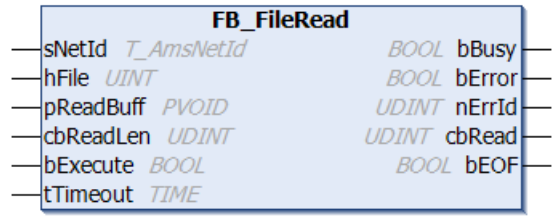
```
    bFilePutsError   : BOOL;
    nFilePutsErrorId : UDINT;
END_VAR
```



前提条件

开发环境	目标平台	待集成的 PLC 库（类别组）
TwinCAT v3.1.0	PC 或 CX（x86、x64、Arm®）	Tc2_System（系统）

3.3.7 FB_FileRead



通过功能块 FB_FileRead 可以读取已打开文件的内容。在读取访问之前，文件必须以相应模式打开。除了 FOPEN_MODEREAD 之外，适当的格式（FOPEN_MODEBINARY 或 FOPEN_MODETEXT）对实现预期效果也很重要。

该功能块仅适用于有限的范围内的实时记录。
为了获得良好的性能，我们建议使用 TF3500 TwinCAT Analytics Logger 产品，该产品需要付费。

输入

```
VAR_INPUT
    sNetId      : T_AmsNetId;
    hFile       : UINT;
    pReadBuff   : PVOID;
    cbReadLen   : UDINT;
    bExecute    : BOOL;
    tTimeout    : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

名称	类型	描述
sNetId	T_AmsNetId	字符串，包含 ADS 命令所指向的目标设备的 AMS NetId（类型：T_AmsNetId [▶ 115]）。
hFile	UINT	文件句柄，在创建功能块 FB_FileOpen 时生成。
pReadBuff	PVOID	要读取数据的缓冲区的地址。缓冲区可以是单个变量、数组或结构体，其地址可以通过 ADR 运算符找到。
cbReadLen	UDINT	要读取的字节数
bExecute	BOOL	功能块由该输入端的上升沿激活。
tTimeout	TIME	规定执行 ADS 命令时不得超过的超时长度。

输出

```
VAR_OUTPUT
    bBusy   : BOOL;
    bError  : BOOL;
    nErrId  : UDINT;
    cbRead  : UDINT;
    bEOF    : BOOL;
END_VAR
```

名称	类型	描述
bBusy	BOOL	激活功能块后，将该输出设置为 TRUE，并保持设置状态，直至收到反馈。只要 bBusy 为 TRUE，就无法执行新命令。
bError	BOOL	如果在命令执行过程中出现错误，则在 bBusy 输出复位后设置该输出。
nErrId	UDINT	在设置 bError 输出时，返回 ADS 错误代码 [▶ 136]或特定命令的错误代码。
cbRead	UDINT	当前读取的字节数
bEOF	BOOL	如果已到达文件末尾，且无法读取更多数据字节（cbRead=0），则会设置该输出。如果可以读取更多数据字节（cbRead>0），则不会设置该输出。

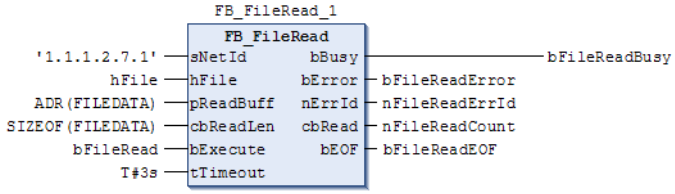
特定命令的错误代码	可能的原因
0x703	无效或未知的文件句柄。
0x70A	读取缓冲区无内存。
0x70E	文件以错误的方法打开（例如，使用过时的 FILEOPEN 功能块）。

使用 ST 语言调用功能块的示例：

- 从 PLC 访问文件 [▶ 131]

在 FBD 中调用功能块的示例：

```
PROGRAM Test
VAR
    fbFileRead         : FB_FileRead;
    hFile              : UINT;
    bFileRead          : BOOL;
    bFileReadBusy      : BOOL;
    bFileReadError     : BOOL;
    nFileReadErrorId   : UDINT;
    nFileReadCount     : UDINT;
    bFileReadEOF       : BOOL;
    fileData           : ARRAY[0..9] OF BYTE;
END_VAR
```

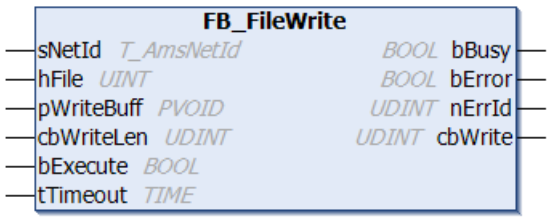


在 bExecute 上升沿并成功执行读取指令后，在 FILEDATA 中将会找到当前从文件中读取的字节。参数 cbRead 可用于确定在最后读取操作中实际已读取的字节数。

前提条件

开发环境	目标平台	待集成的 PLC 库（类别组）
TwinCAT v3.1.0	PC 或 CX（x86、x64、Arm®）	Tc2_System（系统）

3.3.8 FB_FileWrite



功能块 FB_FileWrite 将数据写入文件。对于写入访问，文件必须以相应模式打开，而且必须再次关闭，以便外部程序进一步处理。

该功能块仅适用于有限的范围内的实时记录。
为了获得良好的性能，我们建议使用 TF3500 TwinCAT Analytics Logger 产品，该产品需要付费。

输入

```
VAR_INPUT
    sNetId      : T_AmsNetId;
    hFile       : UINT;
    pWriteBuff  : PVOID;
    cbWriteLen  : UDINT;
    bExecute    : BOOL;
    tTimeout    : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

名称	类型	描述
sNetId	T_AmsNetId	字符串，包含 ADS 命令所指向的目标设备的 AMS NetId（类型：T_AmsNetId [▶ 115]）。
hFile	UINT	文件句柄，在创建功能块 FB_FileOpen 时生成。
pWriteBuff	PVOID	包含要写入数据的缓冲区的地址。缓冲区可以是单个变量、数组或结构体，其地址可以通过 ADR 运算符找到。
cbWriteLen	UDINT	要写入的字节数
bExecute	BOOL	功能块由该输入端的上升沿激活。
tTimeout	TIME	规定执行 ADS 命令时不得超过的超时长度。

输出

```
VAR_OUTPUT
    bBusy     : BOOL;
    bError    : BOOL;
    nErrId    : UDINT;
    cbWrite   : UDINT;
END_VAR
```

名称	类型	描述
bBusy	BOOL	激活功能块后，将该输出设置为 TRUE，并保持设置状态，直至收到反馈。只要 bBusy 为 TRUE，就无法执行新命令。
bError	BOOL	如果在命令执行过程中出现错误，则在 bBusy 输出复位后设置该输出。
nErrId	UDINT	在设置 bError 输出时，返回 ADS 错误代码 [▶ 136]或特定命令的错误代码。
cbWrite	UDINT	包含最后成功写入的数据字节数。写入错误表示成功写入的数据字节数小于所要求的长度（cbWriteLen）或为零。例如，如果数据载体已满，就会发生写入错误。如果发生写入错误，则不会设置 bError 和 nErrID 输出。由于 PLC 应用程序已知要写入的数据字节数，因此可以比较实际写入长度与所要求的长度，并检测写入错误。在发生写入错误时，内部文件指针将处于未定义位置。

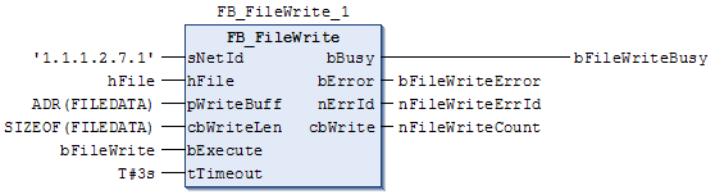
特定命令的错误代码	可能的原因
0x703	无效或未知的文件句柄。
0x70E	文件以错误的方法打开（例如，使用过时的 FILEOPEN 功能块）。

使用 ST 语言调用功能块的示例：

- 从 PLC 访问文件 [▶ 131]

在 FBD 中调用功能块的示例：

```
PROGRAM Test
VAR
    fbFileWrite      : FB_FileWrite;
    hFile            : UINT;
    bFileWrite       : BOOL;
    bFileWriteBusy   : BOOL;
    bFileWriteError  : BOOL;
    nFileWriteErrorId: UDINT;
    nFileWriteCount  : UDINT;
    fileData         : ARRAY[0..9] OF BYTE;
END_VAR
```



在此示例中，在 bFileWrite 的上升沿后，数组 FILEDATA 的 10 字节被写入文件。

前提条件

开发环境	目标平台	待集成的 PLC 库（类别组）
TwinCAT v3.1.0	PC 或 CX（x86、x64、Arm®）	Tc2_System（系统）

3.3.9 FB_FileSeek

功能块 FB_FileSeek 将打开文件的文件指针设置到可定义的位置。

该功能块仅适用于有限的范围内的实时记录。
为了获得良好的性能，我们建议使用 TF3500 TwinCAT Analytics Logger 产品，该产品需要付费。

输入

```
VAR_INPUT
    sNetId    : T_AmsNetId;
    hFile     : UINT;
    nSeekPos  : DINT;(* new seek pointer position *)
    eOrigin   : E_SeekOrigin:= SEEK_SET;
    bExecute  : BOOL;
    tTimeout  : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

名称	类型	描述
sNetId	T_AmsNetId	字符串，包含 ADS 命令所指向的目标设备的 AMS NetId（类型：T_AmsNetId [▶ 115]）。
hFile	UINT	文件句柄，在创建功能块 FB_FileOpen 时生成。
nSeekPos	DINT	文件指针的新位置
eOrigin	E_SeekOrigin	文件指针要移动到的相对位置（类型：E_SeekOrigin [▶ 112]）。
bExecute	BOOL	功能块由该输入端的上升沿激活。
tTimeout	TIME	规定执行 ADS 命令时不得超过的超时长度。

输出

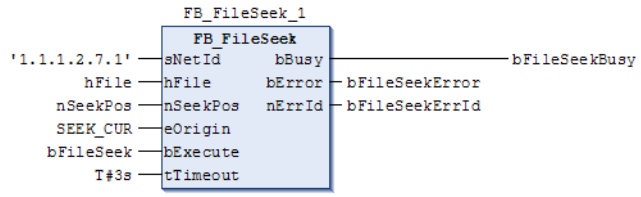
```
VAR_OUTPUT
    bBusy  : BOOL;
    bError : BOOL;
    nErrId : UDINT;
END_VAR
```

名称	类型	描述
bBusy	BOOL	激活功能块后，将该输出设置为 TRUE，并保持设置状态，直至收到反馈。只要 bBusy 为 TRUE，就无法执行新命令。
bError	BOOL	如果在命令执行过程中出现错误，则在 bBusy 输出复位后设置该输出。
nErrId	UDINT	在设置 bError 输出时，返回 ADS 错误代码 [▶ 136]或特定命令的错误代码。

特定命令的错误代码	可能的原因
0x703	无效或未知的文件句柄。
0x70E	文件以错误的方法打开（例如，使用过时的 FILEOPEN 功能块）。

在 FBD 中调用功能块的示例：

```
PROGRAM Test
VAR
    fbFileSeek       : FB_FileSeek;
    hFile            : UINT;
    nSeekPos         : DINT;
    bFileSeek        : BOOL;
    bFileSeekBusy    : BOOL;
    bFileSeekError   : BOOL;
    nFileSeekErrorId : UDINT;
END_VAR
```



前提条件

开发环境	目标平台	待集成的 PLC 库（类别组）
TwinCAT v3.1.0	PC 或 CX（x86、x64、Arm®）	Tc2_System（系统）

3.3.10 FB_FileTell

功能块 FB_FileTell 可确定文件指针的当前位置。位置表示相对于文件起始点的偏移量。

请注意，对于以“在文件末尾添加”模式打开的文件，当前位置由上一次 I/O 操作决定，而不是由下一次写入访问的位置决定。例如，在读取操作后，文件指针位于下一次读取访问的位置，而不是下一次写入访问的位置。在添加模式下，文件指针总是在写入操作前移动到末尾。

如果之前未执行任何 I/O 操作，且以添加模式打开文件，则文件指针位于文件的起始点。

该功能块仅适用于有限的范围内的实时记录。
为了获得良好的性能，我们建议使用 TF3500 TwinCAT Analytics Logger 产品，该产品需要付费。

输入

```
VAR_INPUT
    sNetId    : T_AmsNetId;
    hFile     : UINT;
    bExecute  : BOOL;
    tTimeout  : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

名称	类型	描述
sNetId	T_AmsNetId	字符串，包含 ADS 命令所指向的目标设备的 AMS NetId（类型：T_AmsNetId [▶ 115]）。
hFile	UINT	文件句柄，在创建功能块 FB_FileOpen 时生成。
bExecute	BOOL	功能块由该输入端的上升沿激活。
tTimeout	TIME	规定执行 ADS 命令时不得超过的超时长度。

输出

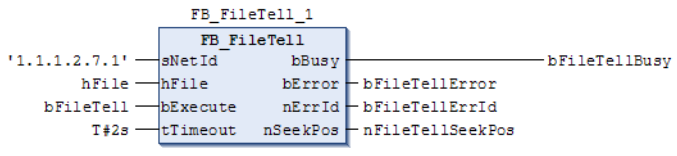
```
VAR_OUTPUT
    bBusy    : BOOL;
    bError   : BOOL;
    nErrId   : UDINT;
    nSeekPos : DINT;(* On error, nSEEKPOS returns -1 *)
END_VAR
```

名称	类型	描述
bBusy	BOOL	激活功能块后，将该输出设置为 TRUE，并保持设置状态，直至收到反馈。只要 bBusy 为 TRUE，就无法执行新命令。
bError	BOOL	如果在命令执行过程中出现错误，则在 bBusy 输出复位后设置该输出。
nErrId	UDINT	在设置 bError 输出时，返回 ADS 错误代码 [▶ 136]或特定命令的错误代码。
nSeekPos	DINT	返回文件指针的当前位置。

特定命令的错误代码	可能的原因
0x703	无效或未知的文件句柄。
0x70E	文件以错误的方法打开（例如，使用过时的 FILEOPEN 功能块）。

在 FBD 中调用功能块的示例：

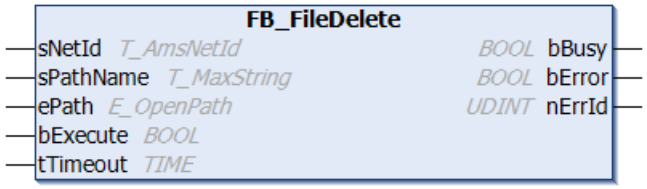
```
PROGRAM Test
VAR
    fbFileTell         : FB_FileTell;
    hFile              : UINT;
    bFileTell          : BOOL;
    bFileTellBusy      : BOOL;
    bFileTellError     : BOOL;
    nFileTellErrorId   : UDINT;
    nFileTellSeekPos   : DINT;
END_VAR
```



前提条件

开发环境	目标平台	待集成的 PLC 库（类别组）
TwinCAT v3.1.0	PC 或 CX（x86、x64、Arm®）	Tc2_System（系统）

3.3.11 FB_FileDelete



功能块 FB_FileDelete 可删除数据存储设备中的文件。

该功能块仅适用于有限的范围内的实时记录。
为了获得良好的性能，我们建议使用 TF3500 TwinCAT Analytics Logger 产品，该产品需要付费。

输入

```
VAR_INPUT
    sNetId    : T_AmsNetId;
    sPathName : T_MaxString;(* file path and name *)
    ePath     : E_OpenPath := PATH_GENERIC;
    bExecute  : BOOL;
    tTimeout  : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

名称	类型	描述
sNetId	T_AmsNetId	字符串，包含 ADS 命令所指向的目标设备的 AMS NetId（类型：T_AmsNetId [▶ 115]）。
sPathName	T_MaxString	文件名，包括完整路径（类型：T_MaxString [▶ 118]）。
ePath	E_OpenPath	该输入可以用于选择目标设备上的 TwinCAT 系统路径，以打开文件（类型：E_OpenPath [▶ 112]）。
bExecute	BOOL	功能块由该输入端的上升沿激活。
tTimeout	TIME	规定执行 ADS 命令时不得超过的超时长度。

输出

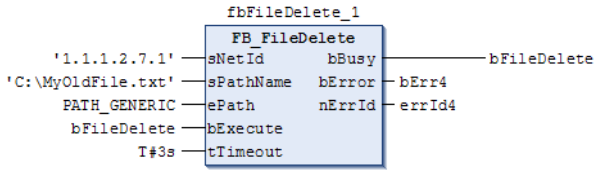
```
VAR_OUTPUT
    bBusy  : BOOL;
    bError : BOOL;
    nErrId : UDINT;
END_VAR
```

名称	类型	描述
bBusy	BOOL	激活功能块后，将该输出设置为 TRUE，并保持设置状态，直至收到反馈。只要 bBusy 为 TRUE，就无法执行新命令。
bError	BOOL	如果在命令执行过程中出现错误，则在 bBusy 输出复位后设置该输出。
nErrId	UDINT	在设置 bError 输出时，返回 ADS 错误代码 [▶ 136]或特定命令的错误代码。

特定命令的错误代码	可能的原因
0x70C	找不到文件。无效的 sPathName 或 ePath 参数。

在 FBD 中调用功能块的示例：

```
PROGRAM Test
VAR
    fbFileDelete      : FB_FileDelete;
    bFileDelete       : BOOL;
    bFileDeleteBusy   : BOOL;
    bFileDeleteError  : BOOL;
    nFileDeleteErrId  : UDINT;
END_VAR
```



前提条件

开发环境	目标平台	待集成的 PLC 库（类别组）
TwinCAT v3.1.0	PC 或 CX（x86、x64、Arm®）	Tc2_System（系统）

3.3.12 FB_FileRename

功能块 FB_FileRename 可用于重命名文件。

该功能块仅适用于有限的范围内的实时记录。
为了获得良好的性能，我们建议使用 TF3500 TwinCAT Analytics Logger 产品，该产品需要付费。

输入

```
VAR_INPUT
    sNetId   : T_AmsNetId;
    sOldName : T_MaxString;
    sNewName : T_MaxString;
    ePath    : E_OpenPath := PATH_GENERIC;      (* Default: generic file path*)
    bExecute : BOOL;
    tTimeout : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

名称	类型	描述
sNetId	T_AmsNetId	字符串，包含 ADS 命令所指向的目标设备的 AMS NetId（类型：T_AmsNetId [▶ 115]）。
sOldName	T_MaxString	旧文件名（类型：T_MaxString [▶ 118]）
sNewName	T_MaxString	新文件名（类型：T_MaxString [▶ 118]）
ePath	E_OpenPath	该输入可以用于选择目标设备上的 TwinCAT 系统路径，以打开文件（类型：E_OpenPath [▶ 112]）。
bExecute	BOOL	功能块由该输入端的上升沿激活。
tTimeout	TIME	规定执行 ADS 命令时不得超过的超时长度。

输出

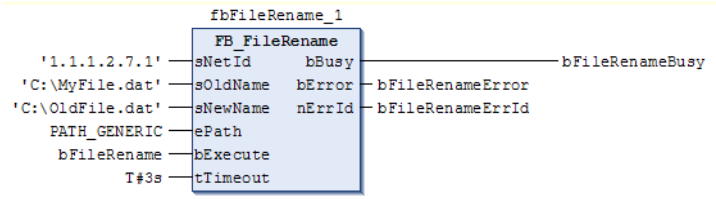
```
VAR_OUTPUT
    bBusy  : BOOL;
    bError : BOOL;
    nErrId : UDINT;
END_VAR
```

名称	类型	描述
bBusy	BOOL	激活功能块后，将该输出设置为 TRUE，并保持设置状态，直至收到反馈。只要 bBusy 为 TRUE，就无法执行新命令。
bError	BOOL	如果在命令执行过程中出现错误，则在 bBusy 输出复位后设置该输出。
nErrId	UDINT	在设置 bError 输出时，返回 ADS 错误代码 [▶ 136]或特定命令的错误代码。

特定命令的错误代码	可能的原因
0x70C	找不到文件。无效的 sOldName、sNewName 或 ePath 参数。

在 FBD 中调用功能块的示例：

```
PROGRAM Test
VAR
    fbFileRename      : FB_FileRename;
    bFileRename       : BOOL;
    bFileRenameBusy   : BOOL;
    bFileRenameError  : BOOL;
    nFileRenameErrId  : UDINT;
END_VAR
```



前提条件

开发环境	目标平台	待集成的 PLC 库（类别组）
TwinCAT v3.1.0	PC 或 CX（x86、x64、Arm®）	Tc2_System（系统）

3.3.13 FB_CreateDir

功能块 FB_CreateDir 可用于在数据存储设备上创建新目录。

该功能块仅适用于有限的范围内的实时记录。
为了获得良好的性能，我们建议使用 TF3500 TwinCAT Analytics Logger 产品，该产品需要付费。

输入

```
VAR_INPUT
    sNetId    : T_AmsNetId;
    sPathName : T_MaxString;
    ePath     : E_OpenPath := PATH_GENERIC; (* Default: generic file path*)
    bExecute  : BOOL;
    tTimeout  : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

名称	类型	描述
sNetId	T_AmsNetId	字符串，包含 ADS 命令所指向的目标设备的 AMS NetId（类型：T_AmsNetId [▶ 115]）。
sPathName	T_MaxString	新目录的名称。在调用功能块时，仅有创建一个新目录这一选项（类型：T_MaxString [▶ 118]）。
ePath	E_OpenPath	该输入可以用于选择目标设备上的新目录的 TwinCAT 系统路径（类型：E_OpenPath [▶ 112]）。
bExecute	BOOL	功能块由该输入端的上升沿激活。
tTimeout	TIME	规定执行 ADS 命令时不得超过的超时长度。

输出

```
VAR_OUTPUT
    bBusy   : BOOL;
    bError  : BOOL;
    nErrId  : UDINT;
END_VAR
```

名称	类型	描述
bBusy	BOOL	激活功能块后，将该输出设置为 TRUE，并保持设置状态，直至收到反馈。只要 bBusy 为 TRUE，就无法执行新命令。
bError	BOOL	如果在命令执行过程中出现错误，则在 bBusy 输出复位后设置该输出。
nErrId	UDINT	在设置 bError 输出时，返回 ADS 错误代码 [▶ 136]或特定命令的错误代码。

特定命令的错误代码	可能的原因
0x723	文件夹已存在或 sPathName 或 ePath 参数无效。

ST 语言示例：

通过 bCreate 的上升沿，在主目录 C:\ 中创建一个名为“PRJDATA”的新目录。通过 bRemove 的上升沿可以删除同名目录。

当 bBootFolder = TRUE 时，可以在 ..\TwinCAT\Boot 目录中创建或删除一个目录。

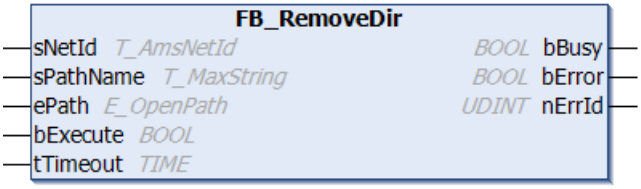
```
PROGRAM MAIN
VAR
    sFolderName   : STRING := 'PRJDATA'; (* folder name *)
    bBootFolder   : BOOL;

    ePath         : E_OpenPath; (* folders root path *)
    sPathName     : STRING;

    fbCreateDir   : FB_CreateDir;
    bCreate       : BOOL;
    bCreate_Busy  : BOOL;
    bCreate_Error : BOOL;
    nCreate_ErrID : UDINT;

    fbRemoveDir   : FB_RemoveDir;
    bRemove       : BOOL;
    bRemove_Busy  : BOOL;
    bRemove_Error : BOOL;
    nRemove_ErrID : UDINT;
END_VAR


ePath := SEL( bBootFolder, PATH_GENERIC, PATH_BOOTPATH );
sPathName := SEL( bBootFolder, CONCAT('C:\', sFolderName), sFolderName );

IF bCreate THEN
    bCreate := FALSE;
    fbCreateDir( bExecute := FALSE );
    fbCreateDir(sNetId:= '',
        sPathName:= sPathName,
        ePath:= ePath,
        bExecute:= TRUE,
        tTimeout:= DEFAULT_ADS_TIMEOUT,
        bBusy=>bCreate_Busy, bError=>bCreate_Error, nErrId=>nCreate_ErrID );
ELSE
    fbCreateDir( bExecute := FALSE, bBusy=>bCreate_Busy, bError=>bCreate_Error, nErrId=>nCreate_ErrI
```

```
D );
END_IF

IF bRemove THEN
    bRemove := FALSE;
    fbRemoveDir( bExecute := FALSE );
    fbRemoveDir(sNetId:= '',
        sPathName:= sPathName,
        ePath:= ePath,
        bExecute:= TRUE,
        tTimeout:= DEFAULT_ADS_TIMEOUT,
        bBusy=>bRemove_Busy, bError=>bRemove_Error, nErrId=>nRemove_ErrID );
ELSE
    fbRemoveDir( bExecute := FALSE, bBusy=>bRemove_Busy, bError=>bRemove_Error, nErrId=>nRemove_ErrI
D );
END_IF
```

前提条件

开发环境	目标平台	待集成的 PLC 库（类别组）
TwinCAT v3.1.0	PC 或 CX（x86、x64、Arm®）	Tc2_System（系统）

3.3.14 FB_RemoveDir



功能块 FB_RemoveDir 可用于从数据存储设备中删除一个目录。无法删除包含文件的目录。

该功能块仅适用于有限的范围内的实时记录。
为了获得良好的性能，我们建议使用 TF3500 TwinCAT Analytics Logger 产品，该产品需要付费。

输入

```
VAR_INPUT
    sNetId    : T_AmsNetId;
    sPathName : T_MaxString;
    ePath     : E_OpenPath := PATH_GENERIC; (* Default: generic file path*)
    bExecute  : BOOL;
    tTimeout  : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

名称	类型	描述
sNetId	T_AmsNetId	字符串，包含 ADS 命令所指向的目标设备的 AMS NetId（类型：T_AmsNetId [▶ 115]）。
sPathName	T_MaxString	要删除的目录名称。在调用该功能块时，只能删除一个目录。sPathName 的最后一个组件必须包含要删除的目录名称（类型：T_MaxString [▶ 118]）。
ePath	E_OpenPath	该输入可以用于选择在目标设备上删除目录的 TwinCAT 系统路径（类型：E_OpenPath [▶ 112]）。
bExecute	BOOL	功能块由该输入端的上升沿激活。
tTimeout	TIME	规定执行 ADS 命令时不得超过的超时长度。

输出

```
VAR_OUTPUT
    bBusy  : BOOL;
    bError : BOOL;
    nErrId : UDINT;
END_VAR
```

名称	类型	描述
bBusy	BOOL	激活功能块后，将该输出设置为 TRUE，并保持设置状态，直至收到反馈。只要 bBusy 为 TRUE，就无法创建新命令。
bError	BOOL	如果在命令执行过程中出现错误，则在 bBusy 输出复位后设置该输出。
nErrId	UDINT	在设置 bError 输出时，返回 ADS 错误代码 [▶ 136]或特定命令的错误代码。

特定命令的错误代码	可能的原因
0x70C	未找到文件夹或 sPathName 或 ePathh 参数无效。

ST 语言示例：

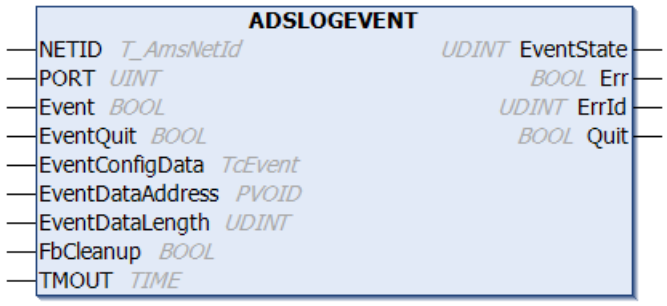
请参见 FB_CreateDir [▶ 60] 的说明。

前提条件

开发环境	目标平台	待集成的 PLC 库（类别组）
TwinCAT v3.1.0	PC 或 CX（x86、x64、Arm®）	Tc2_System（系统）

3.4 EventLogger 功能块

3.4.1 ADSLOGEVENT



该功能块允许向 TwinCAT EventLogger 发送和确认消息。

TwinCAT EventLogger 与 TwinCAT 3 EventLogger

TwinCAT EventLogger 已被其后续版本 TwinCAT 3 EventLogger 取代。TwinCAT 3 最高支持 3.1.4024 版本的旧版 TwinCAT EventLogger 。较新版本的 TwinCAT（>= 3.1.4026.0）仅支持较新的 TwinCAT 3 EventLogger。相关 PLC 功能块可在 PLC 库 Tc3_EventLogger 中找到。

输入

```
VAR_INPUT
    NETID             : T_AmsNetId;
    PORT              : T_AmsPort;
    Event             : BOOL;
    EventQuit         : BOOL;
    EventConfigData   : TcEvent;
    EventDataAddress  : PVOID;
    EventDataLength   : UDINT;
    FbCleanup         : BOOL;
    TMOUT             : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

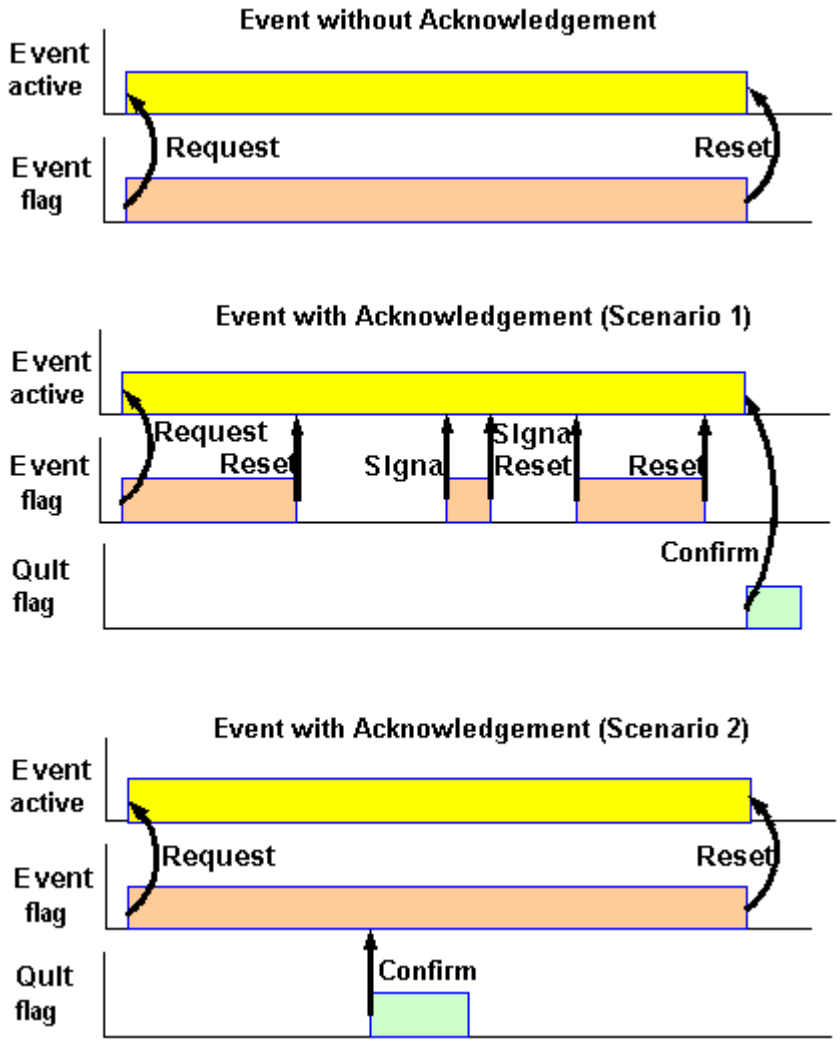
名称	类型	描述
NETID	T_AmsNetId	字符串，包含 ADS 命令所指向的目标设备的 AMS NetId（类型：T_AmsNetId [▶ 115]）。
PORT	T_AmsPort	ADS 设备的端口号。TwinCAT EventLogger 端口号为 110（类型：T_AmsPort [▶ 116]）。
Event	BOOL	事件的“传入”通过上升沿发出信号，事件的“传出”通过下降沿发出信号。
EventQuit	BOOL	通过上升沿确认事件。
EventConfig Data	TcEvent	包含事件参数的数据结构（类型：TcEvent [▶ 118]）。
EventData Address	PVOID	包含要与事件一起发送的数据的地址。
EventData Length	UDINT	要与事件一起发送的数据的长度。
FbCleanup	BOOL	如果为 TRUE，则功能块已完全初始化。
TMOUT	TIME	表示函数被取消前的时间。

输出

```
VAR_OUTPUT
    EventState : UDINT;
    Err        : BOOL;
    ErrId      : UDINT;
    Quit       : BOOL;
END_VAR
```

名称	类型	描述
EventState	UDINT	事件的状态。
Err	BOOL	一旦在执行命令过程中出错，该输出将切换为 TRUE。特定命令的错误代码包含在 ErrId 中。如果功能块出现超时错误，Err 为 TRUE，ErrId 为 1861（十六进制 0x745）。在输入端执行命令后重置为 FALSE。
ErrId	UDINT	ADS 错误代码 [▶ 136]或最后执行命令的特定命令错误代码。在输入端执行命令后重置为 0。
Quit	BOOL	确认事件。

确认消息



上图表示一般顺序。

对于无需确认的消息，通过功能块事件输入端的上升沿将发布事件，从而在 EventLogger 事件。事件输入端的下降沿启动重置。该信号会再次删除 EventLogger 事件。

对于需要确认的消息，通过事件输入端的上升沿将再次激活事件。通过两种方式取消激活事件

- 通过事件输入端的下降沿（如果先前 PLC 已通过 Quit 输入端或可视化界面发出确认信号）或
- 通过 Quit 输入端的上升沿（如果先前已通过事件输入端的下降沿启动重置）。

如果在事件激活和确认到达之间有一次事件重置，则事件输入的下一次到达被称为“信号”。因此，在已激活事件的情况下，将会发布请求。

在 ST 中使用 ADSLOGEVENT 功能块的示例：

- 从 PLC 发送/确认 EventLogger [▶ 130]

要求

开发环境	目标平台	待集成的 PLC 库（类别组）
TwinCAT v3.1.0 最高 TwinCAT v3.1.4024	PC 或 CX（x86、x64、ARM）	Tc2_System（系统）

3.4.2 ADSCLEAREVENTS

该功能块向 TwinCAT EventLogger 和确认消息。

> **TwinCAT EventLogger 与 TwinCAT 3 EventLogger**
>
> TwinCAT EventLogger 已被其后续版本 TwinCAT 3 EventLogger 取代。TwinCAT 3 最高支持 3.1.4024 版本的旧版 TwinCAT EventLogger 。较新版本的 TwinCAT（>= 3.1.4026.0）仅支持较新的 TwinCAT 3 EventLogger。相关 PLC 功能块可在 PLC 库 Tc3_EventLogger 中找到。

输入

```
VAR_INPUT
    NETID    : T_AmsNetId;
    bClear   : BOOL;
    iMode    : UDINT;
    tTimeout : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

名称	类型	描述
NETID	T_AmsNetId	字符串，包含 ADS 命令所指向的目标设备的 AMS NetId（类型：T_AmsNetId [▶ 115]）。
bClear	BOOL	通过上升沿删除事件。
iMode	UDINT	删除事件的模式。在枚举 E_TcEventClearModes [▶ 113] 中进行定义。
tTimeout	TIME	表示函数被取消前的时间。

输出

```
VAR_OUTPUT
    bBusy  : BOOL;
    bErr   : BOOL;
    iErrId : UDINT;
END_VAR
```

名称	类型	描述
bBusy	BOOL	激活功能块后，将该输出设置为 TRUE，并保持设置状态，直至收到反馈。只要 bBusy 为 TRUE，就无法执行新命令。
bErr	BOOL	一旦在执行命令过程中出错，该输出将切换为 TRUE。特定命令的错误代码包含在 iErrId 中。如果功能块出现超时错误，bErr 为 TRUE，iErrId 为 1861（十六进制 0x745）。在输入端执行命令后重置为 FALSE。
iErrId	UDINT	ADS 错误代码 [▶ 136]或最后执行命令的特定命令错误代码。在输入端执行命令后重置为 0。

要求

开发环境	目标平台	待集成的 PLC 库（类别组）
TwinCAT v3.1.0 最高 TwinCAT v3.1.4024	PC 或 CX（x86、x64、ARM）	Tc2_System（系统）

3.4.3 FB_SimpleAdsLogEvent

该功能块允许向 TwinCAT EventLogger 发送和确认消息。与 ADSLOGEVENT 程序块不同，通过 FB_SimpleAdsLogEvent 程序块不能从 PLC 对事件进行参数设置；但可以通过简单的方式对事件进行设置、重置和确认。

TwinCAT EventLogger 与 TwinCAT 3 EventLogger

TwinCAT EventLogger 已被其后续版本 TwinCAT 3 EventLogger 取代。TwinCAT 3 最高支持 3.1.4024 版本的旧版 TwinCAT EventLogger 。较新版本的 TwinCAT（>= 3.1.4026.0）仅支持较新的 TwinCAT 3 EventLogger。相关 PLC 功能块可在 PLC 库 Tc3_EventLogger 中找到。

输入

```
VAR_INPUT
    SourceId    : INT;
    EventId     : INT;
    bSetEvent   : BOOL;
    bQuit       : BOOL;
END_VAR
```

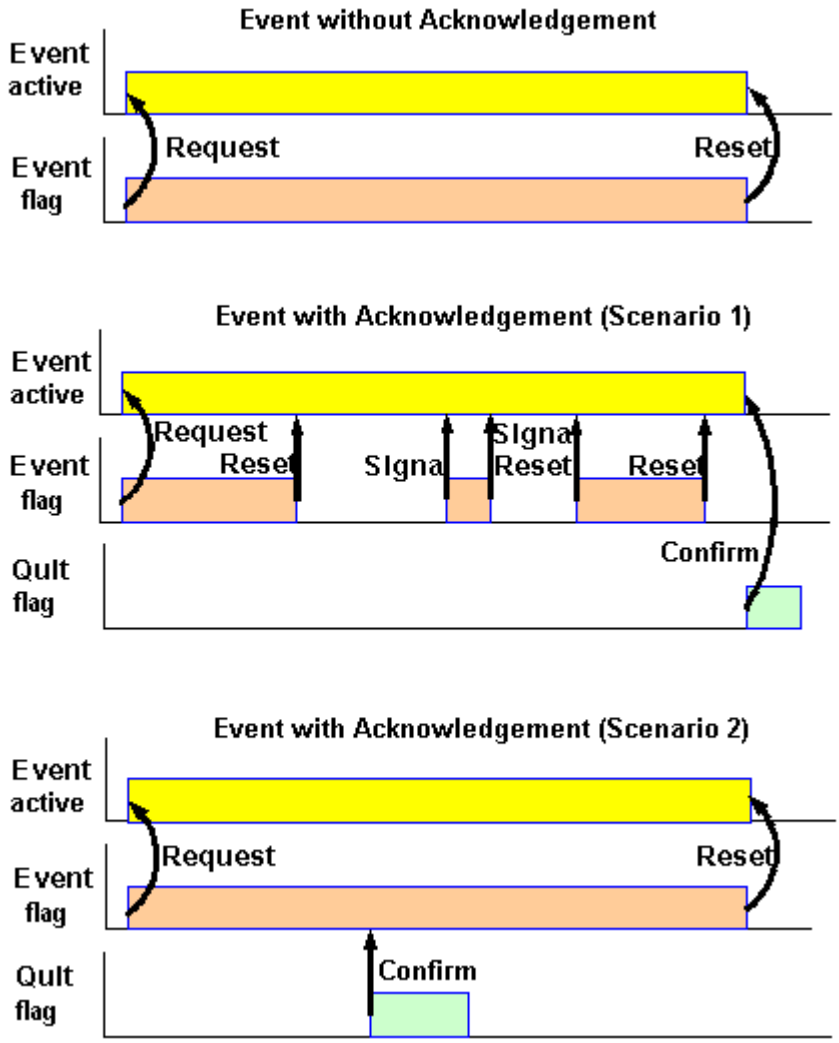
名称	类型	描述
SourceId	INT	源 ID。用于在 EventLogger 确标识源。
EventId	INT	事件 ID。用于在 EventLogger 确标识事件。
bSetEvent	BOOL	事件的“传入”通过上升沿发出信号，事件的“传出”通过下降沿发出信号。
bQuit	BOOL	通过上升沿确认事件。

输出

```
VAR_OUTPUT
    ErrId       : UDINT;
    Error       : BOOL;
END_VAR
```

名称	类型	描述
ErrId	UDINT	ADS 错误代码 [▶ 136]或最后执行命令的特定命令错误代码。在输入端执行命令后重置为 0。
错误	BOOL	一旦在执行命令过程中出错，该输出将切换为 TRUE。特定命令的错误代码包含在 ErrId 中。如果功能块出现超时错误，Error 为 TRUE，ErrId 为 1861（十六进制 0x745）。在输入端执行命令后重置为 FALSE。

确认消息



上图表示一般顺序。

对于无需确认的消息，通过功能块事件输入端的上升沿将发布事件，从而在 EventLogger 事件。事件输入端的下降沿启动重置。该信号会再次删除 EventLogger 事件。

对于需要确认的消息，通过事件输入端的上升沿将再次激活事件。通过两种方式取消激活事件

- 通过事件输入端的下降沿（如果先前 PLC 已通过 Quit 输入端或可视化界面发出确认信号）或
- 通过 Quit 输入端的上升沿（如果先前已通过事件输入端的下降沿启动重置）。

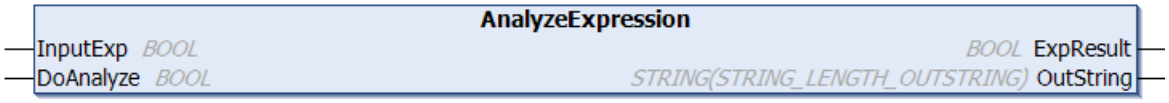
如果在事件激活和确认到达之间有一次事件重置，则事件输入的下一次到达被称为“信号”。因此，在已激活事件的情况下，将会发布请求。

要求

开发环境	目标平台	待集成的 PLC 库（类别组）
TwinCAT v3.1.0 最高 TwinCAT v3.1.4024	PC 或 CX（x86、x64、ARM）	Tc2_System（系统）

3.5 IEC 步骤/SFC 标志位功能块

3.5.1 AnalyzeExpression



该功能块可以用于使用 SFC 标志的 PLC 项目。不会生成任何实例。在项目中必须集成相应的 PLC 库。有关更多配置要求，请参见以下解释性说明。

AnalyzeExpression 和 AnalyzeExpressionTable 功能块可以用于分析转换和步进允许条件。如果在设定时间后没有触发转换，则可使用这些功能块对转换进行分析。

功能块只能用于分析以 ST 编程语言实现的表达式或转换。

- AnalyzeExpression：
  - 该功能块以 STRING 形式捆绑输出分析结果，即未发生切换的原因（即哪些局部条件未满足）。构成局部条件的变量通过运算符（例如 `bVar1 AND (bVar2 OR bVar3)`）相互关联。
  - SFC 标志“SFCErrorAnalyzation”可用于输出。
- AnalyzeExpressionTable：
  - 该功能块逐个输出所有未切换的变量。独立变量作为数组元素记录或输出。每个数组元素列出的信息包括变量的名称、地址、注释和当前值。
  - SFC 标志“SFCErrorAnalyzationTable”可用于输出。

配置要求

为 SFC 启用 AnalyzeExpression 或 AnalyzeExpressionTable 需要以下设置：

- 包含 PLC 库 Tc2_System。
- 在 SFC-POU 中声明以下变量：
  `SFCEnableLimit: BOOL := TRUE;`
- 在属性窗口中，为 SFC 图中需要分析其后续转换/切换条件的步骤配置最长有效时间（另请参见 SFC 元素属性）。
- 在 PLC 项目属性或 POU 属性中配置 SFC 设置（另请参见 SFC 标志和命令属性（PLC 项目） > 类别 SFC）：
  - **标志**选项卡：
    选中以下 SFC 标志的**激活**和**声明**复选框：
    SFCError、SFCEnableLimit、SFCErrorAnalyzation、SFCErrorAnalyzationTable
  - **构建**选项卡：
    启用**仅计算活动转换**选项。

示例

在以下示例中已实现上述配置。对于“Step1”，最长有效时间被设置为 1 s。如果相关的传出转换“Trans_ST”在 1 s 后仍未触发，则通过功能块 AnalyzeExpression 和 AnalyzeExpressionTable 对该转换进行分析。变量 SFCError 被设置为 TRUE，变量 SFCErrorStep 的值为“Step1”。

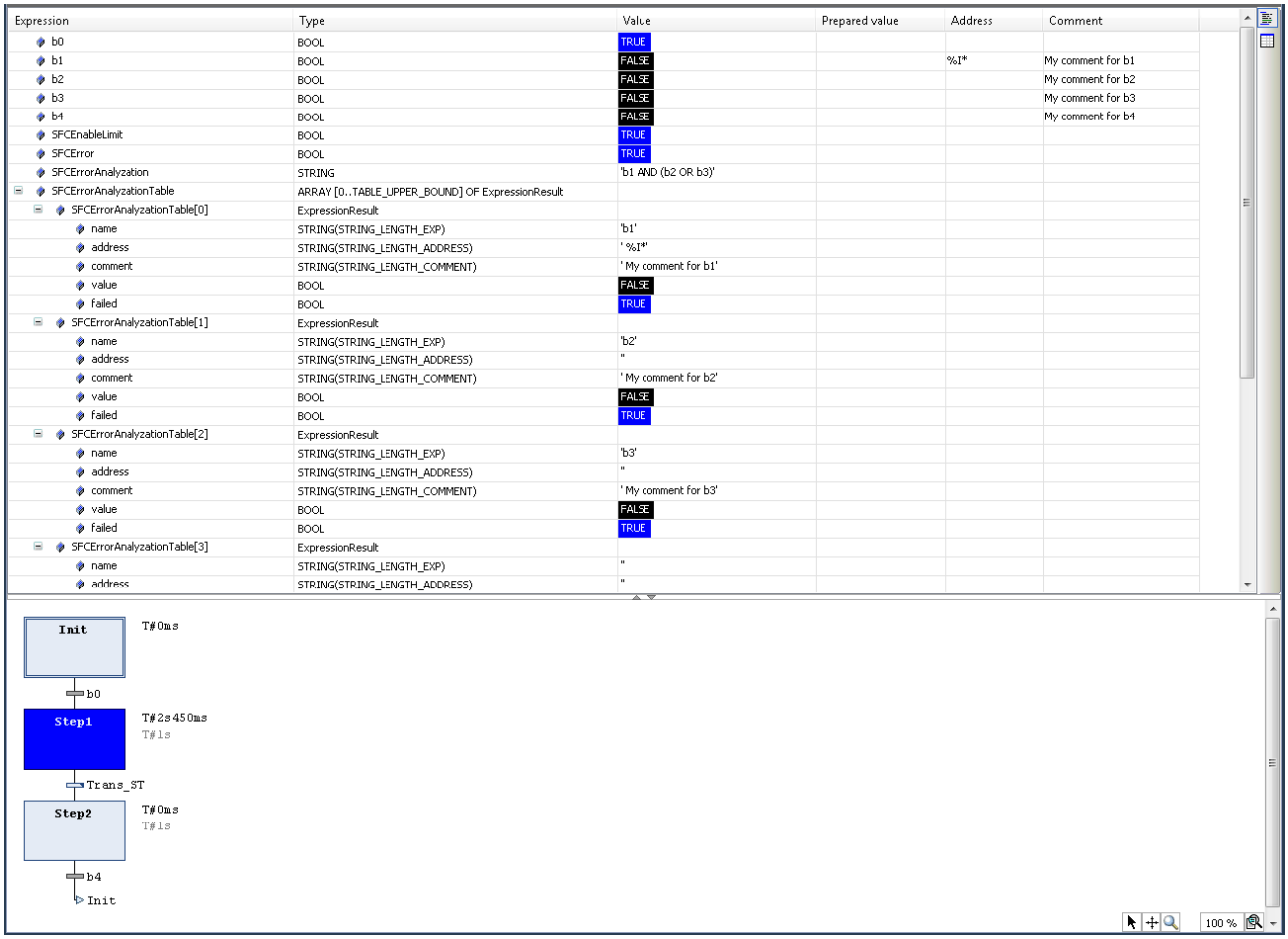
分析结果“SFCErrorAnalyzation”或“SFCErrorAnalyzationTable”指明哪些（局部）表达式尚未触发，从而允许退出“Step1”。

转换“Trans_ST”由表达式组成

```
b1 AND (b2 OR b3);
```

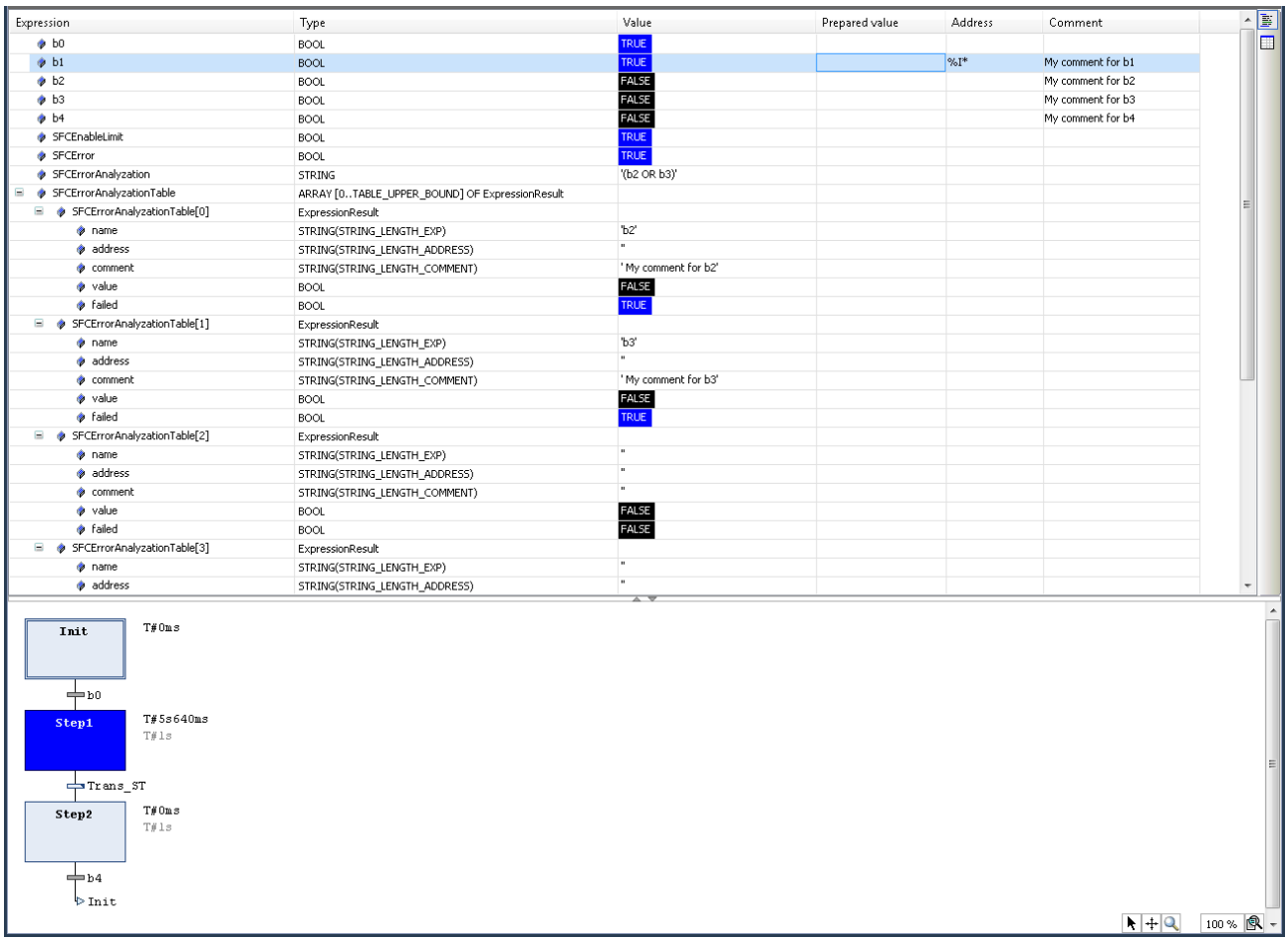
- 情况 1：三个变量 b1、b2、b3 均不为 TRUE。
  - “SFCErrorAnalyzation”显示分析结果“b1 AND (b2 OR b3)”。
  - “SFCErrorAnalyzationTable”列出了所有三个变量 b1、b2、b3 以及详细的变量信息。
  - 另请参见图 1。
- 情况 2：变量 b1 被设置为 TRUE。分析结果会相应改变。
  - “SFCErrorAnalyzation”显示分析结果“(b2 OR b3)”。
  - “SFCErrorAnalyzationTable”只列出了两个变量 b2 和 b3 以及相应的变量信息。
  - 另请参见图 2。

图 1：



Expression	Type	Value	Prepared value	Address	Comment
b0	BOOL	TRUE			
b1	BOOL	FALSE		%I*	My comment for b1
b2	BOOL	FALSE			My comment for b2
b3	BOOL	FALSE			My comment for b3
b4	BOOL	FALSE			My comment for b4
SFCEnableLimit	BOOL	TRUE			
SFCError	BOOL	TRUE			
SFCErrorAnalyzation	STRING	'b1 AND (b2 OR b3)'			
SFCErrorAnalyzationTable	ARRAY [0..TABLE_UPPER_BOUND] OF ExpressionResult				
SFCErrorAnalyzationTable[0]	ExpressionResult				
name	STRING(STRING_LENGTH_EXP)	'b1'			
address	STRING(STRING_LENGTH_ADDRESS)	'%I*'			
comment	STRING(STRING_LENGTH_COMMENT)	' My comment for b1'			
value	BOOL	FALSE			
failed	BOOL	TRUE			
SFCErrorAnalyzationTable[1]	ExpressionResult				
name	STRING(STRING_LENGTH_EXP)	'b2'			
address	STRING(STRING_LENGTH_ADDRESS)	''			
comment	STRING(STRING_LENGTH_COMMENT)	' My comment for b2'			
value	BOOL	FALSE			
failed	BOOL	TRUE			
SFCErrorAnalyzationTable[2]	ExpressionResult				
name	STRING(STRING_LENGTH_EXP)	'b3'			
address	STRING(STRING_LENGTH_ADDRESS)	''			
comment	STRING(STRING_LENGTH_COMMENT)	' My comment for b3'			
value	BOOL	FALSE			
failed	BOOL	TRUE			
SFCErrorAnalyzationTable[3]	ExpressionResult				
name	STRING(STRING_LENGTH_EXP)	''			
address	STRING(STRING_LENGTH_ADDRESS)	''			

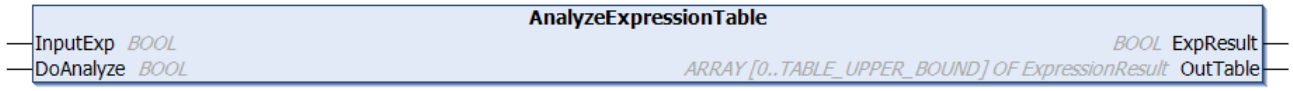
图 2：



前提条件

开发环境	目标平台	待集成的 PLC 库（类别组）
TwinCAT v3.1.0	PC 或 CX（x86、x64、Arm®）	Tc2_System（系统）

3.5.2 AnalyzeExpressionTable



该功能块可以用于使用 SFC 标志的 PLC 项目。不会生成任何实例。在项目中必须集成相应的 PLC 库。有关更多配置要求，请参见以下解释性说明。

AnalyzeExpression 和 AnalyzeExpressionTable 功能块可以用于分析转换和步进允许条件。如果在设定时间后没有触发转换，则可使用这些功能块对转换进行分析。

功能块只能用于分析以 ST 编程语言实现的表达式或转换。

- AnalyzeExpression：
  - 该功能块以 STRING 形式捆绑输出分析结果，即未发生切换的原因（即哪些局部条件未满足）。构成局部条件的变量通过运算符（例如 `bVar1 AND (bVar2 OR bVar3)`）相互关联。
  - SFC 标志“SFCErrorAnalyzation”可用于输出。
- AnalyzeExpressionTable：
  - 该功能块逐个输出所有未切换的变量。独立变量作为数组元素记录或输出。每个数组元素列出的信息包括变量的名称、地址、注释和当前值。

- SFC 标志“SFCErrorAnalyzationTable”可用于输出。

配置要求

为 SFC 启用 AnalyzeExpression 或 AnalyzeExpressionTable 需要以下设置：

- 包含 PLC 库 Tc2_System。
- 在 SFC-POU 中声明以下变量：
  ```
  SFCEnableLimit: BOOL := TRUE;
  ```
- 在属性窗口中，为 SFC 图中需要分析其后续转换/切换条件的步骤配置最长有效时间（另请参见 SFC 元素属性）。
- 在 PLC 项目属性或 POU 属性中配置 SFC 设置（另请参见 SFC 标志和命令属性（PLC 项目） > 类别 SFC）：
  - **标志**选项卡：
    选中以下 SFC 标志的**激活**和**声明**复选框：
    SFCError、SFCEnableLimit、SFCErrorAnalyzation、SFCErrorAnalyzationTable
  - **构建**选项卡：
    启用**仅计算活动转换**选项。

示例

在以下示例中已实现上述配置。对于“Step1”，最长有效时间被设置为 1 s。如果相关的传出转换“Trans_ST”在 1 s 后仍未触发，则通过功能块 AnalyzeExpression 和 AnalyzeExpressionTable 对该转换进行分析。变量 SFCError 被设置为 TRUE，变量 SFCErrorStep 的值为“Step1”。

分析结果“SFCErrorAnalyzation”或“SFCErrorAnalyzationTable”指明哪些（局部）表达式尚未触发，从而允许退出“Step1”。

转换“Trans_ST”由表达式组成

```
b1 AND (b2 OR b3);
```

- 情况 1：三个变量 b1、b2、b3 均不为 TRUE。
  - “SFCErrorAnalyzation”显示分析结果“b1 AND (b2 OR b3)”。
  - “SFCErrorAnalyzationTable”列出了所有三个变量 b1、b2、b3 以及详细的变量信息。
  - 另请参见图 1。
- 情况 2：变量 b1 被设置为 TRUE。分析结果会相应改变。
  - “SFCErrorAnalyzation”显示分析结果“(b2 OR b3)”。
  - “SFCErrorAnalyzationTable”只列出了两个变量 b2 和 b3 以及相应的变量信息。
  - 另请参见图 2。

图 1：

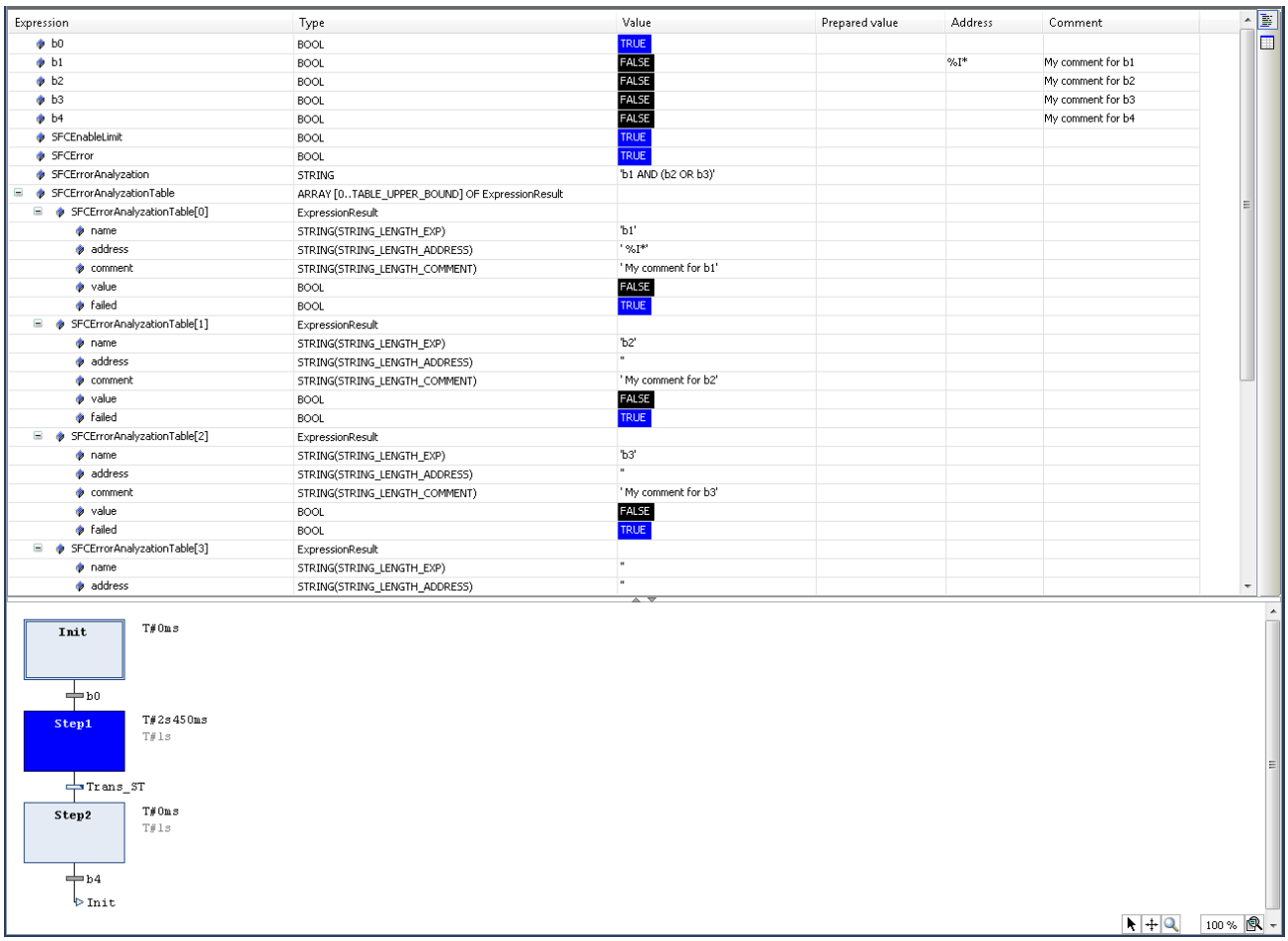
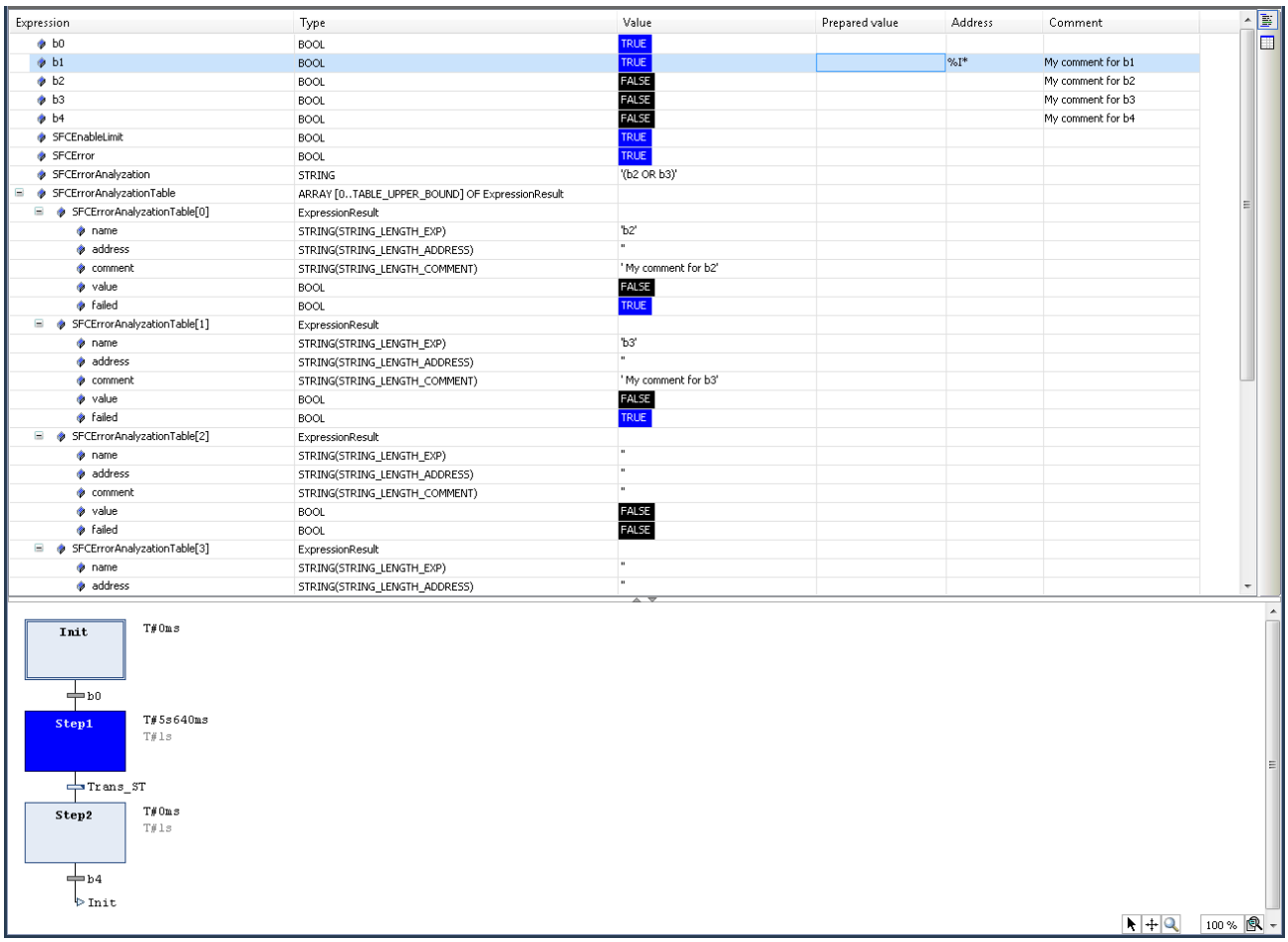

Expression
Type
Value
Prepared value
Address
Comment
b0 BOOL TRUE
b1 BOOL FALSE %I* My comment for b1
b2 BOOL FALSE My comment for b2
b3 BOOL FALSE My comment for b3
b4 BOOL FALSE My comment for b4
SFCEnableLimit BOOL TRUE
SFCError BOOL TRUE
SFCErrorAnalyzation STRING 'b1 AND (b2 OR b3)'
SFCErrorAnalyzationTable ARRAY [0..TABLE_UPPER_BOUND] OF ExpressionResult
SFCErrorAnalyzationTable[0] ExpressionResult
name STRING(STRING_LENGTH_EXP) 'b1'
address STRING(STRING_LENGTH_ADDRESS) '%I*'
comment STRING(STRING_LENGTH_COMMENT) ' My comment for b1'
value BOOL FALSE
failed BOOL TRUE
SFCErrorAnalyzationTable[1] ExpressionResult
name STRING(STRING_LENGTH_EXP) 'b2'
address STRING(STRING_LENGTH_ADDRESS) ''
comment STRING(STRING_LENGTH_COMMENT) ' My comment for b2'
value BOOL FALSE
failed BOOL TRUE
SFCErrorAnalyzationTable[2] ExpressionResult
name STRING(STRING_LENGTH_EXP) 'b3'
address STRING(STRING_LENGTH_ADDRESS) ''
comment STRING(STRING_LENGTH_COMMENT) ' My comment for b3'
value BOOL FALSE
failed BOOL TRUE
SFCErrorAnalyzationTable[3] ExpressionResult
name STRING(STRING_LENGTH_EXP) ''
address STRING(STRING_LENGTH_ADDRESS) ''
Init
T#0ms
b0
Step1
T#2s450ms
T#1s
Trans_ST
Step2
T#0ms
T#1s
b4
Init
100 %

图 2:


Expression
Type
Value
Prepared value
Address
Comment
b0 BOOL TRUE
b1 BOOL TRUE %I* My comment for b1
b2 BOOL FALSE My comment for b2
b3 BOOL FALSE My comment for b3
b4 BOOL FALSE My comment for b4
SFCEnableLimit BOOL TRUE
SFCError BOOL TRUE
SFCErrorAnalyzation STRING '(b2 OR b3)'
SFCErrorAnalyzationTable ARRAY [0..TABLE_UPPER_BOUND] OF ExpressionResult
SFCErrorAnalyzationTable[0] ExpressionResult
name STRING(STRING_LENGTH_EXP) 'b2'
address STRING(STRING_LENGTH_ADDRESS) ''
comment STRING(STRING_LENGTH_COMMENT) ' My comment for b2'
value BOOL FALSE
failed BOOL TRUE
SFCErrorAnalyzationTable[1] ExpressionResult
name STRING(STRING_LENGTH_EXP) 'b3'
address STRING(STRING_LENGTH_ADDRESS) ''
comment STRING(STRING_LENGTH_COMMENT) ' My comment for b3'
value BOOL FALSE
failed BOOL TRUE
SFCErrorAnalyzationTable[2] ExpressionResult
name STRING(STRING_LENGTH_EXP) ''
address STRING(STRING_LENGTH_ADDRESS) ''
comment STRING(STRING_LENGTH_COMMENT) ''
value BOOL FALSE
failed BOOL FALSE
SFCErrorAnalyzationTable[3] ExpressionResult
name STRING(STRING_LENGTH_EXP) ''
address STRING(STRING_LENGTH_ADDRESS) ''
Init
T#0ms
b0
Step1
T#5s640ms
T#1s
Trans_ST
Step2
T#0ms
T#1s
b4
Init
100 %

前提条件

开发环境	目标平台	待集成的 PLC 库（类别组）
TwinCAT v3.1.0	PC 或 CX（x86、x64、Arm®）	Tc2_System（系统）

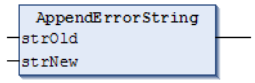
3.5.3 AnalyzeExpressionCombined

在使用 sfc 标志的 PLC 项目中需要使用该功能块。不会创建任何实例。项目中必须包含相应的 PC 库。

前提条件

开发环境	目标平台	待集成的 PLC 库（类别组）
TwinCAT v3.1.0	PC 或 CX（x86、x64、Arm®）	Tc2_System（系统）

3.5.4 AppendErrorString

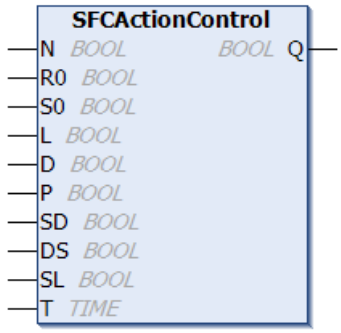


使用 SFC 标志的 PLC 项目需要该函数。在项目中不得调用该函数。项目中仅需包含相应的 PLC 库。

前提条件

开发环境	目标平台	待集成的 PLC 库（类别组）
TwinCAT v3.1.0	PC 或 CX（x86、x64、Arm®）	Tc2_System（系统）

3.5.5 SFCActionControl



在 SFC 程序/项目中使用 IEC 步骤时需要该函数。项目中仅需包含带有 FB 的库，不需要任何实例。

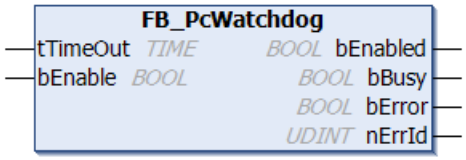
前提条件

开发环境	目标平台	待集成的 PLC 库（类别组）
TwinCAT v3.1.0	PC 或 CX（x86、x64、Arm®）	Tc2_System（系统）

3.6 看门狗功能块

3.6.1 FB_PcWatchdog

该功能仅适用于配备以下主板的 IPC：IP-4GVI63、CB1050、CB2050、CB3050、CB1051、CB2051、CB3051。



功能块 FB_PcWatchdog 在 PC 上激活硬件看门狗。通过 bEnable = TRUE 和超时时间激活看门狗。超时时间范围为 1 至 255 秒。通过 bEnable = TRUE 和 tTimeOut >= 1 s 激活看门狗。

一旦看门狗被激活，该功能块必须循环调用，并且间隔时间要比 tTimeOut 短，因为当超过 tTimeOut 时，PC 会自动重新启动。因此，看门狗可以用于自动重启已进入无限循环的系统或 PLC 已经卡住的系统。

通过 bEnable = FALSE 或 tTimeOut = 0 可以停用看门狗。

注意

PC 重启

在使用断点、PLC 重置或整体重置、TwinCAT 停止、切换到配置模式或者激活配置之前，必须停用看门狗，否则一旦超时，PC 就会立即重新启动。

输入

```
VAR_INPUT
    tTimeOut : TIME;
    bEnable  : BOOL;
END_VAR
```

名称	类型	描述
tTimeOut	TIME	看门狗时间，之后将重新启动。
bEnable	BOOL	启用/禁用看门狗。

输出

```
VAR_OUTPUT
    bEnabled : BOOL;
    bBusy    : BOOL;
    bError   : BOOL;
    nErrId   : UDINT;
END_VAR
```

名称	类型	描述
bEnabled	BOOL	TRUE = 看门狗已激活，FALSE = 看门狗已停用
bBusy	BOOL	在功能块执行命令之前，该输出保持为 TRUE。
bError	BOOL	一旦在执行命令过程中出错，该输出将切换为 TRUE。特定命令的错误代码包含在 nErrId 中。在输入端执行命令后重置为 FALSE。
nErrId	UDINT	ADS 错误代码 [▶ 136]或最后执行命令的特定命令错误代码。在输入端执行命令后重置为 0。

在 ST 中调用功能块的示例：

```
PROGRAM MAIN
VAR
    fbPcWatchDog  : FB_PcWatchdog;
    tWDTime       : TIME := T#10s;
```

```
    bEnableWD      : BOOL;
    bWDActive      : BOOL;
END_VAR

IF bEnableWD OR bWDActive THEN
    fbPcWatchDog(tTimeOut := tWDTime, bEnable := bEnableWD);
    bWDActive := fbPcWatchDog.bEnabled;
END_IF
```

要求

开发环境	目标平台	待集成的 PLC 库（类别组）
TwinCAT v3.1.0	配备以下主板的 IPC：IP 4GVI63、CB1050、CB2050、CB3050、CB1051、CB2051、CB3051	PLC 库 Tc2_System

3.6.2 FB_PcWatchDog_BAPI

该功能仅适用于 BIOS 版本支持 BIOS-API 的 IPC 和嵌入式控制器。

功能块 FB_PcWatchdog_BAPI 在 PC 上激活硬件看门狗。通过 bExecute = TRUE 和看门狗时间激活看门狗。看门狗时间范围为 1 至 15300 秒（255 分钟）。通过 bEnable = TRUE 和 nWatchdogTimeS >=1 s 激活看门狗。

一旦看门狗被激活，该功能块必须循环调用，并且间隔时间要比 nWatchdogTimeS 短，因为当超过 nWatchdogTimeS 时，PC 会自动重新启动。因此，看门狗可以用于自动重启已进入无限循环的系统或 PLC 已经卡住的系统。

注意

PC 重启

在使用断点、PLC 重置或整体重置、TwinCAT 停止、切换到配置模式或者激活配置之前，必须停用看门狗，否则一旦超过 nWatchdogTimeS，PC 就会立即重新启动。

输入

```
VAR_INPUT
    sNetID         : T_AmsNetID;
    nWatchdogTimeS : UDINT;
    bExecute       : BOOL;
    tTimeout       : TIME;
END_VAR
```

名称	类型	描述
sNetID	T_AmsNetID	设备的 AMS NetId（空字符串或本地网络 ID）
nWatchdogTimeS	UDINT	看门狗时间（以秒为单位），0 = 停用，>0（最大 15300）= 激活。
bExecute	BOOL	上升沿触发命令执行。一旦功能块不再处于活动状态，必须立即复位输入端 (bBusy=FALSE)。
tTimeout	TIME	指明 ADS 内部通信终止前的时间。

输出

```
VAR_OUTPUT
    bEnabled : BOOL;
    bBusy    : BOOL;
```

```
    bError   : BOOL;
    nErrId   : UDINT;
END_VAR
```

名称	类型	描述
bEnabled	BOOL	TRUE = 看门狗已激活，FALSE = 看门狗已停用
bBusy	BOOL	在功能块执行命令之前，该输出保持为 TRUE。
bError	BOOL	一旦在执行命令过程中出错，该输出将切换为 TRUE。特定命令的错误代码包含在 nErrId 中。在输入端执行命令后重置为 FALSE。
nErrId	UDINT	ADS 错误代码 [▶ 136]或最后执行命令的特定命令错误代码。在输入端执行命令后重置为 0。

在 ST 中调用功能块的示例：

```
PROGRAM MAIN
VAR
    fbWatchdog     : FB_PcWatchdog_BAPI;
    nWatchdogTimeS : UDINT := 10;  (* 10s *)
    bEnabled       : BOOL; (* TRUE: watchdog is activated *)
    bError         : BOOL;
    nErrID         : UDINT;
    fbTimer        : TON := (IN := TRUE, PT := T#0S);
END_VAR
```

```
fbTimer();

(* 1st enable, then refresh watchdog every 1s *)
IF fbTimer.Q THEN
    fbWatchdog(
        sNetID         := '',
        nWatchdogTimeS := nWatchdogTimeS,
        bExecute       := TRUE,
        tTimeout       := T#5S,
    );

    IF NOT fbWatchdog.bBusy THEN
        bEnabled   := fbWatchdog.bEnabled;
        bError     := fbWatchdog.bError;
        nErrID     := fbWatchdog.nErrID;

        fbWatchdog(bExecute := FALSE);

        (* restart timer*)
        fbTimer(IN := FALSE);
        fbTimer(IN := TRUE, PT := T#1S); (* refresh watchdog every s *)
    END_IF
END_IF
```

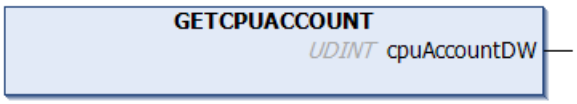
要求

开发环境	目标平台	待集成的 PLC 库（类别组）
TwinCAT v3.1.0	BIOS 版本支持 BIOS-API 的 IPC 和嵌入式控制器。	PLC 库 Tc2_System-版本 >=3.4.14.0

3.7 时间功能块

3.7.1 GETCPUACCOUNT

在 Windows CE 下的 PLC Runtime 系统中不提供该功能。



通过该功能块可以读取 PLC 任务周期计数器。PLC 任务周期计数器仅在执行任务期间递增。该数值是一个 32 位整数，其输出形式与 CPU 内部时钟频率无关，将被转换为 100 ns 时间刻度。每次调用 PLC 任务时，该数值都会刷新，精度为 100 ns，可用于定时等目的。一个单位相当于 100 ns。

输入

```
VAR_INPUT
(*none*)
END_VAR
```

输出

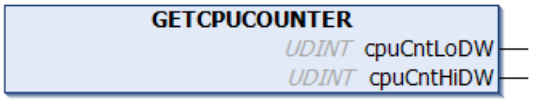
```
VAR_OUTPUT
    cpuAccountDW : UDINT;
END_VAR
```

名称	类型	描述
cpuAccountDW	UDINT	PLC 任务计数器的当前值

要求

开发环境	目标系统类型	待集成的 PLC 库
TwinCAT v3.1.0	PC 或 CX（x86、x64）	Tc2_System（系统）

3.7.2 GETCPUCOUNTER



通过该功能块可以读取 CPU 周期计数器。该数值是一个 64 位相对整数，其输出形式与 CPU 内部时钟频率无关，均被转换为 100 ns 时间刻度。PLC 系统每次调用时，该数字都会刷新，精度为 100 ns，可用于定时任务等目的。一个单位相当于 100 ns。这项服务以功能块而非以函数的形式实现，其原因很简单，因为必须返回两个值，而根据定义，函数无法做到这一点。

输入

```
VAR_INPUT
(*none*)
END_VAR
```

输出

```
VAR_OUTPUT
    cpuCntLoDW : UDINT;
    cpuCntHiDW : UDINT;
END_VAR
```

名称	类型	描述
cpuCntLoDW	UDINT	计数值的最低有效 4 字节
cpuCntHiDW	UDINT	计数值的最高有效 4 字节

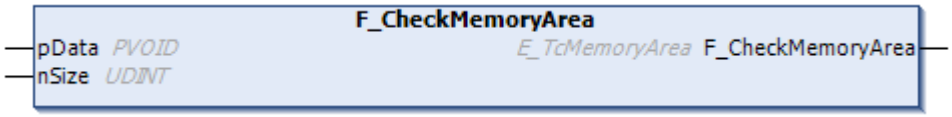
前提条件

开发环境	目标平台	待集成的 PLC 库（类别组）
TwinCAT v3.1.0	PC 或 CX（x86、x64、Arm®）	Tc2_System（系统）

4 函数

4.1 通用函数

4.1.1 F_CheckMemoryArea



该函数返回有关所请求的指定大小的变量所在内存区域的信息。为此，将使用 E_TcMemoryArea [▶ 114] 类型的返回值。

函数 F_CheckMemoryArea：E_TcMemoryArea

输入

```
VAR_INPUT
    pData : PVOID;
    nSize : UDINT;
END_VAR
```

名称	类型	描述
pData	PVOID	变量的内存地址
nSize	UDINT	变量的大小，以字节为单位

示例

```
PROGRAM MAIN
VAR
    nCounter : USINT;
    eMemAreaStatic : E_TcMemoryArea;
    pDynamicVariable : POINTER TO LREAL;
    eMemAreaDynamic : E_TcMemoryArea;
    pNull : PVOID := 0;
    eMemAreaUnknown : E_TcMemoryArea;
END_VAR
-------------------------------------------------------------------

nCounter := nCounter + 1;
eMemAreaStatic := F_CheckMemoryArea( pData:=ADR(nCounter), nSize:=SIZEOF(nCounter) );

IF nCounter = 100 THEN
    pDynamicVariable := __NEW(LREAL);
    IF pDynamicVariable <> 0 THEN
        pDynamicVariable^ := 7 * 4.5;
        eMemAreaDynamic := F_CheckMemoryArea( pData:=pDynamicVariable, nSize:=SIZEOF(LREAL) );
        __DELETE(pDynamicVariable);
    END_IF
END_IF

eMemAreaUnknown := F_CheckMemoryArea( pData:=pNull, nSize:=1 );
```

开发环境	目标平台	待集成的 PLC 库（类别组）
TwinCAT v3.1.4022	PC 或 CX（x86、x64、Arm®）	Tc2_System（系统）

4.1.2 F_CmpLibVersion

函数 F_CmpLibVersion 用于比较现有库的版本与所需版本。每个库均拥有自身版本信息，其常量类型：ST_LibVersion。常量名称的格式：stLibVersion_libraryname。

函数 F_CmpLibVersion：DINT

输入

```
VAR_INPUT
    stVersion   : ST_LibVersion;
    iMajor      : UINT;
    iMinor      : UINT;
    iBuild      : UINT;
    iRevision   : UINT;
END_VAR
```

名称	类型	描述
stVersion	ST_LibVersion	现有库的版本（类型：ST_LibVersion）
iMajor	UINT	所需的主版本号
iMinor	UINT	所需的次版本号
iBuild	UINT	所需的构建版本号
iRevision	UINT	所需的修订版本号

返回参数	版本关系
-1	您的版本低于所需的版本。
0	您的版本为所需的版本。
+1	您的版本高于所需的版本。

ST 中的示例：

```
IF F_CmpLibVersion( stLibVersion_Tc2_System, 3, 3, 8, 0) >= 0 THEN
    (* newer lib ...*)
ELSE
    (* older lib... *)
END_IF
```

前提条件

开发环境	目标平台	待集成的 PLC 库（类别组）
TwinCAT v3.1.0	PC 或 CX（x86、x64、Arm®）	Tc2_System（系统）

4.1.3 F_CreateIPv4Addr

该函数生成格式化的（IPv4）互联网协议网络地址，并将其作为字符串类型的返回参数（例如 172.16.7.199）返回。

函数 F_CreateIPv4Addr：T_IPv4Addr

输入

```
VAR_INPUT
    nIds : T_IPv4AddrArr;
END_VAR
```

名称	类型	描述
nIds	T_IPv4AddrArr	字节数组：每个字节与（IPv4）互联网协议网络地址的一个地址字节相对应。地址字节采用网络字节序（类型：T_IPv4AddrArr [▶ 117]）。

结构化文本示例：

```
PROGRAM MAIN
VAR
    ids   : T_IPv4AddrArr := 172, 16, 7, 199;
    sIPv4 : T_IPv4Addr := '';
END_VAR

sIPv4 := F_CreateIPv4Addr( ids );(* Result: '172.16.7.199' *)
```

前提条件

开发环境	目标平台	待集成的 PLC 库（类别组）
TwinCAT v3.1.0	PC 或 CX（x86、x64、Arm®）	Tc2_System（系统）

4.1.4 F_ScanIPv4AddrIds

函数 F_ScanIPv4AddrIds 将（IPv4）互联网协议网络地址的字符串转换为单个地址字节。单个地址字节自左向右转换。它们以字节数组的形式返回。地址字节采用网络字节序。

函数 F_ScanIPv4AddrIds：T_IPv4AddrArr

输入

```
VAR_INPUT
    sIPv4 : T_IPv4Addr;
END_VAR
```

名称	类型	描述
sIPv4	T_IPv4AddrArr	字符串形式的互联网协议网络地址（类型：T_IPv4Addr [▶ 117]）。例如 172.16.7.199。

输入参数	返回参数	描述
sIPv4 ≠ ''（空字符串） sIPv4 ≠ '0.0.0.0'	所有字节均为空值	在转换过程中出现错误，检查 sIPv4 字符串的格式。

结构化文本示例：

互联网协议（IPv4）网络地址字符串：“172.16.7.199”转换为地址字节数组。

```
PROGRAM MAIN
VAR
    ids    : T_IPv4AddrArr;
    sIPv4  : T_IPv4Addr := '172.16.7.199';
END_VAR

ids := F_ScanIPv4AddrIds( sIPv4 ); (* Result: ids[0]:=172, ids[1]:=16, ids[2]:=7, ids[3]:=199 *)
```

前提条件

开发环境	目标平台	待集成的 PLC 库（类别组）
TwinCAT v3.1.0	PC 或 CX（x86、x64、Arm®）	Tc2_System（系统）

4.1.5 F_GetCpuCoreIndex

对于任务索引，函数 F_GetCpuCoreIndex 可返回任务运行所在 CPU 内核的索引。

如果传递的任务索引为 0，则会确认调用该函数的任务所对应的 CPU 内核索引。如果传递的任务索引无效，则该函数将返回 CPU 内核索引 -1。

函数将已确认的 CPU 内核索引作为返回参数返回。它与 SYSTEM（系统）节点下方 **Real-time（实时）**子节点中所示**内核**列的值相对应。

函数 F_GetCpuCoreIndex：DINT

输入

```
VAR_INPUT
    nTaskIndex : DINT;
END_VAR
```

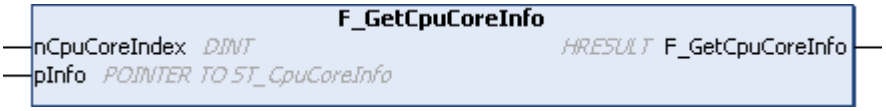
名称	类型	描述
nTaskIndex	DINT	确定任务索引及其相关的 CPU 索引。如果传递的任务索引为 0，则会确认调用该函数的任务所对应的 CPU 内核索引。

另请参见：

- GETCURTASKINDEX [▶ 18]

开发环境	目标平台	待集成的 PLC 库
TwinCAT v3.1.4024.11	PC 或 CX（x86、x64、Arm®）	Tc2_System（系统） >= 3.4.24.0

4.1.6 F_GetCpuCoreInfo



函数 F_GetCpuCoreInfo 返回所传递索引对应的 CPU 内核信息。读取信息包括特定 CPU 内核的基准时间和内核限值。

通过函数 F_GetCpuCoreIndex [▶ 81] 可以确定要传递的 CPU 内核索引。

CPU 内核索引与 SYSTEM（系统）节点下方 **Real-time（实时）**子节点中所示**内核**列的值相对应。在该视图中也会显示通过函数 F_GetCpuCoreInfo 读取的 CPU 内核信息。

该函数以 HRESULT 返回错误代码（另请参见 ADS 返回代码 [▶ 136]）。它指明了函数调用是否成功。如果传递的 CPU 内核索引无效，则该函数将返回错误（0x9811070B = 无效的参数值）。

函数 F_GetCpuCoreInfo：HRESULT

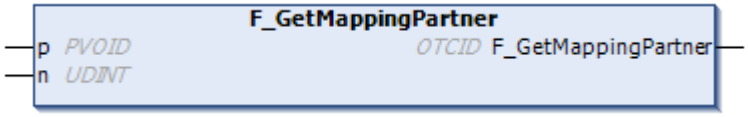
输入

```
VAR_INPUT
    nCpuCoreIndex : DINT;
    pInfo         : POINTER TO ST_CpuCoreInfo;
END_VAR
```

名称	类型	描述
nCpuCoreIndex	DINT	要读取其信息的 CPU 内核的索引。
pInfo	POINTER TO ST_CpuCoreInfo	接收读取数据的变量地址。地址必须指向 ST_CpuCoreInfo [▶ 115] 类型的实例。

开发环境	目标平台	待集成的 PLC 库
TwinCAT v3.1.4024.11	PC 或 CX（x86、x64、Arm®）	Tc2_System（系统） >= 3.4.24.0

4.1.7 F_GetMappingPartner



函数 F_GetMappingPartner 返回映射中对应方的对象 ID（数据类型：OTCID）。

函数 F_GetMappingPartner：OTCID

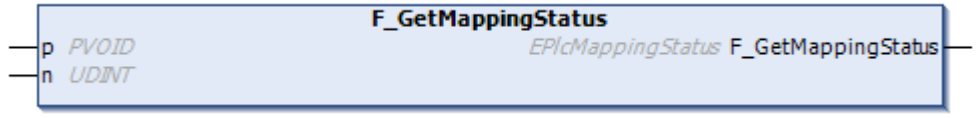
输入

```
VAR_INPUT
    p : PVOID;
    n : UDINT;
END_VAR
```

名称	类型	描述
P	PVOID	变量的内存地址
n	UDINT	变量的大小，以字节为单位

开发环境	目标平台	待集成的 PLC 库
TwinCAT v3.1.4020	PC 或 CX（x86、x64、Arm®）	Tc2_System（系统）

4.1.8 F_GetMappingStatus



函数 F_GetMappingStatus 返回 PLC 变量的当前映射状态。该函数返回一个 ENUM 值（数据类型：EPlcMappingStatus [▶ 114]）以及值 MS_Unmapped、MS_Mapped 或 MS_Partial。

FUNCTION F_GetMappingStatus：EPlcMappingStatus

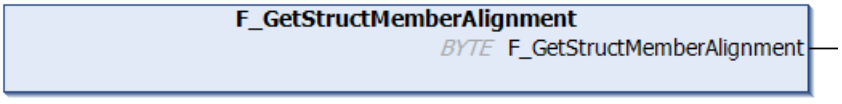
输入

```
VAR_INPUT
    p : PVOID;
    n : UDINT;
END_VAR
```

名称	类型	描述
P	PVOID	变量的内存地址
n	UDINT	变量的大小，以字节为单位

开发环境	目标平台	待集成的 PLC 库
TwinCAT v3.1.4020	PC 或 CX（x86、x64、Arm®）	Tc2_System（系统）

4.1.9 F_GetStructMemberAlignment



该函数返回已用数据结构成员的对齐方式设置信息。对齐方式会影响数据结构元素在计算机内存中的排列方式。

FUNCTION F_GetStructMemberAlignment：BYTE

输入

```
VAR_INPUT
    (* keine Eingangsparameter *)
END_VAR
```

返回参数	描述
1	1 字节对齐（例如 TwinCAT v2.11，x86 目标平台）
2	2 字节对齐
4	4 字节对齐（例如 TwinCAT v2.11，Arm® 目标平台）
8	8 字节对齐

下面的示例显示了数据结构元素在内存中的排列，这具体取决于所采用的内存对齐方式。

?? := 填充字节

示例 1

```
TYPE ST_TEST1
STRUCT
    ui8 : BYTE := 16#FF;(* FF *)
    f64 : LREAL := 1234.5678;(* AD FA 5C 6D 45 4A 93 40 *)
END_STRUCT
END_TYPE

test1 : ST_TEST1;
```

对齐方式	SIZEOF(test1)	内存内容
1 字节	9	FF AD FA 5C 6D 45 4A 93 40
2 字节	10	FF ?? AD FA 5C 6D 45 4A 93 40
4 字节	12	FF ?? ?? ?? AD FA 5C 6D 45 4A 93 40
8 字节	16	FF ?? ?? ?? ?? ?? ?? ?? AD FA 5C 6D 45 4A 93 40

示例 2

转换结构元素的对齐方式会改变填充字节的排列。填充字节将被添加至末尾。

```
TYPE ST_TEST2
STRUCT
    f64 : LREAL := 1234.5678;(* AD FA 5C 6D 45 4A 93 40 *)
    ui8 : BYTE := 16#FF;(* FF *)
END_STRUCT
END_TYPE

test2 : ST_TEST2;
```

对齐方式	SIZEOF(test2)	内存内容
1 字节	9	AD FA 5C 6D 45 4A 93 40 FF
2 字节	10	AD FA 5C 6D 45 4A 93 40 FF ??
4 字节	12	AD FA 5C 6D 45 4A 93 40 FF ?? ?? ??
8 字节	16	AD FA 5C 6D 45 4A 93 40 FF ?? ?? ?? ?? ?? ?? ??

示例 3

在 2、4 和 8 字节对齐的情况下，元素 ui32 和 f64 已正确对齐，因此无需添加填充字节。

```
TYPE ST_TEST3
STRUCT
    ui8  : BYTE := 16#FF;(* FF *)
    ui16 : WORD := 16#1234;(* 34 12 *)
    ui32 : DWORD := 16#AABBCCDD;(* DD CC BB AA *)
    f64  : LREAL := 1234.5678;(* AD FA 5C 6D 45 4A 93 40 *)
END_STRUCT
```

```
END_TYPE

test3 : ST_TEST3;
```

对齐方式	SIZEOF(test3)	内存内容
1 字节	15	FF 34 12 DD CC BB AA AD FA 5C 6D 45 4A 93 40
2 字节	16	FF ?? 34 12 DD CC BB AA AD FA 5C 6D 45 4A 93 40
4 字节	16	FF ?? 34 12 DD CC BB AA AD FA 5C 6D 45 4A 93 40
8 字节	16	FF ?? 34 12 DD CC BB AA AD FA 5C 6D 45 4A 93 40

示例 4

```
TYPE ST_A1
STRUCT
    ui8  : BYTE := 16#FF;(* FF *)
    ui32 : DWORD := 16#AABBCCDD;(* DD CC BB AA *)
    rsv  : BYTE := 16#EE;(* EE *)
END_STRUCT
END_TYPE

TYPE ST_A2
STRUCT
    ui16 : WORD := 16#1234;(* 34 12 *)
    ui8  : BYTE := 16#55;(* 55 *)
END_STRUCT
END_TYPE

TYPE ST_TEST4
STRUCT
    a1 : ST_A1;
    a2 : ST_A2;
END_STRUCT
END_TYPE

test4 : ST_TEST4;
```

对齐方式	SIZEOF(test4)	SIZEOF(test4.a1)	a1/a2 填充字节	SIZEOF(test4.a2)	内存内容
1 字节	9	6	-	3	FF DD CC BB AA EE 34 12 55
2 字节	12	8	-	4	FF ?? DD CC BB AA EE ?? 34 12 55 ??
4 字节	16	12	-	4	FF ?? ?? ?? DD CC BB AA EE ?? ?? ?? 34 12 55 ??
8 字节	16	12	-	4	FF ?? ?? ?? DD CC BB AA EE ?? ?? ?? 34 12 55 ??

示例 5

```
TYPE ST_D1
STRUCT
    ui16 : WORD := 16#1234;(* 34 12 *)
    ui8 : BYTE := 16#55;(* 55 *)
END_STRUCT
END_TYPE

TYPE ST_D2
STRUCT
    ui8 : BYTE := 16#FF;(* FF *)
    f64 : LREAL := 1234.5678;(* AD FA 5C 6D 45 4A 93 40 *)
    rsv : BYTE := 16#EE;(* EE *)
END_STRUCT
END_TYPE

TYPE ST_TEST5
STRUCT
    d1 : ST_D1;
    d2 : ST_D2;
END_STRUCT
```

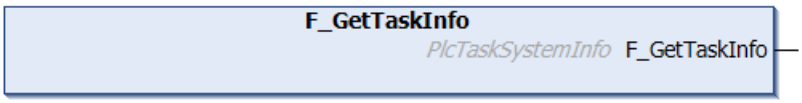
```
END_TYPE

test5 : ST_TEST5;
```

对齐方式	SIZEOF(test5)	SIZEOF(test5.d1)	d1/d2 填充字节	SIZEOF(test5.d2)	内存内容
1 字节	13	3	-	10	34 12 55 FF AD FA 5C 6D 45 4A 93 40 EE
2 字节	16	4	-	12	34 12 55 ?? FF ?? AD FA 5C 6D 45 4A 93 40 EE ??
4 字节	20	4	-	16	34 12 55 ?? FF ?? ?? ?? AD FA 5C 6D 45 4A 93 40 EE ?? ?? ??
8 字节	32	4	4	24	34 12 55 ?? ?? ?? ?? ?? FF ?? ?? ?? ?? ?? ?? ?? AD FA 5C 6D 45 4A 93 40 EE ?? ?? ?? ?? ?? ?? ??

前提条件

开发环境	目标平台	待集成的 PLC 库（类别组）
TwinCAT v3.1.0	PC 或 CX（x86、x64、Arm®）	Tc2_System（系统）

4.1.10 F_GetTaskInfo



该函数可确定调用该函数的任务自身的任务系统信息。

FUNCTION F_GetTaskInfo：PlcTaskSystemInfo

输入

```
VAR_INPUT
    (*keine*)
END_VAR
```

返回参数	描述
PlcTaskSystemInfo	调用该函数的任务自身的任务系统信息。

前提条件

开发环境	目标平台	待集成的 PLC 库（类别组）
TwinCAT v3.1.4026	PC 或 CX（x86、x64、Arm®）	Tc2_System（系统） >= 3.9.1.0

4.1.11 F_RaiseException

调用该函数可以抛出带有特定异常代码的运行时异常。

如果在 __TRY 程序块之外使用 F_RaiseException() 函数，则 TwinCAT Exception Handling 会捕获异常并停止执行控制。

有关 __TRY、__CATCH、__FINALLY、__ENDTRY 的基本信息

请参见__TRY、__CATCH、__FINALLY、__ENDTRY 的相关文档，以获得对异常处理的基本了解。

FUNCTION F_RaiseException

输入

```
VAR_INPUT
    ExceptionCode : UDINT;
END_VAR
```

名称	类型	描述
ExceptionCode	UDINT	ExceptionCode 的规范。请参见 __SYSTEM.ExceptionCode

示例

```
PROGRAM MAIN
VAR
    nCounter1           : INT;
    nCounter2           : INT;
    nCounter_TRY1       : INT;
    nCounter_TRY2       : INT;
    nCounter_CATCH      : INT;
    aData               : ARRAY[1..cMax] OF INT;
    nIndex              : UINT;

    exc                 : __SYSTEM.ExceptionCode;
    lastExc             : __SYSTEM.ExceptionCode;
END_VAR
VAR CONSTANT
    cMax                : UINT := 10;
END_VAR
```

```
// Counter 1
nCounter1 := nCounter1 + 1;

nIndex := nIndex + 1;

// TRY-CATCH block
__TRY
    nCounter_TRY1 := nCounter_TRY1 + 1;
    IF nIndex > cMax THEN
        F_RaiseException(__SYSTEM.ExceptionCode.RTSEXCPT_ARRAYBOUNDS);
    END_IF
    nCounter_TRY2 := nCounter_TRY2 + 1;

__CATCH(exc)
    nCounter_CATCH := nCounter_CATCH + 1;

    // Exception logging
    lastExc := exc;

    // Correct the faulty variable values
    IF (exc = __SYSTEM.ExceptionCode.RTSEXCPT_ARRAYBOUNDS) AND (nIndex > cMax) THEN
        nIndex := cMax;
    END_IF

__ENDTRY

aData[nIndex] := 123;

// Counter 2
nCounter2 := nCounter2 + 1;
```

采用用户特定的 ExceptionCode 的示例

用户特定的异常代码在使用前应与 RTSEXCPT_VENDOR_EXCEPTION_BASE 进行逻辑或处理。

```
PROGRAM MAIN
VAR
    nCounter1           : INT;
    nCounter2           : INT;
    nCounter_TRY1       : INT;
    nCounter_TRY2       : INT;
```

```
    nCounter_CATCH      : INT;
    aData               : ARRAY[1..cMax] OF INT;
    nIndex              : UINT;

    exc                 : __SYSTEM.ExceptionCode;
    lastExc             : __SYSTEM.ExceptionCode;
END_VAR
VAR CONSTANT
    cMyOwnExceptionCode : UDINT := 123;
    cMax                : UINT := 10;
END_VAR
```

```
// Counter 1
nCounter1 := nCounter1 + 1;

nIndex := nIndex + 1;

// TRY-CATCH block
__TRY
    nCounter_TRY1 := nCounter_TRY1 + 1;
    IF nIndex > cMax THEN
        F_RaiseException(__SYSTEM.ExceptionCode.RTSEXCPT_VENDOR_EXCEPTION_BASE OR
cMyOwnExceptionCode);
    END_IF
    nCounter_TRY2 := nCounter_TRY2 + 1;

__CATCH(exc)
    nCounter_CATCH := nCounter_CATCH + 1;

    // Exception logging
    lastExc := exc;

    // Correct the faulty variable values
    IF (exc = __SYSTEM.ExceptionCode.RTSEXCPT_VENDOR_EXCEPTION_BASE OR
cMyOwnExceptionCode) AND (nIndex > cMax) THEN
        nIndex := cMax;
    END_IF

__ENDTRY

aData[nIndex] := 123;

// Counter 2
nCounter2 := nCounter2 + 1;
```

开发环境	目标平台	待集成的 PLC 库（类别组）
TwinCAT v3.1.4026.15	PC 或 CX（x86、x64、Arm®）	Tc2_System（系统） >= 3.8.1.0

4.1.12 F_SplitPathName

该函数将完整的路径名拆分成四个组成部分。分别存储在名为 sDrive、sDir、sFileName 和 sExt 的字符串中。

FUNCTION F_SplitPathName：BOOL

输入

```
VAR_INPUT
    sPathName : T_MaxString;
END_VAR
```

名称	类型	描述
sPathName	T_MaxString	字符串形式的完整文件名（类型：T_MaxString [▶ 118]）： “X:\DIR\SUBDIR\FILENAME.EXT”。

输入/输出

```
VAR_IN_OUT
    sDrive    : STRING(3);
    sDir      : T_MaxString;
```

```
    sFileName : T_MaxString;
    sExt      : T_MaxString;
END_VAR
```

名称	类型	描述
sDrive	STRING	带冒号的驱动器 ID（类型：T_MaxString [▶ 118]）（“C:”、“A:”等）
sDir	T_MaxString	目录名称（类型：T_MaxString [▶ 118]），包含首尾反斜杠（“\BC \INCLUDE\”、“\SOURCE\”等）
sFileName	T_MaxString	文件名（类型：T_MaxString [▶ 118]）
sExt	T_MaxString	包含点和文件扩展名（类型：T_MaxString [▶ 118]）（示例：“.C”、“.EXE”等）。

返回参数	描述
TRUE	无错误
FALSE	错误。检查函数参数。

ST 调用示例：

路径名：C:\TwinCAT\Plc\Project01\Data.txt 被拆分成以下独立的组成部分：

sDrive: = 'C:'
sDir: '\TwinCAT\Plc\Project01\'
sFileName: 'Data'
sExt: '.txt'

```
PROGRAM MAIN
VAR
    bSplit     : BOOL;
    sPathName  : T_MaxString := 'C:\TwinCAT\Plc\Project01\Data.txt';
    sDrive     : STRING(3);
    sDir       : T_MaxString;
    sFileName  : T_MaxString;
    sExt       : T_MaxString;
    bSuccess   : BOOL;
END_VAR

IF bSplit THEN
    bSplit := FALSE;
    bSuccess := F_SplitPathName( sPathName := sPathName,
                sDrive := sDrive,
                sDir := sDir,
                sFileName := sFileName,
                sExt := sExt );
END_IF
```

前提条件

开发环境	目标平台	待集成的 PLC 库（类别组）
TwinCAT v3.1.0	PC 或 CX（x86、x64、Arm®）	Tc2_System（系统）

4.1.13 SETBIT32

该函数在传入的 32 位值中设置由位编号指定的位，并返回结果值。

FUNCTION SETBIT32：DWORD

输入

```
VAR_INPUT
    inVal32 : DWORD;
    bitNo   : SINT;
END_VAR
```

名称	类型	描述
inVal32	DWORD	要更改的 32 位值。
bitNo	SINT	要设置的位编号（0-31）。在执行之前，该编号会在内部转换为对 32 取模后的值。

在 FBD 中调用函数的示例：

此处，在输入值 0 中设置位 31。结果为（十六进制）值“80000000”。

前提条件

开发环境	目标平台	待集成的 PLC 库（类别组）
TwinCAT v3.1.0	PC 或 CX（x86、x64、Arm®）	Tc2_System（系统）

4.1.14 CSETBIT32

该函数在传入的 32 位值中设置/重置由位编号指定的位，并返回结果值。

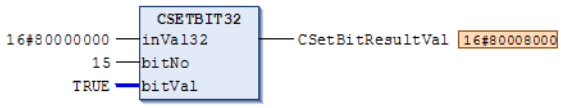
FUNCTION CSETBIT32：DWORD

输入

```
VAR_INPUT
    inVal32 : DWORD;
    bitNo   : SINT;
    bitVal  : BOOL;
END_VAR
```

名称	类型	描述
inVal32	DWORD	32 位值
bitNo	SINT	要设置或重置的位编号（0-31）。在执行之前，该编号会在内部转换为对 32 取模后的值。
bitVal	BOOL	该位要设置或重置的目标值（TRUE = 1，FALSE = 0）。

在 FBD 中调用函数的示例：



此处，在输入值“16#80000000”中将位 15 置为 1。结果（16#80008000）被分配给变量 CSetBitResultVal。

前提条件

开发环境	目标平台	待集成的 PLC 库（类别组）
TwinCAT v3.1.0	PC 或 CX（x86、x64、Arm®）	Tc2_System（系统）

4.1.15 GETBIT32

该函数将传入的 32 位值中由位编号指定的位状态，以布尔型的结果值返回。输入值不会改变。

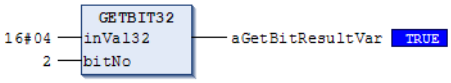
FUNCTION GETBIT32 : BOOL

输入

```
VAR_INPUT
    inVal32 : DWORD;
    bitNo   : SINT;
END_VAR
```

名称	类型	描述
inVal32	DWORD	32 位值
bitNo	SINT	要读取的位编号（0-31）。在执行之前，该编号会在内部转换为对 32 取模后的值。

在 FBD 中调用函数的示例：

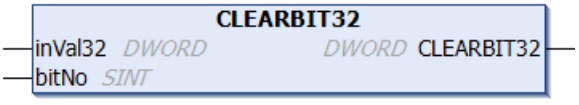


此处，在输入值“16#04”中查询位 2，并将其分配给布尔变量 aGetBitResultVar。在此示例中，查询返回 TRUE。

前提条件

开发环境	目标平台	待集成的 PLC 库（类别组）
TwinCAT v3.1.0	PC 或 CX（x86、x64、Arm®）	Tc2_System（系统）

4.1.16 CLEARBIT32



该函数在传入的 32 位值中将由位编号指定的位重置为零，并返回结果值。

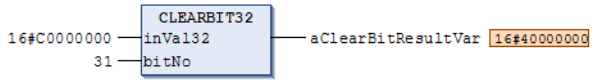
FUNCTION CLEARBIT32 : DWORD

输入

```
VAR_INPUT
    inVal32 : DWORD;
    bitNo   : SINT;
END_VAR
```

名称	类型	描述
inVal32	DWORD	要更改的 32 位值。
bitNo	SINT	要设置的位编号（0-31）。在执行之前，该编号会在内部转换为对 32 取模后的值。

在 FBD 中调用函数的示例：

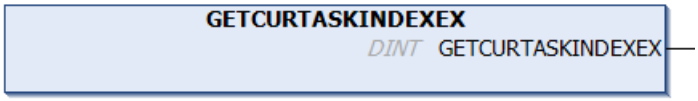


此处，在输入值“C0000000”中重置第 31 位。结果位（十六进制）值“40000000”。

前提条件

开发环境	目标平台	待集成的 PLC 库（类别组）
TwinCAT v3.1.0	PC 或 CX（x86、x64、Arm®）	Tc2_System（系统）

4.1.17 GETCURTASKINDEXEX



该函数用于确定调用它的任务自身的任务索引。相较于功能块 GETCURTASKINDEX [▶ 18]，可以区分该函数是否在周期性实时上下文中被调用。

FUNCTION GETCURTASKINDEXEX : DINT

输入

```
VAR_INPUT
    (*keine*)
END_VAR
```

返回参数	描述
-1	该函数于 Windows 上下文中被调用。
0	该函数于实时上下文中被调用，但并非从 PLC 周期任务中调用。例如，在初始化阶段自动调用 FB_init 方法即属于此种情况。
1 至 n	该函数从 PLC 周期任务中调用。返回值为任务索引。

前提条件

开发环境	目标平台	待集成的 PLC 库（类别组）
TwinCAT v3.1.0	PC 或 CX（x86、x64、Arm®）	Tc2_System（系统）

4.1.18 LPTSIGNAL

该函数将 Centronics 接口中定义的输出位设置为逻辑高电平或低电平，并且可用于配合示波器进行运行时测量等。

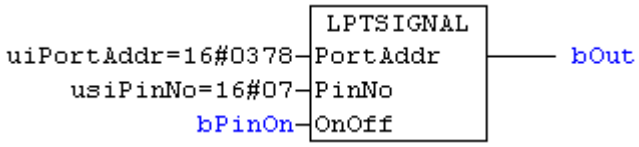
FUNCTION LPTSIGNAL : BOOL

输入

```
VAR_INPUT
    PortAddr : UINT;
    PinNo    : INT;
    OnOff    : BOOL;
END_VAR
```

名称	类型	描述
PortAddr	UINT	可用于所需 LPT 接口的端口地址。
PinNo	INT	引脚编号（引脚 0 .. 7），由 PLC 写入。
OnOff	BOOL	要写入引脚的状态。

在 FBD 中调用函数的示例：



在此示例中，端口 378 的第 7 位（十六进制）被设置为 1。

前提条件

开发环境	目标平台	待集成的 PLC 库（类别组）
TwinCAT v3.1.0	PC 或 CX（x86、x64、Arm®）	Tc2_System（系统）

4.1.19 TestAndSet

您可以使用该函数来检查和设置标志。没有中断进程的选项。这样可以同步数据访问。通过 TestAndSet 可以实现信号的运行模式。

如果函数调用成功，则函数会返回 TRUE，并且可以访问所需的数据。如果函数调用不成功，则函数会返回 FALSE，并且无法访问所需的数据。在这种情况下，必须提供替代处理方法。

输入/输出

```
VAR_IN_OUT
    Flag : BOOL; (* Flag to check if TRUE or FALSE *)
END_VAR
```

名称	类型	描述
标志	BOOL	要检查的 Boolean 标志 • 如果它为 FALSE，则标志尚未被分配并会被设置（因此，从现在起被阻塞），而且函数会返回 TRUE • 如果它为 TRUE，则标志已被分配（因此被阻塞），而且函数会返回 FALSE

示例

```
VAR_GLOBAL
    bGlobalTestFlag : BOOL;
END_VAR
```

```
VAR
    nLocalBlockedCounter : DINT;
END_VAR
```

```
IF TestAndSet(GVL.bGlobalTestFlag) THEN
    (* bGlobalTestFlag was FALSE, nobody was blocking, NOW
    bGlobalTestFlag is set to TRUE and blocking others *)

    (* ... *)

    (* remove blocking by resetting the flag *)
    GVL.bGlobalTestFlag := FALSE;
ELSE
    (* bGlobalTestFlag was TRUE, somebody is blocking *)
    nLocalBlockedCounter := nLocalBlockedCounter + 1;

    (* ... *)
END_IF
```

NEGATIVE 示例

在进一步封装（例如，在功能块中）时应谨慎，因为这可能会破坏所需的原子操作。这样就无法再进行数据访问的安全同步。下面包含的 NEGATIVE 示例说明了不得如何使用该功能。如果在此项实现中 2 个上下文同时请求访问数据，那么这 2 个上下文都会认为可以进行访问，从而同时对数据进行不安全的访问。

```
FUNCTION_BLOCK FB_MyGlobalLock
VAR_INPUT
    bLock    : BOOL; // set TRUE to lock & set FALSE to unlock
END_VAR
VAR_OUTPUT
    bLocked : BOOL;
END_VAR

IF bLock THEN
    TestAndSet(bLocked);
ELSE // unlock
    bLocked := FALSE;
END_IF
```

```
IF NOT GVL.fbGlobalLock.bLocked THEN
    GVL.fbGlobalLock(bLock := TRUE);

    (* ... *)

    GVL.fbGlobalLock(bLock := FALSE);
END_IF
```

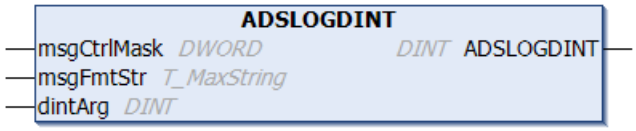
功能块 FB_IecCriticalSection [▶ 13] 可提供临界区的应用，作为 1 种替代的 Mutex 方法。

前提条件

开发环境	目标平台	待集成的 PLC 库（类别组）
TwinCAT v3.1.0	PC 或 CX（x86、x64、Arm®）	Tc2_System（系统）

4.2 ADS 函数

4.2.1 ADSLOGDINT



调用时，该函数会向屏幕输出一个带有可预定义文本的消息框，并在系统事件日志中写入一个条目。由于 PLC 程序采用循环处理方式，因此有必要对消息框等项进行边沿触发输出。最简便的实现方式是串联使用 R_TRIG 或 F_TRIG 功能块（另请参见下文示例）。

使用 ADSLOGDINT 函数，用户可在待输出文本中指定位置插入 DINT 值（4 字节有符号整数）。为此，所创建的格式字符串在目标位置必须包含 %d 字符串。返回参数包含函数错误代码或 0（如成功）。

FUNCTION ADSLOGDINT : DINT

输入

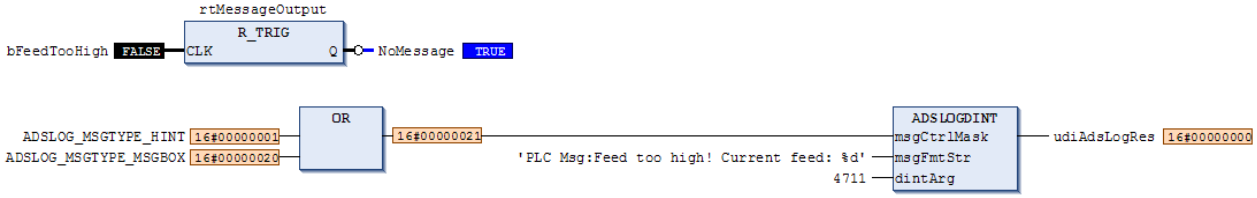
```
VAR_INPUT
    msgCtrlMask : DWORD;
    msgFmtStr   : T_MaxString;
    dintArg     : DINT;
END_VAR
```

名称	类型	描述
msgCtrlMask	DWORD	控制掩码，决定消息输出类型和作用（请参见单独表格）。
msgFmtStr	T_MaxString [▶ 118]	包含要输出的消息。它可以包含格式化字符 %d，用于在任意位置输出 DINT 值。 **补充说明：** 消息长度限值为 253 字节（对应标准格式字符串的 253 个字符）。
dintArg	DINT	包含消息中待插入的数值。

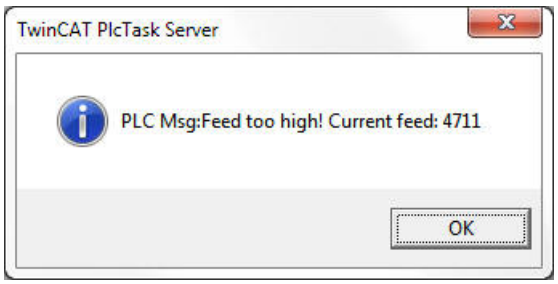
常量	描述
ADSLOG_MSGTYPE_HINT	消息类型为提示。
ADSLOG_MSGTYPE_WARN	消息类型为警告。
ADSLOG_MSGTYPE_ERROR	消息类型为错误。
ADSLOG_MSGTYPE_LOG	消息被写入日志。
ADSLOG_MSGTYPE_MSGBOX	在消息框中输出消息。 **补充说明：** 该功能不适用于 Windows CE。
ADSLOG_MSGTYPE_STRING	消息为直接给出的字符串（默认）。

控制掩码可以按期望组合进行逻辑或处理。

在 FBD 中调用函数的示例：



最终的消息框：



此处将 DINT 值 4711 插入消息。插入点由格式字符串中的 %d 字符标记。

在 ST 中调用函数的示例：

```
PROGRAM MAIN
VAR
    rtMessageOutput: R_TRIG; (* Declaration *)
    bFeedTooHigh: BOOL;
    udiAdsLogRes: UDINT;
END_VAR

rtMessageOutput(CLK := bFeedTooHigh);
IF rtMessageOutput.Q THEN
    UdiAdsLogRes := ADSLOGDINT( msgCtrlMask := ADSLOG_MSGTYPE_HINT OR ADSLOG_MSGTYPE_MSGBOX,
                    msgFmtStr := 'PLC Msg: Feed too high! Current feed: %d', dintArg:= 4711);
END_IF
```

前提条件

开发环境	目标平台	待集成的 PLC 库（类别组）
TwinCAT v3.1.0	PC 或 CX（x86、x64、Arm®）	Tc2_System（系统）

4.2.2 ADSLOGLREAL

调用时，该函数会向屏幕输出一个带有可预定义文本的消息框，并在系统事件日志中写入一个条目。用户可在待输出文本中指定位置加入 LREAL 值（浮点数）。为此，所创建的格式字符串在目标位置必须包含“%f”字符串。请注意，该函数必须以边沿触发的方式调用（另请参见 ADSLOGDINT [▶ 95] 说明中的注释）。返回参数包含函数错误代码或 0（如成功）。

FUNCTION ADSLOGLREAL : DINT

输入

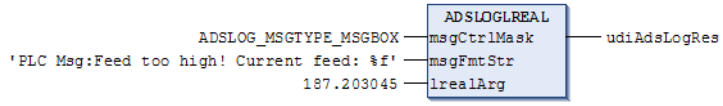
```
VAR_INPUT
    msgCtrlMask : DWORD;
    msgFmtStr   : T_MaxString;
    lrealArg    : LREAL;
END_VAR
```

名称	类型	描述
msgCtrlMask	DWORD	控制掩码，决定消息输出类型和作用（请参见单独表格）。
msgFmtStr	T_MaxString [▶ 118]	包含要输出的消息。它可以包含格式化代码 %f，用于在任意位置输出 LREAL 值。 **补充说明：**消息长度限值为 253 字节（对应标准格式字符串的 253 个字符）。
lrealArg	LREAL	包含消息中待插入的数值。

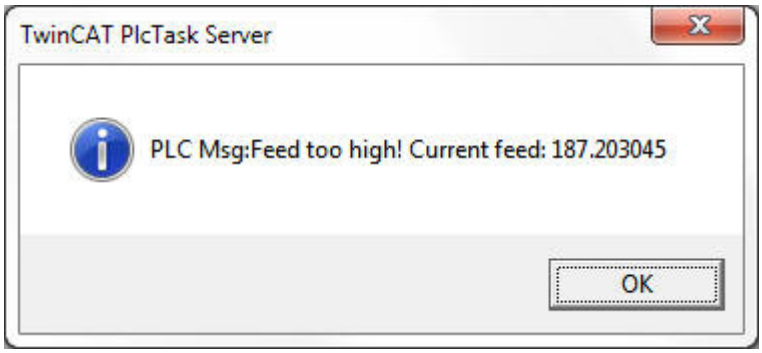
常量	描述
ADSLOG_MSGTYPE_HINT	消息类型为提示。
ADSLOG_MSGTYPE_WARN	消息类型为警告。
ADSLOG_MSGTYPE_ERROR	消息类型为错误。
ADSLOG_MSGTYPE_LOG	消息被写入日志。
ADSLOG_MSGTYPE_MSGBOX	在消息框中输出消息。 **补充说明：** 该功能不适用于 Windows CE。
ADSLOG_MSGTYPE_STRING	消息为直接给出的字符串（默认）。

控制掩码可以按期望组合进行逻辑或处理。

在 FBD 中调用函数的示例：



最终的消息框：



此处将 LREAL 值 187.203045 插入消息。插入点由格式字符串中的 %f 字符标记。输出时，该数值的小数部分会在第六位后被截断。

在 ST 中调用函数的示例：

```
PROGRAM MAIN
VAR
    rtMessageOutput: R_TRIG; (* Declaration *)
    bTemperatureTooHigh: BOOL;
    udiAdsLogRes: UDINT;
END_VAR

rtMessageOutput(CLK := bTemperatureTooHigh);
IF rtMessageOutput.Q THEN
    udiAdsLogRes := ADSLOGLREAL( msgCtrlMask := ADSLOG_MSGTYPE_HINT OR ADSLOG_MSGTYPE_MSGBOX, msgFmt
Str := 'PLC Msg.: Max Temp. reached ! Temperature: %f', lrealArg := 187.203045);
END_IF;
```

前提条件

开发环境	目标平台	待集成的 PLC 库（类别组）
TwinCAT v3.1.0	PC 或 CX（x86、x64、Arm®）	Tc2_System（系统）

4.2.3 ADSLOGSTR

调用时，该函数会向屏幕输出一个带有可预定义文本的消息框，并在系统事件日志中写入一个条目。用户可在待输出文本中指定位置插入一个字符串。为此，所创建的格式字符串在目标位置必须包含 %s 字符串。请注意，该函数必须以边沿触发的方式调用（另请参见 ADSLOGDINT [▶ 95] 说明中的注释）。返回参数包含函数错误代码或 0（如成功）。

FUNCTION ADSLOGSTR : DINT

输入

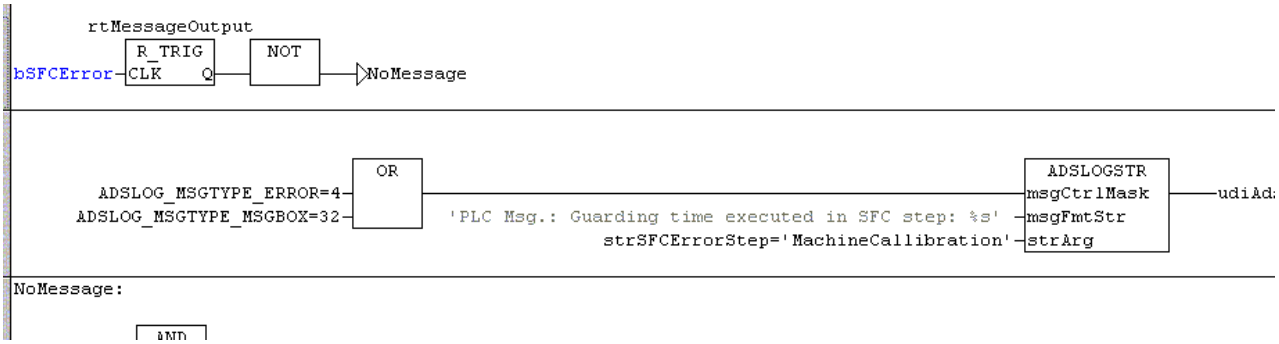
```
VAR_INPUT
    msgCtrlMask   : DWORD;
    msgFmtStr     : T_MaxString;
    strArg        : T_MaxString;
END_VAR
```

名称	类型	描述
msgCtrlMask	DWORD	控制掩码，决定消息输出类型和作用（请参见单独表格）。
msgFmtStr	T_MaxString [▶ 118]	包含要输出的消息。它可以包含格式化字符 %s，用于在任意位置输出文本参数。 **补充说明：** 消息长度限值为 253 字节（对应标准格式字符串的 253 个字符）。
strArg	T_MaxString [▶ 118]	包含要插入消息中的字符串。

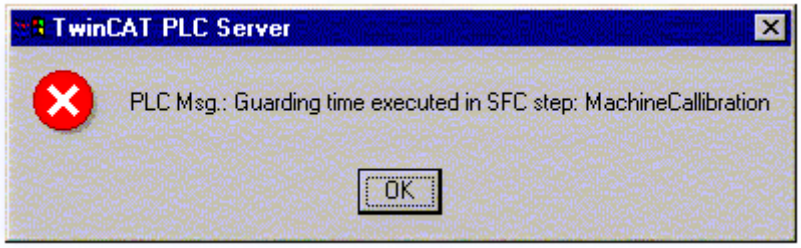
常量	描述
ADSLOG_MSGTYPE_HINT	消息类型为提示。
ADSLOG_MSGTYPE_WARN	消息类型为警告。
ADSLOG_MSGTYPE_ERROR	消息类型为错误。
ADSLOG_MSGTYPE_LOG	消息被写入日志。
ADSLOG_MSGTYPE_MSGBOX	在消息框中输出消息。 **补充说明：** 该功能不适用于 Windows CE。
ADSLOG_MSGTYPE_STRING	消息为直接给出的字符串（默认）。

控制掩码可以按期望组合进行逻辑或处理。

在 FBD 中调用函数的示例：



最终的消息框：



这样，PLC 程序员就会在消息中插入已存储在变量 strSFCErrorStep 中的字符串。插入点由格式字符串中的 %s 字符标记。

在 ST 中调用函数的示例：

```
PROGRAM MAIN
VAR
    strSFCErrorStep : STRING; (* Declaration*)
    rtMessageOutput: R_TRIG;
    bSFCError: BOOL;
END_VAR

rtMessageOutput(CLK := bSFCError);
IF rtMessageOutput.Q THEN
    udiAdsLogRes := ADSLOGSTR( msgCtrlMask := ADSLOG_MSGTYPE_ERROR OR ADSLOG_MSGTYPE_MSGBOX, msgFmtS
tr := 'PLC Msg.: Guarding time executed in SFC step: %s', strArg := strSFCErrorStep);
END_IF;
```

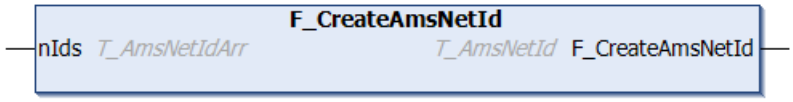
前提条件

开发环境	目标平台	待集成的 PLC 库（类别组）
TwinCAT v3.1.0	PC 或 CX（x86、x64、Arm®）	Tc2_System（系统）

还请参阅有关此

ADSLOGDINT [▶ 95]

4.2.4 F_CreateAmsNetId



该函数生成格式化的 NetID 字符串（类型：T_AmsNetID [▶ 115]），并将其作为返回参数（例如“127.16.17.3.1.1”）返回。

FUNCTION F_CreateAmsNetId：T_AmsNetId

输入

```
VAR_INPUT
    nIds : T_AmsNetIdArr;
END_VAR
```

名称	类型	描述
nIds	T_AmsNetIdArr	字节数组（类型：T_AmsNetIdArr [▶ 116]）。每个字节与网络地址的一个数字相对应。地址字节采用网络字节序。

ST 中的调用示例：

```
PROGRAM MAIN
VAR
    ids    : T_AmsNetIdArr := 127, 16, 17, 3, 1, 1;
    sNetID : T_AmsNetID := '';
END_VAR

sNetID := F_CreateAmsNetId( ids );(* Result: '127.16.17.3.1.1' *)
```

前提条件

开发环境	目标平台	待集成的 PLC 库（类别组）
TwinCAT v3.1.0	PC 或 CX（x86、x64、Arm®）	Tc2_System（系统）

4.2.5 F_ScanAmsNetIds

函数 F_ScanAmsNetIds 可以用于将包含 TwinCAT 网络地址的字符串转换为单个地址字节。单个地址字节从左到右转换，并以字节数组的形式返回（类型：T_AmsNetIdArr [▶ 116]）。地址字节采用网络字节序。

FUNCTION F_ScanAmsNetIds：T_AmsNetIdArr

输入

```
VAR_INPUT
    sNetID : T_AmsNetID;
END_VAR
```

名称	类型	描述
sNetID	T_AmsNetID	字符串形式的 TwinCAT 网络地址（类型：T_AmsNetId [▶ 115]）。例如“127.16.17.3.1.1”

输入参数	返回参数	描述
sNetID ≠ ''（空字符串） 和 sNetID ≠ '0.0.0.0.0.0'	所有字节均为空值	在转换过程中出现错误，检查 sNetID 字符串的格式。

ST 中的调用示例：

在下面的示例中，包含网络地址“127.16.17.3.1.1”的字符串被转换为地址字节数组。

```
PROGRAM MAIN
VAR
    ids    : T_AmsNetIDArr;
    sNetID : T_AmsNetID := '127.16.17.3.1.1';
END_VAR

ids := F_ScanAmsNetIds( sNetID );
(* Result: ids[0]:=127, ids[1]:=16, ids[2]:=17, ids[3]:=3, ids[4]:=1, ids[5]:=1 *)
```

前提条件

开发环境	目标平台	待集成的 PLC 库（类别组）
TwinCAT v3.1.0	PC 或 CX（x86、x64、Arm®）	Tc2_System（系统）

4.3 字符函数

4.3.1 F_ToCHR

该函数将 ASCII 码转换为字符串。

FUNCTION F_ToCHR：STRING

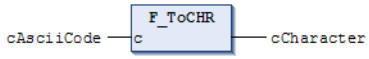
输入

```
VAR_INPUT
    c : BYTE;
END_VAR
```

名称	类型	描述
c	BYTE	要转换的 ASCII 码

在 FBD 中调用函数的示例：

```
PROGRAM P_TEST
VAR
    sCharacter : STRING(1) := '';
    cAsciiCode : BYTE := 16#31;
END_VAR
```



前提条件

开发环境	目标平台	待集成的 PLC 库（类别组）
TwinCAT v3.1.0	PC 或 CX（x86、x64、Arm®）	Tc2_System（系统）

4.3.2 F_ToASC

该函数将字符串转换为 ASCII 码。仅转换字符串的首个字符。空字符串返回零。

FUNCTION F_ToASC：BYTE

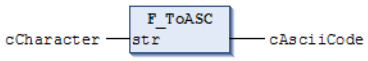
输入

```
VAR_INPUT
    str : STRING;
END_VAR
```

名称	类型	描述
str	STRING	要转换的字符串

在 FBD 中调用函数的示例：

```
PROGRAM P_TEST
VAR
    sCharacter : STRING(1) := '1';
    cAsciiCode : BYTE := 0;
END_VAR
```

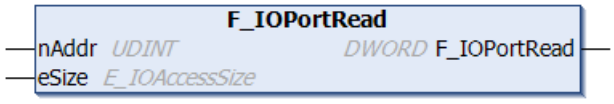


前提条件

开发环境	目标平台	待集成的 PLC 库（类别组）
TwinCAT v3.1.0	PC 或 CX（x86、x64、Arm®）	Tc2_System（系统）

4.4 I/O 端口访问

4.4.1 F_IOPortRead



数字量 I/O 端口通常是一个宽度为 1 字节的 I/O 地址位，它可以映射在内存中，也可以作为端口。若在此写入一个值，输出引脚上的电信号将根据写入位发生相应改变。若从输入位置读取一个值，则输入引脚上当前的逻辑电平将作为一个独立位值返回。

函数 F_IOPortRead 可以用于读取一个宽度为 eSize 的 I/O 地址位。函数将读取的值作为返回参数返回。另请参见 F_IOPortWrite [▶ 102] 函数的说明。

FUNCTION F_IOPortRead : DWORD

输入

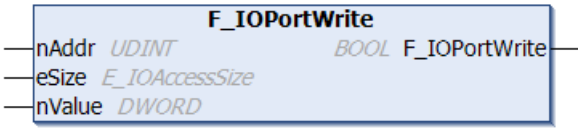
```
VAR_INPUT
    nAddr : UDINT;
    eSize : E_IOAccessSize;
END_VAR
```

名称	类型	描述
nAddr	UDINT	I/O 端口地址
eSize	E_IOAccessSize	要读取的数据字节数（类型：E_IOAccessSize [▶ 112]）

前提条件

开发环境	目标平台	待集成的 PLC 库（类别组）
TwinCAT v3.1.0	PC 或 CX（x86、x64、Arm®）	Tc2_System（系统）

4.4.2 F_IOPortWrite



函数 F_IOPortWrite 可以用于写入一个宽度为 eSize 的 I/O 地址位。另请参见 F_IOPortRead [▶ 101] 函数的说明。

注意

硬件损坏

只要仅执行数据读取操作，直接硬件访问便不会产生问题。写入访问可能会导致系统崩溃和/或破坏硬件或存储介质上的数据。该函数可能会损坏硬件，使其无法启动。

FUNCTION F_IOPortWrite : BOOL

输入

```
VAR_INPUT
    nAddr  : UDINT;
    eSize  : E_IOAccessSize;
    nValue : DWORD;
END_VAR
```

名称	类型	描述
nAddr	UDINT	I/O 端口地址
eSize	E_IOAccessSize	要写入的数据字节数（类型：E_IOAccessSize [▶ 112]）。
nValue	DWORD	要写入的值。

返回参数	描述
TRUE	无错误
FALSE	错误

ST 中的示例代码：

在下面的示例中，利用 I/O 端口函数实现了用于直接控制 PC 扬声器的 PLC 功能块。

接口：

```
FUNCTION_BLOCK FB_Speaker
(* Sample code from: "PC INTERN 2.0", ISBN 3-89011-331-1, Data Becker *)
VAR_INPUT
    freq     : DWORD := 10000; (* Frequency [Hz] *)
 tDuration   : TIME := T#1s; (* Tone duration *)
    bExecute : BOOL; (* Rising edge starts function block execution *)
END_VAR
VAR_OUTPUT
    bBusy  : BOOL;
    bError : BOOL;
    nErrID : UDINT;
END_VAR
VAR
    fbTrig : R_TRIG;
    nState : BYTE;
    sts61H : DWORD;
    cnt42H : DWORD;
    cntLo  : DWORD;
    cntHi  : DWORD;
    timer  : TON;
END_VAR
```

实施：

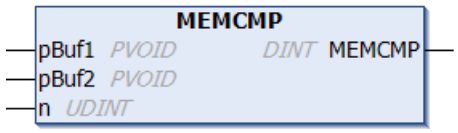
```
fbTrig( CLK := bExecute );
CASE nState OF
0:
IF fbTrig.Q THEN
    bBusy := TRUE;
    bError := FALSE;
    nErrID := 0;
    timer( IN := FALSE );

    IF F_IOPortWrite( 16#43, NoOfByte_Byte, 182 ) THEN

        cnt42H := 1193180 / freq;
        cntLo := cnt42H AND 16#FF;
        cntHi := SHR( cnt42H, 8 ) AND 16#FF;

        F_IOPortWrite( 16#42, NoOfByte_Byte, cntLo ); (* LoByte *)
        F_IOPortWrite( 16#42, NoOfByte_Byte, cntHi ); (* HiByte *)

        timer( IN := TRUE, PT := tDuration );

        sts61H := F_IOPortRead( 16#61, NoOfByte_Byte );
        sts61H := sts61H OR 2#11;
        F_IOPortWrite( 16#61, NoOfByte_Byte, sts61H ); (* speaker ON *)

        nState := 1;
    ELSE
        nState := 100;
    END_IF
END_IF

1:
timer( );
IF timer.Q THEN

    sts61H := F_IOPortRead( 16#61, NoOfByte_Byte );
    sts61H := sts61H AND 2#11111100;
    F_IOPortWrite( 16#61, NoOfByte_Byte, sts61H ); (* speaker off *)
    bBusy := FALSE;
    nState := 0;
END_IF

100:
bBusy := FALSE;
bError := TRUE;
nErrID := 16#8000;
nState := 0;

END_CASE

Test application:

PROGRAM MAIN
VAR
    fbSpeaker : FB_Speaker;
    bStart : BOOL;
END_VAR


fbSpeaker( freq:= 5000,
tDuration:= t#1s,
bExecute:= bStart );
```

前提条件

开发环境	目标平台	待集成的 PLC 库（类别组）
TwinCAT v3.1.0	PC 或 CX（x86、x64、Arm®）	Tc2_System（系统）

4.5 内存函数

4.5.1 MEMCMP



函数 MEMCMP 允许比较两个不同内存区域中的 PLC 变量值。

MEMCMP FUNCTION：DINT

输入

```
VAR_INPUT
    pBuf1 : PVOID;
    pBuf2 : PVOID;
    n     : UDINT;
END_VAR
```

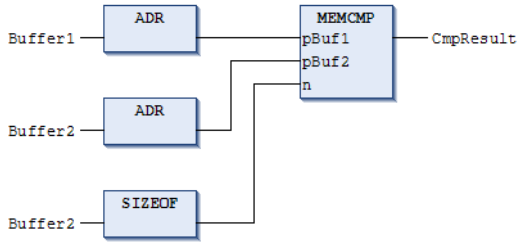
名称	类型	描述
pBuf1	PVOID	第一个内存区域（第一个数据缓冲区）的起始地址。
pBuf2	PVOID	第二个内存区域（第二个数据缓冲区）的起始地址。
n	UDINT	要比较的字节数。

该函数比较两个数据缓冲区中的前 n 个字节，并返回一个与二者比值相对应的值。

返回参数	第一个和第二个数据缓冲区中首个不同字节的比值
-1	pBuf1 小于 pBuf2
0	pBuf1 与 pBuf2 相同
1	pBuf1 大于 pBuf2
0xFF	不正确的参数值。pBuff1 = 0 或 pBuff2 = 0 或 n = 0

FBD 中的调用示例：

```
PROGRAM MAIN
VAR
    Buffer1   : ARRAY[0..3] OF BYTE;
    Buffer2   : ARRAY[0..3] OF BYTE;
    CmpResult : DINT;
END_VAR
```

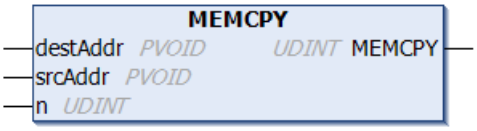


在此示例中，比较 Buffer2 中的 4 字节数据与 Buffer1 中的数据。比较第一个不同的字节，Buffer1 中的对应字节比 Buffer2 中的大。

前提条件

开发环境	目标平台	待集成的 PLC 库（类别组）
TwinCAT v3.1.0	PC 或 CX（x86、x64、Arm®）	Tc2_System（系统）

4.5.2 MEMCPY



函数 MEMCPY 可以用于将 PLC 变量的值从一个内存区域复制到另一个内存区域。

注意

系统崩溃或访问未经授权的内存区域

由于该函数直接访问物理内存，因此在使用时务必谨慎。不正确的参数值可能会导致系统崩溃或访问到禁止访问的内存区域。

如果目标内存区域和源内存区域存在重叠，则 MEMCPY 的行为是未定义的。例如，当需要将在数组中存储的多个值向前或向后移动一个位置时，就会出现这种情况。在这种情况下，使用函数 MEMMOVE [▶ 106]。

FUNCTION MEMCPY : UDINT

输入

```
VAR_INPUT
    destAddr : PVOID;
    srcAddr  : PVOID;
    n        : UDINT;
END_VAR
```

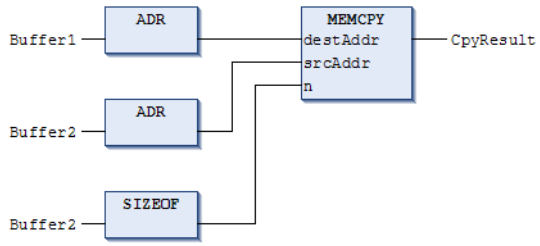
名称	类型	描述
destAddr	PVOID	目标内存区域的起始地址。
srcAddr	PVOID	源内存区域的起始地址。
n	UDINT	要复制的字节数。

该函数将 n 字节从起始地址为 srcAddr 的内存区域复制到起始地址为 destAddr 的内存区域。

返回参数	含义
0	不正确的参数值。destAddr == 0 或 srcAddr == 0 或 n == 0
> 0	如果成功，则为复制的字节数（n）。

FBD 中的调用示例：

```
PROGRAM MAIN
VAR
    Buffer1   : ARRAY[0..3] OF BYTE;
    Buffer2   : ARRAY[0..3] OF BYTE;
    CpyResult : UDINT;
END_VAR
```



在此示例中，将 4 字节从 Buffer2 复制到 Buffer1。

前提条件

开发环境	目标平台	待集成的 PLC 库（类别组）
TwinCAT v3.1.0	PC 或 CX（x86、x64、Arm®）	Tc2_System（系统）

4.5.3 MEMMOVE

使用 MEMMOVE 函数将值从一个内存区域复制到另一个内存区域；内存区域可能会重叠。

注意

系统崩溃或访问未经授权的内存区域

由于该函数直接访问物理内存，因此在使用时务必谨慎。不正确的参数值可能会导致系统崩溃或访问到禁止访问的内存区域。

FUNCTION MEMMOVE : UDINT

输入

```
VAR_INPUT
    destAddr : PVOID;
    srcAddr  : PVOID;
    n        : UDINT;
END_VAR
```

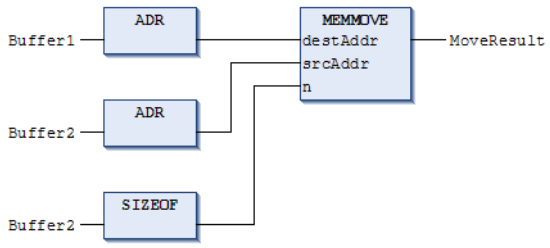
名称	类型	描述
destAddr	PVOID	目标内存区域的起始地址。
srcAddr	PVOID	源内存区域的起始地址。
n	UDINT	要复制的字节数。

该函数将 n 字节从起始地址为 srcAddr 的内存区域复制到起始地址为 destAddr 的内存区域。

返回参数	含义
0	不正确的参数值。destAddr == 0 或 srcAddr == 0 或 n == 0
> 0	如果成功，则为复制的字节数（n）。

FBD 中的调用示例：

```
PROGRAM MAIN
VAR
    Buffer1    : ARRAY[0..3] OF BYTE;
    Buffer2    : ARRAY[0..3] OF BYTE;
    MoveResult : UDINT;
END_VAR
```

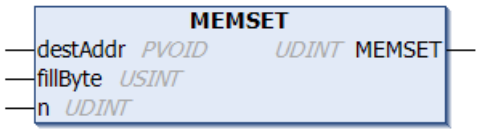


在此示例中，将 4 字节从 Buffer2 移至 Buffer1。

前提条件

开发环境	目标平台	待集成的 PLC 库（类别组）
TwinCAT v3.1.0	PC 或 CX（x86、x64、Arm®）	Tc2_System（系统）

4.5.4 MEMSET



MEMSET 函数允许将特定内存区域中的 PLC 变量设置为特定值。

注意

系统崩溃或访问未经授权的内存区域

由于该函数直接访问物理内存，因此在使用时务必谨慎。不正确的参数值可能会导致系统崩溃或访问到禁止访问的内存区域。

MEMSET FUNCTION：UDINT

输入

```
VAR_INPUT
    destAddr : PVOID;
    fillByte : USINT;
    n        : UDINT;
END_VAR
```

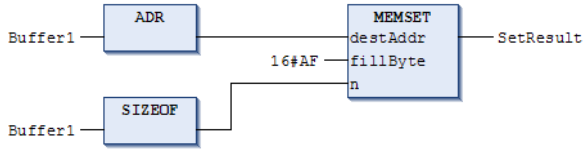
名称	类型	描述
destAddr	PVOID	要设置的内存区域的起始地址。
fillByte	USINT	填充字节的值。
n	UDINT	要设置的字节数。

该函数使用 fillByte 值填充 n 字节；从起始地址为 destAddr 的内存区域开始。

返回参数	含义
0	不正确的参数值。destAddr == 0 或 n == 0
> 0	如果成功，则为设置的字节数（n）。

FBD 中的调用示例：

```
PROGRAM MAIN
VAR
    Buffer1 : ARRAY[0..3] OF BYTE;
    SetResult : UDINT;
END_VAR
```



在此示例中，将 Buffer1 中的 4 字节设置为值 0xAF。

前提条件

开发环境	目标平台	待集成的 PLC 库（类别组）
TwinCAT v3.1.0	PC 或 CX（x86、x64、Arm®）	Tc2_System（系统）

4.6 时间函数

4.6.1 F_GetSystemTime

该函数可以用于读取操作系统时间戳。该时间戳是一个 64 位整数值，精度为 100 ns，每次调用 PLC 时都会更新。此外，它还可以用于定时任务或时间测量。一个单位对应 100 ns。该时间表示自 1601 年 1 月 1 日（UTC）以来的 100 ns 间隔数。

输出

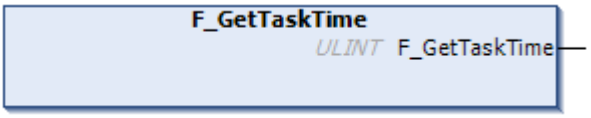
```
VAR_OUTPUT
    F_GetSystemTime    : ULINT;
END_VAR
```

返回值包含时间戳。

要求

开发环境	目标平台	待集成的 PLC 库（类别组）
TwinCAT v3.1	PC 或 CX（x86、x64、ARM）	Tc2_System（系统） >= 3.4.17.0

4.6.2 F_GetTaskTime



该函数可以用于读取任务的开始时间（任务应开始的时间）。该函数始终会返回调用该函数的任务的开始时间。该时间戳是一个 64 位整数值，精度为 100 ns。此外，它还可以用于定时任务或时间测量。一个单位对应 100 ns。该时间表示自 1601 年 1 月 1 日以来的 100 ns 间隔数。

输出

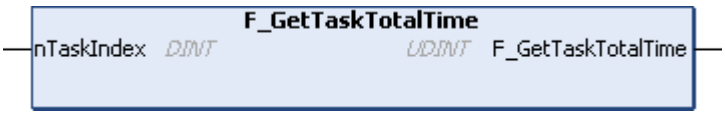
```
VAR_OUTPUT
    F_GetTaskTime : ULINT;
END_VAR
```

返回值包含时间戳。

要求

开发环境	目标平台	待集成的 PLC 库（类别组）
TwinCAT v3.1	PC 或 CX（x86、x64、ARM）	Tc2_System（系统） >= 3.4.17.0

4.6.3 F_GetTaskTotalTime



对于一个任务索引，函数 F_GetTaskTotalTime 返回该任务在上一个周期内的总执行时间。总执行时间是该任务上所有已注册模块的计算时间总和。

如果传递的任务索引为 0，则会确定调用该函数的任务的值。如果传递的任务索引无效，则该函数将返回 0 作为总执行时间。

确定的总执行时间以 100 ns 的倍数显示，并作为返回参数由函数返回。

FUNCTION F_GetTaskTotalTime：UDINT

输入

```
VAR_INPUT
    nTaskIndex : DINT;
END_VAR
```

名称	类型	描述
nTaskIndex	DINT	要确定其总执行时间的任务的索引。如果传递的任务索引为 0，则会确定调用该函数的任务的值。

另请参见：

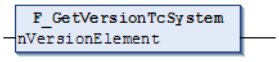
- GETCURTASKINDEX [▶ 18]

开发环境	目标平台	待集成的 PLC 库
TwinCAT v3.1.4024.11	PC 或 CX（x86、x64、Arm®）	Tc2_System（系统） >= 3.4.24.0

4.7 [废弃]

4.7.1 F_GetVersionTcSystem

该函数已废弃，不予使用。可使用全局常量 stLibVersion_Tc2_System [▶ 125]，读取 PLC 库的版本信息。



该函数可以用于读取 PLC 库版本信息。

FUNCTION F_GetVersionTcSystem：UINT

输入

```
VAR_INPUT
    nVersionElement : INT;
END_VAR
```

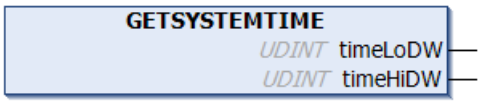
名称	类型	描述
nVersionElement	INT	**nVersionElement**：要读取的版本元素。可选参数： • 1：主版本号； • 2：次版本号； • 3：修订版本号；

前提条件

开发环境	目标平台	待集成的 PLC 库（类别组）
TwinCAT v3.1.0	PC 或 CX（x86、x64、Arm®）	Tc2_System（系统）

4.7.2 GETSYSTEMTIME

该功能块已被更新的函数 F_GetSystemTime() 取代，后者仅需一个返回值，而非两个。



通过该功能块可以读取操作系统时间戳。该时间戳是一个 64 位整数值，精度为 100ns，每次调用 PLC 时都会更新。除其他用途外，它还可以用于定时任务或时间测量。一个单位对应 100 ns。该时间表示自 1601 年 1 月 1 日以来的 100 ns 间隔数。

参见：F_GetSystemTime [▶ 108]

输入

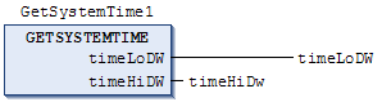
```
VAR_INPUT
(*none*)
END_VAR
```

输出

```
VAR_OUTPUT
    timeLoDW    : UDINT;
    timeHiDW    : UDINT;
END_VAR
```

名称	类型	描述
timeLoDW	UDINT	包含时间戳的低值 4 字节。
timeHiDW	UDINT	包含时间戳的高值 4 字节。

在 FBD 中调用功能块的示例：



该示例通过实例“GetSystemTime1”调用功能块，并返回 64 位整数值（十六进制）1BCD6EAB05C4E60 作为时间戳。

前提条件

开发环境	目标平台	待集成的 PLC 库（类别组）
TwinCAT v3.1.0	PC 或 CX（x86、x64、Arm®）	Tc2_System（系统）

4.7.3 GETTASKTIME

该功能块已被更新的函数 F_GetTaskTime() 取代，后者仅需一个返回值，而非两个。

该功能块可以用于读取任务的开始时间（任务应开始的时间）。该功能块将会返回调用该功能块实例的任务的开始时间。该时间戳是一个 64 位整数值，精度为 100 ns。此外，它还可以用于定时任务或时间测量。一个单位对应 100 ns。该时间表示自 1601 年 1 月 1 日以来的 100 ns 间隔数。

参见：F_GetTaskTime [▶ 108]

输入

```
VAR_INPUT
(*none*)
END_VAR
```

输出

```
VAR_OUTPUT
    timeLoDW : UDINT;
    timeHiDW : UDINT;
END_VAR
```

名称	类型	描述
timeLoDW	UDINT	包含时间戳的低值 4 字节。
timeHiDW	UDINT	包含时间戳的高值 4 字节。

前提条件

开发环境	目标平台	待集成的 PLC 库（类别组）
TwinCAT v3.1.0	PC 或 CX（x86、x64、Arm®）	Tc2_System（系统）

5 数据类型

5.1 E_IOAccessSize

I/O 地址位的字节大小（要写入或读取的字节数）。

```
TYPE E_IOAccessSize :
(
    NoOfByte_Byte  := 1,
    NoOfByte_Word  := 2,
    NoOfByte_DWord := 4
);
END_TYPE
```

前提条件

开发环境	目标平台	待集成的 PLC 库（类别组）
TwinCAT v3.1.0	PC 或 CX（x86、x64、Arm®）	Tc2_System（系统）

5.2 E_OpenPath

该类型变量可在目标设备上选择通用路径或任一 TwinCAT 系统路径，以执行文件打开操作。

```
TYPE E_OpenPath :
(
    PATH_GENERIC :=1, (* search/open/create files in selected/generic folder *)
    PATH_BOOTPRJ,     (* search/open/create files in the TwinCAT/
Boot directory (adds the extension .wbp) *)
    PATH_BOOTDATA,    (* reserved for future use*)
    PATH_BOOTPATH,    (* refers to the TwinCAT/Boot directory without adding an extension (.wbp) *)
    PATH_USERPATH1 :=11, (*reserved for future use*)
    PATH_USERPATH2,   (*reserved for future use*)
    PATH_USERPATH3,   (*reserved for future use*)
    PATH_USERPATH4,   (*reserved for future use*)
    PATH_USERPATH5,   (*reserved for future use*)
    PATH_USERPATH6,   (*reserved for future use*)
    PATH_USERPATH7,   (*reserved for future use*)
    PATH_USERPATH8,   (*reserved for future use*)
    PATH_USERPATH9    (*reserved for future use*)
);
END_TYPE
```

前提条件

开发环境	目标平台	待集成的 PLC 库（类别组）
TwinCAT v3.1.0	PC 或 CX（x86、x64、Arm®）	Tc2_System（系统）

5.3 E_SeekOrigin

该类型的变量通过移动指针来指示起始参考点。

```
TYPE E_SeekOrigin :
(
    SEEK_SET   := 0, (* Seek from beginning of file *)
    SEEK_CUR,        (* Seek from current position of file pointer *)
    SEEK_END         (* Seek from the end of file *)
);
END_TYPE
```

值	描述
SEEK_SET	从文件起始位置查找
SEEK_CUR	从文件指针的当前位置查找
SEEK_END	从文件末尾查找

前提条件

开发环境	目标平台	待集成的 PLC 库（类别组）
TwinCAT v3.1.0	PC 或 CX（x86、x64、Arm®）	Tc2_System（系统）

5.4 E_TcEventClass

```
TYPE E_TcEventClass :
(
    TCEVENTCLASS_NONE         :=0, (* No class *)
    TCEVENTCLASS_MAINTENANCE :=1, (* Maintenance hint *)
    TCEVENTCLASS_MESSAGE      :=2, (* Message *)
    TCEVENTCLASS_HINT         :=3, (* Hint *)
    TCEVENTCLASS_STATEINFO    :=4, (* State information *)
    TCEVENTCLASS_INSTRUCTION :=5, (* Instruction *)
    TCEVENTCLASS_WARNING      :=6, (* Warning *)
    TCEVENTCLASS_ALARM        :=7, (* Alarm *)
    TCEVENTCLASS_PARAMERROR   :=8 (* Parameter error *)
);
END_TYPE
```

前提条件

开发环境	目标平台	待集成的 PLC 库（类别组）
TwinCAT v3.1.0	PC 或 CX（x86、x64、Arm®）	Tc2_System（系统）

5.5 E_TcEventClearModes

```
TYPE E_TcEventClearModes :
(
    TCEVENTLOGIOFFS_CLEARACTIVE := 1,
    TCEVENTLOGIOFFS_CLEARLOGGED ,
    TCEVENTLOGIOFFS_CLEARALL
);
END_TYPE
```

前提条件

开发环境	目标平台	待集成的 PLC 库（类别组）
TwinCAT v3.1.0	PC 或 CX（x86、x64、Arm®）	Tc2_System（系统）

5.6 E_TcEventPriority

```
TYPE E_TcEventPriority :
(
    TCEVENTPRIO_IMPLICIT := 0
);
END_TYPE
```

前提条件

开发环境	目标平台	待集成的 PLC 库（类别组）
TwinCAT v3.1.0	PC 或 CX（x86、x64、Arm®）	Tc2_System（系统）

5.7 E_TcEventStreamType

```
TYPE E_TcEventStreamType :
(
    TCEVENTSTREAM_INVALID       := 0, (* no source name, no prog id *)
    TCEVENTSTREAM_SIMPLE,             (* no source name, no prog id *)
    TCEVENTSTREAM_NORMAL,             (* source name AND prog id *)
    TCEVENTSTREAM_NOSOURCE,           (* no source name, but prog id *)
    TCEVENTSTREAM_CLASSID,            (* source name AND class id *)
    TCEVENTSTREAM_CLSNOSRC,           (* no source name but class id *)
    TCEVENTSTREAM_READCLASSCOUNT,     (* *)
    TCEVENTSTREAM_MAXTYPE             (* no source name but class id *)
);
END_TYPE
```

前提条件

开发环境	目标平台	待集成的 PLC 库（类别组）
TwinCAT v3.1.0	PC 或 CX（x86、x64、Arm®）	Tc2_System（系统）

5.8 E_TcMemoryArea

F_CheckMemoryArea [▶ 79] 函数返回有关所请求的指定大小的变量所在内存区域的信息。为此，将使用 E_TcMemoryArea 类型的返回值。

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_TcMemoryArea :
(
    Unknown := 0,
    Static  := 1, // static PLC memory
    Dynamic := 2, // dynamic memory
    CNC     := 3
)UDINT;
END_TYPE
```

名称	描述
Unknown	内存区域未知。例如，这可能是 Windows 上下文中的内存。如果指定的内存大小涉及两个不同的内存区域，则内存区域也会输出为未知。此外，如果内存区域为堆栈内存，则内存区域输出为未知。
Static	静态 PLC 内存。
Dynamic	动态分配内存，在运行时或 PLC 的初始化阶段进行分配。
CNC	CNC 驱动器的内存。

开发环境	目标平台	待集成的 PLC 库（类别组）
TwinCAT v3.1.4022	PC 或 CX（x86、x64、Arm®）	Tc2_System（系统）

5.9 E_UsrLED_Color

```
TYPE E_UsrLED_Color :
(
    eUsrLED_Off   := 0,
    eUsrLED_Red   := 1,
    eUsrLED_Blue  := 2,
    eUsrLED_Green := 3
);
END_TYPE
```

5.10 EPlcMappingStatus

```
Type EPlcMappingStatus :
(
    MS_Unmapped,
    MS_Mapped,
    MS_Partial
);
End_TYPE
```

该数据类型在全局类型系统中定义。

开发环境	目标平台	待集成的 PLC 库
TwinCAT v3.1.4020	PC 或 CX（x86、x64、Arm®）	Tc2_System（系统）

5.11 ST_AmsAddr

该类型的变量包含 TwinCAT 网络地址。

```
TYPE ST_AmsAddr :
STRUCT
    netId   : T_AmsNetIdArr;
```

```
    port    : T_AmsPort;
END_STRUCT
END_TYPE
```

名称	类型	描述
netId	T_AmsNetIdArr	AMS 网络地址（类型：T_AmsNetIdArr [▶ 116]）
port	T_AmsPort	AMS 端口号（类型：T_AmsPort [▶ 116]）

前提条件

开发环境	目标平台	待集成的 PLC 库（类别组）
TwinCAT v3.1.0	PC 或 CX（x86、x64、Arm®）	Tc2_System（系统）

5.12 ST_CpuCoreInfo

该类型的变量包含有关 CPU 内核的信息。借助函数 F_GetCpuCoreInfo [▶ 82] 和对应的 CPU 内核索引，可以读取特定 CPU 内核的信息。

```
TYPE ST_CpuCoreInfo :
STRUCT
    bRTCore       : BOOL;
    bIsolatedCore : BOOL;
    nBaseTime     : UDINT;
    nCoreLimit    : UDINT;
END_STRUCT
END_TYPE
```

名称	类型	描述
bRTCore	BOOL	如果是实时内核，则该变量的值为 TRUE。
bIsolatedCore	BOOL	如果是隔离核，则该变量的值为 TRUE。
nBaseTime	UDINT	CPU 内核的基准时间，指定为 100 ns 的整数倍。
nCoreLimit	UDINT	CPU 内核的内核限值，单位为 %。

开发环境	目标平台	待集成的 PLC 库
TwinCAT v3.1.4024.11	PC 或 CX（x86、x64、Arm®）	Tc2_System（系统） >= 3.4.24.0

5.13 SYSTEMINFOTYPE

该 TwinCAT 2 数据类型在 TwinCAT 3 中已不再存在。

SystemInfoType 由 PlcAppSystemInfo 取代，后者为全局数据类型。

5.14 SYSTEMTASKINFOTYPE

该 TwinCAT 2 数据类型在 TwinCAT 3 中已不再存在。

SystemTaskInfoType 由 PlcTaskSystemInfo 取代，后者为全局数据类型。

5.15 T_AmsNetID

该类型的 PLC 变量是一个字符串，其中包含 ADS 命令所指向的目标设备的 AMS 网络 ID。该字符串由 6 个数字字段组成，以点隔开。例如，有效的 AMS 网络地址为“1.1.1.2.7.1”或“200.5.7.170.1.7”。如果传递的是空字符串，则自动采用本地设备的 AMS 网络 ID。

命名空间：Tc2_System
库：Tc2_System（Tc2_System.compiled-library）

```
TYPE T_AmsNetID : STRING(23);
END_TYPE
```

5.16 T_AmsNetIdArr

```
TYPE T_AmsNetIdArr : ARRAY[0..5] OF BYTE;
END_TYPE
```

该类型的变量是一个包含 AMS 网络标识符的字节数组。地址字节采用网络字节序。例如，“127.16.17.3.1.1”表示为：

byte[0] = 127
byte[1] = 16
byte[2] = 17
byte[3] = 3
byte[4] = 1
byte[5] = 1

示例： F_ScanAmsNetIds [▶ 100]

前提条件

开发环境	目标平台	待集成的 PLC 库（类别组）
TwinCAT v3.1.0	PC 或 CX（x86、x64、Arm®）	Tc2_System（系统）

5.17 T_AmsPort

该类型的变量包含 ADS 端口号。TwinCAT 网络中的 ADS 设备由 AMS 网络地址和端口号标识。每个独立的 TwinCAT 系统均须指定以下端口号。

```
TYPE T_AmsPort : UINT;
END_TYPE
```

指定 ADS 端口号对照表：

ADS 设备	端口号
Cam controller	900
Runtime system 1 Runtime system 2 Runtime system 3 Runtime system 4 Runtime system 5 Runtime system n	Runtime system 1：851（在 TwinCAT 2 中：801） Runtime system 2：852（在 TwinCAT 2 中：811） Runtime system 3：853（在 TwinCAT 2 中：821） Runtime system 4：854（在 TwinCAT 2 中：831） Runtime system 5：855 Runtime system n：850 + n，以此类推
NC	500
Reserved	400
I/O	300
Real-time core	200
Event System (logger)	100

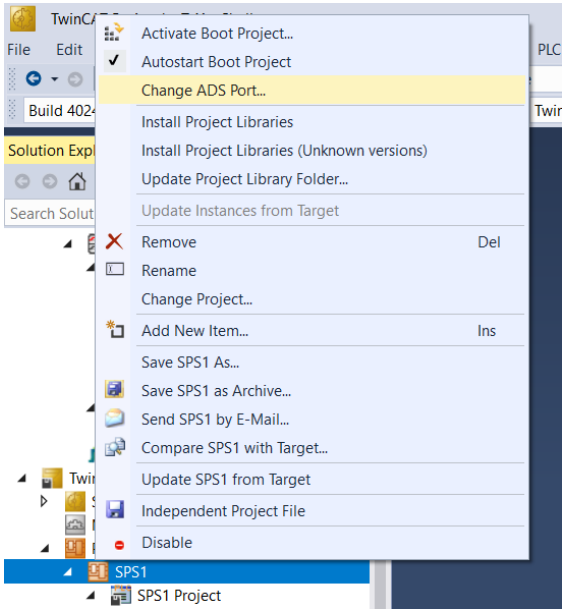
在一个 TwinCAT 2 系统上最多可以运行四个独立的 PLC Runtime 系统（PLC 项目）；每个 PLC 项目都被视为一个独立的 PLC。在调用 ADS 块时，端口号和网络地址都需要作为输入参数。

在 TwinCAT 3 中，ADS 端口 800 至 899 通常分配给 PLC 使用。不过，为了区分 TwinCAT 2 和 TwinCAT 3 系统，我们建议仅使用端口 851 至 899。

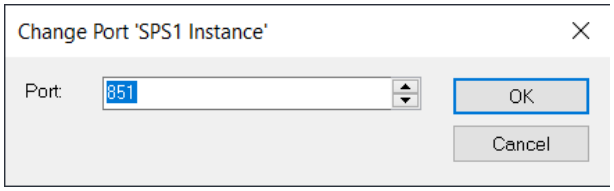
如果通过对话框设置 ADS 端口，则可设置的最低端口为 851 端口。若要使用 800-850 的端口，则需要手动输入端口号。

通过对话框输入端口号的步骤如下：

1. 右键点击所需的 PLC 项目。
2. 点击 **Change ADS Port（更改 ADS 端口）**。



⇨ 对话框随即打开。



3. 使用箭头键选择所需的端口号，或输入所需的端口号。
4. 选择 **OK（确定）**，确认输入。

⇨ 端口号已输入系统。

为了确保能够区分 TwinCAT 2 和 TwinCAT 3 系统，我们建议仅使用 ADS 端口 851 至 899。

输入系统不接受 800 至 899 范围之外的 ADS 端口。

前提条件

开发环境	目标平台	待集成的 PLC 库（类别组）
TwinCAT v3.1.0	PC 或 CX（x86、x64、Arm®）	Tc2_System（系统）

5.18 T_IPv4Addr

该类型的变量是一个包含（Ipv4）互联网协议网络地址的字符串。例如，“172.16.7.199”。

```
TYPE T_IPv4Addr : STRING(15);
END_TYPE
```

前提条件

开发环境	目标平台	待集成的 PLC 库（类别组）
TwinCAT v3.1.0	PC 或 CX（x86、x64、Arm®）	Tc2_System（系统）

5.19 T_IPv4AddrArr

该类型的变量是一个包含（IPv4）互联网协议网络地址的字节数组。

```
TYPE T_IPv4AddrArr: ARRAY[0..3] OF BYTE;
END_TYPE
```

地址字节按网络字节序表示。
例如，“172.16.7.199”表示为：

byte[0] := 172
byte[1] := 16
byte[2] := 7
byte[3] := 199

示例： F_ScanIPv4AddrIds [▶ 81]

前提条件

开发环境	目标平台	待集成的 PLC 库（类别组）
TwinCAT v3.1.0	PC 或 CX（x86、x64、Arm®）	Tc2_System（系统）

5.20 T_MaxString

此类变量为 PLC 字符串，具有最大长度。允许使用更长的字符串，但字符串函数的限值为 255 个字符。

```
TYPE T_MaxString : STRING(MAX_STRING_LENGTH);
END_TYPE
```

```
VAR_GLOBAL CONSTANT
    MAX_STRING_LENGTH : UDINT := 255;
ENd_VAR
```

前提条件

开发环境	目标平台	待集成的 PLC 库（类别组）
TwinCAT v3.1.0	PC 或 CX（x86、x64、Arm®）	Tc2_System（系统）

5.21 TcEvent

TwinCAT EventLogger 与 TwinCAT 3 EventLogger

TwinCAT EventLogger 已被其后续版本 TwinCAT 3 EventLogger 取代。TwinCAT 3 最高支持 3.1.4024 版本的旧版 TwinCAT EventLogger 。较新版本的 TwinCAT（>= 3.1.4026.0）仅支持较新的 TwinCAT 3 EventLogger。相关 PLC 功能块可在 PLC 库 Tc3_EventLogger 中找到。

```
TYPE TcEvent
STRUCT
    Class               : UDINT;
    Prio                : UDINT;
    Id                  : UDINT;
    bQuitRequired       : BOOL;
    DataFormatStrAddress : PVOID;
    UserFlags           : DWORD;
    Flags               : DWORD;
    StreamType          : UDINT;
    SourceString        : STRING[15]; (* TCEVENT_SRCNAMESIZE *)
    SourceId            : UDINT;
    ProgId              : STRING[31]; (* TCEVENT_FMTPRGSIZE *)
END_STRUCT
END_TYPE
```

名称	类型	描述
Class	UDINT	事件类，从枚举 E_TcEventClass [▶ 113] 中获取值。
Prio	UDINT	事件在类中的优先级，可自由选择数值（1..MaxUDINT）
Id	UDINT	事件的 Id，用于在 EventLogger 中进行唯一标识。
bQuitRequired	BOOL	用于切换确认要求的标志（TRUE → 需要确认）
DataFormatStrAddress	PVOID	字符串的地址，字符串包含格式化指令（例如，%d%f 格式化整数和实数（浮点数）值）。
UserFlags	DWORD	可自定义的 32 位数值
Flags	DWORD	标识事件的 32 位数值，在库的全局变量 [▶ 120]中声明了各个位的含义。
StreamType	UDINT	事件的类型，从枚举 E_TcEventStreamType [▶ 113] 中获取值。
SourceString	STRING	包含源名称的字符串（最多 15 个字符 [▶ 120]）
SourceId	UDINT	源 ID
ProgId	STRING	包含格式化程序名称的字符串（Prog-Id）（最多 31 个字符 [▶ 120]）。默认： “TcEventLogger.TcLogFormatter”或 “TcEventFormatter.TcXmlFormatter”

要求

开发环境	目标平台	待集成的 PLC 库（类别组）
TwinCAT v3.1.0 最高 TwinCAT v3.1.4024	PC 或 CX（x86、x64、ARM）	Tc2_System（系统）

6 全局常量

6.1 常量

端口号

端口号	值	描述
AMSPORT_LOGGER	100	标准记录器的端口号。
AMSPORT_EVENTLOG	110	TwinCAT EventLogger 的端口号。
AMSPORT_R0_RTIME	200	TwinCAT Realtime Server 的端口号。
AMSPORT_R0_IO	300	TwinCAT I/O Server 的端口号。
AMSPORT_R0_NC	500	TwinCAT NC Server 的端口号。
AMSPORT_R0_NCSAF	501	TwinCAT NC Server（任务 SAF）的端口号。
AMSPORT_R0_NCSVB	511	TwinCAT NC Server（任务 SVB）的端口号。
AMSPORT_R0_ISG	550	内部使用。
AMSPORT_R0_CNC	600	TwinCAT NC I Server 的端口号。
AMSPORT_R0_LINE	700	内部使用。
AMSPORT_R0_PLC	800	TwinCAT PLC Server 的端口号（仅限总线控制器）。
AMSPORT_R0_PLC_RTS1	801	TwinCAT 2.xx PLC Server runtime 1 的端口号。
AMSPORT_R0_PLC_RTS2	811	TwinCAT 2.xx PLC Server runtime 2 的端口号。
AMSPORT_R0_PLC_RTS3	821	TwinCAT 2.xx PLC Server runtime 3 的端口号。
AMSPORT_R0_PLC_RTS4	831	TwinCAT 2.xx PLC Server runtime 4 的端口号。
AMSPORT_R0_CAM	900	TwinCAT CAM Server 的端口号。
AMSPORT_R0_CAMTOOL	950	TwinCAT CAMTOOL Server 的端口号。
AMSPORT_R3_SYSSERV	10000	TwinCAT System Service 的端口号。
AMSPORT_R3_SCOPESERVER	14001	TwinCAT Scope Server 的端口号。

Ads 状态

ADS 状态	值	描述
ADSSTATE_INVALID	0	ADS 状态：无效
ADSSTATE_IDLE	1	ADS 状态：空闲
ADSSTATE_RESET	2	ADS 状态：重置
ADSSTATE_INIT	3	ADS 状态：初始化
ADSSTATE_START	4	ADS 状态：启动
ADSSTATE_RUN	5	ADS 状态：运行
ADSSTATE_STOP	6	ADS 状态：停止
ADSSTATE_SAVECFG	7	ADS 状态：保存配置
ADSSTATE_LOADCFG	8	ADS 状态：加载配置
ADSSTATE_POWERFAILURE	9	ADS 状态：电源故障
ADSSTATE_POWERGOOD	10	ADS 状态：电源良好
ADSSTATE_ERROR	11	ADS 状态：错误
ADSSTATE_SHUTDOWN	12	ADS 状态：关闭
ADSSTATE_SUSPEND	13	ADS 状态：暂停
ADSSTATE_RESUME	14	ADS 状态：恢复
ADSSTATE_CONFIG	15	ADS 状态：配置（系统处于配置模式）
ADSSTATE_RECONFIG	16	ADS 状态：重新配置（系统应在配置模式下重新启动）
ADSSTATE_STOPPING	17	
ADSSTATE_INCOMPATIBLE	18	
ADSSTATE_EXCEPTION	19	
ADSSTATE_MAXSTATES	20	可用 ads 状态的最大数量

ADS/系统服务

预留索引组	值	描述
ADSIGRP_SYMTAB	16#F000	
ADSIGRP_SYMNAME	16#F001	
ADSIGRP_SYMVAL	16#F002	
ADSIGRP_SYM_HNDBYNAME	16#F003	
ADSIGRP_SYM_VALBYNAME	16#F004	
ADSIGRP_SYM_VALBYHND	16#F005	
ADSIGRP_SYM_RELEASEHND	16#F006	
ADSIGRP_SYM_INFOBYNAME	16#F007	
ADSIGRP_SYM_VERSION	16#F008	
ADSIGRP_SYM_INFOBYNAMEEX	16#F009	
ADSIGRP_SYM_DOWNLOAD	16#F00A	
ADSIGRP_SYM_UPLOAD	16#F00B	
ADSIGRP_SYM_UPLOADINFO	16#F00C	
ADSIGRP_SYMNOTE	16#F010	
ADSIGRP_IOIMAGE_RWIB	16#F020	
ADSIGRP_IOIMAGE_RWIX	16#F021	
ADSIGRP_IOIMAGE_RISIZE	16#F025	
ADSIGRP_IOIMAGE_RWOB	16#F030	
ADSIGRP_IOIMAGE_RWOX	16#F031	
ADSIGRP_IOIMAGE_RWOSIZE	16#F035	
ADSIGRP_IOIMAGE_CLEARI	16#F040	
ADSIGRP_IOIMAGE_CLEARO	16#F050	
ADSIGRP_IOIMAGE_RWIOB	16#F060	
ADSIGRP_DEVICE_DATA	16#F100	
ADSIOFFS_DEVDATA_ADSSTATE	16#0000	
ADSIOFFS_DEVDATA_DEVSTATE	16#0002	

系统服务文件服务

系统服务索引组	值	描述
SYSTEMSERVICE_OPENCREATE	100	
SYSTEMSERVICE_OPENREAD	101	
SYSTEMSERVICE_OPENWRITE	102	
SYSTEMSERVICE_CREATEFILE	110	
SYSTEMSERVICE_CLOSEHANDLE	111	
SYSTEMSERVICE_FOPEN	120	
SYSTEMSERVICE_FCLOSE	121	
SYSTEMSERVICE_FREAD	122	
SYSTEMSERVICE_FWRITE	123	
SYSTEMSERVICE_FSEEK	124	
SYSTEMSERVICE_FTELL	125	
SYSTEMSERVICE_FGETS	126	
SYSTEMSERVICE_FPUTS	127	
SYSTEMSERVICE_FSCANF	128	
SYSTEMSERVICE_FPRINTF	129	
SYSTEMSERVICE_FEOF	130	
SYSTEMSERVICE_FDELETE	131	
SYSTEMSERVICE_FRENAME	132	
SYSTEMSERVICE_REG_HKEYLOCALMACHINE	200	
SYSTEMSERVICE_SENDEMAIL	300	
SYSTEMSERVICE_TIMESERVICES	400	
SYSTEMSERVICE_STARTPROCESS	500	
SYSTEMSERVICE_CHANGENETID	600	

系统服务时间服务

系统服务索引偏移（时间服务）	值	描述
TIMESERVICE_DATEANDTIME	1	
TIMESERVICE_SYSTEMTIMES	2	
TIMESERVICE_RTCTIMEDIFF	3	
TIMESERVICE_ADJUSTTIMETORTC	4	

ADSLOG 消息类型

日志输出掩码	值	描述
ADSLOG_MSGTYPE_HINT	16#01	
ADSLOG_MSGTYPE_WARN	16#02	
ADSLOG_MSGTYPE_ERROR	16#04	
ADSLOG_MSGTYPE_LOG	16#10	
ADSLOG_MSGTYPE_MSGBOX	16#20	
ADSLOG_MSGTYPE_RESOURCE	16#40	
ADSLOG_MSGTYPE_STRING	16#80	

BOOTDATA 标志

Bootdata 标志的掩码	值	描述
BOOTDATAFLAGS_RETAIN_LOADED	16#01	
BOOTDATAFLAGS_RETAIN_INVALID	16#02	
BOOTDATAFLAGS_RETAIN_REQUESTED	16#04	
BOOTDATAFLAGS_PERSISTENT_LOADED	16#10	
BOOTDATAFLAGS_PERSISTENT_INVALID	16#20	

BSOD 标志的掩码	值	描述
SYSTEMSTATEFLAGS_BSOD	16#01	BSOD：蓝屏死机
SYSTEMSTATEFLAGS_RTVIOLATION	16#02	实时性违规延迟超限

文件输出模式

文件输出的掩码	值	描述
FOPEN_MODEREAD	16#0001	“r”：打开文件进行读取
FOPEN_MODEWRITE	16#0002	“w”：打开文件进行读取，（如有）现有文件被覆盖。
FOPEN_MODEAPPEND	16#0004	“a”：打开文件进行读取，附加到（如有）现有文件。如果不存在文件，则将创建文件。
FOPEN_MODEPLUS	16#0008	“+”：打开文件进行读取和写入。
FOPEN_MODEBINARY	16#0010	“b”：打开文件，以二进制形式读取和写入。
FOPEN_MODETEXT	16#0020	“t”：打开文件，以文本形式读取和写入：

EventLogger 常量

EventLogger 标志的掩码	值	描述
TCEVENTFLAG_PRIOCLASS	16#0010	类和优先级由格式化程序定义。
TCEVENTFLAG_FMTSELF	16#0020	事件附带的格式化信息。
TCEVENTFLAG_LOG	16#0040	日志记录。
TCEVENTFLAG_MSGBOX	16#0080	显示消息框。
TCEVENTFLAG_SRCID	16#0100	使用源 Id 取代源名称。

TwinCAT EventLogger 状态消息	值	描述
TCEVENTSTATE_INVALID	16#0000	无效，如果未报告事件也会出现。
TCEVENTSTATE_SIGNALED	16#0001	已报告事件，但未注销或确认。
TCEVENTSTATE_RESET	16#0002	事件已注销（“已清除”）。
TCEVENTSTATE_CONFIRMED	16#0010	事件已确认。
TCEVENTSTATE_RESETCON	16#0012	事件已注销并确认

TwinCAT EventLogger 状态消息	值	描述
TCEVENT_SRCNAMESIZE	15	源名称的最大长度。
TCEVENT_FMTPRGSIZE	31	格式化程序名称的最大长度。

其它	值	描述
PI	3.14159265358979323846264338 32795	Pi 数值
DEFAULT_ADS_TIMEOUT	T#5s	默认 ADS 超时
MAX_STRING_LENGTH	255	T_MaxString 数据类型的最大字符串长度

前提条件

开发环境	目标平台	待集成的 PLC 库（类别组）
TwinCAT v3.1.0	PC 或 CX（x86、x64、Arm®）	Tc2_System（系统）

6.2 库版本

所有库都有特定版本。例如，版本信息会在 PLC 功能库中标注。全局常量包含有关库版本的信息：

Global_Version

```
VAR_GLOBAL CONSTANT
    stLibVersion_Tc2_System : ST_LibVersion;
END_VAR
```

名称	类型	描述
stLibVersion_Tc2_System	ST_LibVersion	Tc2_System 库的版本号（类型：ST_LibVersion）

要检查您拥有的版本是否为所需版本，请使用 F_CmpLibVersion [▶ 79] 函数。

在 TwinCAT 2 中您所熟知的所有库版本对比选项均已过时。

前提条件

开发环境	目标平台	待集成的 PLC 库（类别组）
TwinCAT v3.1.0	PC 或 CX（x86、x64、Arm®）	Tc2_System（系统）

7 示例

7.1 使用 AdsReadInd/AdsReadRes 功能块的示例

该示例展示了在 PLC 上实施一个简单的 ADS Server 应用程序。该服务端应用可以处理 ADS Client 应用程序的 ADSREAD 请求。

在该示例应用中，ADSREAD 请求用于在 PLC 任务中增加/减少或重置 PLC 计数器变量。如果成功，计数器变量的值将返回至 ADS Client 应用程序。

ADS 服务端应用程序的完整源代码可以在此处解压：https://infosys.beckhoff.com/content/1033/TcPlcLib_Tc2_System/Resources/710574987.zip

在此处可以找到适用于 ADS Server 的 ADS Client 应用程序：使用 ADSREAD 功能块的示例。 [▶ 129]

ADS Client 应用程序

PLC 任务中所需的服务通过索引组参数进行编码：

IG:0x80000001 → 增加计数器变量；

IG:0x80000002 → 减少计数器变量；

IG:0x80000003 → 设置计数器变量 = 0；

索引偏移参数为零。

为了使请求能够路由到 PLC 任务，在索引组参数中必须设置最高有效位。

PLC 程序

作为 PLC 任务中的指示，ADSREAD 请求由 ADSREADIND [▶ 25] 功能块实例拦截。然后，检查索引组和索引偏移参数以及所需的数据长度和有效性。在 CASE 指令中，对 PLC 变量执行所需操作。如果成功，ADSREADRES [▶ 32] 功能块实例将向调用者发回一个响应，其中包含 PLC 变量的当前值。如果出现错误，则会显示相应的错误信息。在下一个周期中，功能块上的 CLEAR 和 RESPOND 标志将被重置，以便能够处理更多指示。

声明部分

```
PROGRAM MAIN
VAR
    fbReadInd    : ADSREADIND;(* Indication function block instance *)
    fbReadRes    : ADSREADRES;(* Response function block instance *)
    sNetId       : T_AmsNetID;
    nPort        : T_AmsPort;
    nInvokeId    : UDINT;
    nIdxGrp      : UDINT;
    nIdxOffs     : UDINT;
    cbLength     : UDINT;(* Requested read data/buffer byte size *)
    cbRead       : UDINT;(* Returned read data/buffer byte size *)
    pRead        : PVOID;(* Pointer to returned read data/buffer *)
    nErrID       : UDINT;(* Read indication result error code *)
    nCounter     : INT;(* Server data *)
END_VAR
```

实施

```
fbReadRes( RESPOND := FALSE );(* Reset response function block *)
 fbReadInd( CLEAR := FALSE );(* Trigger indication function block *)
 IF fbReadInd.VALID THEN(* Check for new indication *)

      sNetID := fbReadInd.NETID;
      nPort := fbReadInd.PORT;
      nInvokeID := fbReadInd.INVOKEID;
      nIdxGrp := fbReadInd.IDXGRP;
      nIdxOffs := fbReadInd.IDXOFFS;
      cbLength := fbReadInd.LENGTH;
```

```
      cbRead := 0;
      pRead := 0;
      nErrID     := DEVICE_SRVNOTSUPP;


     CASE nIdxGrp OF
          (*-----------------------------------------------------------------------------------------*)
          16#80000001:
               CASE nIdxOffs OF
                    0:(* Increment counter value *)
                         IF cbLength >= SIZEOF(nCounter) THEN
                              nCounter := nCounter + 1;
                              cbRead := SIZEOF(nCounter);
                              pRead := ADR(nCounter);
                              nErrID := NOERR;
                         ELSE (* ADS error (example): Invalid size *)
                              nErrID := DEVICE_INVALIDSIZE;
                         END_IF
               ELSE (* ADS error (example): Invalid index offset *)
                    nErrID := DEVICE_INVALIDOFFSET;
               END_CASE
          (*-----------------------------------------------------------------------------------------*)
          16#80000002:
               CASE nIdxOffs OF
                    0:(* Decrement counter value *)
                         IF cbLength >= SIZEOF(nCounter) THEN
                              nCounter := nCounter - 1;
                              cbRead := SIZEOF(nCounter);
                              pRead := ADR(nCounter);
                              nErrID := NOERR;
                         ELSE(* ADS error (example): Invalid size *)
                              nErrID := DEVICE_INVALIDSIZE;
                         END_IF
               ELSE (* ADS error (example): Invalid index offset *)
                    nErrID := DEVICE_INVALIDOFFSET;
               END_CASE
          (*-----------------------------------------------------------------------------------------*)
          16#80000003:
               CASE nIdxOffs OF
                    0:(* Reset counter value *)
                         IF cbLength >= SIZEOF(nCounter) THEN
                              nCounter := 0;
                              cbRead := SIZEOF(nCounter);
                              pRead := ADR(nCounter);
                              nErrID := NOERR;
                         ELSE(* ADS error (example): Invalid size *)
                              nErrID := DEVICE_INVALIDSIZE;
                         END_IF
               ELSE (* ADS error (example): ervice is not supported by server *)
                    nErrID := DEVICE_SRVNOTSUPP;
               END_CASE

     ELSE (* ADS error (example): Invalid index group *)
          nErrID := DEVICE_INVALIDGRP;
     END_CASE

     fbReadRes(     NETID := sNetID,
                    PORT := nPort,
                    INVOKEID := nInvokeID,
                    LEN := cbRead,
                    DATAADDR := pRead,
                    RESULT := nErrID,
                    RESPOND := TRUE );(* Send read response *)

     fbReadInd( CLEAR := TRUE ); (* Clear indication entry *)
END_IF
```

7.2 使用 AdsWriteInd/AdsWriteRes 功能块的示例

该示例展示了在 PLC 上实施一个简单的 ADS Server 应用程序。该服务端应用可以处理 ADS Client 应用程序的 ADSWRITE 请求。

在该示例应用中，ADSWRITE 请求用于将包含整数值的数组传输到 PLC 任务。接收到的数据被复制到 PLC 中的相应数组变量中。

ADS 服务端应用程序的完整源代码可以在此处解压：https://infosys.beckhoff.com/content/1033/TcPlcLib_Tc2_System/Resources/710582923.zip

在此处可以找到适用于 ADS Server 的 ADS Client 应用程序：使用 ADSWRITE 功能块的示例。 [▶ 129]

ADS Client 应用程序

PLC 任务所需的服务/命令通过索引组和索引偏移参数进行编码。例如：

IG:0x80000005 和 IO:0x00000007→ 将发送的数据复制到 PLC 中的数组。

为了使请求能够路由到 PLC 任务，在索引组参数中必须设置最高有效位。

PLC 程序

作为 PLC 任务中的指示，请求由 ADSWRITEIND [▶ 28] 功能块实例拦截。随后，检查索引组、索引偏移和传输数据长度参数是否有效，并对 PLC 变量执行所需的操作。接着通过 ADSWRITERES [▶ 33] 功能块实例向调用者返回响应（包括错误代码，如适用）。在下一个周期中，CLEAR 和 RESPOND 标志将被重置，以便能够处理更多指示。

当 ADSWRITEIND 功能块的 CLEAR 输入端出现上升沿时，指向最近发送数据的地址指针将失效（== ZERO）。

因此，在将 CLEAR 输入端设置为 TRUE 之前，首先要将发送的数据复制到 PLC 变量。

声明部分

```
PROGRAM MAIN
VAR
   fbWriteInd  : ADSWRITEIND;
   fbWriteRes  : ADSWRITERES;
   sNetId      : T_AmsNetID;
   nPort       : T_AmsPort;
   nInvokeId   : UDINT;
   nIdxGrp     : UDINT;
   nIdxOffs    : UDINT;
   cbWrite     : UDINT;(* Byte size of written data *)
   pWrite      : PVOID;(* Pointer to written data buffer *)
   nResult     : UDINT;(* Write indication result error code *)
   arrInt      : ARRAY[0..9] OF INT;(* Server data *)
END_VAR
```

实施

```
fbWriteRes( RESPOND := FALSE );(* Reset response function block *)
fbWriteInd( CLEAR := FALSE );(* Trigger indication function block *)
IF ( fbWriteInd.VALID ) THEN

   sNetId       := fbWriteInd.NETID;
   nPort        := fbWriteInd.PORT;
   nInvokeId    := fbWriteInd.INVOKEID;
   nIdxGrp      := fbWriteInd.IDXGRP;
   nIdxOffs     := fbWriteInd.IDXOFFS;
   cbWrite      := fbWriteInd.LENGTH;
   pWrite       := fbWriteInd.DATAADDR;
   nResult      := DEVICE_SRVNOTSUPP;

   CASE nIdxGrp OF
      16#80000005:
         CASE nIdxOffs OF
            16#00000007:
               IF cbWrite <= SIZEOF( arrInt ) THEN
                  MEMCPY( ADR( arrInt ), pWrite, MIN( cbWrite, SIZEOF(arrInt) ) );
                  nResult := NOERR;
               ELSE(* ADS error (example): Invalid size *)
                  nResult := DEVICE_INVALIDSIZE;
               END_IF
         ELSE(* ADS error (example): Invalid index offset *)
            nResult := DEVICE_INVALIDOFFSET;
         END_CASE
   ELSE(* ADS error (example): Invalid index group *)
      nResult := DEVICE_INVALIDGRP;
   END_CASE

   fbWriteRes( NETID := sNetId,
            PORT := nPort,
            INVOKEID := nInvokeId,
            RESULT := nResult,
            RESPOND := TRUE ); (* Send write response *)

   fbWriteInd( CLEAR := TRUE ); (* Clear indication entry *)
END_IF
```

7.3 使用 AdsRead 功能块的示例

该示例演示了 ADS Client 应用程序中 ADSREAD 功能块的使用方法。

ADS Client 应用程序的完整源代码可以在此处解压：https://infosys.beckhoff.com/content/1033/TcPlcLib_Tc2_System/Resources/710578827.zip

声明部分

```
PROGRAM MAIN
VAR
    fbReadReq  : ADSREADEX := ( NETID := '', PORT := 851, TMOUT := DEFAULT_ADS_TIMEOUT );
    bIncrement : BOOL;(* Rising edge at this variable starts command execution *)
    bDecrement : BOOL;(* Rising edge at this variable starts command execution *)
    bReset     : BOOL;(* Rising edge at this variable starts command execution *)
    bOther     : BOOL;(* Rising edge at this variable starts command execution *)

    nState     : BYTE;
    bBusy      : BOOL;
    bError     : BOOL;
    nErrID     : UDINT;
    cbRead     : UDINT;

    nCounter   : INT;(* Server data to be read *)
END_VAR
```

实施

```
CASE nState OF
    0:
        IF bIncrement OR bDecrement OR bReset OR bOther THEN
            bBusy := TRUE;
            bError := FALSE;
            nErrID := 0;

            fbReadReq( READ := FALSE );

            IF bIncrement THEN(* Incement counter value *)
                bIncrement := FALSE;
                fbReadReq( IDXGRP := 16#80000001, IDXOFFS := 0, LEN := SIZEOF(nCounter), DESTADDR :=
 ADR(nCounter), READ := TRUE );
            ELSIF bDecrement THEN(* Decrement counter value *)
                bDecrement := FALSE;
                fbReadReq( IDXGRP := 16#80000002, IDXOFFS := 0, LEN := SIZEOF(nCounter), DESTADDR :=
 ADR(nCounter), READ := TRUE );
            ELSIF bReset THEN(* Reset counter value *)
                bReset := FALSE;
                fbReadReq( IDXGRP := 16#80000003, IDXOFFS := 0, LEN := SIZEOF(nCounter), DESTADDR :=
 ADR(nCounter), READ := TRUE );
            ELSIF bOther THEN(* Call unsupported function *)
                bOther := FALSE;
                fbReadReq( IDXGRP := 16#80000004, IDXOFFS := 0, LEN := SIZEOF(nCounter), DESTADDR :=
 ADR(nCounter), READ := TRUE );
            END_IF

            nState := 1;
        END_IF
    1:
        fbReadReq( READ := FALSE, BUSY=>bBusy, ERR=>bError, ERRID=>nErrID, COUNT_R=>cbRead );
        IF NOT bBusy THEN
            IF NOT bError THEN
                nState := 0;(* Success *)
            ELSE
                nState := 100;(* Error *)
            END_IF
        END_IF

    100:(* TODO::Implement error handler *)
        nState := 0;

END_CASE
```

7.4 使用 AdsWrite 功能块的示例

该示例演示了 ADS 客户端应用程序中 ADSWRITE 功能块的使用。

ADS Client 应用程序的完整源代码可以在此处解压缩：https://infosys.beckhoff.com/content/1033/TcPlcLib_Tc2_System/Resources/710586763.zip

声明部分

```
PROGRAM MAIN
VAR
   fbWriteReq   : ADSWRITE := ( NETID := '', PORT := 851, TMOUT := DEFAULT_ADS_TIMEOUT );
   bWrite       : BOOL;(* Rising edge at this variable starts command execution *)
   nState       : BYTE;
   bBusy        : BOOL;
   bError       : BOOL;
   nErrID       : UDINT;
   arrInt       : ARRAY[0..9] OF INT;(* Server data to be written *)
   i            : INT;
END_VAR
```

实施

```
FOR i:=0 TO 9 BY 1 DO (* modify/simulate new data *)
   arrInt[i] := arrInt[i] + 1;
END_FOR


CASE nState OF
   0:
      IF bWrite THEN
         bWrite := FALSE;

         bBusy := TRUE;
         bError := FALSE;
         nErrID := 0;

         fbWriteReq( WRITE := FALSE );
         fbWriteReq( IDXGRP := 16#80000005, IDXOFFS := 7,
                  LEN := SIZEOF( arrInt ), SRCADDR := ADR( arrInt ),
                  WRITE := TRUE );
         nState := 1;
      END_IF

   1:
      fbWriteReq( WRITE := FALSE, BUSY=>bBusy, ERR=>bError, ERRID=>nErrID );
      IF NOT bBusy THEN
         IF NOT bError THEN
            nState := 0;(* Success *)
         ELSE
            nState := 100;(* Error *)
         END_IF
      END_IF

   100:(* TODO::Implement error handler *)
      nState := 0;

END_CASE
```

7.5 从 PLC 发送/确认 EventLogger

该示例演示了 ADSLOGEVENT 功能块的使用方法。

示例应用程序的完整源代码可以在此处解压缩：https://infosys.beckhoff.com/content/1033/TcPlcLib_Tc2_System/Resources/711421451.zip

分步执行顺序

事件配置：

EventConfigData [▶ 118] 结构参数化

参数传输：

使用 ADR 运算符在 EventDataAddress 处生成结构体、数组或单个变量的地址。使用 SIZEOF 运算符确定结构体、数组或单个变量的长度，并将其应用于 EventDataLength 输入。例如，如果要将一个包含 INT 和 LREAL 变量的结构体与事件一起传输，就必须创建并实例化一个包含这两部分的结构体。必须传输该实例的地址和长度。

设置事件：

事件输入端的上升沿

重置事件：

事件输入端的下降沿

确认事件：

退出输入端的上升沿

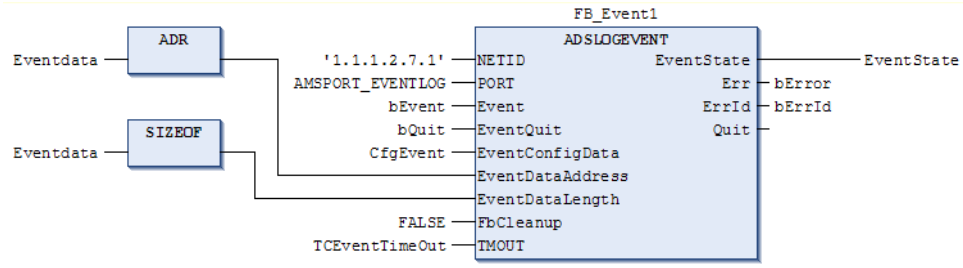
完全删除实例：

通过 FbCleanup 输入端的上升沿完全删除实例的内容。此操作不会将 EventLogger 现有事件直接删除。

在将事件发送到 EventLogger 事件的状态 [▶ 120]会在 Eventstate 输出端发生明显变化。

调用 ADSLOGEVENT 功能块

```
PROGRAM MAIN
VAR
    FB_Event1    : ADSLOGEVENT;
    CfgEvent     : TcEvent;
    Eventdata    : ParaStruct;
    EventState   : UDINT;
    bEvent       : BOOL;
    bQuit        : BOOL;
END_VAR
VAR CONSTANT
    TCEventDataFormatString : STRING:='%f%d';
    TcEventTimeOut      : TIME:=T#1s;
END_VAR
```



声明部分

```
PROGRAM MAIN
VAR
    CfgEvent                : TcEvent;
    fbEvent                 : ADSLOGEVENT;
    bSetEvent               : BOOL;(* Rising edge sets event *)

    eventData               : ST_EventData;
    TcEventDataFormatString : STRING    := '%f%d';
END_VAR
```

实施

```
CfgEvent.Class := TCEVENTCLASS_ALARM;
CfgEvent.Prio := 2;
CfgEvent.Id := 1;
CfgEvent.SourceId := 100;
CfgEvent.bQuitRequired := TRUE;
CfgEvent.DataFormatStrAddress := ADR(TcEventDataFormatString);
CfgEvent.Flags := TCEVENTFLAG_LOG OR TCEVENTFLAG_SRCID OR TCEVENTFLAG_AUTOFMTALL;
CfgEvent.StreamType := TCEVENTSTREAM_SIMPLE;
CfgEvent.ProgId :='TcEventFormatter.TcXMLFormatter' ;

eventData.rVal     := 2.65;
eventData.iVal    := 3;

fbEvent( NETID:= '',
        PORT:= 110,
        Event:= bSetEvent,
        EventConfigData:= CfgEvent,
        EventDataAddress := ADR(eventData) ,
        EventDataLength := SIZEOF(eventData),
        TMOUT:= t#3s);
```

7.6 从 PLC 访问文件

本示例介绍了如何使用 PLC 功能块从 Tc2_system 库中访问数据。借助现有功能块可以实现新功能块 FB_FileCopy。使用 FB_FileCopy 功能块，可以在本地 TwinCAT 系统或本地与远程 TwinCAT 系统之间复制二进制文件。

示例项目的完整源代码可以从此处解压缩：https://infosys.beckhoff.com/content/1033/TcPlcLib_Tc2_System/Resources/707895179.zip

使用 FB_FileCopy 功能块无法访问网络驱动器。

当 FB_FileCopy 功能块的 bExecute 输入端出现上升沿时，将执行以下步骤。

a) 打开源文件和目标文件

b) 将源文件读入缓冲区

c) 将从缓冲区读取的字节写入目标文件

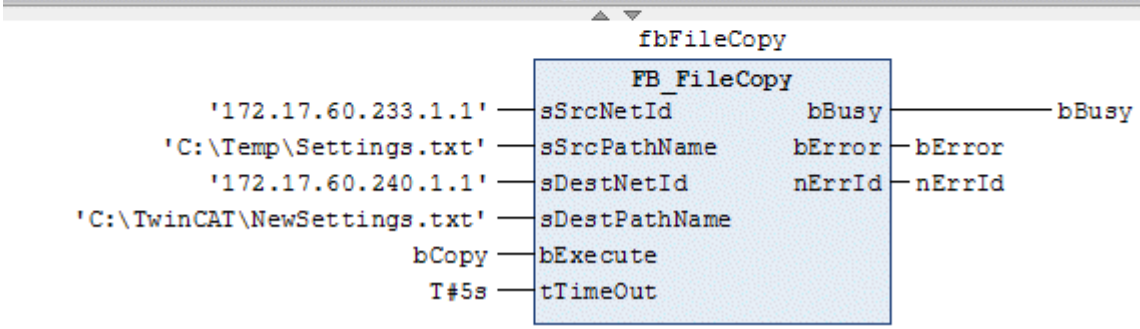
d) 检查是否已到达源文件末尾。如果不是，则重复 b) 和 c)。如果是，则跳至 e)

e) 关闭源文件和目标文件；

文件按段逐次复制。在此示例中，缓冲区的大小指定为 1000 字节，但可以进行修改。

PLC 程序

```
PROGRAM MAIN
VAR
    fbFileCopy  : FB_FileCopy;
    bCopy       : BOOL;
    bBusy       : BOOL;
    bError      : BOOL;
    nErrId      : UDINT;
END_VAR
```



声明部分

```
FUNCTION_BLOCK FB_FileCopy
VAR_INPUT
    sSrcNetId           : T_AmsNetId;
    sSrcPathName        : T_MaxString;
    sDestNetId          : T_AmsNetId;
    sDestPathName       : T_MaxString;
    bExecute            : BOOL;
    tTimeOut            : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
VAR_OUTPUT
    bBusy               : BOOL;
    bError              : BOOL;
    nErrId              : UDINT;
END_VAR
VAR
    fbFileOpen          : FB_FileOpen;
    fbFileClose         : FB_FileClose;
    fbFileRead          : FB_FileRead;
    fbFileWrite         : FB_FileWrite;
    hSrcFile            : UINT := 0;(* File handle of the source file *)
    hDestFile           : UINT := 0;(* File handle of the destination file *)

    Step                : DWORD;
    RisingEdge          : R_TRIG;
```

```
    buffRead             : ARRAY[1..1000] OF BYTE;(* Buffer *)
    cbReadLength         : UDINT := 0;
END_VAR
```

实施

```
RisingEdge(CLK:=bExecute);

CASE Step OF
     0:     (* Idle state *)
          IF RisingEdge.Q THEN
               bBusy := TRUE;
               bError:= FALSE;
               nErrId:=0;
               Step := 1;
               cbReadLength:=0;
               hSrcFile:=0;
               hDestFile:=0;
          END_IF

     1:     (* Open source file *)
          fbFileOpen( bExecute := FALSE );
          fbFileOpen( sNetId := sSrcNetId, sPathName := sSrcPathName,
                         nMode := FOPEN_MODEREAD OR FOPEN_MODEBINARY,
                         ePath := PATH_GENERIC, tTimeout := tTimeOut, bExecute := TRUE );
          Step := Step + 1;
     2:
          fbFileOpen( bExecute := FALSE );
          IF NOT fbFileOpen.bBusy THEN
               IF fbFileOpen.bError THEN
                    nErrId := fbFileOpen.nErrId;
                    bError := TRUE;
                    Step := 50;
               ELSE
                    hSrcFile := fbFileOpen.hFile;
                    Step := Step + 1;
               END_IF
          END_IF

     3:     (* Open destination file *)
          fbFileOpen( bExecute := FALSE );
          fbFileOpen( sNetId := sDestNetId, sPathName := sDestPathName,
                         nMode := FOPEN_MODEWRITE OR FOPEN_MODEBINARY,
                         ePath := PATH_GENERIC, tTimeout := tTimeOut, bExecute := TRUE );
          Step := Step+1;
     4:
          fbFileOpen( bExecute := FALSE );
          IF NOT fbFileOpen.bBusy THEN
               IF fbFileOpen.bError THEN
                    nErrId := fbFileOpen.nErrId;
                    bError := TRUE;
                    Step := 50;
               ELSE
                    hDestFile := fbFileOpen.hFile;
                    Step := Step + 1;
               END_IF
          END_IF

     5:     (* Read data from source file *)
          cbReadLength := 0;
          fbFileRead( bExecute:= FALSE );
          fbFileRead( sNetId:=sSrcNetId, hFile:=hSrcFile,
                         pReadBuff:= ADR(buffRead), cbReadLen:= SIZEOF(buffRead),
                         bExecute:=TRUE, tTimeout:=tTimeOut );
          Step := Step + 1;
     6:
          fbFileRead( bExecute:= FALSE );
          IF NOT fbFileRead.bBusy THEN
               IF fbFileRead.bError THEN
                    nErrId := fbFileRead.nErrId;
                    bError := TRUE;
                    Step := 50;
               ELSE
                    cbReadLength := fbFileRead.cbRead;
                    Step := Step + 1;
               END_IF
          END_IF

     7:     (* Write data to destination file *)
          fbFileWrite( bExecute := FALSE );
          fbFileWrite( sNetId:=sDestNetId, hFile:=hDestFile,
                         pWriteBuff:= ADR(buffRead), cbWriteLen:= cbReadLength,
                         bExecute:=TRUE, tTimeout:=tTimeOut );
          Step := Step + 1;
     8:
          fbFileWrite( bExecute := FALSE );
          IF NOT fbFileWrite.bBusy THEN
               IF fbFileWrite.bError THEN
                    nErrId := fbFileWrite.nErrId;
                    bError := TRUE;
                    Step := 50;
               ELSE
                    IF fbFileRead.bEOF THEN (* Check if the EOF flag ist set *)
```

```
                    Step := 50;     (* Cleanup: close the destination and source files *)
                ELSE
                    Step := 5; (* Repeat reading/writing *)
                END_IF
            END_IF
        END_IF

    30:     (* Close the destination file *)
        fbFileClose( bExecute := FALSE );
        fbFileClose( sNetId:=sDestNetId, hFile:=hDestFile, bExecute:=TRUE, tTimeout:=tTimeOut );
        Step := Step + 1;
    31:
        fbFileClose( bExecute := FALSE );
        IF NOT fbFileClose.bBusy THEN
            IF fbFileClose.bError THEN
                nErrId := fbFileClose.nErrId;
                bError := TRUE;
            END_IF
            Step := 50;
            hDestFile := 0;
        END_IF

    40: (* Close source file *)
        fbFileClose( bExecute := FALSE );
        fbFileClose( sNetId:=sSrcNetId, hFile:=hSrcFile, bExecute:=TRUE, tTimeout:=tTimeOut );
        Step := Step + 1;
    41:
        fbFileClose( bExecute := FALSE );
        IF NOT fbFileClose.bBusy THEN
            IF fbFileClose.bError THEN
                nErrId := fbFileClose.nErrId;
                bError := TRUE;
            END_IF
            Step := 50;
            hSrcFile := 0;
        END_IF

    50: (* Error or ready => Cleanup *)
        IF ( hDestFile <> 0 ) THEN
            Step := 30; (* Close the destination file*)
        ELSIF (hSrcFile <> 0 ) THEN
            Step := 40; (* Close the source file *)
        ELSE
            Step := 0;      (* Ready *)
            bBusy := FALSE;
        END_IF

END_CASE
```

7.7 测试 CX70xx 的 CPU 剩余性能

此示例是一个测试项目，用于测试 CX70xx 的 CPU 剩余性能。所配备的功能块 FB_Test_CPU_Performance 可测量当前应用程序的 CPU 剩余性能。该功能块可读取当前 CPU 算力和周期时间。随后，该功能块会增加 CPU 负载，直至 CX7k 不再实时运行。接着，将再次减少负载，直至达到稳定的实时状态。最后，该功能块会确定 CPU 算力和周期时间，并将其与测量初始时间和负载进行对比，从而得出变化量。该功能块仅可用于测试目的，而不能在实际环境中使用。

如果 CPU 剩余性能大于 20%，则可以调整任务周期时间，使其快于当前使用速度。任务速度更快的好处是对输入的反应更快，而且应用程序速度也更快，具体取决于程序内容。数毫秒的累积即可提升机器的输出效率。理想的剩余算力为 20%。

示例项目的源代码可以在此处解压缩：https://infosys.beckhoff.com/content/1033/TcPlcLib_Tc2_System/Resources/11298888331.zip

失真的测量结果

如果在低优先级任务中使用该功能块，结果将会失真，无法确定真实数据。由于测量时间较长，慢速任务也会被考虑在内。

- 如果可能，应当始终在速度最快的高优先级任务中使用该功能块。

测量过程中有未运行的循环或程序段会导致测量结果失真

循环、多任务以及由此产生的存在剧烈波动的周期时间会导致 CPU 负载剧烈波动。

- 将功能块的 TimeOut 设置为一个较大的值，因为功能块会搜索最高的 CPU 负载，该过程比处理恒定的 CPU 负载耗时更长。

中止测量

只有在配置不存在实时性违规的情况下，才能使用该功能块。如果该功能块在启动时已经读取到实时性违规数据，则该功能块会中止测量。

关于功能块 `FB_Test_CPU_Performance` 的信息：

VAR_INPUT

名称	类型	描述
bExecute	BOOL	上升沿激活功能块。
tTimeOut	TIME	超过限值时，停止测量。

VAR_OUTPUT

名称	类型	描述
bBusy	BOOL	功能块处于活动状态并开始工作
bError	BOOL	功能块发生错误。
nErrorID	UDINT	ADS 错误代码
nCpuLoadReserve	UDINT	CPU 的剩余性能，[%]
fCycleTimeReserve	LREAL	周期时间余量，[ms]

开发环境	目标平台	待集成的 PLC 库
TwinCAT v3.1.4024.22	PC 或 CX（x86、x64、ARM）	Tc2_System（系统） >= 3.4.25.0

8 附录

8.1 ADS 返回代码

错误代码分组：
全局错误代码：--- FEHLENDER LINK ---... （0x9811_0000 ...）
路由器错误代码：--- FEHLENDER LINK ---... （0x9811_0500 ...）
一般 ADS 错误：--- FEHLENDER LINK ---... （0x9811_0700 ...）
RTime 错误代码：--- FEHLENDER LINK ---... （0x9811_1000 ...）

全局错误代码

Hex	Dec	HRESULT	名称	描述
0x0	0	0x98110000	ERR_NOERROR	无错误。
0x1	1	0x98110001	ERR_INTERNAL	内部错误。
0x2	2	0x98110002	ERR_NORTIME	不具有实时性。
0x3	3	0x98110003	ERR_ALLOCLOCKEDMEM	分配已锁定 – 内存错误。
0x4	4	0x98110004	ERR_INSERTMAILBOX	邮箱已满 – 无法发送 ADS 消息。减少每个周期的 ADS 消息数量将有所帮助。
0x5	5	0x98110005	ERR_WRONGRECEIVEHMSG	HMSG 错误。
0x6	6	0x98110006	ERR_TARGETPORTNOTFOUND	未找到目标端口 – ADS 服务器未启动或无法访问。
0x7	7	0x98110007	ERR_TARGETMACHINENOTFOUND	未找到目标计算机 – 未找到 AMS 路由。
0x8	8	0x98110008	ERR_UNKNOWNCMDID	未知命令 ID。
0x9	9	0x98110009	ERR_BADTASKID	任务 ID 无效。
0xA	10	0x9811000A	ERR_NOIO	No IO。
0xB	11	0x9811000B	ERR_UNKNOWNAMSCMD	未知 AMS 命令。
0xC	12	0x9811000C	ERR_WIN32ERROR	Win32 错误。
0xD	13	0x9811000D	ERR_PORTNOTCONNECTED	端口未连接。
0xE	14	0x9811000E	ERR_INVALIDAMSLENGTH	AMS 长度无效。
0xF	15	0x9811000F	ERR_INVALIDAMSNETID	AMS Net ID 无效。
0x10	16	0x98110010	ERR_LOWINSTLEVEL	安装级别 – TwinCAT 2 授权错误。
0x11	17	0x98110011	ERR_NODEBUGINTAVAILABLE	无调试可用。
0x12	18	0x98110012	ERR_PORTDISABLED	端口已禁用 – TwinCAT 系统服务未启动。
0x13	19	0x98110013	ERR_PORTALREADYCONNECTED	端口已连接。
0x14	20	0x98110014	ERR_AMSSYNC_W32ERROR	AMS Sync Win32 错误。
0x15	21	0x98110015	ERR_AMSSYNC_TIMEOUT	AMS Sync 超时。
0x16	22	0x98110016	ERR_AMSSYNC_AMSERROR	AMS Sync 错误。
0x17	23	0x98110017	ERR_AMSSYNC_NOINDEXINMAP	不存在适用于 AMS Sync 的索引映射。
0x18	24	0x98110018	ERR_INVALIDAMSPORT	AMS 端口无效。
0x19	25	0x98110019	ERR_NOMEMORY	无内存。
0x1A	26	0x9811001A	ERR_TCPSEND	TCP 发送错误。
0x1B	27	0x9811001B	ERR_HOSTUNREACHABLE	主机无法访问。
0x1C	28	0x9811001C	ERR_INVALIDAMSFRAGMENT	AMS 片段无效。
0x1D	29	0x9811001D	ERR_TLSSEND	TLS 发送错误 – ADS 安全连接失败。
0x1E	30	0x9811001E	ERR_ACCESSDENIED	拒绝访问 – 拒绝 ADS 安全访问。

路由器错误代码

Hex	Dec	HRESULT	名称	描述
0x500	1280	0x98110500	ROUTERERR_NOLOCKEDMEMORY	无法分配锁定的内存。
0x501	1281	0x98110501	ROUTERERR_RESIZEMEMORY	路由器内存大小无法更改。
0x502	1282	0x98110502	ROUTERERR_MAILBOXFULL	邮箱已达到最大消息数。
0x503	1283	0x98110503	ROUTERERR_DEBUGBOXFULL	调试邮箱已达到最大消息数。
0x504	1284	0x98110504	ROUTERERR_UNKNOWNPORTTYPE	端口类型未知。
0x505	1285	0x98110505	ROUTERERR_NOTINITIALIZED	路由器未初始化。
0x506	1286	0x98110506	ROUTERERR_PORTALREADYINUSE	端口号已分配。
0x507	1287	0x98110507	ROUTERERR_NOTREGISTERED	端口未注册。
0x508	1288	0x98110508	ROUTERERR_NOMOREQUEUES	已达到最大端口数。
0x509	1289	0x98110509	ROUTERERR_INVALIDPORT	端口无效。
0x50A	1290	0x9811050A	ROUTERERR_NOTACTIVATED	路由未激活。
0x50B	1291	0x9811050B	ROUTERERR_FRAGMENTBOXFULL	邮箱已达到最大片段消息数。
0x50C	1292	0x9811050C	ROUTERERR_FRAGMENTTIMEOUT	发生片段超时。
0x50D	1293	0x9811050D	ROUTERERR_TOBEREMOVED	端口已移除。

一般性 ADS 错误代码

Hex	Dec	HRESULT	名称	描述
0x700	1792	0x98110700	ADSERR_DEVICE_ERROR	一般性设备错误。
0x701	1793	0x98110701	ADSERR_DEVICE_SRVNOTSUPP	服务器不支持该服务。
0x702	1794	0x98110702	ADSERR_DEVICE_INVALIDGRP	索引组无效。
0x703	1795	0x98110703	ADSERR_DEVICE_INVALIDOFFSET	索引偏移量无效。
0x704	1796	0x98110704	ADSERR_DEVICE_INVALIDACCESS	不允许读取或写入。 可能有几种原因。例如，创建路由时输入了错误的密码。
0x705	1797	0x98110705	ADSERR_DEVICE_INVALIDSIZE	参数大小不正确。
0x706	1798	0x98110706	ADSERR_DEVICE_INVALIDDATA	无效数据值。
0x707	1799	0x98110707	ADSERR_DEVICE_NOTREADY	设备尚未准备好运行。
0x708	1800	0x98110708	ADSERR_DEVICE_BUSY	设备正忙。
0x709	1801	0x98110709	ADSERR_DEVICE_INVALIDCONTEXT	操作系统上下文无效。这可能是由于在不同的任务中使用 ADS 函数块造成的。可以通过在 PLC 中进行多任务同步解决这个问题。
0x70A	1802	0x9811070A	ADSERR_DEVICE_NOMEMORY	内存不足。
0x70B	1803	0x9811070B	ADSERR_DEVICE_INVALIDPARM	参数值无效。
0x70C	1804	0x9811070C	ADSERR_DEVICE_NOTFOUND	未找到（文件…）。
0x70D	1805	0x9811070D	ADSERR_DEVICE_SYNTAX	文件或命令中存在语法错误。
0x70E	1806	0x9811070E	ADSERR_DEVICE_INCOMPATIBLE	对象不匹配。
0x70F	1807	0x9811070F	ADSERR_DEVICE_EXISTS	对象已存在。
0x710	1808	0x98110710	ADSERR_DEVICE_SYMBOLNOTFOUND	符号未找到。
0x711	1809	0x98110711	ADSERR_DEVICE_SYMBOLVERSIONINVALID	符号版本无效。这可能是由于联机更改造成的。创建新句柄。
0x712	1810	0x98110712	ADSERR_DEVICE_INVALIDSTATE	设备（服务器）处于无效状态。
0x713	1811	0x98110713	ADSERR_DEVICE_TRANSMODENOTSUPP	不支持 AdsTransMode。
0x714	1812	0x98110714	ADSERR_DEVICE_NOTIFYHNDINVALID	通知句柄无效。
0x715	1813	0x98110715	ADSERR_DEVICE_CLIENTUNKNOWN	通知客户端未注册。
0x716	1814	0x98110716	ADSERR_DEVICE_NOMOREHDLS	没有更多的句柄可用。
0x717	1815	0x98110717	ADSERR_DEVICE_INVALIDWATCHSIZE	通知大小过大。
0x718	1816	0x98110718	ADSERR_DEVICE_NOTINIT	设备未初始化。
0x719	1817	0x98110719	ADSERR_DEVICE_TIMEOUT	设备超时。
0x71A	1818	0x9811071A	ADSERR_DEVICE_NOINTERFACE	接口查询失败。
0x71B	1819	0x9811071B	ADSERR_DEVICE_INVALIDINTERFACE	请求的接口错误。
0x71C	1820	0x9811071C	ADSERR_DEVICE_INVALIDCLSID	Class ID 无效。
0x71D	1821	0x9811071D	ADSERR_DEVICE_INVALIDOBJID	Object ID 无效。
0x71E	1822	0x9811071E	ADSERR_DEVICE_PENDING	请求待定。
0x71F	1823	0x9811071F	ADSERR_DEVICE_ABORTED	请求已中止。
0x720	1824	0x98110720	ADSERR_DEVICE_WARNING	警告信号。
0x721	1825	0x98110721	ADSERR_DEVICE_INVALIDARRAYIDX	数组索引无效。
0x722	1826	0x98110722	ADSERR_DEVICE_SYMBOLNOTACTIVE	符号未激活。
0x723	1827	0x98110723	ADSERR_DEVICE_ACCESSDENIED	拒绝访问。 有几种可能的原因。例如，单向 ADS 路由用于相反方向。
0x724	1828	0x98110724	ADSERR_DEVICE_LICENSENOTFOUND	缺少授权。
0x725	1829	0x98110725	ADSERR_DEVICE_LICENSEEXPIRED	授权已过期。
0x726	1830	0x98110726	ADSERR_DEVICE_LICENSEEXCEEDED	超出授权。
0x727	1831	0x98110727	ADSERR_DEVICE_LICENSEINVALID	授权无效。
0x728	1832	0x98110728	ADSERR_DEVICE_LICENSESYSTEMID	授权问题：系统 ID 无效。
0x729	1833	0x98110729	ADSERR_DEVICE_LICENSENOTIMELIMIT	授权不受时间限制。
0x72A	1834	0x9811072A	ADSERR_DEVICE_LICENSEFUTUREISSUE	授权问题：未来的时间。
0x72B	1835	0x9811072B	ADSERR_DEVICE_LICENSETIMETOLONG	授权期限太长。
0x72C	1836	0x9811072C	ADSERR_DEVICE_EXCEPTION	系统启动异常。
0x72D	1837	0x9811072D	ADSERR_DEVICE_LICENSEDUPLICATED	授权文件读取了两次。
0x72E	1838	0x9811072E	ADSERR_DEVICE_SIGNATUREINVALID	签名无效。
0x72F	1839	0x9811072F	ADSERR_DEVICE_CERTIFICATEINVALID	证书无效。
0x730	1840	0x98110730	ADSERR_DEVICE_LICENSEOEMNOTFOUND	OEM未知公钥。
0x731	1841	0x98110731	ADSERR_DEVICE_LICENSERESTRICTED	此系统 ID 授权无效。

Hex	Dec	HRESULT	名称	描述
0x732	1842	0x98110732	ADSERR_DEVICE_LICENSEDEMODENIED	演示授权已禁止。
0x733	1843	0x98110733	ADSERR_DEVICE_INVALIDFNCID	无效函数 ID。
0x734	1844	0x98110734	ADSERR_DEVICE_OUTOFRANGE	超出有效范围。
0x735	1845	0x98110735	ADSERR_DEVICE_INVALIDALIGNMENT	无效对齐。
0x736	1846	0x98110736	ADSERR_DEVICE_LICENSEPLATFORM	无效平台级别。
0x737	1847	0x98110737	ADSERR_DEVICE_FORWARD_PL	上下文 – 转到被动级别。
0x738	1848	0x98110738	ADSERR_DEVICE_FORWARD_DL	上下文 – 转到调度级别。
0x739	1849	0x98110739	ADSERR_DEVICE_FORWARD_RT	上下文 – 转到实时。
0x740	1856	0x98110740	ADSERR_CLIENT_ERROR	客户端错误。
0x741	1857	0x98110741	ADSERR_CLIENT_INVALIDPARM	服务包含无效参数。
0x742	1858	0x98110742	ADSERR_CLIENT_LISTEMPTY	轮询列表为空。
0x743	1859	0x98110743	ADSERR_CLIENT_VARUSED	Var 连接已在使用中。
0x744	1860	0x98110744	ADSERR_CLIENT_DUPLINVOKEID	调用的 ID 已在使用中。
0x745	1861	0x98110745	ADSERR_CLIENT_SYNCTIMEOUT	已发生超时 – 远程终端在指定的 ADS 超时中未响应。远程终端的路由设置可能配置不正确。
0x746	1862	0x98110746	ADSERR_CLIENT_W32ERROR	Win32 子系统中发生错误。
0x747	1863	0x98110747	ADSERR_CLIENT_TIMEOUTINVALID	客户端超时值无效。
0x748	1864	0x98110748	ADSERR_CLIENT_PORTNOTOPEN	端口未打开。
0x749	1865	0x98110749	ADSERR_CLIENT_NOAMSADDR	无 AMS 地址。
0x750	1872	0x98110750	ADSERR_CLIENT_SYNCINTERNAL	Ads sync 中发生内部错误。
0x751	1873	0x98110751	ADSERR_CLIENT_ADDHASH	哈希表溢出。
0x752	1874	0x98110752	ADSERR_CLIENT_REMOVEHASH	未在表格中找到密钥。
0x753	1875	0x98110753	ADSERR_CLIENT_NOMORESYM	缓存中没有符号。
0x754	1876	0x98110754	ADSERR_CLIENT_SYNCRESINVALID	收到无效响应。
0x755	1877	0x98110755	ADSERR_CLIENT_SYNCPORTLOCKED	同步端口已锁定。
0x756	1878	0x98110756	ADSERR_CLIENT_REQUESTCANCELLED	请求被取消。

RTime 错误代码

Hex	Dec	HRESULT	名称	描述
0x1000	4096	0x98111000	RTERR_INTERNAL	实时系统中发生内部错误。
0x1001	4097	0x98111001	RTERR_BADTIMERPERIODS	计时器值无效。
0x1002	4098	0x98111002	RTERR_INVALIDTASKPTR	任务指针提供了无效值 0（零）。
0x1003	4099	0x98111003	RTERR_INVALIDSTACKPTR	堆栈指针提供了无效值 0（零）。
0x1004	4100	0x98111004	RTERR_PRIOEXISTS	请求任务优先级已分配。
0x1005	4101	0x98111005	RTERR_NOMORETCB	没有可用的空闲 TCB（任务控制块）。TCB 的最大数量为 64。
0x1006	4102	0x98111006	RTERR_NOMORESEMAS	没有可用的空闲信号。信号的最大数量为 64。
0x1007	4103	0x98111007	RTERR_NOMOREQUEUES	队列中没有可用的空闲空间。队列中的最大位置数为 64。
0x100D	4109	0x9811100D	RTERR_EXTIRQALREADYDEF	已应用外部同步中断。
0x100E	4110	0x9811100E	RTERR_EXTIRQNOTDEF	未应用外部同步中断。
0x100F	4111	0x9811100F	RTERR_EXTIRQINSTALLFAILED	外部同步中断应用失败。
0x1010	4112	0x98111010	RTERR_IRQLNOTLESSOREQUAL	在错误的上下文中调用服务函数
0x1017	4119	0x98111017	RTERR_VMXNOTSUPPORTED	不支持 Intel VT-x 扩展。
0x1018	4120	0x98111018	RTERR_VMXDISABLED	BIOS 中未启用 Intel VT-x 扩展。
0x1019	4121	0x98111019	RTERR_VMXCONTROLSMISSING	Intel VT-x 扩展中缺少函数。
0x101A	4122	0x9811101A	RTERR_VMXENABLEFAILS	Intel VT-x 激活失败。

具体的正向 HRESULT 返回代码：

HRESULT	名称	描述
0x0000_0000	S_OK	无错误。
0x0000_0001	S_FALSE	无错误。 示例：处理成功，但有一个负面或不完整的结果。
0x0000_0203	S_PENDING	无错误。 示例：处理成功，但还没有结果。
0x0000_0256	S_WATCHDOG_TIMEOUT	无错误。 示例：处理成功，但发生了超时。

TCP Winsock 错误代码

Hex	Dec	名称	描述
0x274C	10060	WSAETIMEDOUT	发生连接超时 - 在创建连接时发生错误，因为远程终端在特定时间后未正确响应，或者所建立连接因所连接主机未响应而无法维持。
0x274D	10061	WSAECONNREFUSED	连接遭到拒绝 - 无法建立连接，因为目标计算机明确拒绝了该连接。此错误通常由尝试连接到外部主机上处于非活动状态的服务（即没有运行服务器应用程序的服务）导致。
0x2751	10065	WSAEHOSTUNREACH	不存在至主机的路由 - 套接字操作引用了不可用的主机。
更多 Winsock 错误代码：Win32 错误代码			

还请参阅有关此

- ADS 返回代码 [▶ 136]
- ADS 返回代码 [▶ 136]
- ADS 返回代码 [▶ 137]
- ADS 返回代码 [▶ 139]

8.2 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.

DALI, DALI-2, D4i, DALI+, and DiiA are trademarks in various countries in the exclusive use of the Digital Illumination Interface Alliance.

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.

更多信息:
www.beckhoff.com/te1000

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