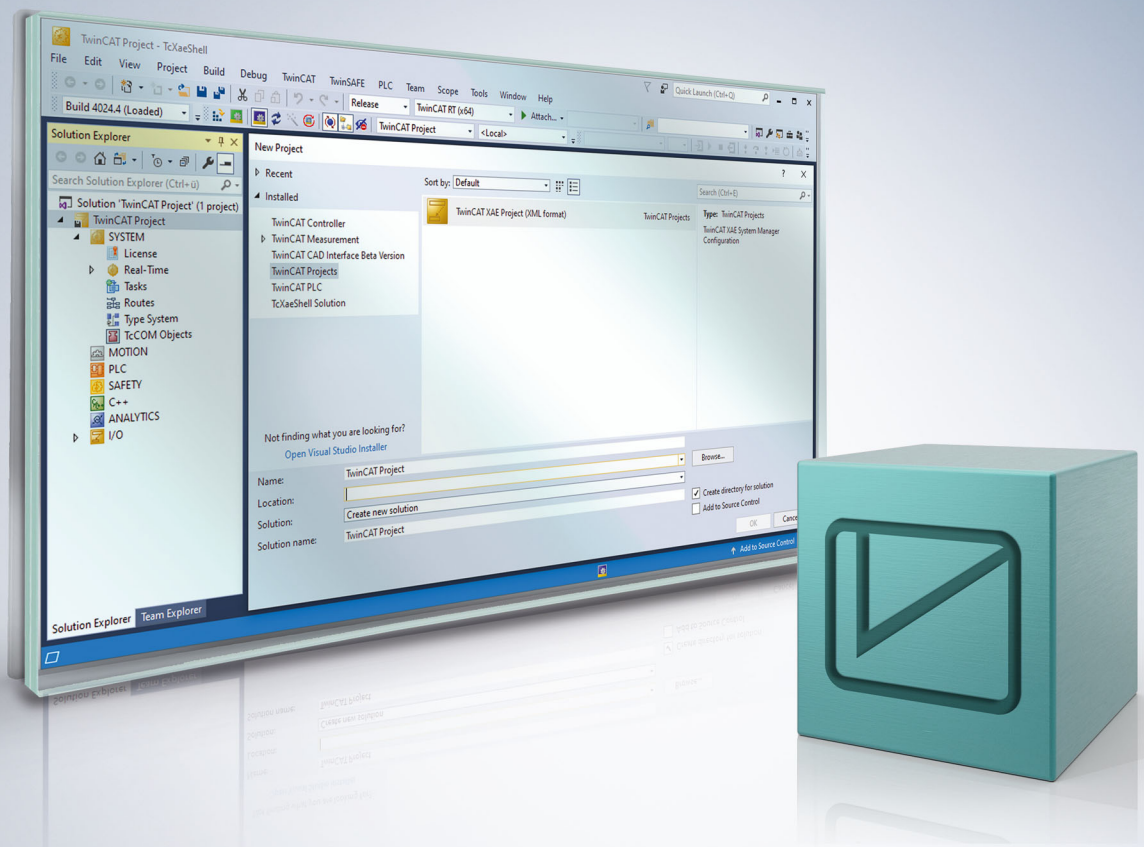
BECKHOFF New Automation Technology

手册 | ZH

# TF4100

## TwinCAT 3 | TwinCAT 3 Controller Toolbox




2026-01-14 | 版本： 2.2.4

# 目录

# 1 前言

## 1.1 文档说明

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

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

### 免责声明

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

### 商标

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

EtherCAT® 是注册商标和专利技术，由德国倍福自动化有限公司授权使用

### 版权所有



### 第三方商标

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

## 1.2 安全信息

### 安全规范

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

### 责任免除

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

### 人员资格

本说明仅供熟悉适用国家标准的控制、自动化和驱动工程专家使用。

### 警示性词语

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

### 人身伤害警告

**⚠ 危险**

存在死亡或重伤的高度风险。

**⚠ 警告**

存在死亡或重伤的中度风险。

**⚠ 谨慎**

存在可能导致中度或轻度伤害的低度风险。

### 财产或环境损害警告

**注意**

可能会损坏环境、设备或数据。

### 操作产品的信息

这些信息包括：
有关产品的操作、帮助或进一步信息的建议。

## 1.3 信息安全说明

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

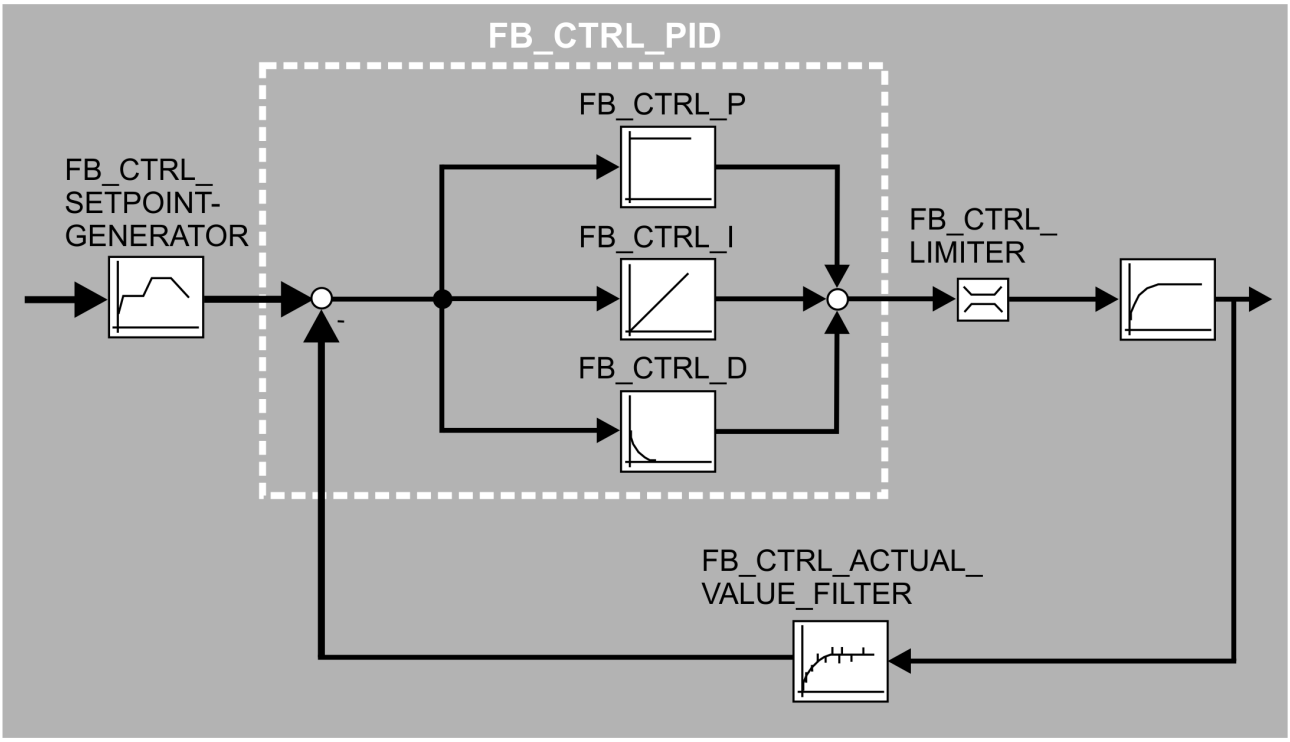
此外，还应遵守 Beckhoff 关于采取适当保护措施的建议。关于信息安全和工业安全的更多信息，请访问本公司网站 https://www.beckhoff.com/secguide。

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

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

# 2 概述

该库包含了一系列功能块，它们在功能图中代表了各种控制工程中的传递环节。其中包括可用于多种应用的复杂控制器，以及可用于针对特殊应用构建独特控制器结构的基本功能块。




**功能块**

| 名称 | 描述 |
|---|---|
| FB_CTRL_2POINT [▶ 37] | 二位式控制器 |
| FB_CTRL_2POINT_PWM_ADAPTIVE [▶ 38] | 带 PWM 输出的自适应二位式控制器 |
| FB_CTRL_3PHASE_SETPOINT_GENERATOR [▶ 139] | 三相设定值发生器 |
| FB_CTRL_3POINT [▶ 41] | 三位式控制器 |
| FB_CTRL_3POINT_EXT [▶ 43] | 扩展型三位式控制器 |
| FB_CTRL_ACTUAL_VALUE_FILTER [▶ 77] | 实际值滤波器 |
| FB_CTRL_ARITHMETIC_MEAN [▶ 78] | 算术平均值滤波器 |
| FB_CTRL_CHECK_IF_IN_BAND [▶ 112] | 区域监控 |
| FB_CTRL_D [▶ 29] | D 环节 |
| FB_CTRL_DEADBAND [▶ 120] | 死区 |
| FB_CTRL_DIGITAL_FILTER [▶ 81] | 数字滤波器 |
| FB_CTRL_FLOW_TEMP_SETPOINT_GEN [▶ 146] | 根据室外温度设定流量温度 |
| FB_CTRL_GET_SYSTEM_TIME [▶ 18] | Windows 系统时间的输出 |
| FB_CTRL_GET_TASK_CYCLETIME [▶ 19] | 确定任务循环时间 |
| FB_CTRL_HYSTERESIS [▶ 23] | 滞后单元 |
| FB_CTRL_I [▶ 26] | 积分单元 |
| FB_CTRL_I_WITH_DRIFTCOMPENSATION [▶ 27] | 带漂移补偿的积分单元 |
| FB_CTRL_LEAD_LAG [▶ 85] | 超前/滞后单元 |
| FB_CTRL_LIMITER [▶ 121] | 控制量限幅器 |
| FB_CTRL_LIN_INTERPOLATION [▶ 107] | 线性插值单元 |
| FB_CTRL_LOG_DATA [▶ 114] | 以 CSV ASCII 格式的数据记录器 |
| FB_CTRL_LOG_MAT_FILE [▶ 116] | Matlab 5 格式的数据记录器 |

| 名称 | 描述 |
|---|---|
| FB_CTRL_LOOP_SCHEDULER [▶ 20] | 多个控制回路场景下的算力分配 |
| FB_CTRL_MOVING_AVERAGE [▶ 84] | 滑动平均滤波器 |
| FB_CTRL_MULTIPLE_PWM_OUT [▶ 123] | 带多个输出的 PWM 单元 |
| FB_CTRL_NORMALIZE [▶ 109] | 特性曲线线性化 |
| FB_CTRL_NOISE_GENERATOR [▶ 88] | 噪声发生器 |
| FB_CTRL_NOTCH_FILTER [▶ 89] | 陷波滤波器 |
| FB_CTRL_nPOINT [▶ 45] | n 位式控制器 |
| FB_CTRL_P [▶ 25] | 比例单元 |
| FB_CTRL_PARAMETER_SWITCH [▶ 47] | 分程控制器的参数切换算法 |
| FB_CTRL_PI [▶ 49] | 比例-积分 控制器 |
| FB_CTRL_PI_PID [▶ 52] | 串级 PI-PID 控制器 |
| FB_CTRL_PID [▶ 55] | PID 控制器 |
| FB_CTRL_PID_EXT [▶ 64] | 扩展型 PID 控制器 |
| FB_CTRL_PID_EXT_SPLITRANGE [▶ 59] | 带参数集切换功能的扩展型 PID 调节器 |
| FB_CTRL_PID_SPLITRANGE [▶ 69] | 带参数集切换功能的 PID 调节器 |
| FB_CTRL_PT1 [▶ 91] | $PT_1$ 单元 |
| FB_CTRL_PT2 [▶ 93] | $PT_2$ 单元 |
| FB_CTRL_PT2oscillation [▶ 94] | 振荡型 $_{PT2}$ 单元 |
| FB_CTRL_PT3 [▶ 96] | $PT_3$ 单元 |
| FB_CTRL_PTn [▶ 98] | $PT_n$ 单元 |
| FB_CTRL_PTt [▶ 100] | $PT_t$ 单元 |
| FB_CTRL_PWM_OUT [▶ 126] | PWM 单元 |
| FB_CTRL_PWM_OUT_EXT [▶ 128] | 扩展型 PWM 单元 |
| FB_CTRL_RAMP_GENERATOR [▶ 148] | 斜坡信号发生器 |
| FB_CTRL_RAMP_GENERATOR_EXT [▶ 150] | 扩展型斜坡信号发生器 |
| FB_CTRL_SCALE [▶ 130] | 范围调整 |
| FB_CTRL_SERVO_MOTOR_OUT [▶ 131] | 执行器控制 |
| FB_CTRL_SERVO_MOTOR_SIMULATION [▶ 101] | 执行器模拟 |
| FB_CTRL_SETPOINT_GENERATOR [▶ 152] | 设定值发生器 |
| FB_CTRL_SIGNAL_GENERATOR [▶ 155] | 信号发生器 |
| FB_CTRL_SPLITRANGE [▶ 134] | 信号的正负分量分解。 |
| FB_CTRL_STEPPING_MOTOR_OUT [▶ 136] | 步进电机控制 |
| FB_CTRL_TRANSFERFUNCTION_1 [▶ 31] | 根据第一标准形式的传递函数 |
| FB_CTRL_TRANSFERFUNCTION_2 [▶ 33] | 根据第二标准形式的传递函数 |
| FB_CTRL_TuTg [▶ 103] | TuTg 单元 |
| FB_CTRL_ZERO_ZONE_DAMPING [▶ 105] | 零阻尼 |

**要求**

| 开发环境 | 目标平台 | 要包含的 PLC 库 |
|---|---|---|
| TwinCAT 3.1.4016 | PC 或 CX | Tc2_ControllerToolbox |

# 3 安装

## 3.1 系统要求

描述工程和/或运行时系统所需的最低要求。

**开发环境**

纯粹的开发环境指的是一种仅用于开发 PLC 程序、但不具备程序执行功能的计算机系统。务必在开发计算机上安装以下组件：

- TwinCAT 3 XAE（工程）版本 4012 或更高版本
- TwinCAT 3 功能组件 TF4100 Controller Toolbox 版本 3.4.0.0 或更高版本
- 请注意：开发环境可使用 7 天试用授权，请参见 授权 [▶ 13]。

**运行时环境**

运行时环境指的是一种用于执行 PLC 程序的计算机系统。务必在运行时计算机上安装以下组件：

- TwinCAT3 XAR 版本 4012 或更高版本
- TC1200 PLC 和 TF4100 Controller Toolbox 的授权
- 请注意：7 天试用授权密钥可用于测试目的，请参见 授权 [▶ 13]。

**一台计算机上集成开发和运行时环境**

如果要在同一台计算机上运行运行时和开发环境（例如，在目标计算机上加载 PLC 程序之前，对其进行测试），则必须满足以下要求：

- TwinCAT3 XAE（工程安装）版本 4012 或更高版本
- TwinCAT 3 功能组件 TF4100 Controller Toolbox 版本 3.4.0.0 或更高版本
- 请注意：7 天试用授权密钥可用于测试目的，请参见 授权 [▶ 13]。

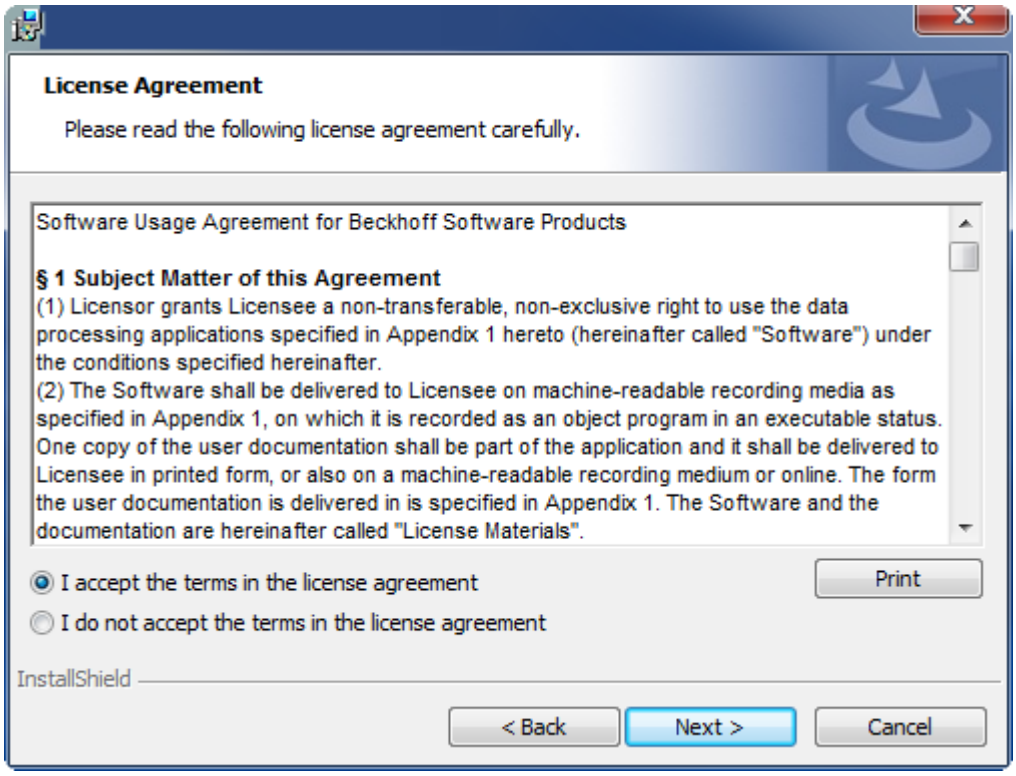
## 3.2 安装

下文将介绍如何在基于 Windows 的操作系统上安装 TwinCAT 3 功能组件。

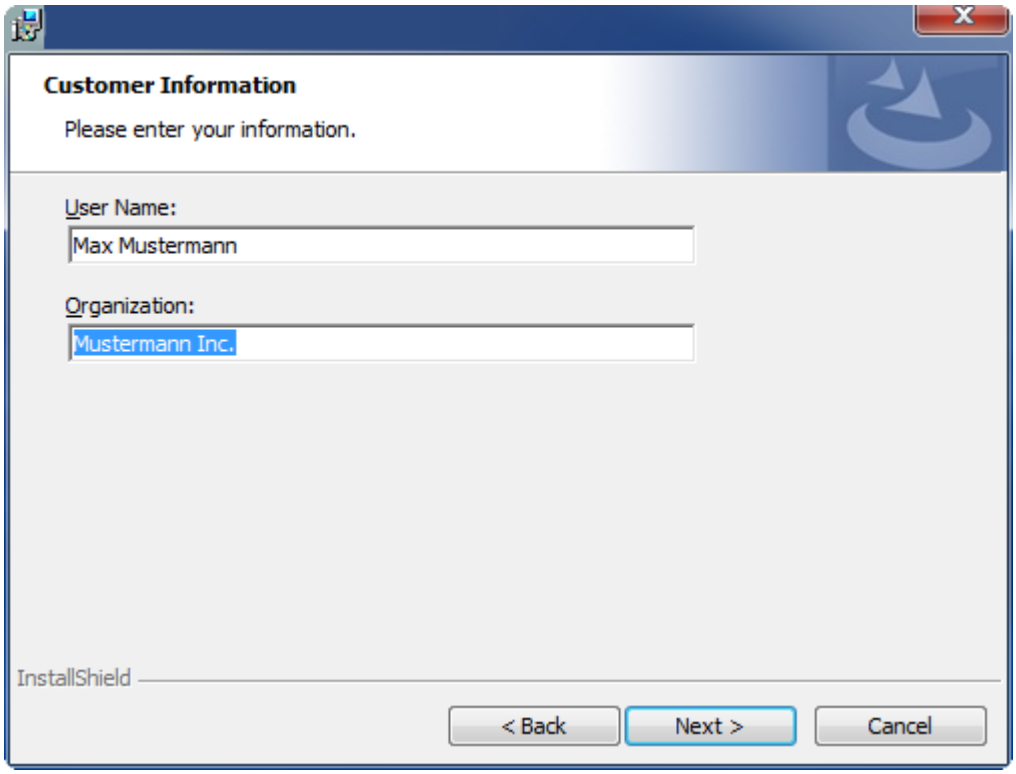
✓ 从倍福网站下载 TwinCAT 3 功能组件的安装文件。

1. 以管理员身份运行安装文件。为此，请右击文件并在菜单中选择“**以管理员身份运行**”命令。
   ⇨ 安装对话框打开。

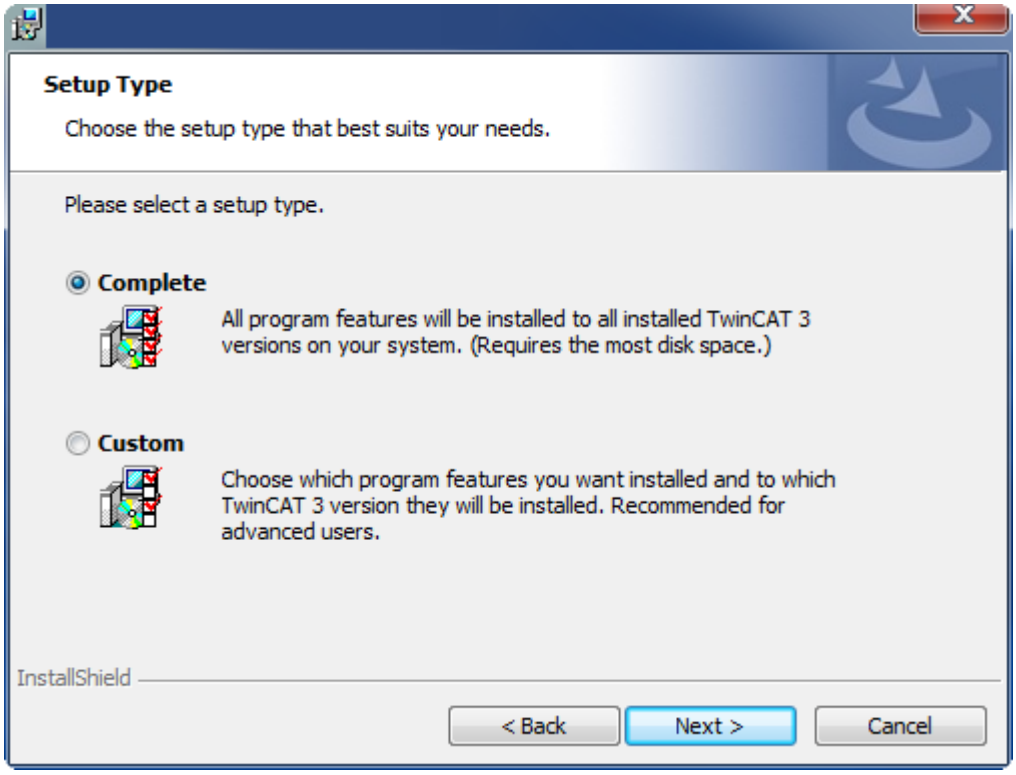
2. 接受最终用户许可协议，然后点击“**Next**”（下一步）。



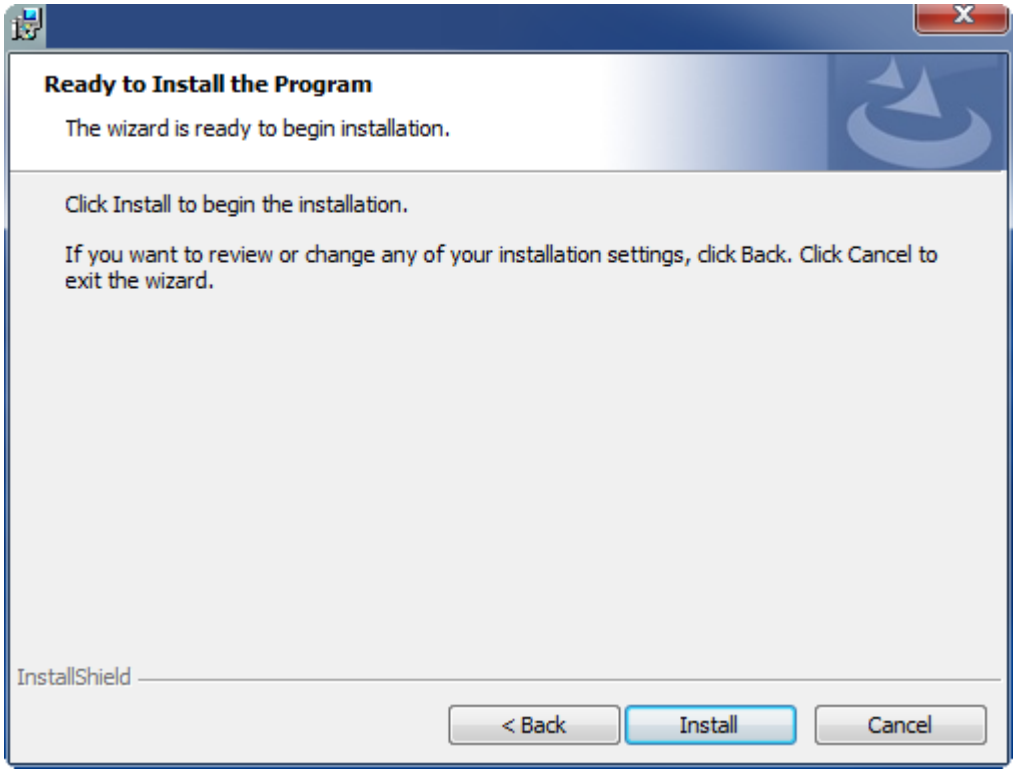

3. 输入用户数据。



4. 如果要安装完整版 TwinCAT 3 功能组件，请选择“**Complete**”（完整版）作为安装类型。如果要单独安装 TwinCAT 3 功能组件，请选择“**Custom**”（自定义）。

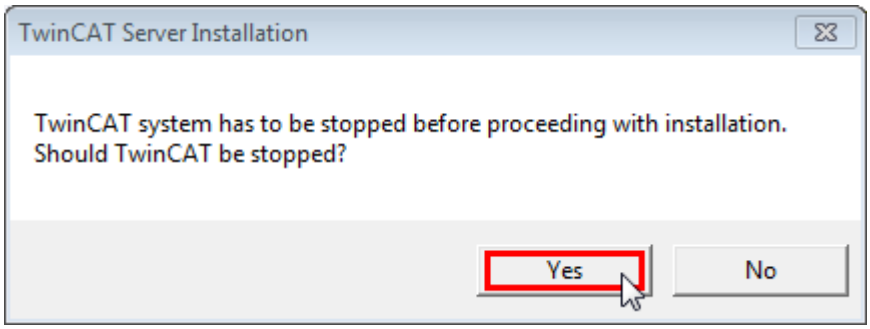



5. 选择“**Next**”（下一步），然后选择“**Install**”（安装）开始安装。

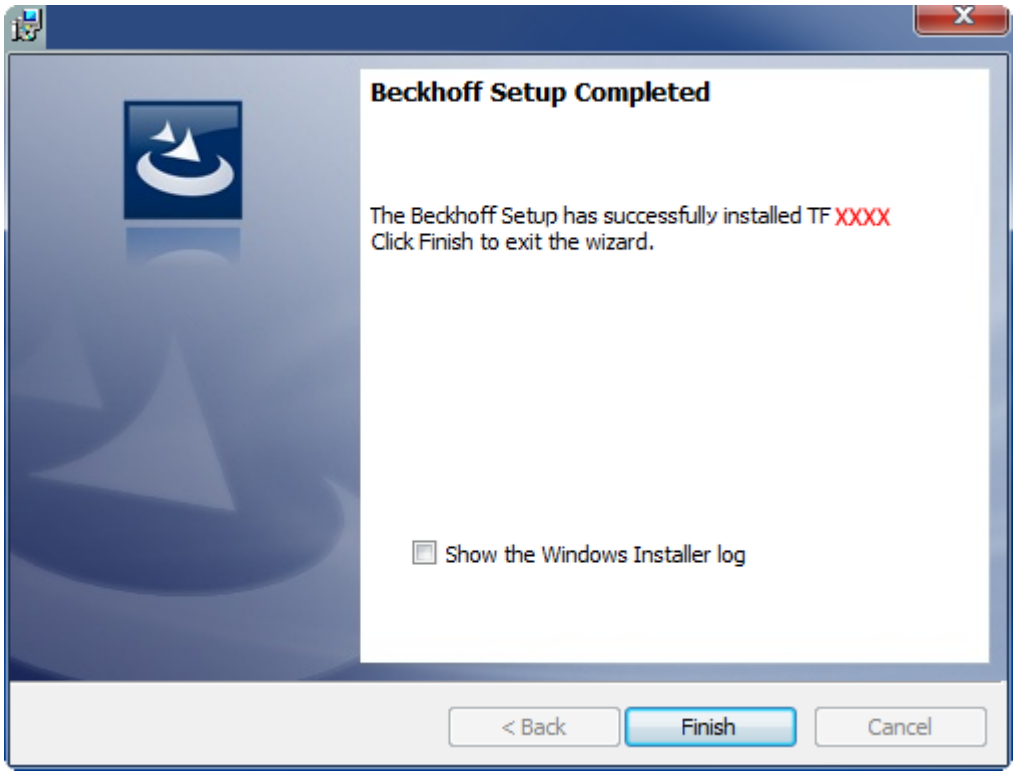



⇨ 将显示一个对话框，提示您必须关闭 TwinCAT 系统才能继续安装。

6. 选择“**Yes**”（是）。




7. 选择“**Finish**”（完成）退出安装。




⇨ TwinCAT 3 功能组件已成功安装。

## 3.3 授权

TwinCAT 3 功能可以作为完整版或 7 天测试版激活。这两种许可证类型都可以通过 TwinCAT 3 开发环境（XAE）激活。

### 授权使用 TwinCAT 3 功能的完整版本

关于许可证完整版本的程序描述，可参见倍福信息系统的文档“TwinCAT 3 授权”。

### 授权一个 TwinCAT 3 功能的 7 天测试版本

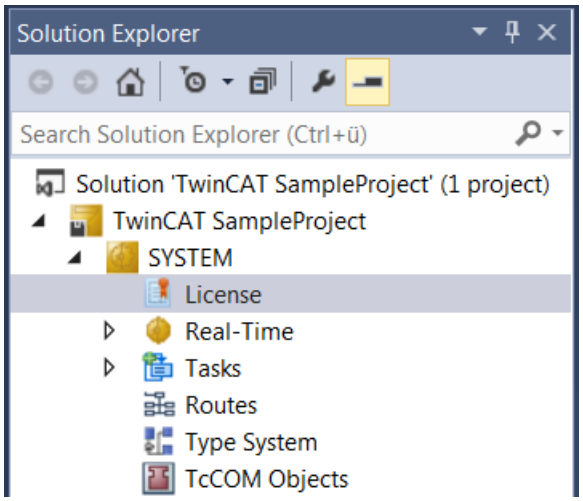
对于TwinCAT 3 许可证加密狗无法启用 7 天测试版本。

1. 启动 TwinCAT 3 开发环境（XAE）。
2. 打开一个现有的 TwinCAT 3 项目或创建一个新项目。

3. 如果想为一个远程设备激活授权，请设置所需的目标系统。为此，从工具栏的**选择目标系统**下拉列表中选择目标系统。

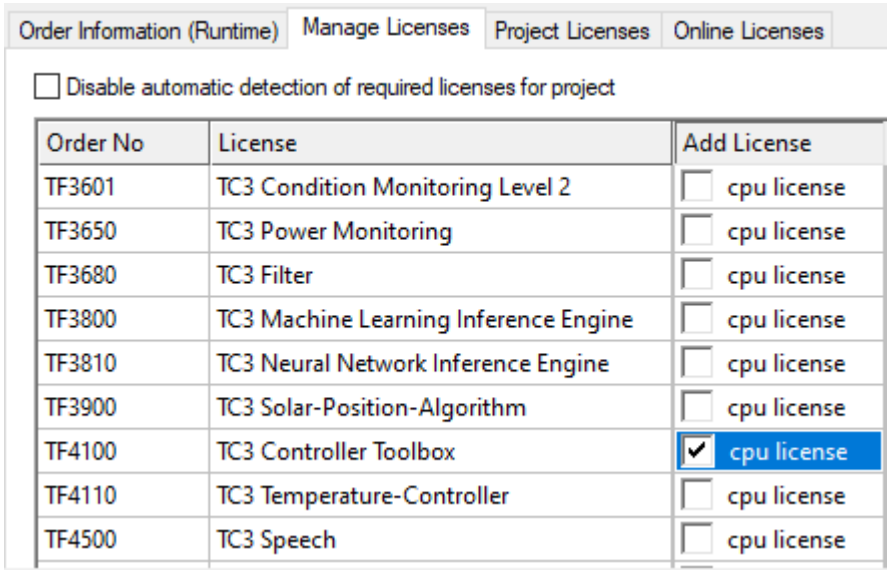
   ⇨ 授权设置总是指选定的目标系统。在目标系统上激活项目时，自动将相应的 TwinCAT 3 许可证复制到该系统中。

4. 在**解决方案资源管理器**中，双击**系统**子目录的**许可证**。




   ⇨ TwinCAT 3 许可证管理器打开。

5. 打开**管理许可证**选项卡。在**添加许可证**栏中，选中希望添加到项目的许可证的复选框（例如“TF4100 TC3 控制器工具箱”）。

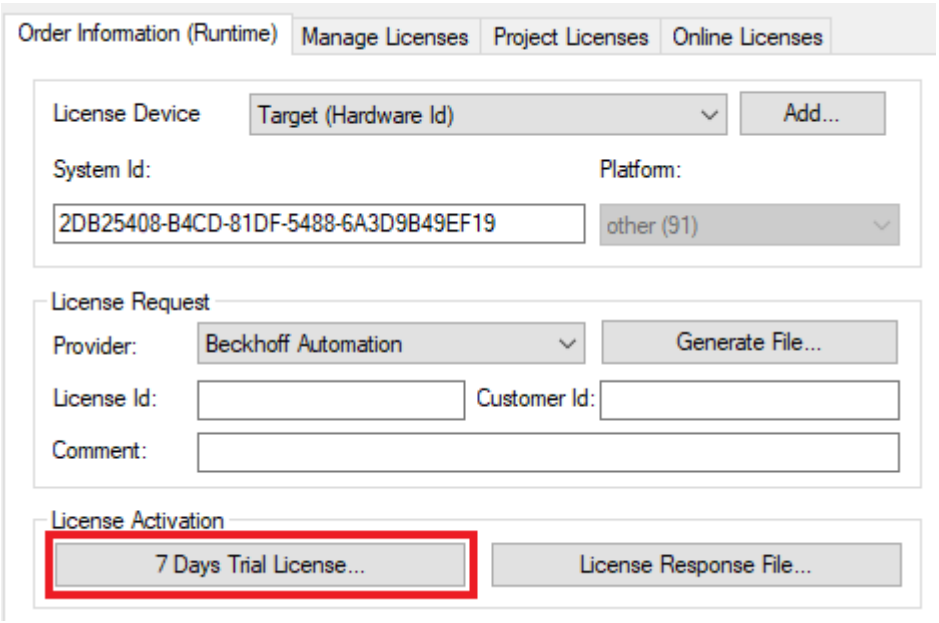



| Order No | License | Add License |
|---|---|---|
| TF3601 | TC3 Condition Monitoring Level 2 | cpu license |
| TF3650 | TC3 Power Monitoring | cpu license |
| TF3680 | TC3 Filter | cpu license |
| TF3800 | TC3 Machine Learning Inference Engine | cpu license |
| TF3810 | TC3 Neural Network Inference Engine | cpu license |
| TF3900 | TC3 Solar-Position-Algorithm | cpu license |
| TF4100 | TC3 Controller Toolbox | cpu license |
| TF4110 | TC3 Temperature-Controller | cpu license |
| TF4500 | TC3 Speech | cpu license |

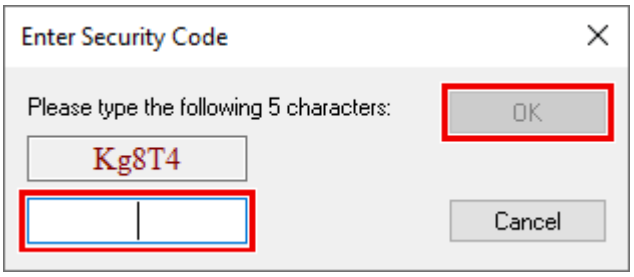
6. 打开**订单信息（运行时间）**选项卡。

   ⇨ 在许可证的表格概览中，以前选择的许可证显示为“缺失”状态。

7. 点击**7 天试用许可证...**，以激活 7 天试用许可证。




⇨ 一个对话框打开，提示输入对话框中显示的安全代码。




8. 准确输入显示的代码并确认输入。
9. 确认随后的对话框，表明激活成功。
   ⇨ 在许可证的表格概览中，许可证状态现在显示许可证的到期日。
10. 重新启动 TwinCAT 系统。

⇨ 7 天试用版启用。

# 4 PLC API

## 4.1 通用工作原理

以下各部分将对 TwinCAT Controller Toolbox 中的功能块进行概述说明。

**离散化**

使用梯形规则（图斯汀公式）将该库中所装配的传递单元的连续传递函数转换为离散值。

图斯汀公式：

$$\frac{1}{s} = \frac{T}{2}\frac{z+1}{z-1}$$

**功能块输入**

**eMode**

大多数功能块的运行模式都可以通过该输入进行选择。由此可以选择以下运行模式之一：

| | |
|---|---|
| eCTRL_MODE_PASSIVE | 功能块的一个或多个输出被设置为零，但内部状态保持不变。 |
| eCTRL_MODE_ACTIVE | 功能块按照其描述执行，并计算相应的输出值（正常运行）。 |
| eCTRL_MODE_RESET | 在此运行模式下，所有内部状态被重置，错误位清零。 |
| eCTRL_MODE_MANUAL | 在输出中提供输入值 `fManSyncValue` 的值（手动操作）。 |

**stParams**

通过此结构将必要的参数传递给功能块。变量 `tTaskCycleTime` 和 `tCtrlCycleTime` 包含在所有参数结构中。这些参数按以下方式起作用：

参数 `tTaskCycleTime` 指定调用功能块的循环时间。如果在每个循环中都调用该功能块，这相当于调用任务的循环时间。如果每隔一个循环才调用一次该功能块，则时间必须同步倍增。参数 `tCtrlCycleTime` 表示控制回路的采样时间。该时间必须大于或等于参数 `tTaskCycleTime`。如果采样时间被设置为等于 `tTaskCycleTime`，则每次调用都会执行功能块。如果所选值扩大为原来的 5 倍，则每五次调用才会处理一次功能块。由此可实现在快速任务中执行慢速控制回路。

参数 `tTaskCycleTime` 和 `tCtrlCycleTime` 属于 TIME 类型，因此不允许输入小于 1ms。为了在循环时间小于 1ms 的快速 PLC 任务中使用控制器，可以指定一个全局基准时间作为指定循环时间的参考。以下示例解释了基准时间的使用情况。

假设在每个任务周期中都调用功能块。

| 任务：配置 | 参数：`tTaskCycleTime` | 参数：`tCtrlCycleTime` | 操作 |
|---|---|---|---|
| T#10ms | T#10ms | T#10ms | 使用 10 ms 的采样时间处理控制回路。 |
| T#10ms | T#10ms | T#50ms | 使用 50 ms 的采样时间处理控制回路。 |
| T#100ms | T#100ms | T#100ms | 使用 100 ms 的采样时间处理控制回路。 |
| T#100ms | T#100ms | T#50ms | **错误**，无法执行！ |
| T#100ms | T#50ms | T#50ms | **错误**，虽然已执行功能块，但计算的输出值不正确！ |

**功能块的输出**

**eState**

该输出表示功能块的当前内部状态。

| | |
|---|---|
| eCTRL_STATE_IDLE | 功能块已成功重置，现在等待选择运行模式。 |
| eCTRL_STATE_PASSIVE | 功能块处于被动状态，不执行任何计算。 |
| eCTRL_STATE_ACTIVE | 功能块处于激活状态，即正常运行状态。 |
| eCTRL_STATE_RESET | 正在处理重置请求，但重置尚未完成。 |
| eCTRL_STATE_MANUAL | 功能块处于手动状态，可在相应的输入中手动指定输出值。 |
| eCTRL_STATE_... | 如果还有任何其他内部状态，它们将与相应的功能块一起说明。 |
| eCTRL_STATE_ERROR | 发生错误；在此状态下不会执行功能块。请参见 **eErrorId**，以了解更多信息。 |

### bError

该布尔输出为 TRUE 时，表示功能块出错。

### eErrorId

如果 `bError` 输出为 TRUE，则会在此输出中提供错误编号 [▶ 157]。

### 使用全局基准时间

为了能够在周期时间小于 1ms 的 PLC 任务中使用这些功能块，可以将指定的循环时间解释为基准时间的节拍。在这种特殊的参数设置中，1ms 的时间单位被解释为 1 个节拍。这种方法相当于在 TwinCAT System Manager 中将 PLC 周期时间设置为小于 1ms。

对于工具箱的所有功能块，基准时间的切换和声明均通过全局结构 `stCtrl_GLOBAL_CycleTimeInterpretation` 完成。

```
VAR_GLOBAL
    stCtrl_GLOBAL_CycleTimeInterpretation :
    ST_CTRL_CYCLE_TIME_INTERPRETATION;
END_VAR

TYPE ST_CTRL_CYCLE_TIME_INTERPRETATION :
STRUCT
bInterpretCycleTimeAsTicks  : BOOL; (* e.g. 2ms -> 2ticks *)
fBaseTime                   : FLOAT; (* Base time in seconds, e.g. 200µs -> 200E-6s *)
END_STRUCT
END_TYPE
```

为了将指定的循环时间解释为节拍，全局结构 `stCtrl_GLOBAL_CycleTimeInterpretation` 中的变量 `bInterpretCycleTimeAsTicks` 被设置为 TRUE。在此结构中，在变量 `fBaseTime` 中必须设置基准时间单位。

通过设置标志 `bInterpretCycleTimeAsTicks`，名称为

- tTaskCycleTime
- tCtrlCycleTime

的参数的解释方式被改变了。所有其他 **TIME** 类型的参数的解释和作用均不受影响。

### 例程

TwinCAT 系统的基准时间单位为 200µs。PLC 任务以及工具箱的功能块以 400µs 为周期进行循环调用。

设置全局结构：

```
stCtrl_GLOBAL_CycleTimeInterpretation.bInterpretCycleTimeAsTicks := TRUE;
stCtrl_GLOBAL_CycleTimeInterpretation.fBaseTime := 200E-6;
```

对工具箱中的功能块进行参数设置：

```
stParams.tTaskCycleTime := T#2ms; (* 2*200µs=400µs *)
stParams.tCtrlCycleTime := T#4ms; (* 4*200µs=800µs *)
stParams. ...
```

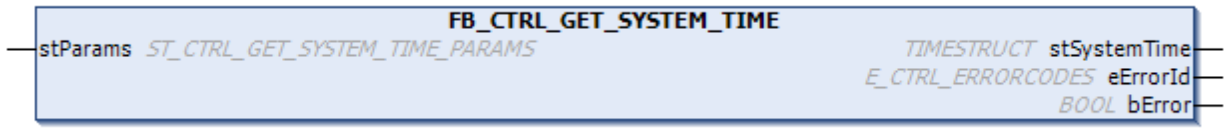
在功能块上指定的 **TaskCycleTime** 为“2 • 200E-6s = 400µs”，因此这相当于设定的 PLC 循环时间。**CtrlCycleTime** 被设置为“800µs = 4 • 200E-6s”，以便控制回路以 800µs 的采样时间运行，即每两个 PLC 循环执行一次处理。

# 4.2 参考

## 4.2.1 功能块

### 4.2.1.1 辅助功能块

#### 4.2.1.1.1 FB_CTRL_GET_SYSTEM_TIME




该功能块读取当前的 Windows 系统时间，并在 **SystemTimeStruct** 中提供它。

**描述**

该功能块在其输出结构中提供当前的系统时间。通过 `tCtrlCycleTime` 参数指定分辨率；最大分辨率为 10 ms，而且必须遵守“tCtrlCycleTime > 2 • tTaskCycleTime”的条件。否则，分辨率将降至“2 • tCtrlCycleTime”。

**输出**

```
VAR_OUTPUT
    stSystemTime    : TIMESTRUCT;
    eErrorId        : E_CTRL_ERRORCODES;
    bError          : BOOL;
END_VAR
TYPE TIMESTRUCT
STRUCT
    wYear           : WORD;
    wMonth          : WORD;
    wDayOfWeek      : WORD;
    wDay            : WORD;
    wHour           : WORD;
    wMinute         : WORD;
    wSecond         : WORD;
    wMilliseconds   : WORD;
END_STRUCT
END_TYPE
```

| 名称 | 类型 | 描述 |
|---|---|---|
| stSystemTime | TIME STRUCT | 输出系统时间的结构。 |
| eErrorId | E_CTRL_ ERROR CODES | 在设置输出 `bError` 时，提供错误编号 [▶ 157]。 |
| bError | BOOL | 一旦出现情况，则变为 TRUE。 |
| wYear | WORD | 年份：1970 ~ 2106； |
| wMonth | WORD | 月份：1 ~ 12（一月 = 1，二月 = 2 等）； |
| wDayOfWeek | WORD | 星期中的某一天：0 ~ 6（星期日 = 0，星期一 = 1 等）； |
| wDay | WORD | 月份中的某一天：1 ~ 31； |
| wHour | WORD | 小时：0 ~ 23； |
| wMinute | WORD | 分钟：0 ~ 59； |
| wSecond | WORD | 秒钟：0 ~ 59； |
| WMilliseconds | WORD | 毫秒：0 ~ 999； |

### 输入/输出

```
VAR_IN_OUT
    stParams    : ST_CTRL_GET_SYSTEM_TIME;
END_VAR
```

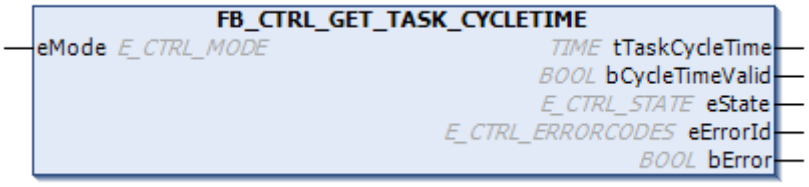
| 名称 | 类型 | 描述 |
|---|---|---|
| stParams | ST_CTRL_GET_SYSTEM_TIME | 功能块的参数结构 |

`stParams` 由以下元素组成：

```
TYPE
ST_CTRL_GET_SYSTEM_TIME:
STRUCT
    tTaskCycleTime : TIME;
    tCtrlCycleTime : TIME;
END_STRUCT
END_TYPE
```

| 名称 | 类型 | 描述 |
|---|---|---|
| tCtrlCycleTime | TIME | 处理控制回路的循环时间。该时间必须大于或等于 **TaskCycleTime**。功能块使用该输入值在内部计算是否需要在当前周期内更新状态和输出值。 |
| tTaskCycleTime | TIME | 调用功能块的循环时间。若功能块在每个任务周期内均被调用，则其周期时间等同于调用任务的任务周期时间。 |

## 4.2.1.1.2 FB_CTRL_GET_TASK_CYCLETIME




该功能块可以确定程序的任务循环时间，分辨率为 1 ms。

**正确使用功能块**

只有当程序中不包含任何活动断点时，才能正确确定 **TaskCycleTime**。
任务循环时间仅确定一次。如果 `bCycleTimeValid` 或 `bError` 输出中任意一个为 TRUE，则不再进行后续测量。

如果您使用的循环时间小于 1 ms 或不是 1 ms 的倍数，请勿使用此功能块

### 输入

```
VAR_INPUT
    eMode : E_CTRL_MODE;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| eMode | E_CTRL_MODE | 指定功能块的运行模式的输入。 |

### 输出

```
VAR_OUTPUT
    tTaskCycleTime    : TIME;
    bCycleTimeValid   : BOOL;
    eState            : E_CTRL_STATE;
    eErrorId          : E_CTRL_ERRORCODES;
    bError            : BOOL;
END_VAR
```

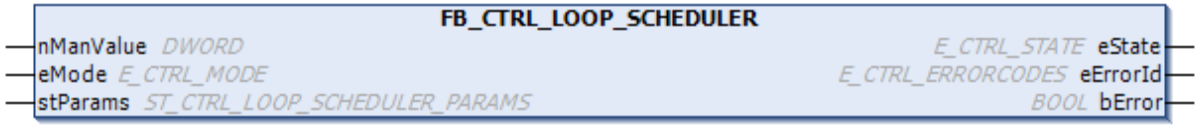
| 名称 | 类型 | 描述 |
|---|---|---|
| tTaskCycleTime | TIME | 该输出表示当前的任务循环时间，分辨率为 1 ms。 |
| bCycleTimeValid | BOOL | 当该输出为 TRUE 时，tTaskCycleTime 输出中包含的时间有效。 |
| eState | E_CTRL_STATE | 功能块的状态 |
| eErrorId | E_CTRL_ERRORCODES | 在设置输出 bError 时，提供错误编号。 |
| bError | BOOL | 一旦发生错误，则变为 TRUE。 |

**示例：**

```
PROGRAM PRG_GET_TASK_CYCLETIME_TEST
VAR
    tTaskCycleTime      : TIME;
    bCycleTimeValid     : BOOL;
    eState              : E_CTRL_STATE;
    eErrorId            : E_CTRL_ERRORCODES;
    bError              : BOOL;
    fbCTRL_GET_TASK_CYCLETIME : FB_CTRL_GET_TASK_CYCLETIME;
    bInit               : BOOL := TRUE;
    fSetpointValue      : FLOAT := 45.0;
    fActualValue        : FLOAT;
    fbCTRL_PI           : FB_CTRL_PI;
    stCTRL_PI_Params    : ST_CTRL_PI_PARAMS;
    fbCTRL_PT1          : FB_CTRL_PT1;
    stCTRL_PT1_Params   : ST_CTRL_PT1_PARAMS;
END_VAR
```

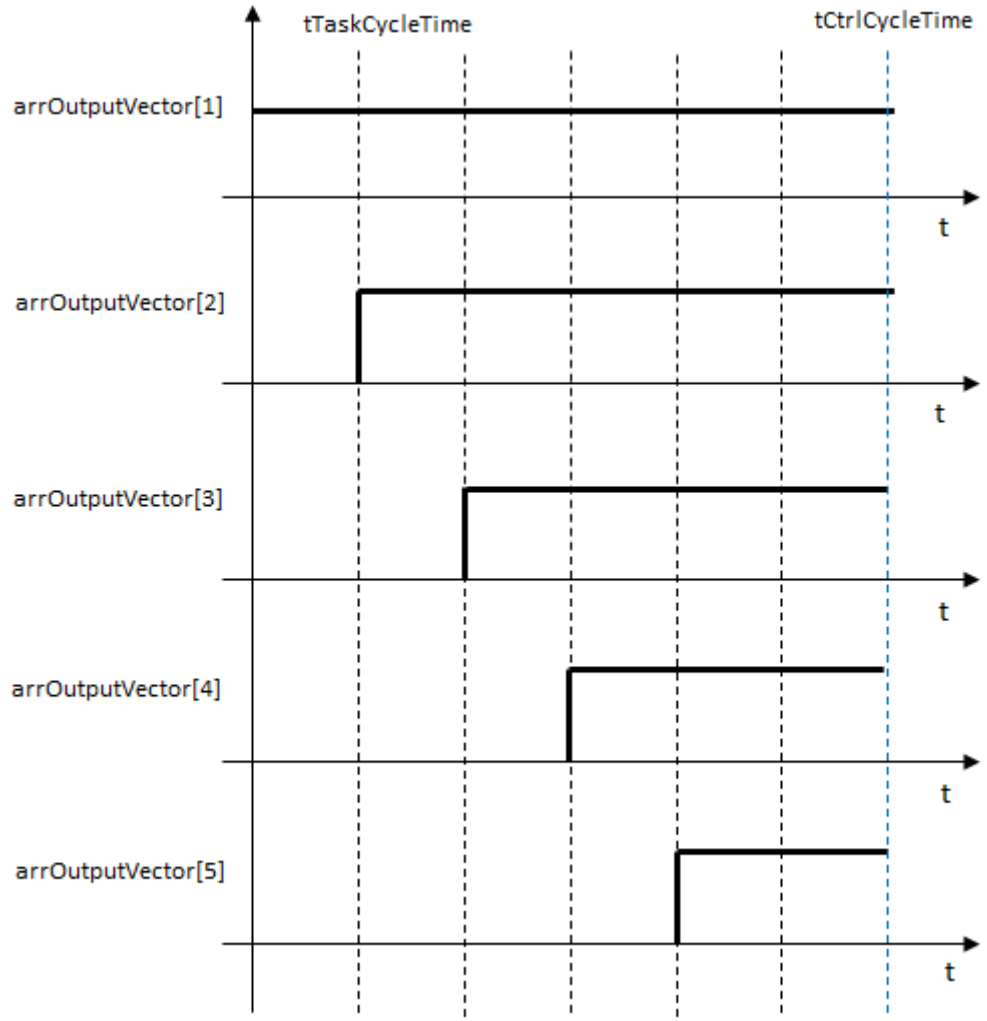
```
fbCTRL_GET_TASK_CYCLETIME( eMode     := eCTRL_MODE_ACTIVE,
                           tTaskCycleTime  => tTaskCycleTime,
                           bCycleTimeValid => bCycleTimeValid,
                           eState          => eCTRL_MODE_ACTIVE,
                           eErrorId        => eErrorId,
                           bError          => bError );
IF fbCTRL_GET_TASK_CYCLETIME.bCycleTimeValid
THEN
    IF bInit
    THEN
        stCTRL_PT1_Params.tTaskCycleTime := fbCTRL_GET_TASK_CYCLETIME.tTaskCycleTime;
        stCTRL_PT1_Params.tCtrlCycleTime := T#100ms;
        stCTRL_PT1_Params.fKp := 1.0;
        stCTRL_PT1_Params.tT1 := T#10s;
        stCTRL_PI_Params.tTaskCycleTime := fbCTRL_GET_TASK_CYCLETIME.tTaskCycleTime;
        stCTRL_PI_Params.tCtrlCycleTime := T#100ms;
        stCTRL_PI_Params.fKp := 0.5;
        stCTRL_PI_Params.tTn := T#5s;
        stCTRL_PI_Params.fOutMaxLimit := 100.0;
        stCTRL_PI_Params.fOutMinLimit := 0.0;
        bInit := FALSE;
END_IF
fbCTRL_PI( fActualValue := fbCTRL_PT1.fOut,
         fSetpointValue := fSetpointValue,
         eMode := eCTRL_MODE_ACTIVE,
         stParams := stCTRL_PI_Params );
fbCTRL_PT1( fIn := fbCTRL_PI.fOut,
            eMode := eCTRL_MODE_ACTIVE,
            stParams := stCTRL_PT1_Params,
            fOut => fActualValue );
END_IF
```

### 4.2.1.1.3 FB_CTRL_LOOP_SCHEDULER




该功能块允许将系统负载分配给多个控制回路，这些回路使用相同的 tCtrlCycleTime 进行参数设置，且满足条件“tCtrlCycleTime > tTaskCycleTime”。该功能块计算出的输出矢量用于在不同时间启动各个控制回路，从而实现系统负载的均衡分配。

**输出矢量的行为**




此图表中管理着 5 个控制回路。在这种情况下，“tCtrlCycleTime = 6 • tTaskCycleTime”。

如果要使用该功能块，程序员必须在 PLC 中创建以下数组：

```
arrOutputVector : ARRAY[1..nNumberOfControlLoops] OF BOOL;
```

该功能块将该矢量中的位设置为 TRUE 或 FALSE。当输出矢量中的相应位为 TRUE 时，使用循环调度程序管理的控制回路将切换到 eCTRL_MODE_ACTIVE 状态。请参见以下示例代码。

### 输入

```
VAR_INPUT
    nManValue : DWORD;
    eMode     : E_CTRL_MODE;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| nManValue | DWORD | 该输入允许以 eCTRL_MODE_MANUAL 方式设置输出矢量中的前 32 位。1 设置第一位，2 设置第二位，3 设置第一位和第二位，… |
| eMode | E_CTRL_MODE | 指定功能块的运行模式 [▶ 157]的输入。 |

### 输出

```
VAR_OUTPUT
    eState      : E_CTRL_STATE;
    eErrorId    : E_CTRL_ERRORCODES;
    bError      : BOOL;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| eState | E_CTRL_STATE | 功能块的状态 |
| eErrorId | E_CTRL_ERRORCODES | 在设置输出 bError 时，提供错误编号 [▶ 157]。 |
| bError | BOOL | 一旦发生错误，则变为 TRUE。 |

### 输入/输出

```
VAR_IN_OUT
    stParams     : ST_CTRL_LOOP_SCHEDULER_PARAMS;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| stParams | ST_CTRL_LOOP_SCHEDULER_PARAMS | 循环调度程序的参数结构。 |

stParams 由以下元素组成：

```
TYPE
ST_CTRL_LOOP_SCHEDULER_PARAMS:
STRUCT
    tCtrlCycleTime          : TIME := T#0ms;
    tTaskCycleTime          : TIME := T#0ms;
    nNumberOfControlLoops   : UINT;
    pOutputVector_ADR       : POINTER TO BOOL := 0;
    nOutputVector_SIZEOF    : UINT := 0;
END_STRUCT
END_TYPE
```

| 名称 | 类型 | 描述 |
|---|---|---|
| tCtrlCycleTime | TIME | 处理由循环调度程序管理的控制回路的循环时间。该时间必须大于或等于 **TaskCycleTime**。 |
| tTaskCycleTime | TIME | 调用与控制回路相关的循环调度程序和功能块的循环时间。如果在每个循环中都调用该功能块，这相当于调用任务的任务循环时间。 |
| nNumberOfControlLoops | UINT | 所管理的控制回路的数量。 |
| pOutputVector_ADR | POINTER TO BOOL | 输出矢量的地址 |
| nOutputVector_SIZEOF | UINT | 输出矢量的大小，以字节为单位 |

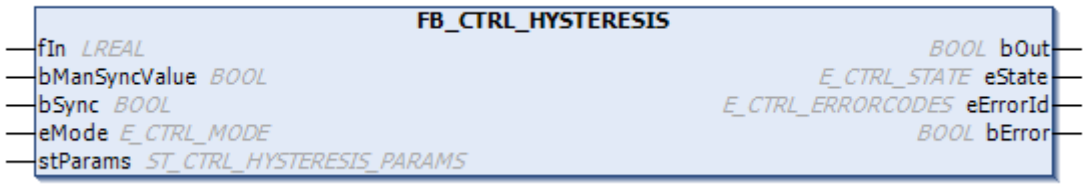
**示例：**

```
PROGRAM PRG_LoopScheduler
VAR
    arrOutputVector         : ARRAY[1..5] OF BOOL;
    eMode                   : E_CTRL_MODE;
    stParams                : ST_CTRL_LOOP_SCHEDULER_PARAMS;
    eErrorId                : E_CTRL_ERRORCODES;
    bError                  : BOOL;
    fbCTRL_LoopScheduler    : FB_CTRL_LOOP_SCHEDULER;
    bInit                   : BOOL := TRUE;
    eMode_CtrlLoop_1        : E_CTRL_MODE;
    eMode_CtrlLoop_2        : E_CTRL_MODE;
    eMode_CtrlLoop_3        : E_CTRL_MODE;
    eMode_CtrlLoop_4        : E_CTRL_MODE;
    eMode_CtrlLoop_5        : E_CTRL_MODE;
END_VAR
IF bInit
THEN
    stParams.tCtrlCycleTime         := T#10ms;
    stParams.tTaskCycleTime         := T#2ms;
```

```
    stParams.nNumberOfControlLoops  := 5;
    bInit                           := FALSE;
END_IF
stParams.nOutputVector_SIZEOF  := SIZEOF(arrOutputVector);
stParams.pOutputVector_ADR     := ADR(arrOutputVector);
fbCTRL_LoopScheduler( eMode     := eMode,
                      stParams  := stParams,
                      eErrorId  => eErrorId,
                      bError    => bError);
IF arrOutputVector[1] THEN
    eMode_CtrlLoop_1 := eCTRL_MODE_ACTIVE;
END_IF
IF arrOutputVector[2] THEN
    eMode_CtrlLoop_2 := eCTRL_MODE_ACTIVE;
END_IF
IF arrOutputVector[3] THEN
    eMode_CtrlLoop_3 := eCTRL_MODE_ACTIVE;
END_IF
IF arrOutputVector[4] THEN
    eMode_CtrlLoop_4 := eCTRL_MODE_ACTIVE;
END_IF
IF arrOutputVector[5]
THEN
    eMode_CtrlLoop_5 := eCTRL_MODE_ACTIVE;
END_IF
```

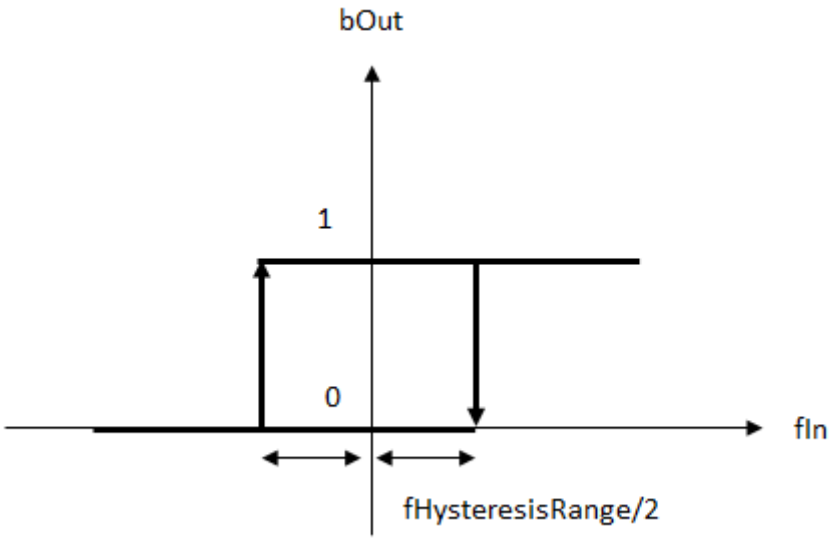
### 4.2.1.2 基础功能

#### 4.2.1.2.1 FB_CTRL_HYSTERESIS




该功能块在功能图中提供了一个滞环传递单元。

**传递函数**




**输入**

```
VAR_INPUT
    fIn              : FLOAT;
    bManSyncValue    : BOOL;
```

```
    bSync            : BOOL;
    eMode            : E_CTRL_MODE;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| fIn | FLOAT | 滞环单元的输入 |
| bManSync Value | BOOL | 用于将滞环单元设置为两个拓扑扩展模块之一的输入。 |
| bSync | BOOL | 该输入端的上升沿将滞环单元设置为值“fManSyncValue”。 |
| eMode | E_CTRL_MODE | 指定功能块的运行模式 [▶ 157]的输入。 |

### 输出

```
VAR_OUTPUT
    bOut         : BOOL;
    eState       : E_CTRL_STATE;
    eErrorId     : E_CTRL_ERRORCODES;
    bError       : BOOL;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| bOut | BOOL | 滞环单元的输出 |
| eState | E_CTRL_STATE | 功能块的状态 |
| eErrorId | E_CTRL_ERROR CODES | 在设置输出 bError 时，提供错误编号 [▶ 157]。 |
| bError | BOOL | 一旦出现情况，则变为 TRUE。 |

### 输入/输出

```
VAR_IN_OUT
    stParams     : ST_CTRL_HYSTERESIS_PARAMS;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| stParams | ST_CTRL_ HYSTERESIS_ PARAMS | 滞环单元的参数结构 |

stParams 由以下元素组成：

```
TYPE
ST_CTRL_HYSTERESIS_PARAMS :
STRUCT
    tCtrlCycleTime      : TIME := T#0ms;
    tTaskCycleTime      : TIME := T#0ms;
    fHysteresisRange    : FLOAT;
END_STRUCT
END_TYPE
```

| 名称 | 类型 | 描述 |
|---|---|---|
| tCtrlCycleTime | TIME | 处理控制回路的循环时间。该时间必须大于或等于 **TaskCycleTime**。功能块使用该输入值在内部计算是否需要在当前周期内更新状态和输出值。 |
| tTaskCycleTime | TIME | 调用功能块的循环时间。如果在每个循环中都调用该功能块，这相当于调用任务的任务循环时间。 |
| fHysteresis Range | FLOAT | 滞环范围，请参见上图。 |

### 4.2.1.2.2 FB_CTRL_P


该功能块在功能图中提供了一个 P 传递单元。

**传递函数：**

$$G(s) = K_p$$

#### 输入

```
VAR_INPUT
    fIn       :FLOAT;
    eMode     :E_CTRL_MODE;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| fIn | FLOAT | P 单元的输入 |
| eMode | E_CTRL_MODE | 指定功能块的运行模式的输入。 |

#### 输出

```
VAR_OUTPUT
    fOut          :FLOAT;
    eState        :E_CTRL_STATE;
    eErrorId      :E_CTRL_ERRORCODES;
    bError        :BOOL;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| fOut | FLOAT | P 单元的输出 |
| eState | E_CTRL_STATE | 功能块的状态 |
| eErrorId | E_CTRL_ERRORCODES | 在设置输出 `bError` 时，提供错误编号 [▶ 157]。 |
| bError | BOOL | 一旦发生错误，则变为 TRUE。 |

#### 输入/输出

```
VAR_IN_OUT
    stParams     :ST_CTRL_P_PARAMS;
END_VAR
```

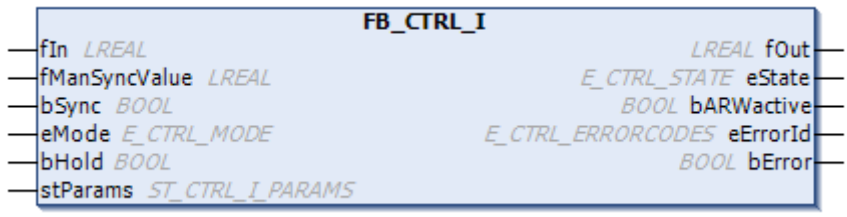
| 名称 | 类型 | 描述 |
|---|---|---|
| stParams | ST_CTRL_P_PARAMS | P 单元的参数结构 |

`stParams` 由以下元素组成：

```
TYPE ST_CTRL_P_PARAMS:
STRUCT
    tCtrlCycleTime     : TIME := T#0ms;
    tTaskCycleTime     : TIME := T#0ms;
    fKp                : FLOAT := 0.0;
END_STRUCT
END_TYPE
```

| 名称 | 类型 | 描述 |
|---|---|---|
| tCtrlCycleTime | TIME | 处理控制回路的循环时间。该时间必须大于或等于 **TaskCycleTime**。功能块使用该输入值在内部计算是否需要在当前周期内更新状态和输出值。 |
| tTaskCycleTime | TIME | 调用功能块的循环时间。如果在每个循环中都调用该功能块，这相当于调用任务的任务循环时间。 |
| fKp | FLOAT | P 单元的比例增益 |

### 4.2.1.2.3 FB_CTRL_I




该功能块在功能图中提供了一个 I 传递单元。

**传递函数**

$$G(s) = \frac{1}{T_I s}$$

**输入**

```
VAR_INPUT
    fIn             : FLOAT;
    fManSyncValue   : FLOAT;
    bSync           : BOOL;
    eMode           : E_CTRL_MODE;
    bHold           : BOOL;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| fIn | FLOAT | I 单元的输入 |
| fManSyncValue | FLOAT | I 单元可与之同步的输入，或者其值在手动模式下存在于输出端。 |
| bSync | BOOL | 该输入端的上升沿将积分器设置为值“fManSyncValue”。 |
| eMode | E_CTRL_MODE | 指定功能块的运行模式的输入。 |
| bHold | BOOL | 该输入为 TRUE 时，积分器将保持当前值不变，且不受输入 `fIn` 影响。 |

**输出**

```
VAR_OUTPUT
    fOut            : FLOAT;
    bARWactive      : BOOL;
    eState          : E_CTRL_STATE;
    eErrorId        : E_CTRL_ERRORCODES;
    bError          : BOOL;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| fOut | FLOAT | I 单元的输出 |
| bARWactive | BOOL | 该输出为 TRUE 时，表示积分器正处于限制状态。 |

| 名称 | 类型 | 描述 |
|---|---|---|
| eState | E_CTRL_STATE | 功能块的状态 |
| eErrorId | E_CTRL_ERRORCODES | 在设置输出 bError 时，提供错误编号。 |
| bError | BOOL | 一旦发生错误，则变为 TRUE。 |

### 输入/输出

```
VAR_IN_OUT
    stParams    : ST_CTRL_I_PARAMS;
END_VAR
```

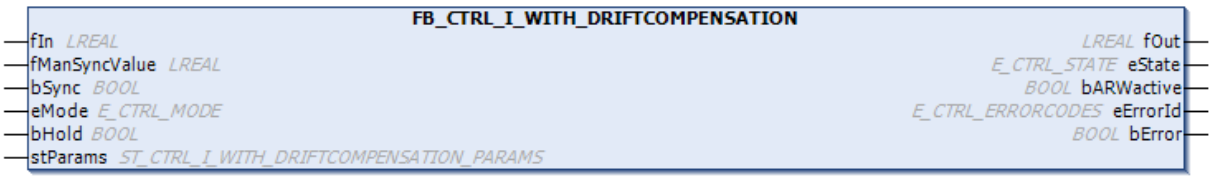
| 名称 | 类型 | 描述 |
|---|---|---|
| stParams | ST_CTRL_I_PARAMS | I 单元的参数结构 |

stParams 由以下元素组成：

```
TYPE ST_CTRL_I_PARAMS:
STRUCT
    tCtrlCycleTime    : TIME := T#0ms;
    tTaskCycleTime    : TIME := T#0ms;
    tTi               : TIME := T#0ms;
    fOutMaxLimit      : FLOAT := 1E38;
    fOutMinLimit      : FLOAT := -1E38;
END_STRUCT
END_TYPE
```

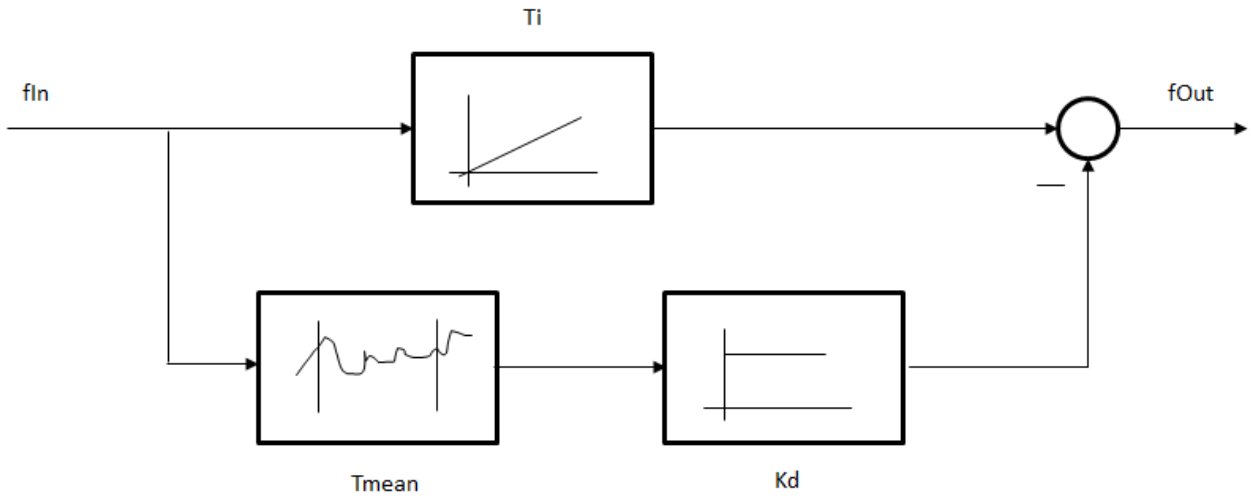
| 名称 | 类型 | 描述 |
|---|---|---|
| tCtrlCycleTime | TIME | 处理控制回路的循环时间。该时间必须大于或等于 TaskCycleTime。功能块使用该输入值在内部计算是否需要在当前周期内更新状态和输出值。 |
| tTaskCycleTime | TIME | 调用功能块的循环时间。如果在每个循环中都调用该功能块，这相当于调用任务的任务循环时间。 |
| tTi | TIME | I 单元的积分时间 |
| fOutMaxLimit | FLOAT | 积分在此上限处被停止（ARW 措施）。bARWActive 输出为 TRUE 时，表示达到该限值。 |
| fOutMinLimit | FLOAT | 积分在此下限处被停止（ARW 措施）。bARWActive 输出为 TRUE 时，表示达到该限值。 |

## 4.2.1.2.4 FB_CTRL_I_WITH_DRIFTCOMPENSATION




该功能块表示带漂移补偿的 I 传递单元。

## 功能图




### 输入

```
VAR_INPUT
    fIn                 : FLOAT;
    fManSyncValue       : FLOAT;
    bSync               : BOOL;
    eMode               : E_CTRL_MODE;
    bHold               : BOOL;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| fIn | FLOAT | I 单元的输入 |
| fManSyncValue | FLOAT | I 单元可与之同步的输入，或者其值在手动模式下存在于输出端。 |
| bSync | BOOL | 该输入端的上升沿将积分器设置为值“fManSyncValue”。 |
| eMode | E_CTRL_MODE | 指定功能块的运行模式的输入。 |
| bHold | BOOL | 该输入为 TRUE 时，积分器将保持当前值不变，且不受输入 `fIn` 影响。 |

### 输出

```
VAR_OUTPUT
    fOut            : FLOAT;
    bARWactive      : BOOL;
    eState          : E_CTRL_STATE;
    eErrorId        : E_CTRL_ERRORCODES;
    bError          : BOOL;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| fOut | FLOAT | I 单元的输出 |
| bARWactive | BOOL | 该输出为 TRUE 时，表示积分器正处于限制状态。 |
| eState | E_CTRL_STATE | 功能块的状态 |
| eErrorId | E_CTRL_ERRORCODES | 在设置输出 `bError` 时，提供错误编号 [▶ 157]。 |
| bError | BOOL | 一旦发生错误，则变为 TRUE。 |

### 输入/输出

```
VAR_IN_OUT
    stParams    : ST_CTRL_I_WITH_DRIFTCOMPENSATION_PARAMS;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| stParams | ST_CTRL_I_WITH_DRIFTCOMPENSATION_PARAMS | I 单元的参数结构 |

`stParams` 由以下元素组成：

```
TYPE
ST_CTRL_I_WITH_DRIFTCOMPENSATION_PARAMS:
STRUCT
    tCtrlCycleTime          : TIME := T#0ms;
    tTaskCycleTime          : TIME := T#0ms;
    tTi                     : TIME := T#0ms;
    fOutMaxLimit            : FLOAT := 1E38;
    fOutMinLimit            : FLOAT := -1E38;
    fDampingCoefficient     : FLOAT := 0.0;
    tAveragingTime          : TIME := T#0ms;
    pWorkArray_ADR          : POINTER TO FLOAT := 0;
    nWorkArray_SIZEOF       : UINT := 0;
END_STRUCT
END_TYPE
```

| 名称 | 类型 | 描述 |
|---|---|---|
| tCtrlCycleTime | TIME | 处理控制回路的循环时间。该时间必须大于或等于 **TaskCycleTime**。功能块使用该输入值在内部计算是否需要在当前周期内更新状态和输出值。 |
| tTaskCycleTime | TIME | 调用功能块的循环时间。如果在每个循环中都调用该功能块，这相当于调用任务的任务循环时间。 |
| tTi | TIME | I 单元的积分时间 |
| fOutMaxLimit | FLOAT | 积分在此上限处被停止（ARW 措施）。`bARWActive` 输出为 TRUE 时，表示达到该限值。 |
| fOutMinLimit | FLOAT | 积分在此下限处被停止（ARW 措施）。`bARWActive` 输出为 TRUE 时，表示达到该限值。 |
| fDampingCoefficient | FLOAT | 功能图中的系数 $k_d$ |
| tAveragingTime | TIME | 滑动平均值滤波器计算平均值所采用的时间跨度。 |
| pWorkArray_ADR | POINTER TO FLOAT | 大小为 `tAveragingTime / tCtrlCycleTime` 的 FLOAT 数组的地址（请参见功能块 FB_CTRL_MOVING_AVERAGE 的描述） |
| nWorkArray_SIZEOF | UINT | 大小为 `tAveragingTime / tCtrlCycleTime` 的 FLOAT 数组的大小（请参见功能块 FB_CTRL_MOVING_AVERAGE 的描述） |

### 4.2.1.2.5 FB_CTRL_D


该功能块在功能图中提供了一个 $DT_1$ 传递单元（一个真正的 D 单元）。

**传递函数（连续）**

$$G(s) = \frac{T_v s}{1 + T_d s}$$

### 输入

```
VAR_INPUT
    fIn                 : FLOAT;
    fManSyncValue       : FLOAT;
    bSync               : BOOL;
    eMode               : E_CTRL_MODE;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| fIn | FLOAT | D 单元的输入 |
| fManSyncValue | FLOAT | D 单元可与之同步的输入，或者其值在手动模式下存在于输出端。 |
| bSync | BOOL | 该输入端的上升沿将 D 单元设置为值“fManSyncValue”。 |
| eMode | E_CTRL_MODE | 指定功能块的运行模式 [▶ 157]的输入。 |

### 输出

```
VAR_OUTPUT
    fOut        : FLOAT;
    eState      : E_CTRL_STATE;
    eErrorId    : E_CTRL_ERRORCODES;
    bError      : BOOL;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| fOut | FLOAT | D 单元的输出 |
| eState | E_CTRL_STATE | 功能块的状态 |
| eErrorId | E_CTRL_ERRORCODES | 在设置输出 bError 时，提供错误编号 [▶ 157]。 |
| bError | BOOL | 一旦发生错误，则变为 TRUE。 |

### 输入/输出

```
VAR_IN_OUT
    stParams    : ST_CTRL_D_PARAMS;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| stParams | ST_CTRL_D_PARAMS | D 单元的参数结构 |

stParams 由以下元素组成：

```
TYPE ST_CTRL_D_PARAMS:
STRUCT
    tCtrlCycleTime  : TIME  := T#0ms;
    tTaskCycleTime  : TIME  := T#0ms;
    tTv             : TIME  := T#0ms;
    tTd             : TIME  := T#0ms;
    fOutMaxLimit    : FLOAT := 1E38;
    fOutMinLimit    : FLOAT := -1E38;
END_STRUCT
END_TYPE
```

| 名称 | 类型 | 描述 |
|---|---|---|
| tCtrlCycleTime | TIME | 处理控制回路的循环时间。该时间必须大于或等于 **TaskCycleTime**。功能块使用该输入值在内部计算是否需要在当前周期内更新状态和输出值。 |
| tTaskCycleTime | TIME | 调用功能块的循环时间。如果在每个循环中都调用该功能块，这相当于调用任务的任务循环时间。 |
| tTv | TIME | 微分时间常数 |
| tTd | TIME | 阻尼时间常数 |
| fOutMaxLimit | FLOAT | D 单元输出的上限。 |
| fOutMinLimit | FLOAT | D 单元输出的下限。 |

## 4.2.1.2.6 FB_CTRL_TRANSFERFUNCTION_1

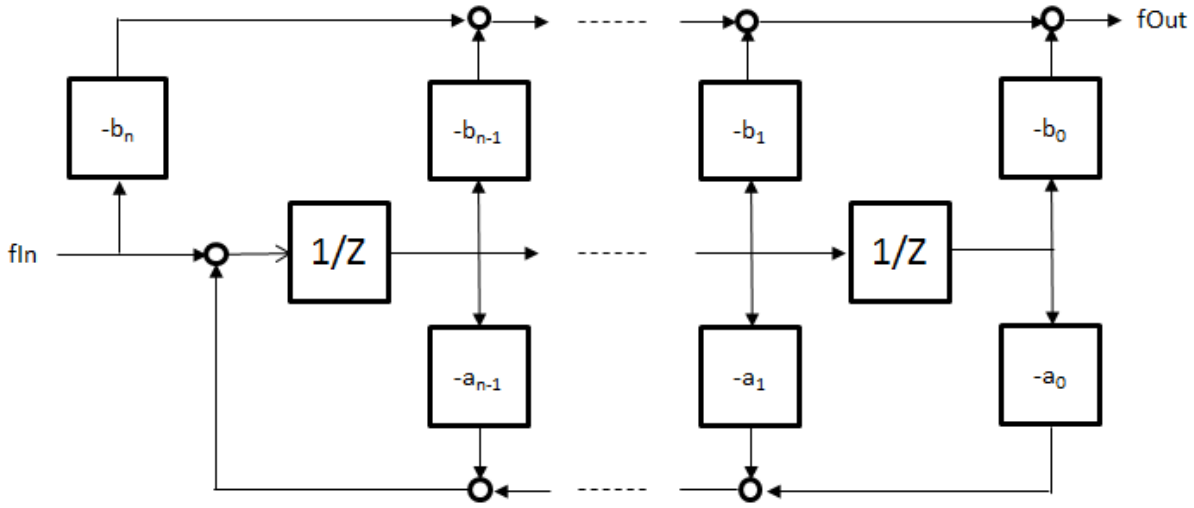

该功能块使用如下所示的第一标准形式计算离散传递函数。此处的传递函数可以是任意阶次 n。

下列传递函数的系数存储在参数数组中：

$$G(z) = \frac{b_n z^n + b_{(n-1)} z^{(n-1)} + ... + b_1 z + b_0}{z^n + a_{(n-1)} z^{(n-1)} + ... + a_1 z + a_0}$$

**传递行为的描述**

在每个扫描周期中，上述传递函数在经过若干变换后，按第一标准形式进行计算。




如果要使用该功能块，程序员必须在 PLC 中创建以下数组：

```
aNumArray         : ARRAY[0..nNumOrder] OF FLOAT;
aDenArray         : ARRAY[0..nDenOrder] OF FLOAT;
aStructTfData     : ARRAY[0..nTfOrder] OF ST_CTRL_TRANSFERFUNCTION_1_DATA;
```

系数“$b_0$”至“$b_n$”存储在数组 `aNumArray` 中。其结构必须按以下方式组织：

```
aNumArray[0] := b0;
aNumArray[1] := b1;
...
aNumArray[n-1] := bn-1;
aNumArray[n] := bn;
```

系数“$a_0$”至“$a_n$”存储在数组 `ar_DenominatorArray` 中。其结构必须按以下方式组织：

```
aDenArray[0] := a0;
aDenArray[1] := a1;
...
aDenArray[n-1] := an-1;
aDenArray[n] := an;
```

功能块所需的内部数据存储在 `ar_stTransferfunction1Data` 数组中。绝不能在 PLC 程序内部修改此数据。此流程也在下方列出的示例程序中予以说明。

### 输入

```
VAR_INPUT
    fIn          : FLOAT;
    eMode        : E_CTRL_MODE;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| fIn | FLOAT | 传递函数的输入 |
| eMode | E_CTRL_MODE | 指定功能块的运行模式的输入。 |

### 输出

```
VAR_OUTPUT
    fOut         : FLOAT;
    eState       : E_CTRL_STATE;
    bError       : BOOL;
    eErrorId     : E_CTRL_ERRORCODES;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| fOut | FLOAT | 传递函数的输出 |
| eState | E_CTRL_STATE | 功能块的状态 |
| bError | BOOL | 在设置输出 bError 时，提供错误编号 [▶ 157]。 |
| eErrorId | E_CTRL_ERRORCODES | 一旦发生错误，则变为 TRUE。 |

### 输入/输出

```
VAR_IN_OUT
    stParams     : ST_CTRL_TRANSFERFUNCTION_1_PARAMS;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| stParams | ST_CTRL_TRANSFERFUNCTION_1_PARAMS | 功能块的参数结构 |

stParams 由以下元素组成：

```
TYPE
ST_CTRL_TRANSFERFUNCTION_1_PARAMS:
STRUCT
    tTaskCycleTime                  : TIME;
    tCtrlCycleTime                  : TIME := T#0ms;
    nOrderOfTheTransferfunction     : USINT;
    pNumeratorArray_ADR             : POINTER TO FLOAT := 0;
    nNumeratorArray_SIZEOF          : UINT;
    pDenominatorArray_ADR           : POINTER TO FLOAT := 0;
    nDenomiantorArray_SIZEOF        : UINT;
    pTransferfunction1Data_ADR      : POINTER TO
ST_CTRL_TRANSFERFUNCTION_1_DATA;
    nTransferfunction1Data_SIZEOF   : UINT;
END_STRUCT
END_TYPE
```

| 名称 | 类型 | 描述 |
|---|---|---|
| tTaskCycleTime | TIME | 调用功能块的循环时间。如果在每个循环中都调用该功能块，这相当于调用任务的任务循环时间。 |
| tCtrlCycleTime | TIME | 处理控制回路的循环时间。该时间必须大于或等于 **TaskCycleTime**。功能块使用该输入值在内部计算是否需要在当前周期内更新状态和输出值。 |
| nOrderOfTheTransferfunction | USINT | 传递函数的阶次 [0...] |
| pNumeratorArray_ADR | POINTER TO FLOAT | 包含分子系数的数组的地址 |

| 名称 | 类型 | 描述 |
|---|---|---|
| nNumeratorArray_SIZEOF | UINT | 包含分子系数的数组的大小，以字节为单位 |
| pDenominatorArray_ADR | POINTER TO FLOAT | 包含分母系数的数组的地址 |
| nDenomiantorArray_SIZEOF | UINT | 包含分母系数的数组的大小，以字节为单位 |
| pTransferfunction1Data_ADR | POINTER TO ST_CTRL_TRANSFER FUNCTION_1_DATA | 数据数组的地址 |
| nTransferfunction1Data_SIZEOF | UINT | 数据数组的大小，以字节为单位 |

**示例：**

```
PROGRAM PRG_TRANSFERFUNCTION_1_TEST
VAR CONSTANT
    nTfOrder    : USINT := 2;
END_VAR
VAR
    aNumArray           : ARRAY[0..nNumOrder] OF FLOAT;
    aDenArray           : ARRAY[0..nDenOrder] OF FLOAT;
    aStTfData           : ARRAY[0..nTfOrder] OF ST_CTRL_TRANSFERFUNCTION_1_DATA;
    eMode               : E_CTRL_MODE;
    stParams            : ST_CTRL_TRANSFERFUNCTION_1_PARAMS;
    eErrorId            : E_CTRL_ERRORCODES;
    bError              : BOOL;
    fbTansferfunction   : FB_CTRL_TRANSFERFUNCTION_1;
    bInit               : BOOL := TRUE;
    fIn                 : FLOAT := 0;
    fOut                : FLOAT;
    b_0, b_1, b_2       : FLOAT;
    a_0,a_1,a_2         : FLOAT;
END_VAR
IF bInit THEN
    aNumArray[0] := 1.24906304658218E-007;
    aNumArray[1] := 2.49812609316437E-007;
    aNumArray[2] := 1.24906304658218E-007;
    aDenArray[0] := 0.998501124344101;
    aDenArray[1] := -1.99850062471888;
    aDenArray[2] := 1.0;
    stParams.tTaskCycleTime         := T#2ms;
    stParams.tCtrlCycleTime         := T#2ms;
    stParams.nOrderOfTheTransferfunction := nTfOrder;
    eMode := eCTRL_MODE_ACTIVE;
    bInit := FALSE;
END_IF
stParams.pNumeratorArray_ADR            := ADR(aNumArray);
stParams.nNumeratorArray_SIZEOF         := SIZEOF(aNumArray);
stParams.pDenominatorArray_ADR          := ADR(aDenArray);
stParams.nDenominatorArray_SIZEOF       := SIZEOF(aDenArray);
stParams.pTransferfunction1Data_ADR     := ADR(aStructTfData);
stParams.nTransferfunction1Data_SIZEOF  := SIZEOF(aStructTfData);
fbTansferfunction (fIn := fIn,
                   eMode := eMode,
                   stParams := stParams,
                   fOut => fOut,
                   eErrorId => eErrorId,
                   bError => bError);
```

## 4.2.1.2.7 FB_CTRL_TRANSFERFUNCTION_2

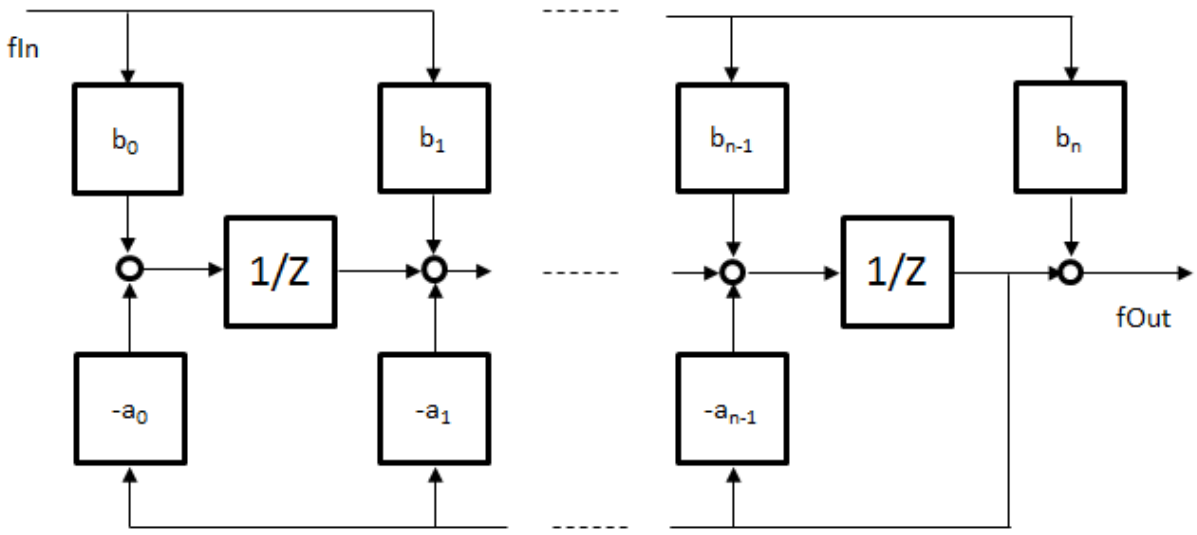

该功能块使用如下所示的第二标准形式计算离散传递函数。此处的传递函数可以是任意阶次 n。

下列传递函数的系数存储在参数数组中：

$$G(z) = \frac{b_n z^n + b_{(n-1)} z^{(n-1)} + \ldots + b_1 z + b_0}{z^n + a_{(n-1)} z^{(n-1)} + \ldots + a_1 z + a_0}$$

**传递行为的描述**

内部计算在每个采样步骤中均按照第二标准形式执行，框图如下：




通过若干变换，该传递函数可以转换为框图所示形式：

$$G(z) = \tilde{b} + \frac{\tilde{b}_{(n-1)} z^{-1} + \ldots + \tilde{b}_1 z^{-(n-1)} + \tilde{b}_0 z^{-n}}{1 + a_{(n-1)} z^{-1} + \ldots + a_1 z^{-(n-1)} + a_0 z^{-n}}$$

分子多项式的系数按以下规则计算：

$$\tilde{b}_i = b_i - b_n a_i \quad \forall 0 \le i < n$$

$$\tilde{b}_n = b_n$$

如果要使用该功能块，程序员必须在 PLC 中创建以下数组：

```
aNumArray    : ARRAY[0..nTfOrder] OF FLOAT;
aDenArray    : ARRAY[0..nTfOrder] OF FLOAT;
aStTfData    : ARRAY[0..nTfOrder] OF ST_CTRL_TRANSFERFUNCTION_2_DATA;
```

系数“$b_0$”至“$b_n$”存储在数组 aNumArray 中。其结构必须按以下方式组织：

```
aNumArray[0]:= b0;
aNumArray[1]:= b1;
...
aNumArray[n-1] := bn-1;
aNumArray[n] := bn;
```

系数“$a_0$”至“$a_n$”存储在数组 aDenArray 中。其结构必须按以下方式组织：

```
aDenArray[0]:= a0;
aDenArray[1]:= a1;
...
aDenArray[n-1] := an-1;
aDenArray[n] := an;
```

功能块所需的内部数据存储在数组 aStTfData 中。绝不能在 PLC 程序内部修改此数据。此流程也在下方列出的示例程序中予以说明。

### 输入

```
VAR_INPUT
    fIn          : FLOAT;
    fManValue    : FLOAT;
    eMode        : E_CTRL_MODE;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| fIn | FLOAT | 传递函数的输入 |
| fManValue | FLOAT | 在手动模式下，其数值直接出现在输出端的输入。 |
| eMode | E_CTRL_MODE | 指定功能块的运行模式的输入。 |

### 输出

```
VAR_OUTPUT
    fOut         : FLOAT;
    eState       : E_CTRL_STATE;
    bError       : BOOL;
    eErrorId     : E_CTRL_ERRORCODES;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| fOut | FLOAT | 传递函数的输出 |
| eState | E_CTRL_STATE | 功能块的状态 |
| bError | BOOL | 在设置输出 bError 时，提供错误编号 [▶ 157]。 |
| eErrorId | E_CTRL_ERRORCODES | 一旦发生错误，则变为 TRUE。 |

### 输入/输出

```
VAR_IN_OUT
    stParams     : ST_CTRL_TRANSFERFUNCTION_2_PARAMS;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| stParams | ST_CTRL_TRANSFERFUNCTION_2_PARAMS | 功能块的参数结构 |

stParams 由以下元素组成：

```
TYPE
ST_CTRL_TRANSFERFUNCTION_2_PARAMS:
STRUCT
    tTaskCycleTime                : TIME;
    tCtrlCycleTime                : TIME := T#0ms;
    nOrderOfTheTransferfunction   : USINT;
    pNumeratorArray_ADR           : POINTER TO FLOAT := 0;
    nNumeratorArray_SIZEOF        : UINT;
    pDenominatorArray_ADR         : POINTER TO FLOAT := 0;
    nDenomiantorArray_SIZEOF      : UINT;
    pTransferfunction2Data_ADR    : POINTER TO
ST_CTRL_TRANSFERFUNCTION_2_DATA;
    nTransferfunction2Data_SIZEOF : UINT;
END_STRUCT
END_TYPE
```

| 名称 | 类型 | 描述 |
|---|---|---|
| tTaskCycleTime | TIME | 调用功能块的循环时间。如果在每个循环中都调用该功能块，这相当于调用任务的任务循环时间。 |
| tCtrlCycleTime | TIME | 处理控制回路的循环时间。该时间必须大于或等于 **TaskCycleTime**。功能块使用该输入值在内部计算是否需要在当前周期内更新状态和输出值。 |
| nOrderOfTheTransferfunction | USINT | 传递函数的阶次 [0...] |

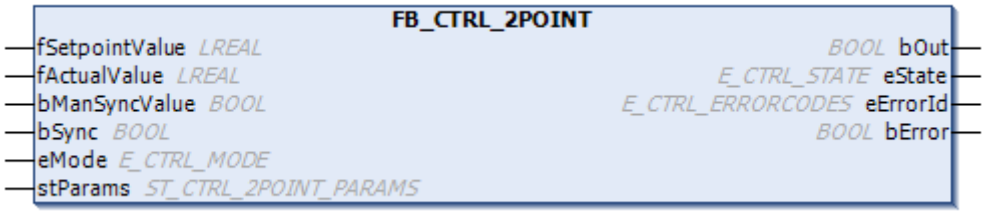
| 名称 | 类型 | 描述 |
|---|---|---|
| pNumeratorArray_ADR | POINTER TO FLOAT | 包含分子系数的数组的地址 |
| nNumeratorArray_SIZEOF | UINT | 包含分子系数的数组的大小，以字节为单位 |
| pDenominatorArray_ADR | POINTER TO FLOAT | 包含分母系数的数组的地址 |
| nDenomiantorArray_SIZEOF | UINT | 包含分母系数的数组的大小，以字节为单位 |
| pTransferfunction2Data_ADR | POINTER TO ST_CTRL_TRANSFERFUNCTION_2_DATA | 数据数组的地址 |
| nTransferfunction2Data_SIZEOF | UINT | 数据数组的大小，以字节为单位 |

**示例：**

```
PROGRAM PRG_TRANSFERFUNCTION_2_TEST
VAR CONSTANT
    nTfOrder : USINT := 2;
END_VAR
VAR
    aNumArray           : ARRAY[0..nTfOrder] OF FLOAT;
    aDenArray           : ARRAY[0..nTfOrder] OF FLOAT;
    aStTfData           : ARRAY[0..nTfOrder] OF ST_CTRL_TRANSFERFUNCTION_2_DATA;
    eMode               : E_CTRL_MODE;
    stParams            : ST_CTRL_TRANSFERFUNCTION_2_PARAMS;
    eErrorId            : E_CTRL_ERRORCODES;
    bError              : BOOL;
    fbTansferfunction   : FB_CTRL_TRANSFERFUNCTION_2;
    bInit               : BOOL := TRUE;
    fIn                 : FLOAT := 0;
    fOut                : FLOAT;
    b_0, b_1, b_2       : FLOAT;
    a_0,a_1,a_2         : FLOAT;
END_VAR
IF bInit THEN
    aNumArray[0] := 1.24906304658218E-007;
    aNumArray[1] := 2.49812609316437E-007;
    aNumArray[2] := 1.24906304658218E-007;
    aDenArray[0] := 0.998501124344101;
    aDenArray[1] := -1.99850062471888;
    aDenArray[2] := 1.0;
    stParams.tTaskCycleTime    := T#2ms;
    stParams.tCtrlCycleTime    := T#2ms;
    stParams.nOrderOfTheTransferfunction := nTfOrder;
    eMode := eCTRL_MODE_ACTIVE;
    bInit := FALSE;
END_IF
stParams.pNumeratorArray_ADR := ADR(aNumArray);
stParams.nNumeratorArray_SIZEOF := SIZEOF(aNumArray);
stParams.pDenominatorArray_ADR := ADR(aDenArray );
stParams.nDenominatorArray_SIZEOF := SIZEOF(aDenArray );
stParams.pTransferfunction2Data_ADR := ADR(aStTfData );
stParams.nTransferfunction2Data_SIZEOF := SIZEOF(aStTfData);
fbTansferfunction (fIn := fIn,
                   eMode := eMode,
                   stParams := stParams,
                   fOut => fOut,
                   eErrorId => eErrorId,
                   bError => bError);
```

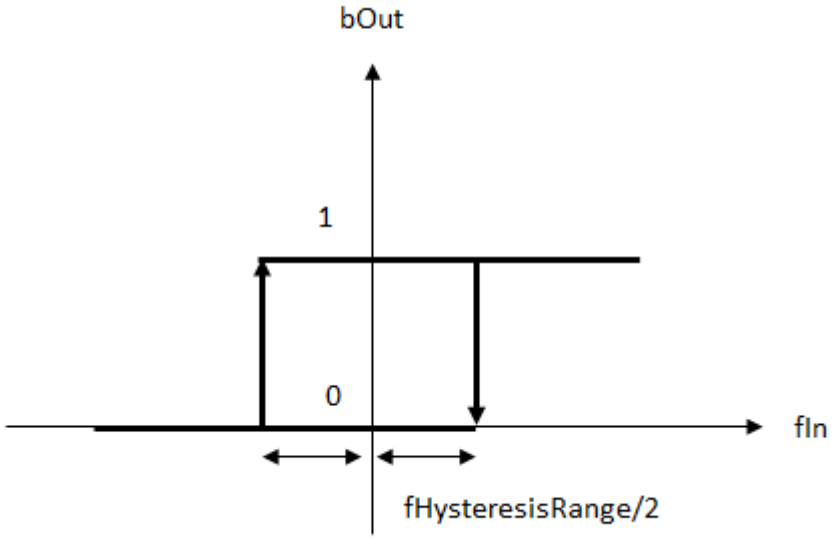
## 4.2.1.3 控制器

### 4.2.1.3.1 FB_CTRL_2POINT




该功能块在功能图中提供了一个两点传递单元。

**输出的行为**




**输入**

```
VAR_INPUT
    fSetpointValue    : FLOAT;
    fActualValue      : FLOAT;
    bManSyncValue     : BOOL;
    bSync             : BOOL;
    eMode             : E_CTRL_MODE;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| fSetpointValue | FLOAT | 控制变量的设定点 |
| fActualValue | FLOAT | 控制变量的实际值 |
| bManSyncValue | BOOL | 通过该输入可将二点传递单元切换至两个状态之一。 |
| bSync | BOOL | 该输入端的上升沿将两点单元设置为值“bManSyncValue”。 |
| eMode | E_CTRL_MODE | 指定功能块的运行模式 [▶ 157]的输入。 |

**输出**

```
VAR_OUTPUT
    bOut      : BOOL;
    eState    : E_CTRL_STATE;
    eErrorId  : E_CTRL_ERRORCODES;
    bError    : BOOL;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| bOut | BOOL | 两点单元的输出 |
| eState | E_CTRL_STATE | 功能块的状态 |
| eErrorId | E_CTRL_ERROR CODES | 在设置输出 bError 时，提供错误编号 [▶ 157]。 |
| bError | BOOL | 一旦发生错误，则变为 TRUE。 |

### 输入/输出

```
VAR_IN_OUT
    stParams    : ST_CTRL_2POINT_PARAMS;
END_VAR
```

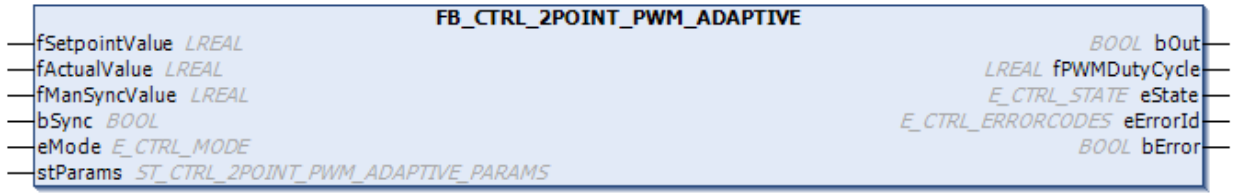
| 名称 | 类型 | 描述 |
|---|---|---|
| stParams | ST_CTRL_ 2POINT_ PARAMS | 两点单元的参数结构 |

stParams 由以下元素组成：

```
TYPE
ST_CTRL_2POINT_PARAMS :
STRUCT
    tCtrlCycleTime      : TIME := T#0ms;
    tTaskCycleTime      : TIME := T#0ms;
    fHysteresisRange    : FLOAT;
END_STRUCT
END_TYPE
```

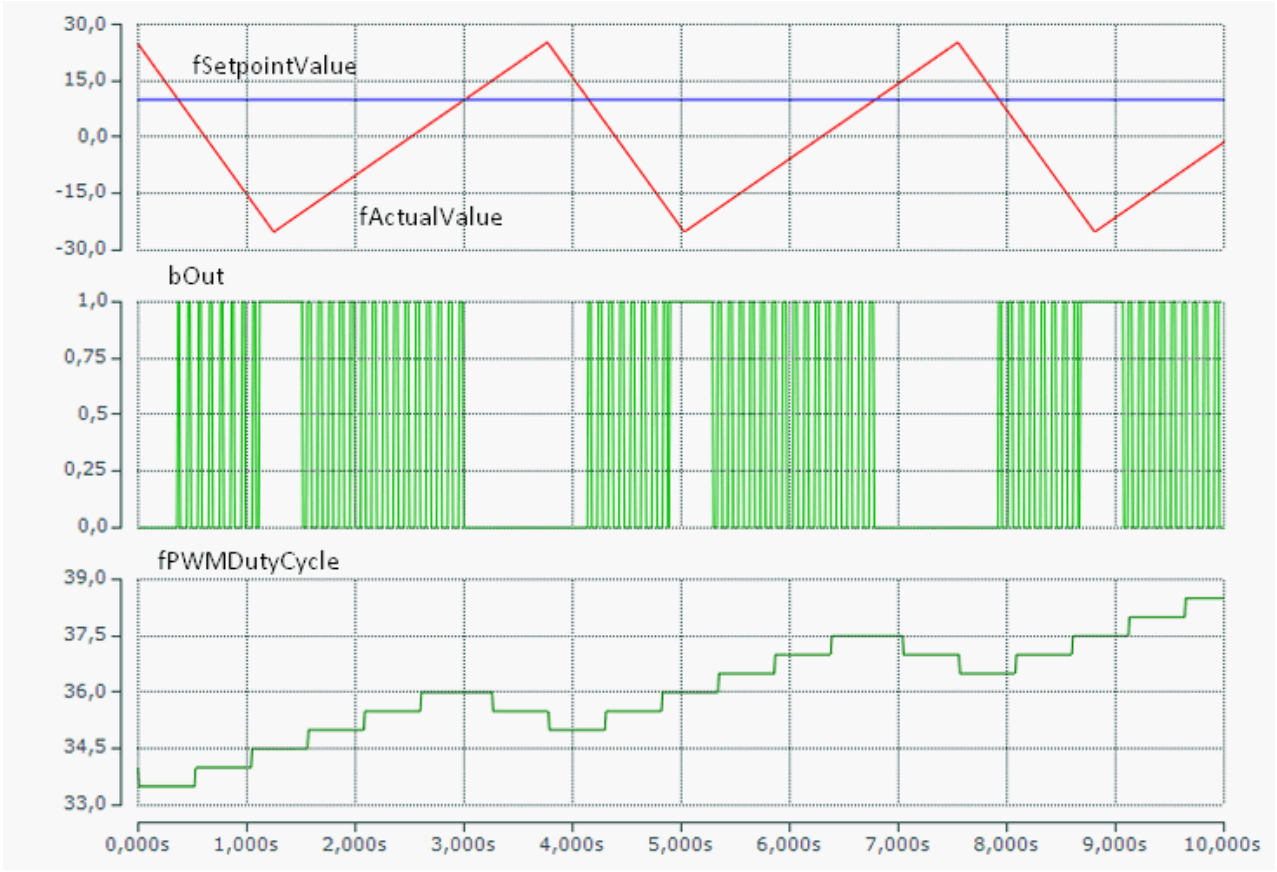
| 名称 | 类型 | 描述 |
|---|---|---|
| tCtrlCycleTime | TIME | 处理控制回路的循环时间。该时间必须大于或等于 **TaskCycleTime**。功能块使用该输入值在内部计算是否需要在当前周期内更新状态和输出值。 |
| tTaskCycle Time | TIME | 调用功能块的循环时间。如果在每个循环中都调用该功能块，这相当于调用任务的任务循环时间。 |
| fHysteresis Range | FLOAT | 滞环范围，请参见上图。 |

## 4.2.1.3.2 FB_CTRL_2POINT_PWM_ADAPTIVE




该功能块提供了一个自适应二位式控制器。它特别适用于进口温度较高且使用热执行器的单区域控制场景。

### 输出的行为




### 功能的描述

控制器内部使用一个 PWM 功能块，用于控制热执行器。PWM 功能块的脉宽占空比可根据受控系统的行为进行自适应调整。当控制偏差“fE = setpoint – actual value”大于“0”时，PWM 输出开启；当控制偏差小于“0”时，PWM 输出关闭。只要控制偏差保持在范围“[-fOkRange ... fOkRange]”之内，脉宽占空比就不会改变。如果“fE > fOkRange”，脉宽占空比将按“fStepSize”增加。此类增加后，必须经过时间 `tWaitTime` 才能再次改变脉宽占空比。如果 fE 低于“-fOkRange”，脉宽占空比将按“fStepSize”减小。脉宽占空比仅限在范围“[fMinLimit ... fMaxLimit]”之内进行修改。PWM 信号的周期由参数 `tPWMPeriod` 指定。

### 输入

```
VAR_INPUT
    fSetpointValue    : FLOAT;
    fActualValue      : FLOAT;
    fManSyncValue     : FLOAT;
    bSync             : BOOL;
    eMode             : E_CTRL_MODE;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| fSetpointValue | FLOAT | 控制变量的设定点 |
| fActualValue | FLOAT | 控制变量的实际值 |
| fManSync Value | FLOAT | 用于设置控制器占空比的输入，或用于在手动模式下设定输出的输入。如果“fManSyncValue > 0.0”，则在手动模式下设置输出。 |
| bSync | BOOL | 该输入端的上升沿将内部 PWM 功能块的脉宽占空比设置为值“fManSyncValue”。 |
| eMode | E_CTRL_MODE | 指定功能块的运行模式 [▶ 157]的输入。 |

### 输出

```
VAR_OUTPUT
    bOut            : BOOL;
    fPWMDutyCycle   : FLOAT;
    eState          : E_CTRL_STATE;
    eErrorId        : E_CTRL_ERRORCODES;
    bError          : BOOL;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| bOut | BOOL | 控制器输出 |
| fPWMDutyCycle | FLOAT | 内部 PWM 功能块的当前脉宽占空比 |
| eState | E_CTRL_STATE | 功能块的状态 |
| eErrorId | E_CTRL_ERROR CODES | 在设置输出 bError 时，提供错误编号 [▶ 157]。 |
| bError | BOOL | 一旦发生错误，则变为 TRUE。 |

### 输入/输出

```
VAR_IN_OUT
    stParams    : ST_CTRL_2POINT_PWM_ADAPTIVE_PARAMS;
END_VAR
```

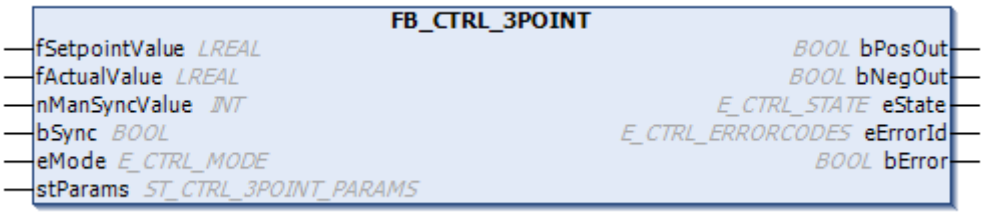
| 名称 | 类型 | 描述 |
|---|---|---|
| stParams | ST_CTRL_2POINT _PWM_ ADAPTIVE_ PARAMS | 两点单元的参数结构 |

stParams 由以下元素组成：

```
TYPE
ST_CTRL_2POINT_PWM_ADAPTIVE_PARAMS:
STRUCT
    tCtrlCycleTime    : TIME := T#0ms;
    tTaskCycleTime    : TIME := T#0ms;
    tPWMPeriod        : TIME;
    fOkRange          : FLOAT;
    fForceRange       : FLOAT;
    fStepSize         : FLOAT;
    fMinLimit         : FLOAT;
    fMaxLimit         : FLOAT;
    tWaitTime         : TIME;
END_STRUCT
END_TYPE
```

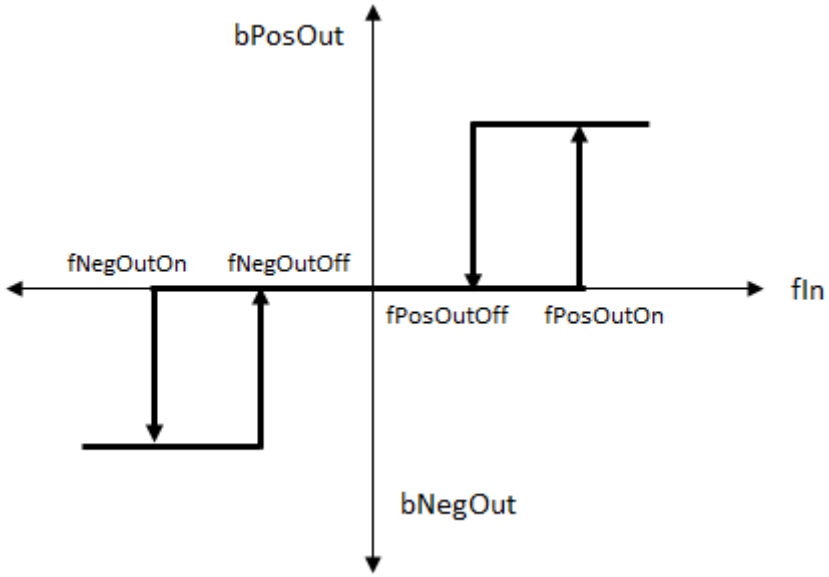
| 名称 | 类型 | 描述 |
|---|---|---|
| tCtrlCycleTime | TIME | 处理控制回路的循环时间。该时间必须大于或等于 **TaskCycleTime**。功能块使用该输入值在内部计算是否需要在当前周期内更新状态和输出值。 |
| tTaskCycleTime | TIME | 调用功能块的循环时间。如果在每个循环中都调用该功能块，这相当于调用任务的任务循环时间。 |
| tPWMPeriod | TIME | PWM 信号的周期 |
| fOkRange | FLOAT | 占空比不发生变化的 “fE” 范围。 |
| fForceRange | FLOAT | 如果 “fE” 超出此范围，输出将永久设置为 TRUE。 |
| fStepSize | FLOAT | 每次调整脉宽占空比时的变化值。[0% ... 100%] |
| fMinLimit | FLOAT | 最大脉宽占空比（以百分比为单位）[0% ... 100%] |
| fMaxLimit | FLOAT | 最小脉宽占空比（以百分比为单位）[0% ... 100%] |
| tWaitTime | TIME | 单次修改脉宽占空比之间的等待时间 |

## 4.2.1.3.3 FB_CTRL_3POINT




该功能块在功能图中提供了一个三点传递单元。

**输出的行为**




### 输入

```
VAR_INPUT
    fSetpointValue  : FLOAT;
    fActualValue    : FLOAT;
    nManSyncValue   : INT;
    bSync           : BOOL;
    eMode           : E_CTRL_MODE;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| fSetpointValue | FLOAT | 控制变量的设定点 |
| fActualValue | FLOAT | 控制变量的实际值 |
| nManSyncValue | INT | 通过该输入可将三点单元切换至三个状态之一。<br>nManSyncValue >= 1 → bPosOut = TRUE, bNegOut = FALSE<br>nManSyncValue <= -1 → bPosOut = FALSE, bNegOut = TRUE<br>otherwise → bPosOut = FALSE, bNegOut = FALSE |
| bSync | BOOL | 该输入端的上升沿将三点单元设置为值“bManSyncValue”。 |
| eMode | E_CTRL_MODE | 指定功能块的运行模式 [▶ 157]的输入。 |

### 输出

```
VAR_OUTPUT
    bPosOut     : BOOL;
    bNegOut     : BOOL;
    eState      : E_CTRL_STATE;
    eErrorId    : E_CTRL_ERRORCODES;
    bError      : BOOL;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| bPosOut | BOOL | 如果特性曲线的上节点处于活动状态，则三点单元的该输出为 TRUE。 |
| bNegOut | BOOL | 如果特性曲线的下节点处于活动状态，则三点单元的该输出为 TRUE。 |
| eState | E_CTRL_STATE | 功能块的状态 |
| eErrorId | E_CTRL_ERRORCODES | 在设置输出 bError 时，提供错误编号 [▶ 157]。 |
| bError | BOOL | 一旦发生错误，则变为 TRUE。 |

### 输入/输出

```
VAR_IN_OUT
    stParams    : ST_CTRL_3POINT_PARAMS;
END_VAR
```

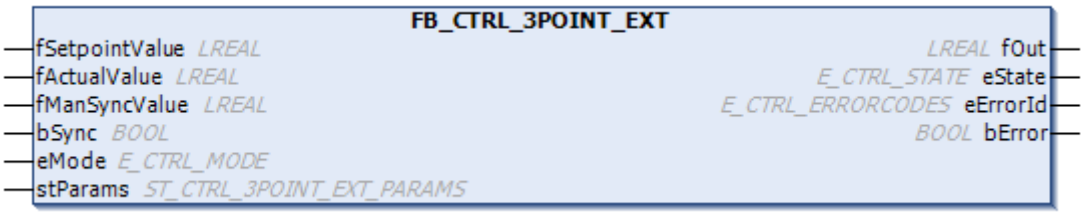
| 名称 | 类型 | 描述 |
|---|---|---|
| stParams | ST_CTRL_3POINT_PARAMS | 三点单元的参数结构 |

stParams 由以下元素组成：

```
TYPE
ST_CTRL_3POINT_PARAMS :
STRUCT
    tCtrlCycleTime    : TIME := T#0ms;
    tTaskCycleTime    : TIME := T#0ms;
    fPosOutOn         : FLOAT;
    fPosOutOff        : FLOAT;
    fNegOutOn         : FLOAT;
    fNegOutOff        : FLOAT;
END_STRUCT
END_TYPE
```

| 名称 | 类型 | 描述 |
|---|---|---|
| tCtrlCycleTime | TIME | 处理控制回路的循环时间。该时间必须大于或等于 **TaskCycleTime**。功能块使用该输入值在内部计算是否需要在当前周期内更新状态和输出值。 |
| tTaskCycleTime | TIME | 调用功能块的循环时间。如果在每个循环中都调用该功能块，这相当于调用任务的任务循环时间。 |
| fPosOutOn | FLOAT | 导致 bPosOut = FALSE 切换为 bPosOut = TRUE（bNegOut = FALSE）所对应的控制偏差。 |
| fPosOutOff | FLOAT | 导致 bPosOut = TRUE 切换为 bPosOut = FALSE（bNegOut = FALSE）所对应的控制偏差。 |
| fNegOutOn | FLOAT | 导致 bNegOut = FALSE 切换为 bNegOut = TRUE（bPosOut = FALSE）所对应的控制偏差。 |
| fNegOutOff | FLOAT | 导致 bNegOut = TRUE 切换为 bNegOut = FALSE（bPosOut = FALSE）所对应的控制偏差。 |

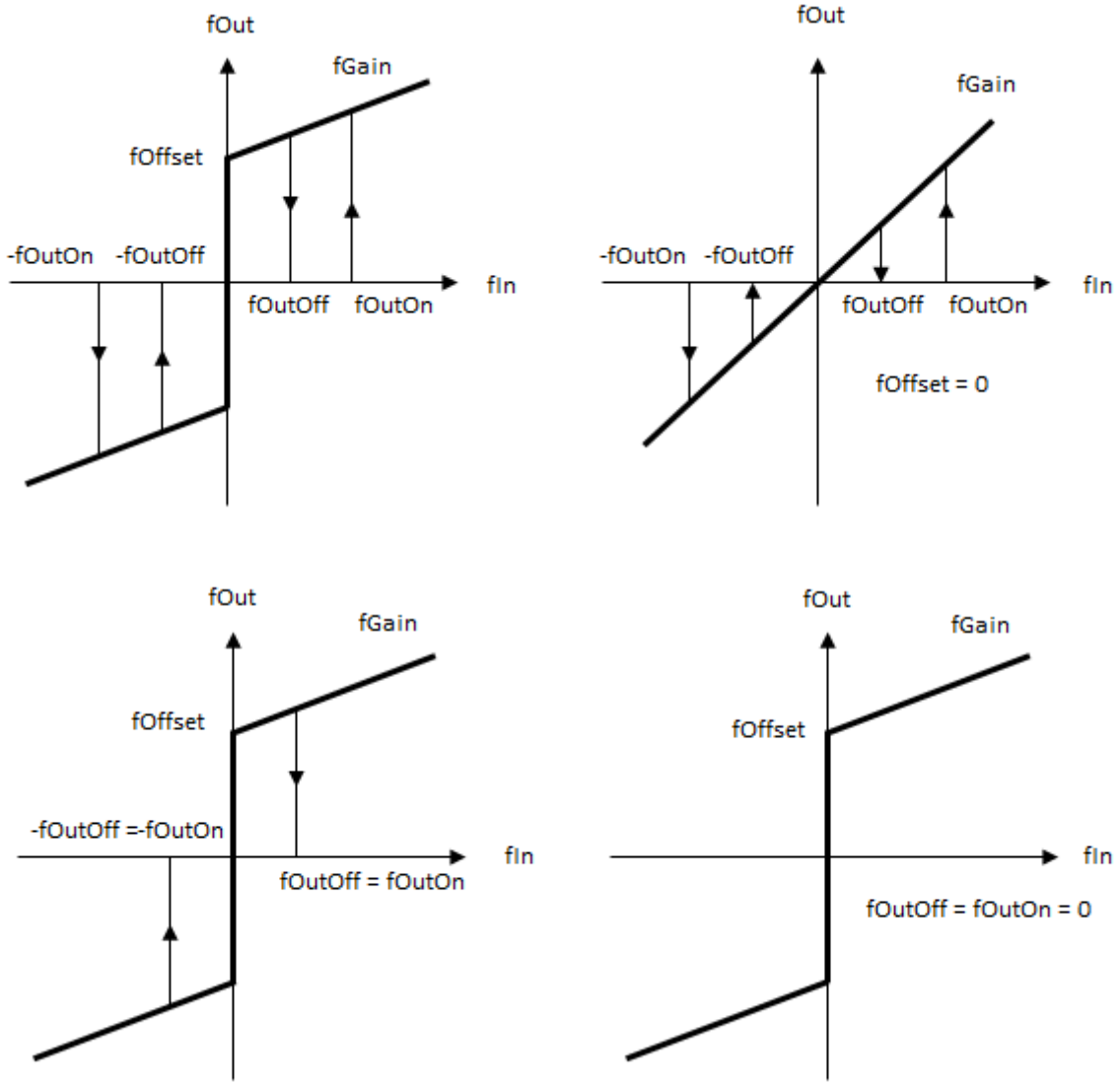
## 4.2.1.3.4 FB_CTRL_3POINT_EXT




该功能块在功能图中提供了一个扩展型三点单元。

**输出的行为**

```
fIn := fSetpointValue - fActualValue;
```




### 输入

```
VAR_INPUT
    fSetpointValue    : FLOAT;
    fActualValue      : FLOAT;
    fManSyncValue     : FLOAT;
```

```
    bSync             : BOOL;
    eMode             : E_CTRL_MODE;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| fSetpointValue | FLOAT | 控制变量的设定点 |
| fActualValue | FLOAT | 控制变量的实际值 |
| fManSyncValue | FLOAT | 用于将扩展型三点单元设置为输出分支之一的输入。<br>`\| fManSyncValue \| < 1 → fOut = 0.0`<br>`\| fManSyncValue \| >= 1 → fOut = fE * fGain + fOffset` |
| bSync | BOOL | 该输入端的上升沿将三点单元设置为值“fManSyncValue”。 |
| eMode | E_CTRL_MODE | 指定功能块的运行模式 [▶ 157]的输入。 |

### 输出

```
VAR_OUTPUT
    fOut          : FLOAT;
    eState        : E_CTRL_STATE;
    eErrorId      : E_CTRL_ERRORCODES;
    bError        : BOOL;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| fOut | FLOAT | 扩展型三点单元的输出 |
| eState | E_CTRL_STATE | 功能块的状态 |
| eErrorId | E_CTRL_ERRORCODES | 在设置输出 `bError` 时，提供错误编号 [▶ 157]。 |
| bError | BOOL | 一旦发生错误，则变为 TRUE。 |

### 输入/输出

```
VAR_IN_OUT
    stParams      : ST_CTRL_3POINT_EXT_PARAMS;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| stParams | ST_CTRL_3POINT_EXT_PARAMS | 扩展型三点单元的参数结构 |

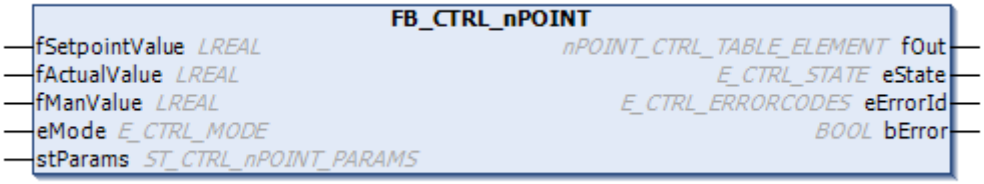
`stParams` 由以下元素组成：

```
TYPE
ST_CTRL_3POINT_EXT_PARAMS :
STRUCT
    tCtrlCycleTime    : TIME := T#0ms;
    tTaskCycleTime    : TIME := T#0ms;
    fOutOff           : FLOAT;
    fOutOn            : FLOAT;
    fGain             : FLOAT;
    fOffset           : FLOAT;
END_STRUCT
END_TYPE
```

| 名称 | 类型 | 描述 |
|---|---|---|
| tCtrlCycleTime | TIME | 处理控制回路的循环时间。该时间必须大于或等于 **TaskCycleTime**。功能块使用该输入值在内部计算是否需要在当前周期内更新状态和输出值。 |
| tTaskCycleTime | TIME | 调用功能块的循环时间。如果在每个循环中都调用该功能块，这相当于调用任务的任务循环时间。 |
| fOutOff | FLOAT | 如果控制偏差低于该值，则输出将被关闭（设置为零）。 |
| fOutOn | FLOAT | 如果控制偏差超过该值，则输出将被开启。 |
| fGain | FLOAT | 增益系数 |

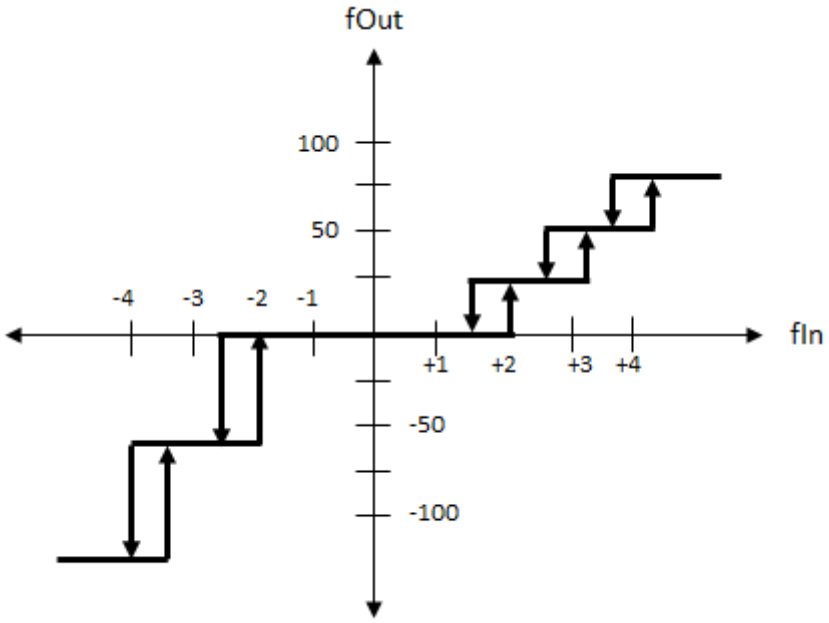
| 名称 | 类型 | 描述 |
|---|---|---|
| fOffset | FLOAT | 偏移 |

### 4.2.1.3.5 FB_CTRL_nPOINT




该功能块在功能图中提供了一个 n 点传递单元。

**输出的行为**




示例数据数组：

| fE | fOut |
|---|---|
| xx | -100 |
| -4 | -50 |
| -2 | 0 |
| 1 | 25 |
| 2 | 50 |
| 3 | 75 |
| 4 | 100 |

索引为（1,1）的数组的值（即第一行左侧的值）可以自由选择，因为该值不会参与计算。

### 输入

```
VAR_INPUT
    fSetpointValue    : FLOAT;
    fActualValue      : FLOAT;
    fManValue         : BOOL;
    eMode             : E_CTRL_MODE;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| fSetpointValue | FLOAT | 控制变量的设定点 |
| fActualValue | FLOAT | 控制变量的实际值 |
| fManValue | BOOL | 在手动模式下，其数值将被输出的输入。 |
| eMode | E_CTRL_MODE | 指定功能块的运行模式 [▶ 157]的输入。 |

### 输出

```
VAR_OUTPUT
    fOut         : nPOINT_CTRL_TABLE_ELEMENT;
    eState       : E_CTRL_STATE;
    eErrorId     : E_CTRL_ERRORCODES;
    bError       : BOOL;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| fOut | nPOINT_CTRL_TABLE_ELEMENT | n 点单元的输出 |
| eState | E_CTRL_STATE | 功能块的状态 |
| eErrorId | E_CTRL_ERRORCODES | 在设置输出 bError 时，提供错误编号 [▶ 157]。 |
| bError | BOOL | 一旦发生错误，则变为 TRUE。 |

### 输入/输出

```
VAR_IN_OUT
    stParams     : ST_CTRL_nPOINT_PARAMS;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| stParams | ST_CTRL_nPOINT_PARAMS | n 点单元的参数结构 |

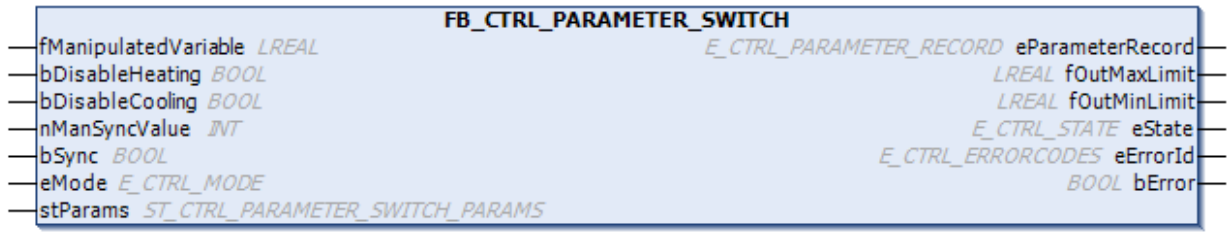
stParams 由以下元素组成：

```
TYPE
ST_CTRL_nPOINT_PARAMS :
STRUCT
    tCtrlCycleTime           : TIME := T#0ms;
    tTaskCycleTime           : TIME := T#0ms;
    pDataTable_ADR           : POINTER TO nPOINT_CTRL_TABLE_ELEMENT
     := 0;
    nDataTable_SIZEOF        : UINT := 0;
    nDataTable_NumberOfRows  : UINT := 0;
    fHysteresisRange         : FLOAT;
END_STRUCT
END_TYPE
```

| 名称 | 类型 | 描述 |
|---|---|---|
| tCtrlCycleTime | TIME | 处理控制回路的循环时间。该时间必须大于或等于 **TaskCycleTime**。功能块使用该输入值在内部计算是否需要在当前周期内更新状态和输出值。 |
| tTaskCycleTime | TIME | 调用功能块的循环时间。如果在每个循环中都调用该功能块，这相当于调用任务的任务循环时间。 |

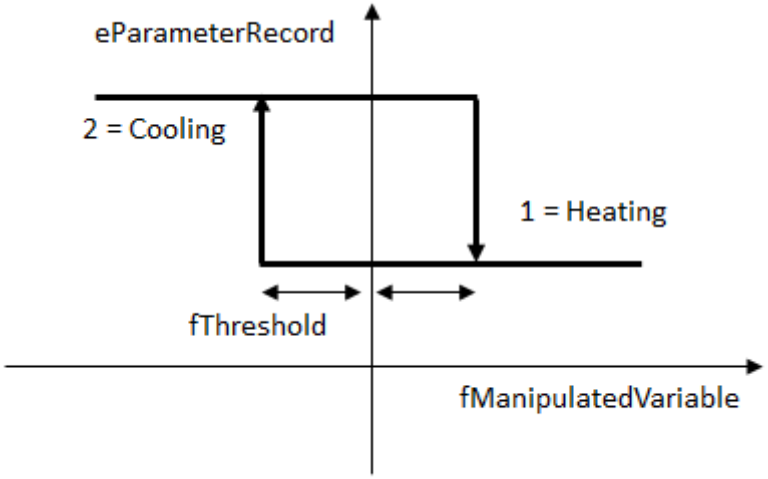
| 名称 | 类型 | 描述 |
|---|---|---|
| pDataTable_ADR | POINTER TO nPOINT_CTRL_TABLE_ELEMENT | 数据表的地址 |
| nDataTable_SIZEOF | UINT | 数据表的大小，以字节为单位 |
| nDataTable_NumberOfRows | UINT | 数据表中的行数 |
| fHysteresisRange | FLOAT | 滞环范围，请参见上图。滞环范围的功能与 FB_CTRL_2POINT [▶ 37] 的描述一致。 |

### 4.2.1.3.6 FB_CTRL_PARAMETER_SWITCH




该功能块可用于切换 FB_CTRL_PID_SPLITRANGE 所使用的参数集。

**输出的行为**




**功能块的描述**

该功能块用于切换 FB_CTRL_PID_SPLITRANGE 所使用的参数集。该功能块专用于切换可使用两个执行器进行加热和冷却的控制器的参数集，并为控制器设置限值。时间 tMinWaitTime 被指定为输入参数。在要求切换时，必须至少经过此时间段，以便更改参数范围，并将控制器限值设置为可以从加热运行切换到冷却运行。此举旨在防止仅因控制器略微超调而立即改变运行模式。

在加热运行时，选择参数范围“**eCTRL_PARAMETER_RECORD_HEATING = heating**”，在冷却运行时，选择参数范围“**eCTRL_PARAMETER_RECORD_COOLING = cooling**”。控制器的参数集必须根据此布局进行指定。

转换请求本身由一个两点单元提供（请参见图示）。控制器的输出值（即控制值）应用作图示特性滞环曲线的输入值。由滞环单元创建的转换请求必须至少在指定的等待时间内存在，以便更改参数范围。

bDisableRange1 和 bDisableRange2 输入可防止切换到其中一个范围。因此，例如，在夏季可以禁用加热运行，在冬季可以禁用冷却运行。此外，还可以根据当前的控制偏差来更改运行模式。例如，在夏季可能需要在超温 2°C 时才能切换至冷却运行。通过正确连接输入也可以实现这一点。

除了提供参数范围的输出外，还会输出最大和最小限值，它们都可以被复制到 PID 控制器的参数集中。如果 FB_CTRL_PARAMETER_SWITCH 处于**加热**运行模式，则限值设置如下：

```
fOutMinLimit = -1.0 stParams.fThreshold;

fOutMaxLimit = stParams.fOutMaxLimit;
```

在**冷却模式**下，限值设置如下：

```
fOutMinLimit = stParams.fOutMaxLimit;

fOutMaxLimit = stParams.fThreshold;
```

### 输入

```
VAR_INPUT
    fManipulatedVariable  : FLOAT;
    nManSyncValue         : eCTRL_PARAMETER_RECORD_HEATING;
    bSync                 : BOOL;
    eMode                 : E_CTRL_MODE;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| fManipulated Variable | FLOAT | FB_Parameter_Switch 的输入值。这应该等于控制器的输出值。 |
| nManSync Value | eCTRL_PARAMETER_RECORD_ HEATING | 用于将功能块设置为参数范围之一的输入。 |
| bSync | BOOL | 该输入端的上升沿将功能块设置为值“nManSyncValue”。 |
| eMode | E_CTRL_MODE | 指定功能块的运行模式 [▶ 157]的输入。 |

### 输出

```
VAR_OUTPUT
    eParameterRecord    : E_CTRL_PARAMETER_RECORD;
    fOutMaxLimit        : FLOAT;
    fOutMinLimit        : FLOAT;
    eState              : E_CTRL_STATE;
    eErrorId            : E_CTRL_ERRORCODES;
    bError              : BOOL;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| eParameterRecord | E_CTRL_ PARAMETER_ RECORD | 用于确定参数范围的功能块的输出。 |
| fOutMaxLimit | FLOAT | 控制器受到限制的最大输出值。（应将其复制到控制器的参数结构中。） |
| fOutMinLimit | FLOAT | 控制器受到限制的最小输出值。（应将其复制到控制器的参数结构中。） |
| eState | E_CTRL_ STATE | 功能块的状态 |
| eErrorId | E_CTRL_ERROR CODES | 在设置输出 bError 时，提供错误编号 [▶ 157]。 |
| bError | BOOL | 一旦发生错误，则变为 TRUE。 |

### 输入/输出

```
VAR_IN_OUT
    stParams    : ST_CTRL_PARAMETER_SWITCH_PARAMS;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| stParams | ST_CTRL_PARAMETER_SWITCH_PARAMS | 功能块的参数结构 |

`stParams` 由以下元素组成：

```
TYPE
ST_CTRL_2POINT_PARAMS :
STRUCT
    tTaskCycleTime    : TIME;
    tCtrlCycleTime    : TIME;
    fThreshold        : FLOAT;
    fOutMaxLimit      : FLOAT;
    fOutMinLimit      : FLOAT;
    tMinWaitTime      : TIME;
END_STRUCT
END_TYPE
```

| 名称 | 类型 | 描述 |
|---|---|---|
| tTaskCycleTime | TIME | 调用功能块的循环时间。如果在每个循环中都调用该功能块，这相当于调用任务的任务循环时间。 |
| tCtrlCycleTime | TIME | 处理控制回路的循环时间。该时间必须大于或等于 **TaskCycleTime**。功能块使用该输入值在内部计算是否需要在当前周期内更新状态和输出值。 |
| fThreshold | FLOAT | 切换阈值，请参见上图。 |
| fOutMaxLimit | FLOAT | 最大限值；该值可传递至控制器。 |
| fOutMinLimit | FLOAT | 最小限值；该值可传递至控制器。 |
| tMinWaitTime | TIME | 等待时间，请参见上文描述。 |

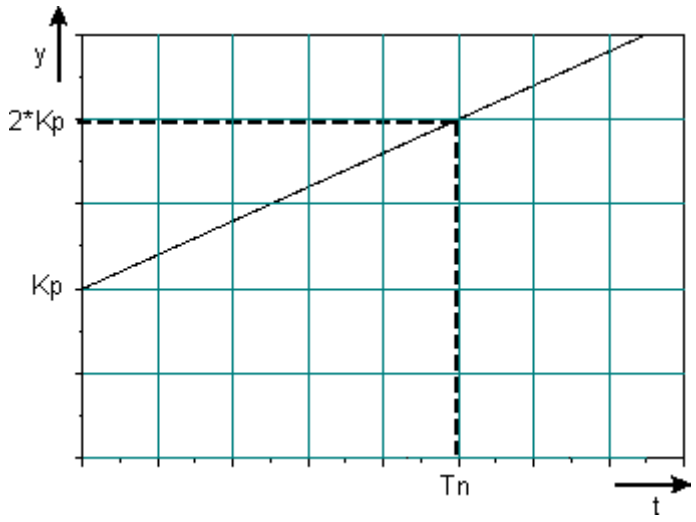
## 4.2.1.3.7 FB_CTRL_PI


该功能块在功能图中提供了一个 PI 传递单元。

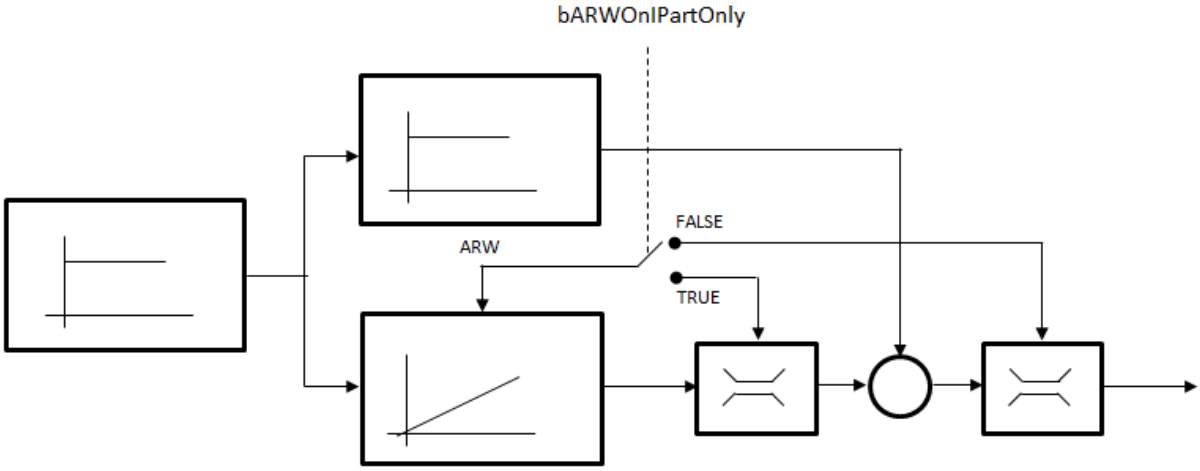
### 输出的行为

$$G(s) = K_p\left(1+\frac{1}{T_n s}\right)$$

**阶跃响应**




**ARW**




## 输入

```
VAR_INPUT
    fSetpointValue    : FLOAT;
    fActualValue      : FLOAT;
    fManSyncValue     : FLOAT;
    bSync             : BOOL;
    eMode             : E_CTRL_MODE;
    bHold             : BOOL;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| fSetpointValue | FLOAT | 控制变量的设定点 |
| fActualValue | FLOAT | 控制变量的实际值 |
| fManSyncValue | FLOAT | 用于设置 PI 单元的输入。 |
| bSync | BOOL | 该输入端的上升沿将 PI 单元设置为值“fManSyncValue”。 |
| eMode | E_CTRL_MODE | 指定功能块的运行模式 [▶ 157]的输入。 |
| bHold | BOOL | 该输入为 TRUE 时，内部状态（以及输出）将保持当前值不变，且不受控制偏差影响。 |

### 输出

```
VAR_OUTPUT
    fOut            : FLOAT;
    bARWactive      : BOOL;
    eState          : E_CTRL_STATE;
    eErrorId        : E_CTRL_ERRORCODES;
    bError          : BOOL;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| fOut | FLOAT | PI 单元的输出 |
| bARWactive | BOOL | 该输出为 TRUE 时，表示 PI 单元正处于限制状态。 |
| eState | E_CTRL_STATE | 功能块的状态 |
| eErrorId | E_CTRL_ERRORCODES | 在设置输出 bError 时，提供错误编号 [▶ 157]。 |
| bError | BOOL | 一旦发生错误，则变为 TRUE。 |

### 输入/输出

```
VAR_IN_OUT
    stParams        : ST_CTRL_PI_PARAMS;
END_VAR
```

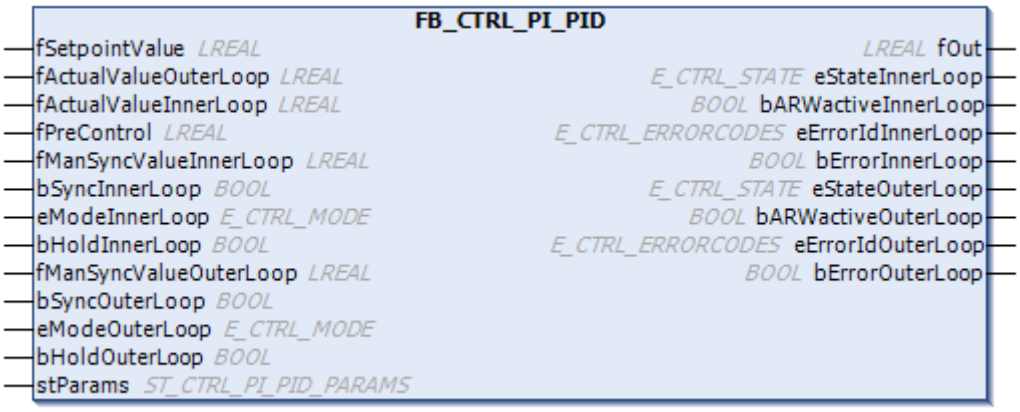
| 名称 | 类型 | 描述 |
|---|---|---|
| stParams | ST_CTRL_PI_PARAMS | PI 单元参数结构 |

stParams 由以下元素组成：

```
TYPE ST_CTRL_PI_PARAMS:
STRUCT
    tCtrlCycleTime      : TIME := T#0ms;
    tTaskCycleTime      : TIME := T#0ms;
    tTn                 : TIME := T#0ms;
    fKp                 : FLOAT := 0;
    fOutMaxLimit        : FLOAT := 1E38;
    fOutMinLimit        : FLOAT := -1E38;
    bARWOnIPartOnly     : BOOL := FALSE;
END_STRUCT
END_TYPE
```

| 名称 | 类型 | 描述 |
|---|---|---|
| tCtrlCycleTime | TIME | 处理控制回路的循环时间。该时间必须大于或等于 **TaskCycleTime**。功能块使用该输入值在内部计算是否需要在当前周期内更新状态和输出值。 |
| tTaskCycleTime | TIME | 调用功能块的循环时间。如果在每个循环中都调用该功能块，这相当于调用任务的任务循环时间。 |
| tTn | TIME | 积分作用时间 |
| fKp | FLOAT | 控制器放大系数/传递系数 |
| fOutMaxLimit | FLOAT | 积分停止和输出限制的上限（ARW 措施）。bARWActive 输出为 TRUE 时，表示达到该限值。 |
| fOutMinLimit | FLOAT | 积分停止和输出限制的下限（ARW 措施）。bARWActive 输出为 TRUE 时，表示达到该限值。 |
| bARWOnIPartOnly | BOOL | 如果该参数为 FALSE（标准设置），则当控制器的完整输出达到上限或下限时，I 分量的积分将停止。如果该参数为 TRUE，则当 I 分量（积分器的输出）达到某个限值时，积分将停止。（请参见功能图。） |

## 4.2.1.3.8 FB_CTRL_PI_PID




该功能块在功能图中提供了一个级联 PI-PID 控制器。在内部，该功能块使用 FB_CTRL_PI，FB_CTRL_LIMITER 和 FB_CTRL_PID 传递单元。

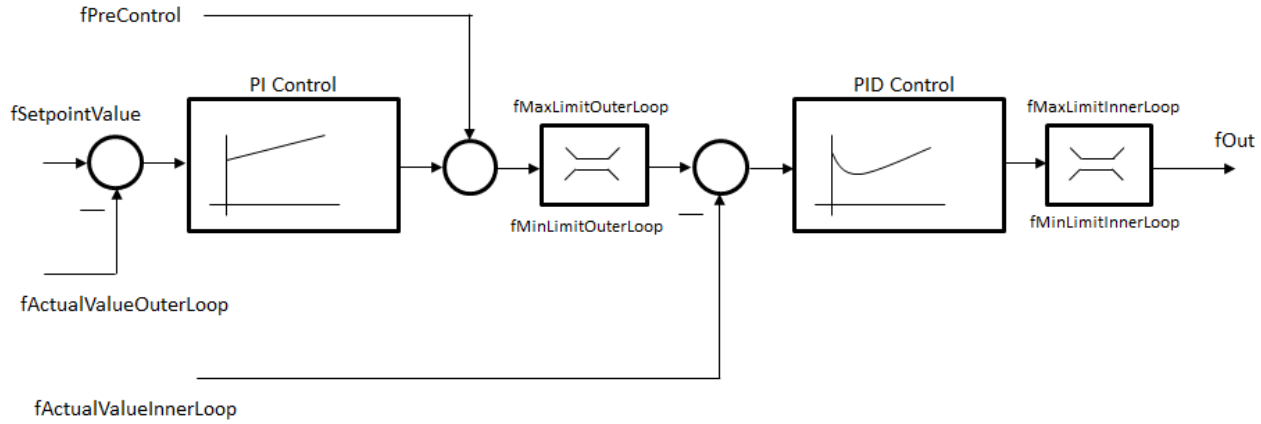
**PI 单元的传递函数**

$$G(s) = K_p\left(1+\frac{1}{T_n s}\right)$$

**PID 单元的传递函数**

$$G(s) = K_p\left(1+\frac{1}{T_n s}+\frac{T_v s}{1+T_d s}\right)$$

**级联传递单元的功能图**




### 输入

```
VAR_INPUT
    fSetpointValue          : FLOAT;
    fActualValueOuterLoop   : FLOAT;
    fActualValueInnerLoop   : FLOAT;
    fPreControl             : FLOAT;
    fManSyncValueInnerLoop  : FLOAT;
    bSyncInnerLoop          : BOOL;
    eModeInnerLoop          : E_CTRL_MODE;
    bHoldInnerLoop          : BOOL;
    fManSyncValueOuterLoop  : FLOAT;
    bSyncOuterLoop          : BOOL;
```

```
    eModeOuterLoop          : E_CTRL_MODE;
    bHoldOuterLoop          : BOOL;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| fSetpointValue | FLOAT | 控制变量的设定点 |
| fActualValueOuterLoop | FLOAT | 反馈至外层控制回路 PI 控制器的控制变量的实际值。 |
| fActualValueInnerLoop | FLOAT | 反馈至内层控制回路 PID 控制器的控制变量的实际值。 |
| fPreControl | FLOAT | 连接在 PI 控制器之后的前馈控制。 |
| fManSyncValueInnerLoop | FLOAT | 其值可用于设置 PID 单元（内层控制回路）的内部状态的输入。 |
| bSyncInnerLoop | BOOL | 该输入端的上升沿将 PID 单元（内层控制回路）设置为值“fManSyncValueInnerLoop”。 |
| eModeInnerLoop | E_CTRL_MODE | 指定 PID 单元（内层控制回路）的运行模式 [▶ 157]的输入。 |
| bHoldInnerLoop | BOOL | 该输入为 TRUE 时，PID 单元（内层控制回路）的内部状态将保持当前值不变。 |
| fManSyncValueOuterLoop | FLOAT | 其值可用于设置 PI 单元（外层控制回路）的内部状态的输入。 |
| bSyncOuterLoop | BOOL | 该输入端的上升沿将 PI 单元（外层控制回路）设置为值“fManSyncValueOuterLoop”。 |
| eModeOuterLoop | E_CTRL_MODE | 指定 PI 单元（外层控制回路）的运行模式 [▶ 157]的输入。 |
| bHoldOuterLoop | BOOL | 该输入为 TRUE 时，PI 单元（外层控制回路）的内部状态将保持当前值不变。 |

### 输出

```
VAR_OUTPUT
    fOut                    : FLOAT;
    eStateInnerLoop         : E_CTRL_STATE;
    bARWactiveInnerLoop     : BOOL;
    eErrorIdInnerLoop       : E_CTRL_ERRORCODES;
    bErrorInnerLoop         : BOOL;
    eStateOuterLoop         : E_CTRL_STATE;
    bARWactiveOuterLoop     : BOOL;
    eErrorIdOuterLoop       : E_CTRL_ERRORCODES;
    bErrorOuterLoop         : BOOL;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| fOut | FLOAT | PI-PID 单元的输出 |
| eStateInnerLoop | E_CTRL_STATE | 内部 PID 单元（内层控制回路）的状态 |
| bARWactiveInnerLoop | BOOL | 该输出为 TRUE 时，表示 PID 单元（内层控制回路）的输出正处于限制状态。 |
| eErrorIdInnerLoop | E_CTRL_ERRORCODES | 在设置 bError 输出时，返回 PID 单元（内层控制回路）的错误编号 [▶ 157]。 |
| bErrorInnerLoop | BOOL | 当 PID 单元（内层控制回路）发生错误时，则立即设置为 TRUE。 |
| eStateOuterLoop | E_CTRL_STATE | 内部 PI 单元（外层控制回路）的状态 |
| bARWactiveOuterLoop | BOOL | 该输出为 TRUE 时，表示 PI 单元（外层控制回路）的输出正处于限制状态。 |
| eErrorIdOuterLoop | E_CTRL_ERRORCODES | 在设置 bError 输出时，返回 PI 单元（外层控制回路）的错误编号 [▶ 157]。 |
| bErrorOuterLoop | BOOL | 当 PI 单元（外层控制回路）发生错误时，则立即设置为 TRUE。 |

### 输入/输出

```
VAR_IN_OUT
    stParams        : ST_CTRL_PI_PID_PARAMS;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| stParams | ST_CTRL_PI_PID_PARAMS | PI-PID 单元的参数结构 |

`stParams` 由以下元素组成：

```
TYPE
ST_CTRL_PI_PID_PARAMS :
STRUCT
    tCtrlCycleTime          : TIME := T#0ms;
    tTaskCycleTime          : TIME := T#0ms;
    fKp_OuterLoop           : FLOAT := 0;
    tTn_OuterLoop           : TIME := T#0s;
    fMaxLimit_OuterLoop     : FLOAT := 1E38;
    fMinLimit_OuterLoop     : FLOAT := -1E38;
    fKp_InnerLoop           : FLOAT := 0;
    tTn_InnerLoop           : TIME := T#0ms;
    tTv_InnerLoop           : TIME := T#0ms;
    tTd_InnerLoop           : TIME := T#0ms;
    fMaxLimit_InnerLoop     : FLOAT := 1E38;
    fMinLimit_InnerLoop     : FLOAT := -1E38;
END_STRUCT
END_TYPE
```

| 名称 | 类型 | 描述 |
|---|---|---|
| tCtrlCycleTime | TIME | 处理控制回路的循环时间。该时间必须大于或等于 **TaskCycleTime**。功能块使用该输入值在内部计算是否需要在当前周期内更新状态和输出值。 |
| tTaskCycleTime | TIME | 调用功能块的循环时间。如果在每个循环中都调用该功能块，这相当于调用任务的任务循环时间。 |
| fKp_OuterLoop | FLOAT | PI 单元（外层控制回路）的控制器放大/控制器系数 |
| tTn_OuterLoop | TIME | 内部 PI 单元（外层控制回路）的积分作用时间。如果将其参数设置为 T#0s，则 I 分量将被停用。 |
| fMaxLimit_OuterLoop | FLOAT | PID 单元积分停止和输出限制的上限（ARW 措施）。`bARWactiveOuterLoop` 输出为 TRUE 时，表示达到该限值。 |
| fMinLimit_OuterLoop | FLOAT | PID 单元积分停止和输出限制的下限（ARW 措施）。`bARWactiveOuterLoop` 输出为 TRUE 时，表示达到该限值。 |
| fKp_InnerLoop | FLOAT | PID 单元（内层控制回路）的控制器放大/控制器系数 |
| tTn_InnerLoop | TIME | PID 单元（内层控制回路）的积分作用时间。如果将其参数设置为 T#0s，则 I 分量将被停用。 |
| tTv_InnerLoop | TIME | PID 单元（内层控制回路）的微分作用时间。如果将其参数设置为 T#0s，则 D 分量将被停用。 |
| tTd_InnerLoop | TIME | PID 单元（内层控制回路）的阻尼时间 |
| fMaxLimit_InnerLoop | FLOAT | PID 单元积分停止和输出限制的上限（ARW 措施）。`bARWactiveInnerLoop` 输出为 TRUE 时，表示达到该限值。 |
| fMinLimit_InnerLoop | FLOAT | PID 单元积分停止和输出限制的下限（ARW 措施）。`bARWactiveInnerLoop` 输出为 TRUE 时，表示达到该限值。 |

## 4.2.1.3.9 FB_CTRL_PID

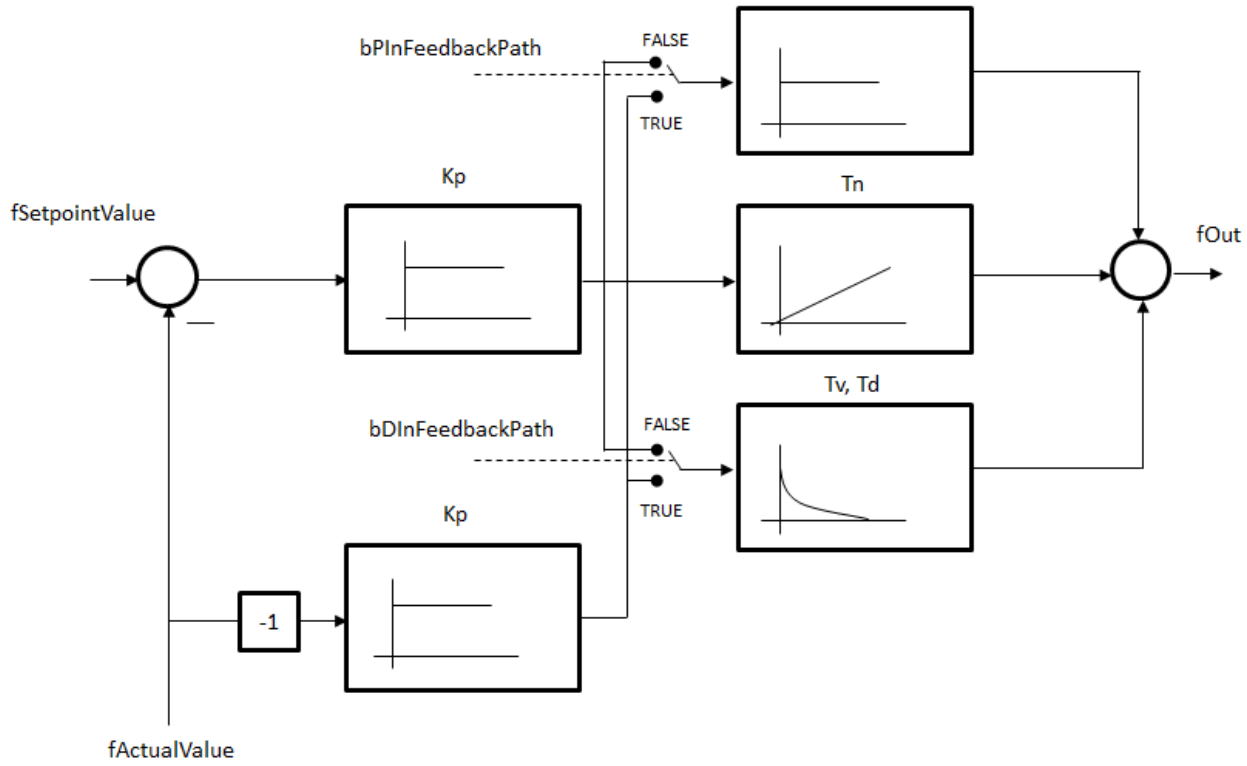

该功能块在功能图中提供了一个 PID 传递单元。

### 传递函数

如果布尔输入 `bPInTheFeedbackPath` 和 `bDInTheFeedbackPath` 为 FALSE，则可指定以下传递函数。否则，传递函数将仅描述功能块的部分传递行为。

$$G_{PID}(s) = K_p\left(1+\frac{1}{T_n s}+\frac{T_v s}{1+T_d s}\right)$$

### 功能图




通过两个有效的布尔输入 `bPInTheFeedbackPath` 和 `bDInTheFeedbackPath`（充当“开关”），对加法形式的 PID 控制器的标准功能图进行扩展，从而可以激活一个修改后的功能图。

控制背景：由于控制算法中的微分分量，在设定点阶跃变化时会产生较大的控制值，这会对控制单元造成负担，而且可能会导致控制系统振荡。微分分量仅应用于控制值（`bDInTheFeedbackPath := TRUE`）的控制算法可以避免此问题。

`bPInTheFeedbackPath` 和 `bDInTheFeedbackPath` 输入允许闭环控制回路实现以下传递函数：

| bPInTheFeedbackPath | bDInTheFeedbackPath | G(s) |
|---|---|---|
| false | false | $G(s)=\frac{G_{PID}(s)\cdot G_S(s)}{1+G_{PID}(s)\cdot G_S(s)}$ |
| true | false | $G(s)=\frac{G_{DI}(s)\cdot G_S(s)}{1+G_{PID}(s)\cdot G_S(s)}$ |
| false | true | $G(s)=\frac{G_{PI}(s)\cdot G_S(s)}{1+G_{PID}(s)\cdot G_S(s)}$ |
| true | true | $G(s)=\frac{G_I(s)\cdot G_S(s)}{1+G_{PID}(s)\cdot G_S(s)}$ |

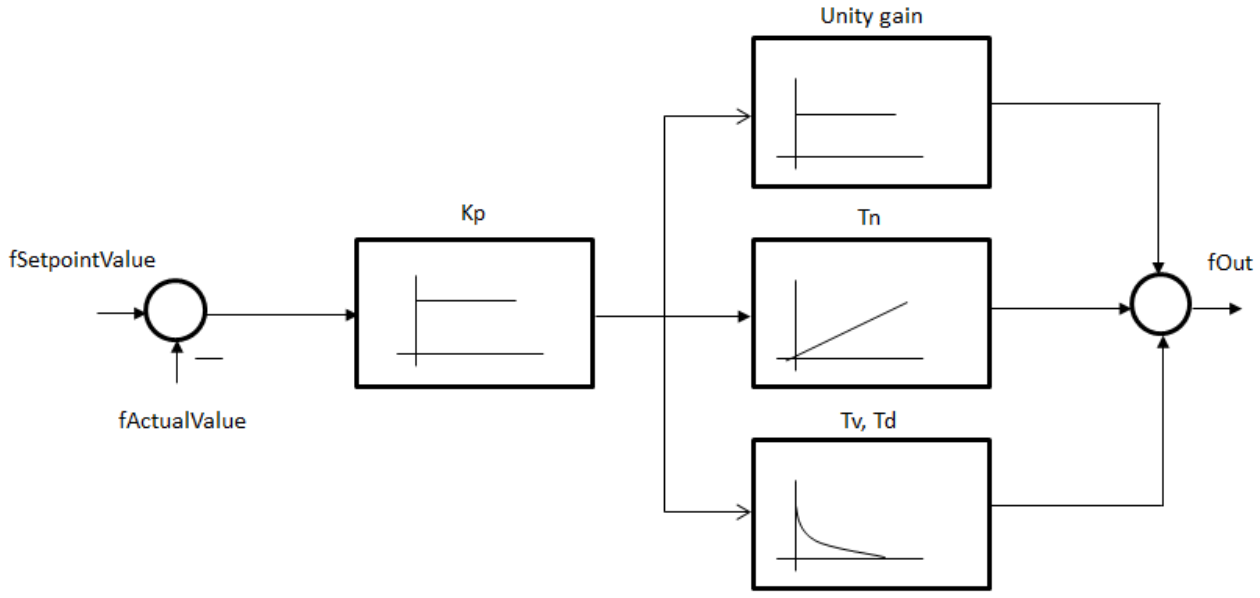
及：

$$G_{DI}(s)=K_p\left(\frac{1}{T_n s}+\frac{T_v s}{1+T_d s}\right)$$

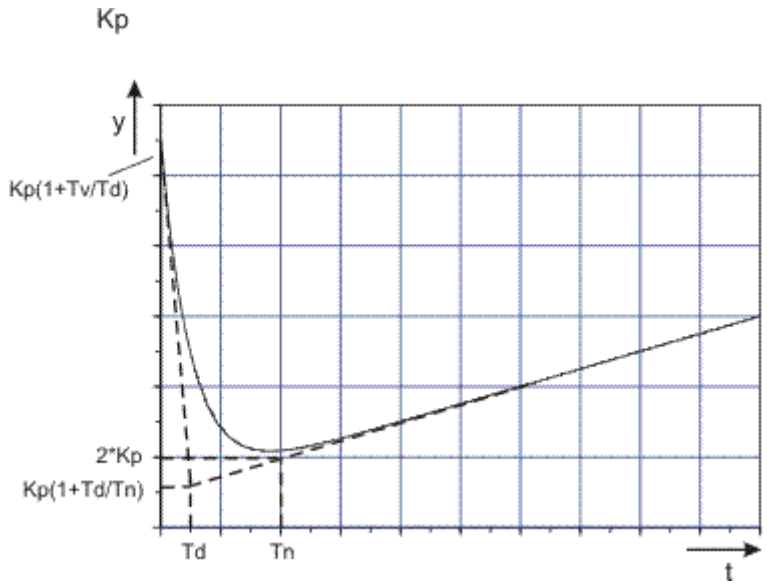
$$G_{PI}(s)=K_p\left(1+\frac{1}{T_n s}\right)$$

$$G_I(s)=K_p\left(\frac{1}{T_n s}\right)$$

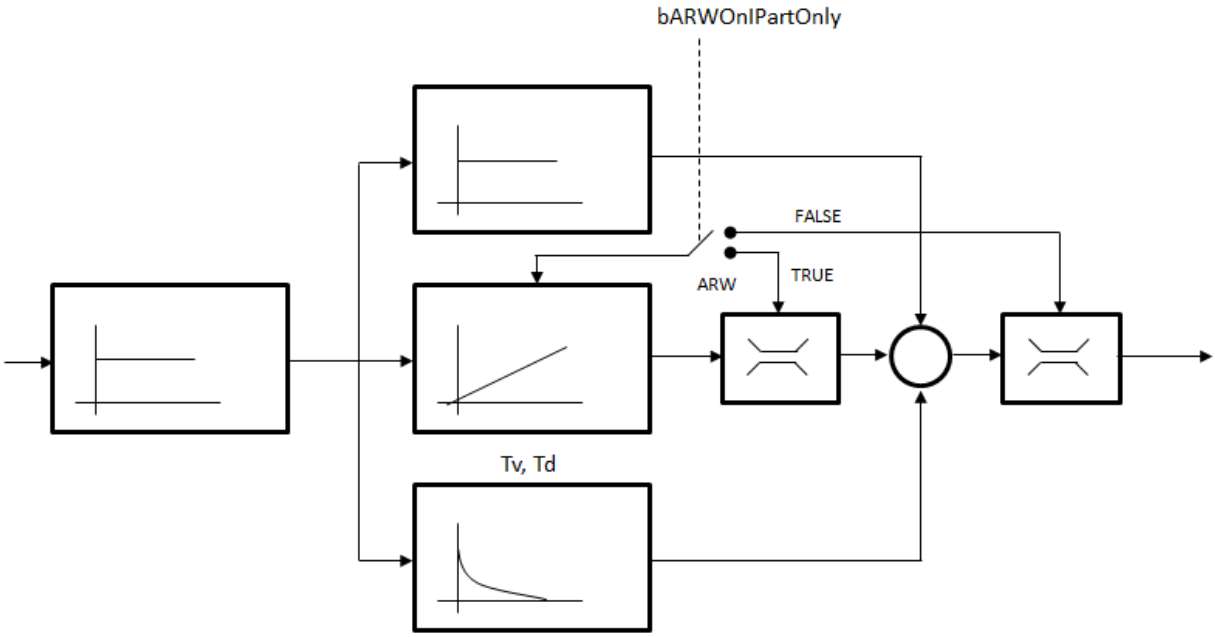
两个 `bPInTheFeedbackPath` 和 `bDInTheFeedbackPath` 输入的标准设置为 FALSE。此时，PID 控制器相当于一个标准加法形式的 PID 控制器。



### 阶跃响应




### ARW




### 输入

```
VAR_INPUT
    fSetpointValue    : FLOAT;
    fActualValue      : FLOAT;
    fManSyncValue     : FLOAT;
    bSync             : BOOL;
    eMode             : E_CTRL_MODE;
    bHold             : BOOL;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| fSetpointValue | FLOAT | 控制变量的设定点 |
| fActualValue | FLOAT | 控制变量的实际值 |
| fManSyncValue | FLOAT | 其值可用于设置 PID 单元的内部状态的输入。 |
| bSync | BOOL | 该输入端的上升沿将 PID 单元设置为值“fManSyncValue”。 |

| 名称 | 类型 | 描述 |
|---|---|---|
| eMode | E_CTRL_MODE | 指定功能块的运行模式 [▶ 157]的输入。 |
| bHold | BOOL | 该输入为 TRUE 时，内部状态（以及输出）将保持当前值不变，且不受控制偏差影响。 |

### 输出

```
VAR_OUTPUT
    fOut           : FLOAT;
    bARWactive     : BOOL;
    eState         : E_CTRL_STATE;
    eErrorId       : E_CTRL_ERRORCODES;
    bError         : BOOL;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| fOut | FLOAT | PID 单元的输出 |
| bARWactive | BOOL | 该输出为 TRUE 时，表示 PID 单元正处于限制状态。 |
| eState | E_CTRL_STATE | 功能块的状态 |
| eErrorId | E_CTRL_ERRORCODES | 在设置输出 bError 时，提供错误编号 [▶ 157]。 |
| bError | BOOL | 一旦发生错误，则变为 TRUE。 |

### 输入/输出

```
VAR_IN_OUT
    stParams        : ST_CTRL_PID_PARAMS;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| stParams | ST_CTRL_PID_PARAMS | PID 单元的参数结构 |

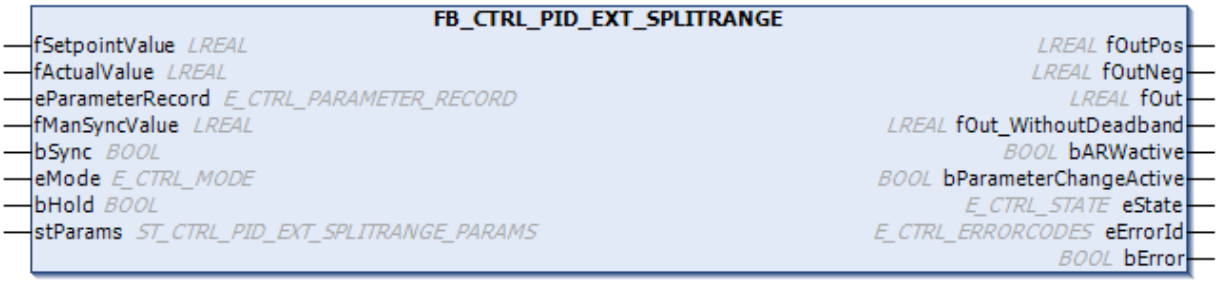
stParams 由以下元素组成：

```
TYPE
ST_CTRL_PID_PARAMS :
STRUCT
    tCtrlCycleTime          : TIME := T#0ms;
    tTaskCycleTime          : TIME := T#0ms;
    fKp                     : FLOAT := 0;
    tTn                     : TIME := T#0ms;
    tTv                     : TIME := T#0ms;
    tTd                     : TIME := T#0ms;
    fOutMaxLimit            : FLOAT := 1E38;
    fOutMinLimit            : FLOAT := -1E38;
    bPInTheFeedbackPath     : BOOL;
    bDInTheFeedbackPath     : BOOL;
    bARWOnIPartOnly         : BOOL;
END_STRUCT
END_TYPE
```

| 名称 | 类型 | 描述 |
|---|---|---|
| tCtrlCycleTime | TIME | 处理控制回路的循环时间。该时间必须大于或等于 **TaskCycleTime**。功能块使用该输入值在内部计算是否需要在当前周期内更新状态和输出值。 |
| tTaskCycleTime | TIME | 调用功能块的循环时间。如果在每个循环中都调用该功能块，这相当于调用任务的任务循环时间。 |
| fKp | FLOAT | 控制器放大/控制器系数 |
| tTn | TIME | 积分作用时间；如果将其参数设置为 T#0s，则 I 分量将被停用。 |
| tTv | TIME | 微分作用时间；如果将其参数设置为 T#0s，则 D 分量将被停用。 |
| tTd | TIME | 阻尼时间 |
| fOutMaxLimit | FLOAT | 积分停止和输出限制的上限（ARW 措施）。bARWActive 输出为 TRUE 时，表示达到该限值。 |

| 名称 | 类型 | 描述 |
|---|---|---|
| fOutMinLimit | FLOAT | 积分停止和输出限制的下限（ARW 措施）。bARWActive 输出为 TRUE 时，表示达到该限值。 |
| bPInTheFeedbackPath | BOOL | 通过该输入可以选择内部 P 单元的输入值（请参见功能图）。标准设置：FALSE |
| bDInTheFeedbackPath | BOOL | 通过该输入可以选择内部 D 单元的输入值（请参见功能图）。标准设置：FALSE |
| bARWOnIPartOnly | BOOL | 如果该参数为 FALSE（标准设置），则当控制器的完整输出达到上限或下限时，I 分量的积分将停止。如果该参数为 TRUE，则当 I 分量（积分器的输出）达到某个限值时，积分将停止。（请参见功能图。） |

## 4.2.1.3.10 FB_CTRL_PID_EXT_SPLITRANGE




该功能块在功能图中提供了一个扩展型 PID 传递单元。在调节过程中，通过该控制器可以在两个不同的参数集之间切换。此外，还提供内外窗口以及输入和输出死区的功能。

### 描述

该功能块是 FB_CTRL_PID_EXT 的扩展，这意味着控制器可用于控制配备两个传递行为不同的受控设备的系统。系统配备一个用于加热的执行器和另一个用于冷却的执行器，这就是典型的应用。为了优化此类布局的调节性能，可以在两个 PID 参数集之间进行切换。参数集转换的方式是，即使在转换期间也能保持连续的控制值。

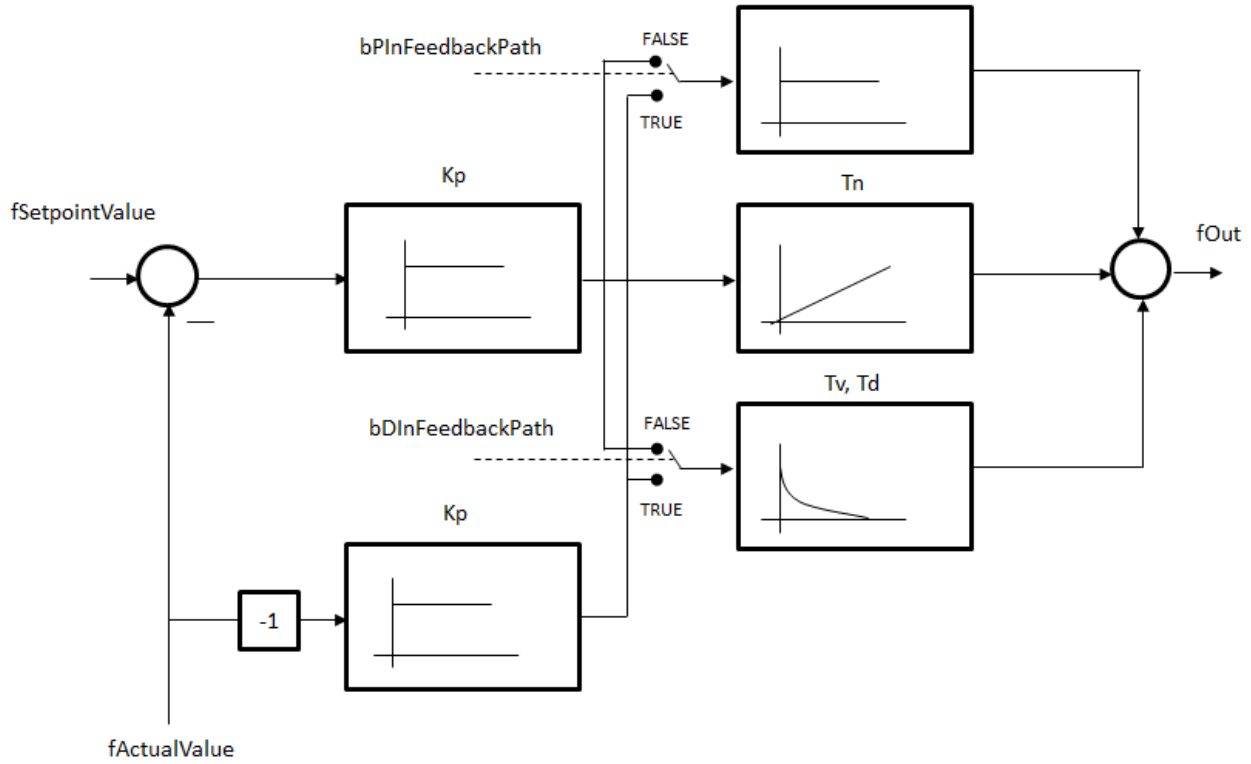
切换算法会计算两个参数集之间依时间线性变化的过渡。nParameterChangeCycleTicks 参数可用于指定两个参数集之间发生连续切换的任务周期数。

### 传递函数

如果布尔输入 bPInTheFeedbackPath 和 bDInTheFeedbackPath 为 FALSE，则可指定以下传递函数。否则，传递函数将仅描述功能块的部分传递行为。

$$G_{PID}(s) = K_p \left( 1 + \frac{1}{T_n s} + \frac{T_v s}{1 + T_d s} \right)$$

**功能图**




通过两个有效的布尔输入 `bPInTheFeedbackPath` 和 `bDInTheFeedbackPath`（充当“开关”），对加法形式的 PID 控制器的标准功能图进行扩展，从而可以激活一个修改后的功能图。

控制背景：由于控制算法中的微分分量，在设定点阶跃变化时会产生较大的控制值，这会对控制单元造成负担，而且可能会导致控制系统振荡。微分分量仅应用于控制变量（`bDInTheFeedbackPath := TRUE`）的控制算法可以避免此问题。

`bPInTheFeedbackPath` 和 `bDInTheFeedbackPath` 输入允许闭环控制回路实现以下传递函数：

| bPInTheFeedbackPath | bDInTheFeedbackPath | G(s) |
|---|---|---|
| false | false | $G(s) = \frac{G_{PID}(s) \cdot G_S(s)}{1 + G_{PID}(s) \cdot G_S(s)}$ |
| true | false | $G(s) = \frac{G_{DI}(s) \cdot G_S(s)}{1 + G_{PID}(s) \cdot G_S(s)}$ |
| false | true | $G(s) = \frac{G_{PI}(s) \cdot G_S(s)}{1 + G_{PID}(s) \cdot G_S(s)}$ |
| true | true | $G(s) = \frac{G_I \cdot G_S(s)}{1 + G_{PID}(s) \cdot G_S(s)}$ |

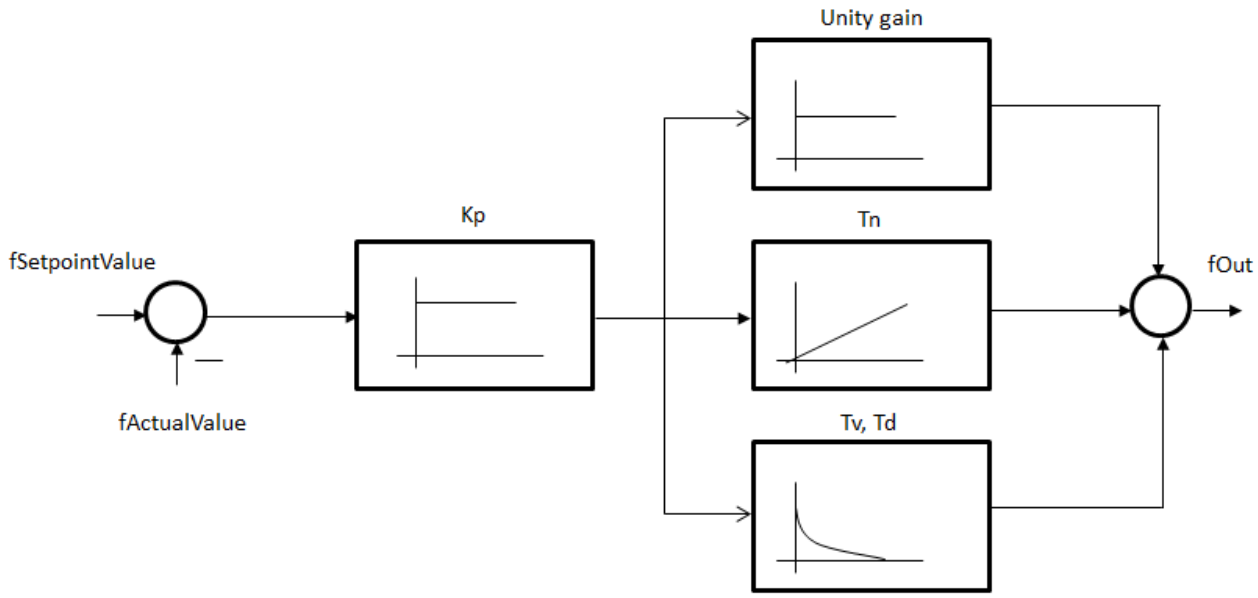
及：

$$G_{DI}(s) = K_p \left( \frac{1}{T_n s} + \frac{T_v s}{1 + T_d s} \right)$$

$$G_{PI}(s) = K_p\left(1+\frac{1}{T_n s}\right)$$

$$G_I(s) = K_p\left(\frac{1}{T_n s}\right)$$

两个 `bPInTheFeedbackPath` 和 `bDInTheFeedbackPath` 输入的标准设置为 FALSE。此时，PID 控制器充当一个标准加法形式的 PID 控制器。




**附加功能**

**关闭外窗口中的 I 分量**

如果控制偏差大于 `fOuterWindow` 参数，则控制偏差的积分将停止。这样可以防止在控制偏差较大时产生极大的 I 分量，因为这可能会导致明显的超调。如果不需要，通过设置 `fOuterWindow:= 0` 可以禁用该功能。

**内窗口中的 I 分量的线性减少**

通过该功能可以在 `fInnerWindow` 参数指定的范围内将 I 分量线性降至零。如果不需要，通过设置 `fInnerWindow := 0` 可以禁用该功能。

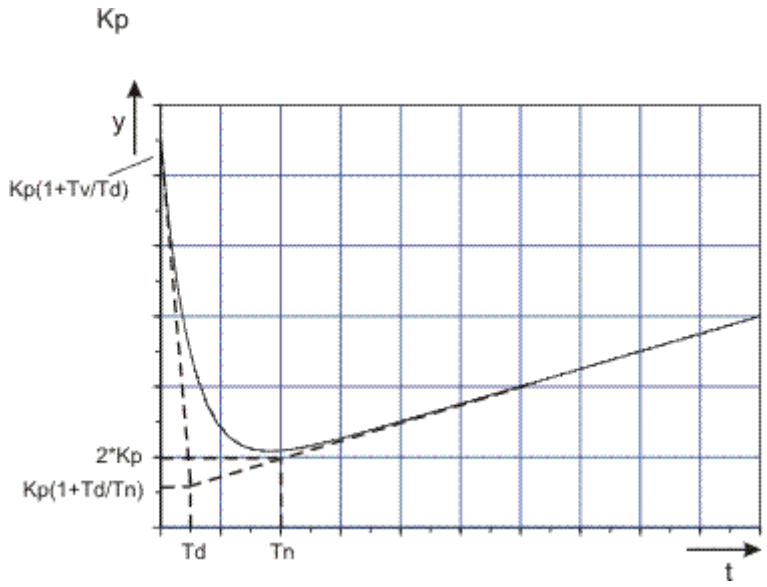
**输出死区**

如果设置了参数 `fDeadBandOutput > 0`，当输出在 `[ -fDeadBandOutput ... fDeadBandOutput ]` 的范围内时，它将被设置为零。

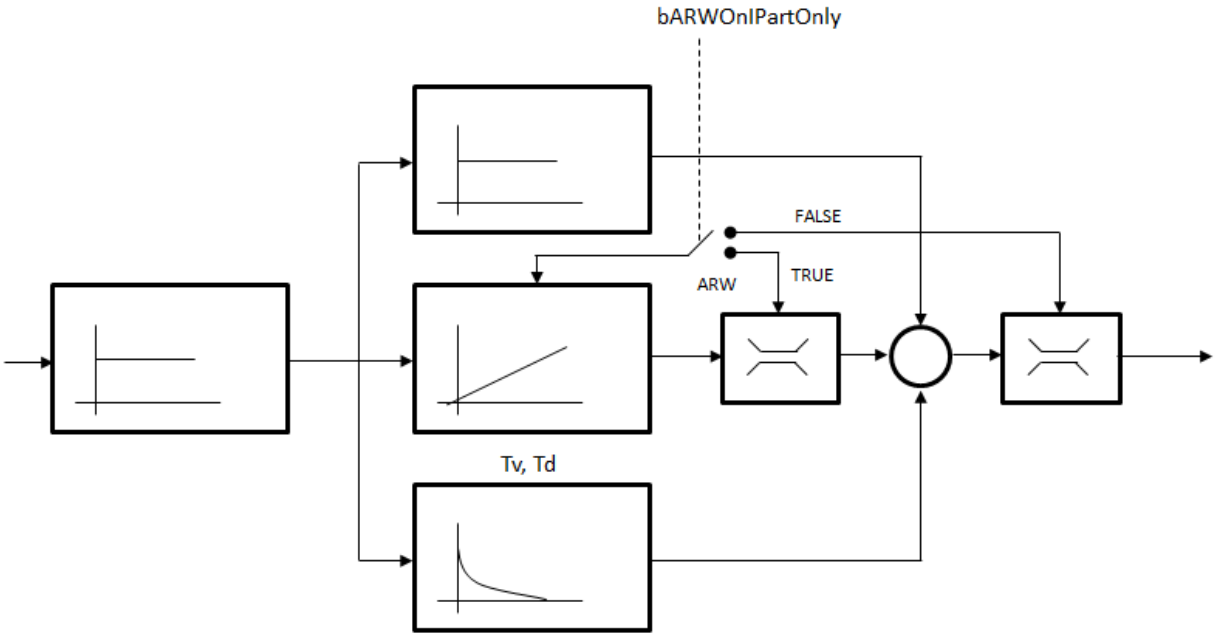
**输入死区**

如果设置了参数 `fDeadBandInput > 0`，只要控制偏差保持在 `[ -fDeadBandInput ... fDeadBandInput ]` 的范围内，输出将保持不变。

### 阶跃响应




### ARW




### 输入

```
VAR_INPUT
    fSetpointValue      : FLOAT;
    fActualValue        : FLOAT;
    eParameterRecord    : E_CTRL_PARAMETER_RECORD;
    fManSyncValue       : FLOAT;
    bSync               : BOOL;
    eMode               : E_CTRL_MODE;
    bHold               : BOOL;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| fSetpointValue | FLOAT | 控制变量的设定点 |
| fActualValue | FLOAT | 控制变量的实际值 |
| eParameterRecord | E_CTRL_PARAMETER_RECORD | 活动参数集的索引 |

| 名称 | 类型 | 描述 |
|---|---|---|
| fManSyncValue | FLOAT | 用于设置 PI 单元的输入。 |
| bSync | BOOL | 该输入端的上升沿将 PI 单元设置为值“fManSyncValue”。 |
| eMode | E_CTRL_MODE | 指定功能块的运行模式 [▶ 157]的输入。 |
| bHold | BOOL | 该输入为 TRUE 时，内部状态（以及输出）将保持当前值不变，且不受控制偏差影响。 |

### 输出

```
VAR_OUTPUT
    fOutPos                 : FLOAT;
    fOutNeg                 : FLOAT;
    fOut                    : FLOAT;
    bARWActive              : BOOL := FALSE;
    bParameterChangeActive  : BOOL;
    bError                  : BOOL;
    eErrorId                : E_CTRL_ERRORCODES;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| fOutPos | FLOAT | 当控制值为正时，PID 单元的输出。否则，输出为零。 |
| fOutNeg | FLOAT | 当控制值为负时，PID 单元的输出。否则，输出为零。 |
| fOut | FLOAT | PID 单元的输出 |
| bARWActive | BOOL | 该输出为 TRUE 时，表示 PID 单元正处于限制状态。 |
| bParameter ChangeActive | BOOL | 该输出为 TRUE 时，表示正在从一个参数集切换到另一个参数集。 |
| bError | BOOL | 一旦发生错误，则变为 TRUE。 |
| eErrorId | E_CTRL_ERRORCODES | 在设置输出 bError 时，提供错误编号 [▶ 157]。 |
| fCtrlDerivation | | 该输出为 TRUE 时，表示 PID 单元正处于限制状态。 |

### 输入/输出

```
VAR_IN_OUT
    stParams    : ST_CTRL_PID_EXT_SPLITRANGE_PARAMS;
END_VAR
```

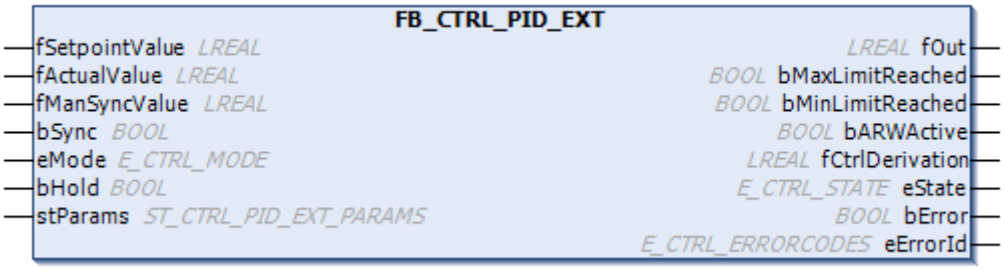
| 名称 | 类型 | 描述 |
|---|---|---|
| stParams | ST_CTRL_PID_EXT_SPLITRANGE_PARAMS | PID 单元的参数结构 |

stParams 由以下元素组成：

```
TYPE
ST_CTRL_PID_EXT_SPLITRANGE_PARAMS :
STRUCT
    tCtrlCycleTime              : TIME := T#0ms;
    tTaskCycleTime              : TIME := T#0ms;
    fKp_heating                 : FLOAT := 0;
    tTn_heating                 : TIME := T#0ms;
    tTv_heating                 : TIME := T#0ms;
    tTd_heating                 : TIME := T#0ms;
    fKp_cooling                 : FLOAT := 0;
    tTn_cooling                 : TIME := T#0ms;
    tTv_cooling                 : TIME := T#0ms;
    tTd_cooling                 : TIME := T#0ms;
    nParameterChangeCycleTicks  : INT;
    fDeadBandInput              : REAL := 0.0;
    fDeadBandOutput             : REAL := 0.0;
    fInnerWindow                : REAL := 0.0;
    fOuterWindow                : REAL := 0.0;
    fOutMaxLimit                : FLOAT := 1E38;
    fOutMinLimit                : FLOAT := -1E38;
    bPInTheFeedbackPath         : BOOL;
    bDInTheFeedbackPath         : BOOL;
    bARWOnIPartOnly             : BOOL;
END_STRUCT
END_TYPE
```

| 名称 | 类型 | 描述 |
|---|---|---|
| tCtrlCycleTime | TIME | 处理控制回路的循环时间。该时间必须大于或等于 **TaskCycleTime**。功能块使用该输入值在内部计算是否需要在当前周期内更新状态和输出值。 |
| tTaskCycleTime | TIME | 调用功能块的循环时间。如果在每个循环中都调用该功能块，这相当于调用任务的任务循环时间。 |
| 范围 **eCTRL_PARAMETER_RECORD_HEATING**： | | |
| fKp_heating | FLOAT | 控制器放大/控制器系数 |
| tTn_heating | TIME | 积分作用时间：如果将其参数设置为 T#0s，则 I 分量将被停用。 |
| tTv_heating | TIME | 微分作用时间：如果将其参数设置为 T#0s，则 D 分量将被停用。 |
| tTd_heating | TIME | 阻尼时间 |
| 范围 **eCTRL_PARAMETER_RECORD_COOLING**： | | |
| fKp_cooling | FLOAT | 控制器放大/控制器系数 |
| tTn_cooling | TIME | 积分作用时间：如果将其参数设置为 T#0s，则 I 分量将被停用。 |
| tTv_cooling | TIME | 微分作用时间：如果将其参数设置为 T#0s，则 D 分量将被停用。 |
| tTd_cooling | TIME | 阻尼时间 |
| nParameterChangeCycleTicks | INT | 从一个参数集切换到另一个参数集的任务周期数。 |
| fDeadBandInput | REAL | 请参见上文描述：输入死区 |
| fDeadBandOutput | REAL | 请参见上文描述：输出死区 |
| fInnerWindow | REAL | 请参见上文描述：内窗口中的 I 分量的线性衰减 |
| fOuterWindow | REAL | 请参见上文描述：关闭外窗口中的 I 分量 |
| fOutMaxLimit | FLOAT | 积分停止和输出限制的上限（ARW 措施）。`bARWActive` 输出为 TRUE 时，表示达到该限值。 |
| fOutMinLimit | FLOAT | 积分停止和输出限制的下限（ARW 措施）。`bARWActive` 输出为 TRUE 时，表示达到该限值。 |
| bPInTheFeedbackPath | BOOL | 通过该输入可以选择 P 单元的输入值（请参见功能图）。 |
| bDInTheFeedbackPath | BOOL | 通过该输入可以选择 D 单元的输入值（请参见功能图）。 |
| bARWOnIPartOnly | BOOL | 如果该参数为 FALSE（标准设置），则当控制器的完整输出达到上限或下限时，I 分量的积分将停止。<br>如果该参数为 TRUE，则当 I 分量（积分器的输出）达到某个限值时，积分将停止。（请参见功能图。） |

### 4.2.1.3.11 FB_CTRL_PID_EXT

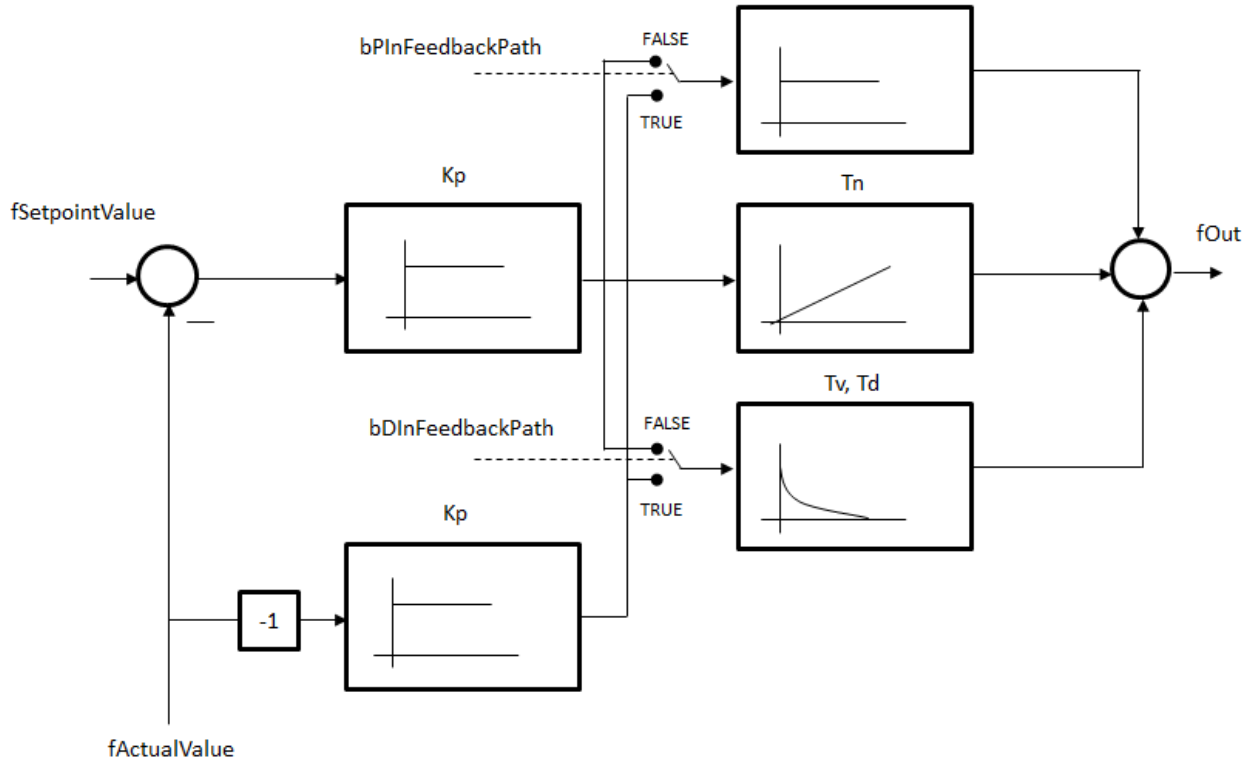


### 传递函数

如果布尔输入 bPInTheFeedbackPath 和 bDInTheFeedbackPath 为 FALSE，则可指定以下传递函数。否则，传递函数将仅描述功能块的部分传递行为。

$$G_{PID}(s) = K_p\left(1+\frac{1}{T_n s}+\frac{T_v s}{1+T_d s}\right)$$

### 功能图




通过两个有效的布尔输入 bPInTheFeedbackPath 和 bDInTheFeedbackPath（充当“开关”），对加法形式的 PID 控制器的标准功能图进行扩展，从而可以激活一个修改后的功能图。

控制背景：由于控制算法中的微分分量，在设定点阶跃变化时会产生较大的控制值，这会对控制单元造成负担，而且可能会导致控制系统振荡。微分分量仅应用于控制变量（bDInTheFeedbackPath := TRUE）的控制算法可以避免此问题。

bPInTheFeedbackPath 和 bDInTheFeedbackPath 输入允许闭环控制回路实现以下传递函数：

| bPInTheFeedbackPath | bDInTheFeedbackPath | G(s) |
|---|---|---|
| false | false | $G(s)=\frac{G_{PID}(s)\cdot G_S(s)}{1+G_{PID}(s)\cdot G_S(s)}$ |
| true | false | $G(s)=\frac{G_{DI}(s)\cdot G_S(s)}{1+G_{PID}(s)\cdot G_S(s)}$ |
| false | true | $G(s)=\frac{G_{PI}(s)\cdot G_S(s)}{1+G_{PID}(s)\cdot G_S(s)}$ |

| bPInTheFeedbackPath | bDInTheFeedbackPath | G(s) |
|---|---|---|
| true | true | $G(s) = \frac{G_I(s) \square G_S(s)}{1 + G_{PID}(s) \square G_S(s)}$ |

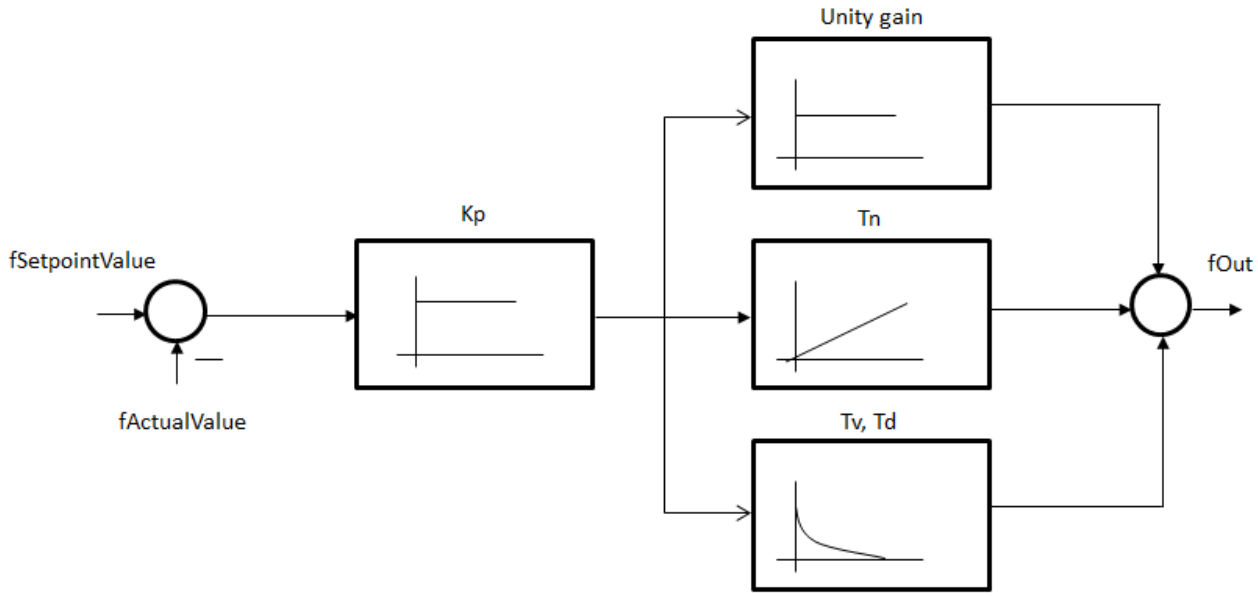
及：

$$G_{DI}(s) = K_p \left( \frac{1}{T_n s} + \frac{T_v s}{1 + T_d s} \right)$$

$$G_{PI}(s) = K_p \left( 1 + \frac{1}{T_n s} \right)$$

$$G_I(s) = K_p \left( \frac{1}{T_n s} \right)$$

两个 `bPInTheFeedbackPath` 和 `bDInTheFeedbackPath` 输入的标准设置为 **FALSE**。此时，PID 控制器充当一个标准加法形式的 PID 控制器。




### 关闭外窗口中的 I 分量

如果控制偏差大于 `fOuterWindow` 参数，则控制偏差的积分将停止。这样可以防止在控制偏差较大时产生极大的 I 分量，因为这可能会导致明显的超调。如果不需要，通过设置 `fOuterWindow := 0` 可以禁用该功能。

### 内窗口中的 I 分量的线性减少

通过该功能可以在 `fInnerWindow` 参数指定的范围内将 I 分量线性降至零。如果不需要，通过设置 `fInnerWindow := 0` 可以禁用该功能。

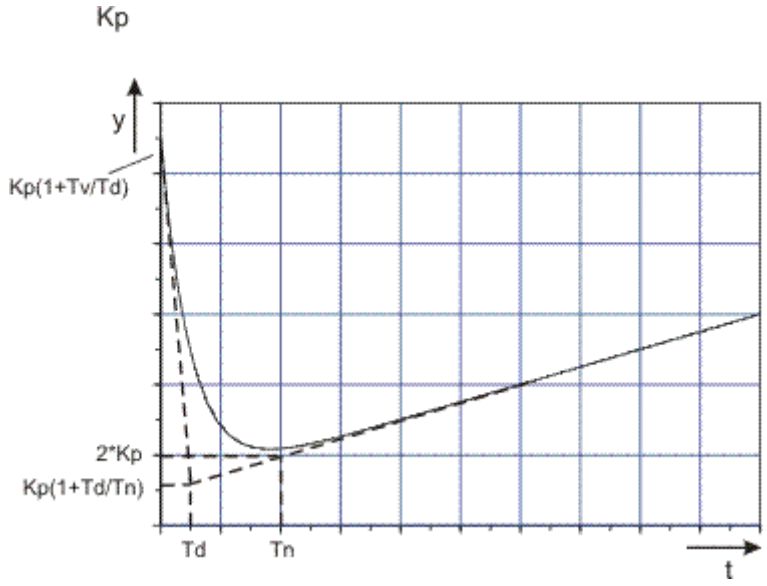
### 输出死区

如果设置了参数 `fDeadBandOutput > 0`，当输出在 `[ -fDeadBandOutput ... fDeadBandOutput ]` 的范围内时，它将被设置为零。

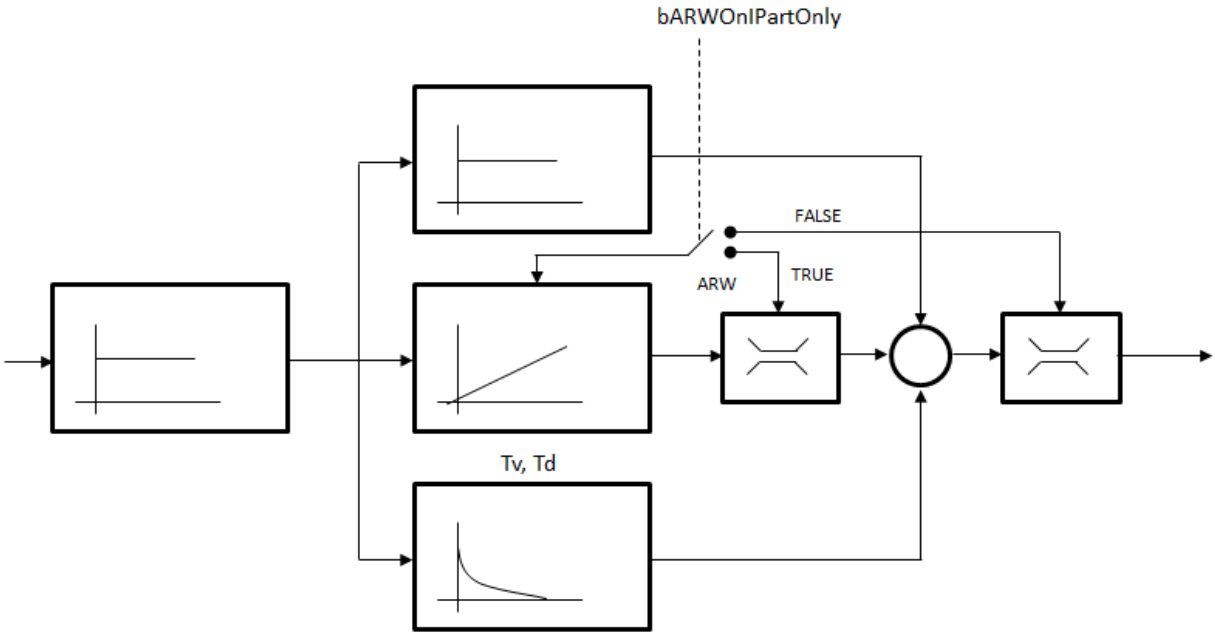
### 输入死区

如果设置了参数 `fDeadBandInput > 0`，只要控制偏差保持在 `[ -fDeadBandInput ... fDeadBandInput ]` 的范围内，输出将保持不变。

### 阶跃响应




### ARW




### 输入

```
VAR_INPUT
    fSetpointValue     : FLOAT;
    fActualValue       : FLOAT;
    fManSyncValue      : FLOAT;
    bSync              : BOOL;
    eMode              : E_CTRL_MODE;
    bHold              : BOOL;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| fSetpointValue | FLOAT | 控制变量的设定点 |
| fActualValue | FLOAT | 控制变量的实际值 |
| fManSyncValue | FLOAT | 用于设置 PID 单元的输入。 |
| bSync | BOOL | 该输入端的上升沿将 PID 单元设置为值“fManSyncValue”。 |
| eMode | E_CTRL_MODE | 指定功能块的运行模式 [▶ 157]的输入。 |
| bHold | BOOL | 该输入为 TRUE 时，内部状态（以及输出）将保持当前值不变，且不受控制偏差影响。 |

### 输出

```
VAR_OUTPUT
    fOut                : FLOAT;
    bMaxLimitReached    : BOOL := FALSE;
    bMinLimitReached    : BOOL := FALSE;
    bARWActive          : BOOL := FALSE;
    fCtrlDerivation     : FLOAT;
    eState              : E_CTRL_STATE;
    bError              : BOOL;
    eErrorId            : E_CTRL_ERRORCODES;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| fOut | FLOAT | PID 单元的输出 |
| bMaxLimit Reached | BOOL | 当功能块处于其上限时，输出为 TRUE。 |
| bMinLimit Reached | BOOL | 当功能块处于其下限时，输出为 TRUE。 |
| bARWActive | BOOL | 该输出为 TRUE 时，表示 PID 单元正处于限制状态。 |
| fCtrlDerivation | FLOAT | 控制偏差的当前值 |
| eState | E_CTRL_STATE | 功能块的状态 |
| bError | BOOL | 在设置输出 bError 时，提供错误编号 [▶ 157]。 |
| eErrorId | E_CTRL_ ERRORCODES | 一旦发生错误，则变为 TRUE。 |

### 输入/输出

```
VAR_IN_OUT
    stParams    : ST_CTRL_PID_EXT_PARAMS;
END_VAR
```

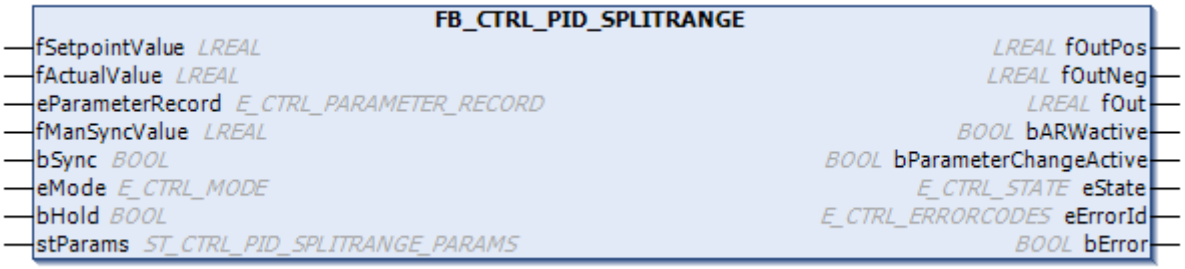
| 名称 | 类型 | 描述 |
|---|---|---|
| stParams | ST_CTRL_PID_ EXT_PARAMS | PID 单元的参数结构 |

stParams 由以下元素组成：

```
TYPE
ST_CTRL_PID_EXT_PARAMS :
STRUCT
    tCtrlCycleTime          : TIME := T#0ms;
    tTaskCycleTime          : TIME := T#0ms;
    fKp                     : FLOAT := 0;
    tTn                     : TIME := T#0ms;
    tTv                     : TIME := T#0ms;
    tTd                     : TIME := T#0ms;
    fDeadBandInput          : REAL := 0.0;
    fDeadBandOutput         : REAL := 0.0;
    fInnerWindow            : REAL := 0.0;
    fOuterWindow            : REAL := 0.0;
    fOutMaxLimit            : FLOAT := 1E38;
    fOutMinLimit            : FLOAT := -1E38;
    bPInTheFeedbackPath     : BOOL;
    bDInTheFeedbackPath     : BOOL;
    bARWOnIPartOnly         : BOOL;
END_STRUCT
END_TYPE
```

| 名称 | 类型 | 描述 |
|---|---|---|
| tCtrlCycleTime | TIME | 处理控制回路的循环时间。该时间必须大于或等于 **TaskCycleTime**。功能块使用该输入值在内部计算是否需要在当前周期内更新状态和输出值。 |
| tTaskCycleTime | TIME | 调用功能块的循环时间。如果在每个循环中都调用该功能块，这相当于调用任务的任务循环时间。 |
| fKp | FLOAT | 控制器放大/控制器系数 |
| tTn | TIME | 积分作用时间；如果将其参数设置为 T#0s，则 I 分量将被停用。 |
| tTv | TIME | 微分作用时间；如果将其参数设置为 T#0s，则 D 分量将被停用。 |
| tTd | TIME | 阻尼时间 |
| fDeadBandInput | REAL | 请参见上文描述。 |
| fDeadBand Output | REAL | 请参见上文描述。 |
| fInnerWindow | REAL | 请参见上文描述。 |
| fOuterWindow | REAL | 请参见上文描述。 |
| fOutMaxLimit | FLOAT | 积分停止和输出限制的上限（ARW 措施）。`bARWActive` 输出为 TRUE 时，表示达到该限值。 |
| fOutMinLimit | FLOAT | 积分停止和输出限制的下限（ARW 措施）。`bARWActive` 输出为 TRUE 时，表示达到该限值。 |
| bPInTheFeed backPath | BOOL | 通过该输入可以选择 P 单元的输入值（请参见功能图）。 |
| bDInTheFeed backPath | BOOL | 通过该输入可以选择 D 单元的输入值（请参见功能图）。 |
| bARWOnIPart Only | BOOL | 如果该参数为 FALSE（标准设置），则当控制器的完整输出达到上限或下限时，I 分量的积分将停止。<br>如果该参数为 TRUE，则当 I 分量（积分器的输出）达到某个限值时，积分将停止。（请参见功能图。） |

## 4.2.1.3.12 FB_CTRL_PID_SPLITRANGE




该功能块在功能图中提供了一个扩展型 PID 传递单元。在控制过程中，该控制器可实现两个参数集之间的切换。

### 描述

该功能块是 `FB_CTRL_PID` 的扩展，这意味着控制器可用于控制配备两个传递行为不同的受控设备的系统。系统配备一个用于加热的执行器和另一个用于冷却的执行器，这就是典型的应用。为了优化此类布局的调节性能，可以在两个 PID 参数集之间进行切换。参数集切换的方式是，即使在更改参数集时也能保持连续的控制值。

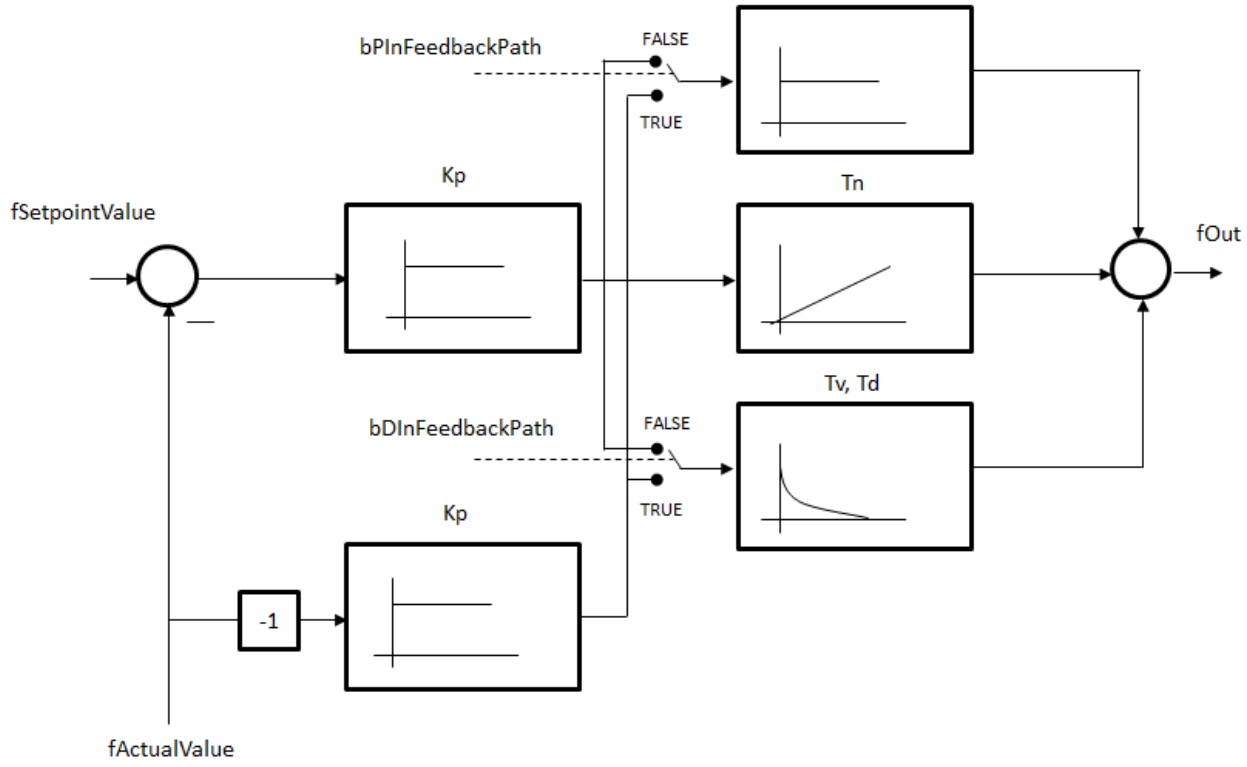
切换算法会计算两个参数集之间依时间线性变化的过渡。`nParameterChangeCycleTicks` 参数可用于指定两个参数集之间发生连续切换的任务周期数。

### 传递函数

如果布尔输入 bPInTheFeedbackPath 和 bDInTheFeedbackPath 为 FALSE，则可指定以下传递函数。否则，传递函数将仅描述功能块的部分传递行为。

$$G_{PID}(s) = K_p\left(1+\frac{1}{T_n s}+\frac{T_v s}{1+T_d s}\right)$$

### 功能图




通过两个有效的布尔输入 bPInTheFeedbackPath 和 bDInTheFeedbackPath（充当“开关”），对加法形式的 PID 控制器的标准功能图进行扩展，从而可以激活一个修改后的功能图。

控制背景：由于控制算法中的微分分量，在设定点阶跃变化时会产生较大的控制值，这会对控制单元造成负担，而且可能会导致控制系统振荡。微分分量仅应用于控制变量（bDInTheFeedbackPath := TRUE）的控制算法可以避免此问题。

bPInTheFeedbackPath 和 bDInTheFeedbackPath 输入允许闭环控制回路实现以下传递函数：

| bPInTheFeedbackPath | bDInTheFeedbackPath | G(s) |
|---|---|---|
| false | false | $G(s)=\frac{G_{PID}(s)\cdot G_S(s)}{1+G_{PID}(s)\cdot G_S(s)}$ |
| true | false | $G(s)=\frac{G_{DI}(s)\cdot G_S(s)}{1+G_{PID}(s)\cdot G_S(s)}$ |
| false | true | $G(s)=\frac{G_{PI}(s)\cdot G_S(s)}{1+G_{PID}(s)\cdot G_S(s)}$ |

| bPInTheFeedbackPath | bDInTheFeedbackPath | G(s) |
|---|---|---|
| true | true | $G(s) = \frac{G_I(s) \cdot G_S(s)}{1 + G_{PID}(s) \cdot G_S(s)}$ |

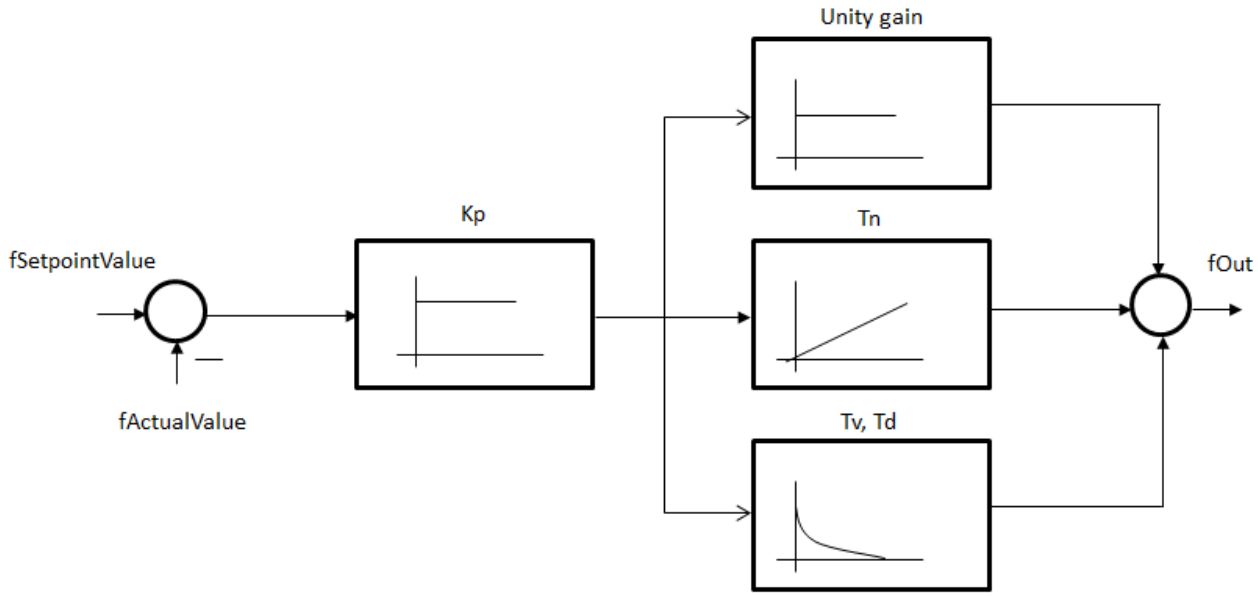
及：

$$G_{DI}(s) = K_p \left( \frac{1}{T_n s} + \frac{T_v s}{1 + T_d s} \right)$$

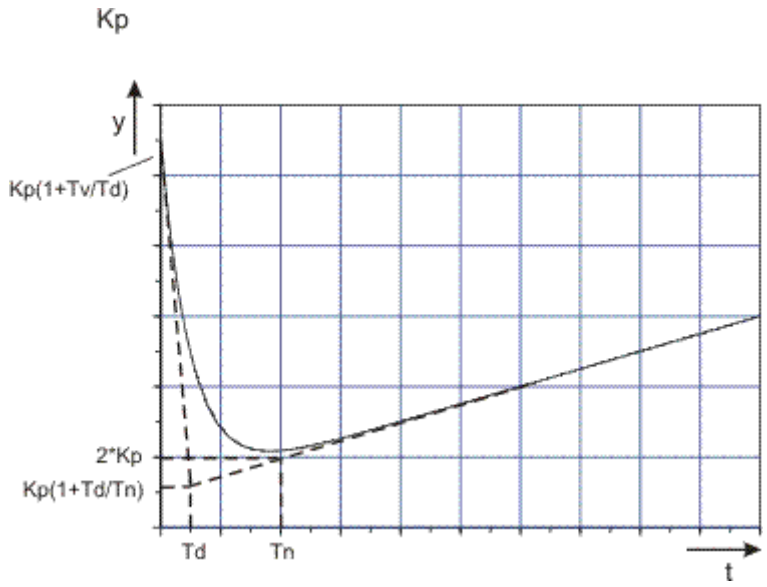
$$G_{PI}(s) = K_p \left( 1 + \frac{1}{T_n s} \right)$$

$$G_I(s) = K_p \left( \frac{1}{T_n s} \right)$$

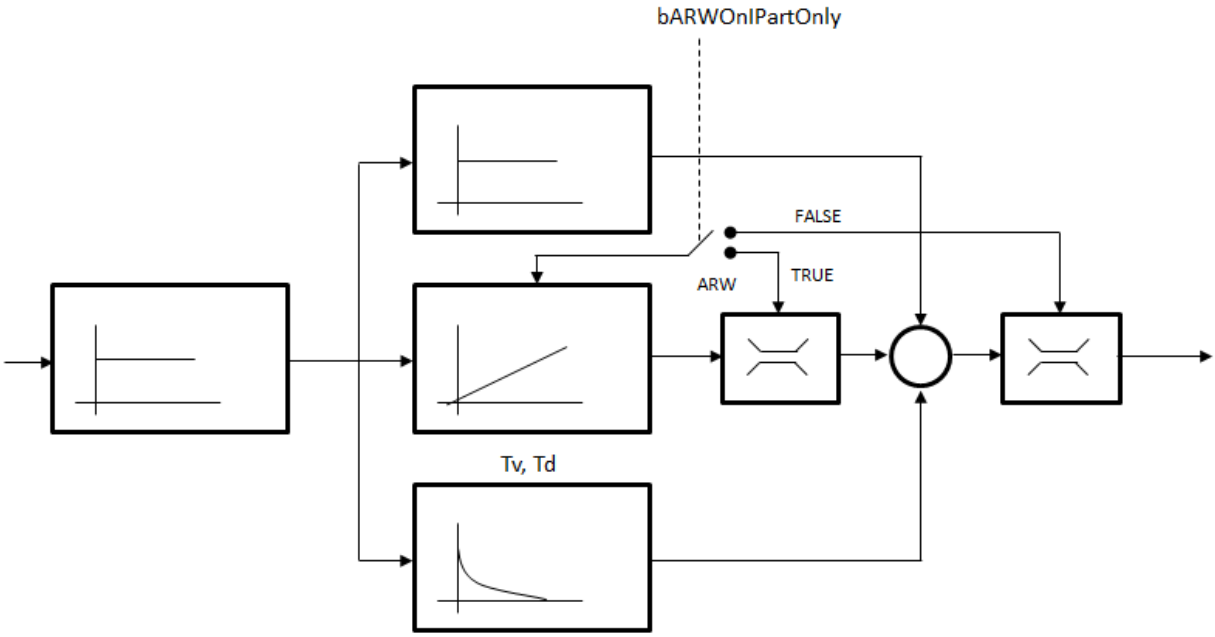
两个 `bPInTheFeedbackPath` 和 `bDInTheFeedbackPath` 输入的标准设置为 **FALSE**。此时，PID 控制器相当于一个标准加法形式的 PID 控制器。



### 阶跃响应




### ARW




### 输入

```
VAR_INPUT
    fSetpointValue      : FLOAT;
    fActualValue        : FLOAT;
    eParameterRecord    : E_CTRL_PARAMETER_RECORD;
    fManSyncValue       : FLOAT;
    bSync               : BOOL;
    eMode               : E_CTRL_MODE;
    bHold               : BOOL;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| fSetpointValue | FLOAT | 控制变量的设定点 |
| fActualValue | FLOAT | 控制变量的实际值 |

| 名称 | 类型 | 描述 |
|---|---|---|
| eParameter Record | E_CTRL_ PARAMETER_ RECORD | 活动参数集的索引 |
| fManSyncValue | FLOAT | 用于设置 PI 单元的输入。 |
| bSync | BOOL | 该输入端的上升沿将 PI 单元设置为值“fManSyncValue”。 |
| eMode | E_CTRL_MODE | 指定功能块的运行模式 [▶ 157]的输入。 |
| bHold | BOOL | 该输入为 TRUE 时，内部状态（以及输出）将保持当前值不变，且不受控制偏差影响。 |

### 输出

```
VAR_OUTPUT
    fOutPos                     : FLOAT;
    fOutNeg                     : FLOAT;
    fOut                        : FLOAT;
    bARWActive                  : BOOL := FALSE;
    bParameterChangeActive      : BOOL;
    eState                      : E_CTRL_STATE;
    bError                      : BOOL;
    eErrorId                    : E_CTRL_ERRORCODES;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| fOutPos | FLOAT | 当控制值为正时，PID 单元的输出。否则，输出为零。 |
| fOutNeg | FLOAT | 当控制值为负时，PID 单元的输出。否则，输出为零。 |
| fOut | FLOAT | PID 单元的输出 |
| bARWActive | BOOL | 该输出为 TRUE 时，表示 PID 单元正处于限制状态。 |
| bParameter ChangeActive | BOOL | 该输出为 TRUE 时，表示正在从一个参数集切换到另一个参数集。 |
| eState | E_CTRL_STATE | 功能块的状态 |
| bError | BOOL | 一旦发生错误，则变为 TRUE。 |
| eErrorId | E_CTRL_ERRORC ODES | 在设置输出 bError 时，提供错误编号 [▶ 157]。 |

### 输入/输出

```
VAR_IN_OUT
    stParams        : ST_CTRL_PID_SPLITRANGE_PARAMS;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| stParams | ST_CTRL_PID_ SPLITRANGE_ PARAMS | PID 单元的参数结构 |

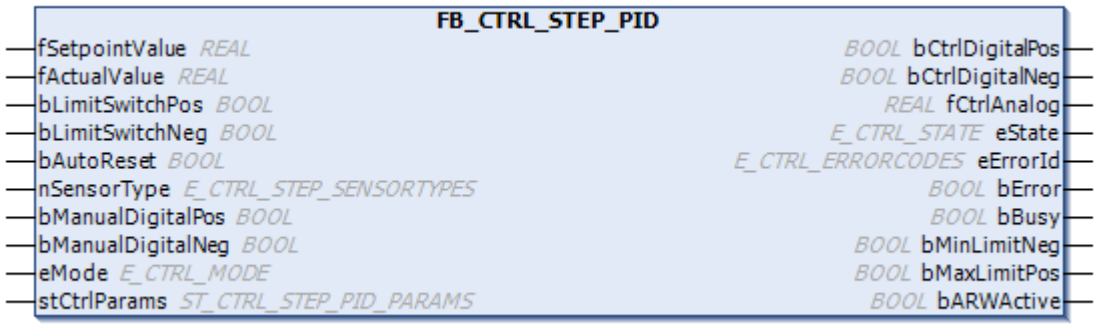
stParams 由以下元素组成：

```
TYPE
ST_CTRL_PID_SPLITRANGE_PARAMS :
STRUCT
    tCtrlCycleTime                : TIME := T#0ms;
    tTaskCycleTime                : TIME := T#0ms;
    fKp_heating                   : FLOAT := 0;
    tTn_heating                   : TIME := T#0ms;
    tTv_heating                   : TIME := T#0ms;
    tTd_heating                   : TIME := T#0ms;
    fKp_cooling                   : FLOAT := 0;
    tTn_cooling                   : TIME := T#0ms;
    tTv_cooling                   : TIME := T#0ms;
    tTd_cooling                   : TIME := T#0ms;
    nParameterChangeCycleTicks    : INT;
    fOutMaxLimit                  : FLOAT := 1E38;
    fOutMinLimit                  : FLOAT := -1E38;
    bPInTheFeedbackPath           : BOOL;
    bDInTheFeedbackPath           : BOOL;
```

```
    bARWOnIPartOnly                 : BOOL;
END_STRUCT
END_TYPE
```

| 名称 | 类型 | 描述 |
|---|---|---|
| tCtrlCycleTime | TIME | 处理控制回路的循环时间。该时间必须大于或等于 **TaskCycleTime**。功能块使用该输入值在内部计算是否需要在当前周期内更新状态和输出值。 |
| tTaskCycleTime | TIME | 调用功能块的循环时间。如果在每个循环中都调用该功能块，这相当于调用任务的任务循环时间。 |
| 范围 **eCTRL_PARAMETER_RECORD_HEATING**： | | |
| fKp_heating | FLOAT | 控制器放大/控制器系数 |
| tTn_heating | TIME | 积分作用时间：如果将其参数设置为 T#0s，则 I 分量将被停用。 |
| tTv_heating | TIME | 微分作用时间：如果将其参数设置为 T#0s，则 D 分量将被停用。 |
| tTd_heating | TIME | 阻尼时间 |
| 范围 **eCTRL_PARAMETER_RECORD_COOLING**： | | |
| fKp_cooling | FLOAT | 控制器放大/控制器系数 |
| tTn_cooling | TIME | 积分作用时间：如果将其参数设置为 T#0s，则 I 分量将被停用。 |
| tTv_cooling | TIME | 微分作用时间：如果将其参数设置为 T#0s，则 D 分量将被停用。 |
| tTd_cooling | TIME | 阻尼时间 |
| nParameterChangeCycleTicks | INT | 从一个参数集切换到另一个参数集的任务周期数。 |
| fOutMaxLimit | FLOAT | 积分停止和输出限制的上限（ARW 措施）。`bARWActive` 输出为 TRUE 时，表示达到该限值。 |
| fOutMinLimit | FLOAT | 积分停止和输出限制的下限（ARW 措施）。`bARWActive` 输出为 TRUE 时，表示达到该限值。 |
| bPInTheFeedbackPath | BOOL | 通过该输入可以选择 P 单元的输入值（请参见功能图）。 |
| bDInTheFeedbackPath | BOOL | 通过该输入可以选择 D 单元的输入值（请参见功能图）。 |
| bARWOnIPartOnly | BOOL | 如果该参数为 FALSE（标准设置），则当控制器的完整输出达到上限或下限时，I 分量的积分将停止。如果该参数为 TRUE，则当 I 分量（积分器的输出）达到某个限值时，积分将停止。（请参见功能图。） |

### 4.2.1.3.13 FB_CTRL_STEP_PID




该功能块控制一个数字式操纵变量，可用于执行器的集成。它基于一个标准 PID 控制器，除模拟量输出外，还附带一个二进制输出信号，无需精确的位置反馈信号即可工作。

计算控制器输出所需的位置反馈信号是通过极限值以及达到这些极限值所需的时间估算得出的。然后，该功能块会产生驱动执行器作为伺服电机运行所需的脉冲。

### 输入

```
VAR_INPUT
fSetpointValue         : LREAL;
fActualValue           : LREAL;
bLimitSwitchPos        : BOOL;
bLimitSwitchNeg        : BOOL;
bAutoReset             : BOOL;
nSensorType            : E_CTRL_STEP_SENSORTYPES;
fDisturbanceValue      : REAL;
bManualDigitalPos      : BOOL;
bManualDigitalNeg      : BOOL;
eMode                  : E_CTRL_MODE;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| fSetpointValue | LREAL | 控制变量的设定点 |
| fActualValue | LREAL | 控制值的实际值 |
| bLimitSwitchPos | BOOL | 限位开关，在达到上限位时变为 TRUE。 |
| bLimitSwitchNeg | BOOL | 限位开关，在达到下限位时变为 TRUE。 |
| bAutoReset | BOOL | 如果该变量为 TRUE，则将控制器重置为初始状态。 |
| nSensorType | E_CTRL_STEP_SENSORTYPES | 用于选择正确的热电偶类型。 |
| fDisturbanceValue | REAL | 扰动变量的实际值 |
| bManualDigitalPos | BOOL | 如果该变量为 TRUE，则手动将电机设置为正转方向 |
| bManualDigitalNeg | BOOL | 如果为 TRUE，则手动将电机设置为反转方向 |
| eMode | E_CTRL_MODE | 指定控制器运行模式的输入 |

### 输出

```
VAR_OUTPUT
bCtrlDigitalPos    : BOOL;
bCtrlDigitalNeg    : BOOL;
fCtrlAnalog        : LREAL;
eState             : E_CTRL_STATE;
eErrorId           : E_CTRL_ERRORCODES;
bError             : BOOL;
bBusy              : BOOL;
bMinLimitNeg       : BOOL;
bMinLimitPos       : BOOL;
bARWActive         : BOOL;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| bCtrlDigitalPos | BOOL | 电机正向运行所需的输出。 |
| bCtrlDigitalNeg | BOOL | 电机反向运行所需的输出。 |
| fCtrlAnalog | LREAL | 控制器的模拟量输出 |
| eState | E_CTRL_STATE | 控制器的实际状态 |
| eErrorId | E_CTRL_ERRORCODES | 如果 `bError` 为 TRUE，则返回错误编号。 |
| bError | BOOL | 如果功能块发生错误，则为 TRUE。 |
| bBusy | BOOL | 当功能块处于活动状态时，则为 TRUE。 |
| bMinLimitNeg | BOOL | 该输出为 TRUE 时，表示控制器已达到最小限值。 |
| bMinLimitPos | BOOL | 该输出为 TRUE 时，表示控制器已达到最大限值。 |
| bARWActive | BOOL | 该输出为 TRUE 时，表示控制器当前处于限制状态。 |

### 输入/输出

```
VAR_IN_OUT
stCtrlParams : ST_CTRL_STEP_PID_PARAMS;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| stCtrlParams | ST_CTRL_STEP_PID_PARAMS | 控制器的参数结构。 |

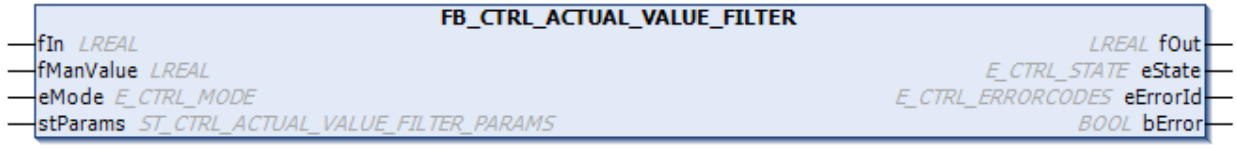
stCtrlParams 由以下元素组成：

```
TYPE ST_CTRL_STEP_PID_PARAMS:
STRUCT
tCtrlCycleTime      : TIME;
tTaskCycleTime      : TIME;
fKp                 : REAL;
fTn                 : REAL;
fTv                 : REAL;
fTd                 : REAL;
fTM                 : REAL;
fDeadBandWidth      : REAL;
tMinimumPulseTime   : TIME;
tFilterTime         : TIME;
fWmax               : REAL;
fWmin               : REAL;
fYMax               : REAL;
fYMin               : REAL;
END_STRUCT
END_TYPE
```

| 名称 | 类型 | 描述 |
|---|---|---|
| tCtrlCycleTime | TIME | 处理控制回路的循环时间。该时间必须大于或等于 **TaskCycleTime**。功能块使用该输入值进行内部计算，以确定在当前周期内是否已更新状态和输出值。 |
| tTaskCycleTime | TIME | 调用功能块的循环时间。如果在每个循环中都调用该功能块，这相当于调用任务的任务循环时间。 |
| fKp | REAL | 控制器增益或比例增益 |
| fTn | REAL | 积分作用时间：如果该参数为零，则 I 分量将被禁用。 |
| fTv | REAL | 微分作用时间；如果该参数为零，则 D 分量将被禁用。 |
| fTd | REAL | 阻尼时间 |
| fTM | REAL | 数字阀门驱动时间，以秒为单位 |
| fDeadBandWidth | REAL | 控制器偏差的允许宽度 |
| tMinimumPulseTime | TIME | 脉冲发生器的最短脉冲时间 |
| tFilterTime | TIME | 实际值滤波器的滤波时间 |
| fWmax | REAL | 设定最大值，以 °C 为单位 |
| fWmin | REAL | 设定最小值，以 °C 为单位 |
| fYMax | REAL | 控制器输出的最大值 |
| fYMin | REAL | 控制器输出的最小值 |

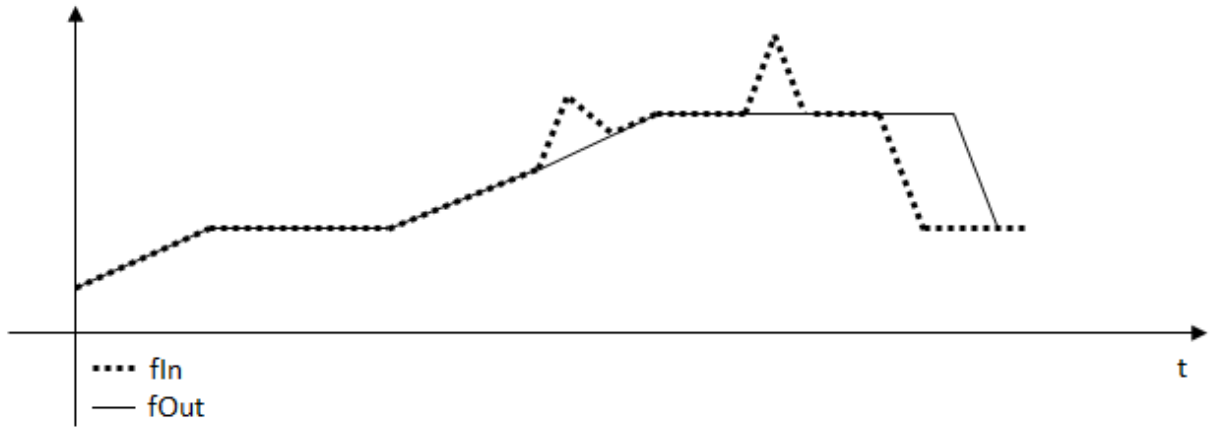
## 4.2.1.4 滤波器/受控系统仿真

### 4.2.1.4.1 FB_CTRL_ACTUAL_VALUE_FILTER




该功能块可对测量的输入值进行合理性检查与滤波。

**输出的行为**




该功能块可对测量的输入值进行合理性检查。如果两个连续采样值（测量值）之间的差值大于指定窗口 `fDeltaMax`，则当前输入值将被抑制最多三个周期。在此期间，输出值将根据先前的输入值进行外插值推算。如果超出指定窗口的状况持续超过三个周期，则输出将再次跟随输入值。输出的行为如上图所示。

### 输入

```
VAR_INPUT
    fIn         : FLOAT;
    fManValue   : FLOAT;
    eMode       : E_CTRL_MODE;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| fIn | FLOAT | 滤波器的输入值 |
| fManValue | FLOAT | 在手动模式下输出的输入值。 |
| eMode | E_CTRL_MODE | 指定功能块的运行模式 [▶ 157]的输入。 |

### 输出

```
VAR_OUTPUT
    fOut        : FLOAT;
    eState      : E_CTRL_STATE;
    eErrorId    : E_CTRL_ERRORCODES;
    bError      : BOOL;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| fOut | FLOAT | 功能块的输出 |

| 名称 | 类型 | 描述 |
|---|---|---|
| eState | E_CTRL_STATE | 功能块的状态 |
| eErrorId | E_CTRL_ERROR CODES | 在设置输出 bError 时，提供错误编号 [▶ 157]。 |
| bError | BOOL | 一旦发生错误，则变为 TRUE。 |

### 输入/输出

```
VAR_IN_OUT
    stParams    : ST_CTRL_ACTUAL_VALUE_FILTER_PARAMS;
END_VAR
```

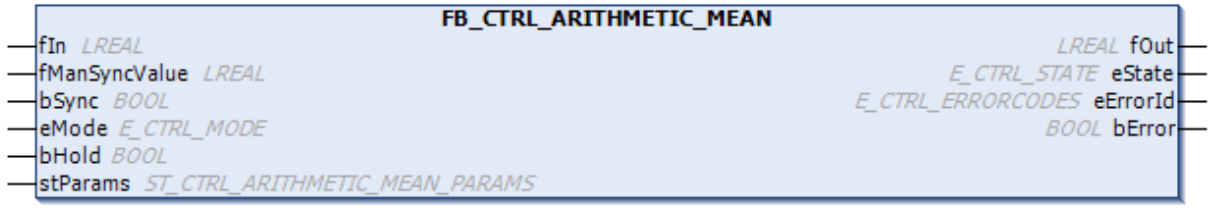
| 名称 | 类型 | 描述 |
|---|---|---|
| stParams | ST_CTRL_ACTUAL_VALUE_FILTER_PARAMS | 实际值滤波单元的参数结构 |

stParams 由以下元素组成：

```
TYPE
ST_CTRL_ACTUAL_VALUE_FILTER_PARAMS:
STRUCT
    tCtrlCycleTime    : TIME := T#0ms;
    tTaskCycleTime    : TIME := T#0ms;
    fDeltaMax         : FLOAT;
END_STRUCT
END_TYPE
```

| 名称 | 类型 | 描述 |
|---|---|---|
| tCtrlCycleTime | TIME | 处理控制回路的循环时间。该时间必须大于或等于 **TaskCycleTime**。功能块使用该输入值在内部计算是否需要在当前周期内更新状态和输出值。 |
| tTaskCycleTime | TIME | 调用功能块的循环时间。如果在每个循环中都调用该功能块，这相当于调用任务的任务循环时间。 |
| fDeltaMax | FLOAT | 两个连续输入值之间的最大差值。请参见上文描述。 |

## 4.2.1.4.2 FB_CTRL_ARITHMETIC_MEAN




$$\bar{\chi} = \frac{1}{n}\sum_{n}\chi_n$$

### 输入

```
VAR_INPUT
    fIn             : FLOAT;
    fManSyncValue   : FLOAT;
    bSync           : BOOL;
```

```
    eMode           : E_CTRL_MODE;
    bHold           : BOOL;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| fIn | FLOAT | 平均值滤波器的输入值 |
| fManSync Value | FLOAT | 可用于设置平均值滤波器的输入值，或在手动模式下从输出端发出的输入值。 |
| bSync | BOOL | 该输入端的上升沿将平均值滤波器设置为值“fManSyncValue”。 |
| eMode | E_CTRL_MODE | 指定功能块的运行模式 [▶ 157]的输入。 |
| bHold | BOOL | 该输入为 TRUE 时，内部状态（以及输出）将保持当前值不变，且不受输入值影响。 |

### 输出

```
VAR_OUTPUT
    fOut       : FLOAT;
    eState     : E_CTRL_STATE;
    eErrorId   : E_CTRL_ERRORCODES;
    bError     : BOOL;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| fOut | FLOAT | 平均值滤波器的输出 |
| eState | E_CTRL_STATE | 功能块的状态 |
| eErrorId | E_CTRL_ERRORCODES | 在设置输出 bError 时，提供错误编号 [▶ 157]。 |
| bError | BOOL | 一旦发生错误，则变为 TRUE。 |

### 输入/输出

```
VAR_IN_OUT
    stParams    : ST_CTRL_ARITHMETIC_MEAN_PARAMS;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| stParams | ST_CTRL_ARITHMETIC_MEAN_PARAMS | 平均值滤波器的参数结构 |

stParams 由以下元素组成：

```
TYPE
ST_CTRL_ARITHMETIC_MEAN_PARAMS:
STRUCT
    tCtrlCycleTime : TIME := T#0ms;
    tTaskCycleTime : TIME := T#0ms;
END_STRUCT
END_TYPE
```

| 名称 | 类型 | 描述 |
|---|---|---|
| tCtrlCycleTime | TIME | 处理控制回路的循环时间。该时间必须大于或等于 **TaskCycleTime**。功能块使用该输入值在内部计算是否需要在当前周期内更新状态和输出值。 |
| tTaskCycleTime | TIME | 调用功能块的循环时间。如果在每个循环中都调用该功能块，这相当于调用任务的任务循环时间。 |

## 4.2.1.4.3 FB_CTRL_MEDIAN_FILTER


该功能块指定中位数。中位数是按大小排序后的一系列数值的平均值。这意味着收集到的数据值有一半小于中位数，另一半大于中位数。

程序员必须创建一个数组“ARRAY [1..2(n+2)] OF LREAL”，功能块可以在其中存储内部所需数据。

### 输入

```
VAR_INPUT
    fIn                 : FLOAT;
    fManSyncValue       : FLOAT;
    eMode               : E_CTRL_MODE;
    bSync               : FLOAT;
    bHold               : BOOL;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| fIn | FLOAT | 中位数滤波器的输入值 |
| fManSyncValue | FLOAT | 可用于设置中位数滤波器的输入值，或在手动模式下输出的值。 |
| eMode | E_CTRL_MODE | 该输入端的上升沿将中位数滤波器设置为值“fManSyncValue”。 |
| bSync | FLOAT | 指定功能块的运行模式的输入。 |
| bHold | BOOL | 该输入为 TRUE 时会保持内部状态，因此也保持输出。 |

### 输出

```
VAR_OUTPUT
    fOut        : FLOAT;
    eState      : E_CTRL_STATE;
    eErrorId    : E_CTRL_ERRORCODES;
    bError      : BOOL;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| fOut | FLOAT | 中位数滤波器的输出 |
| eState | E_CTRL_STATE | 功能块的状态 |
| eErrorId | E_CTRL_ERRORCODES | 在设置输出 bError 时，提供错误编号。 |
| bError | BOOL | 一旦发生错误，则变为 TRUE。 |

### 输入/输出

```
VAR_IN_OUT
    tCtrlCycleTime          : TIME := T#0ms;
    tTaskCycleTime          : TIME := T#0ms;
    nSamplesToFilter        : UINT;
    pWorkArray_ADR          : POINTER TO FLOAT := 0;
    nWorkArray_SIZEOF       : UINT := 0;
END_VAR
```

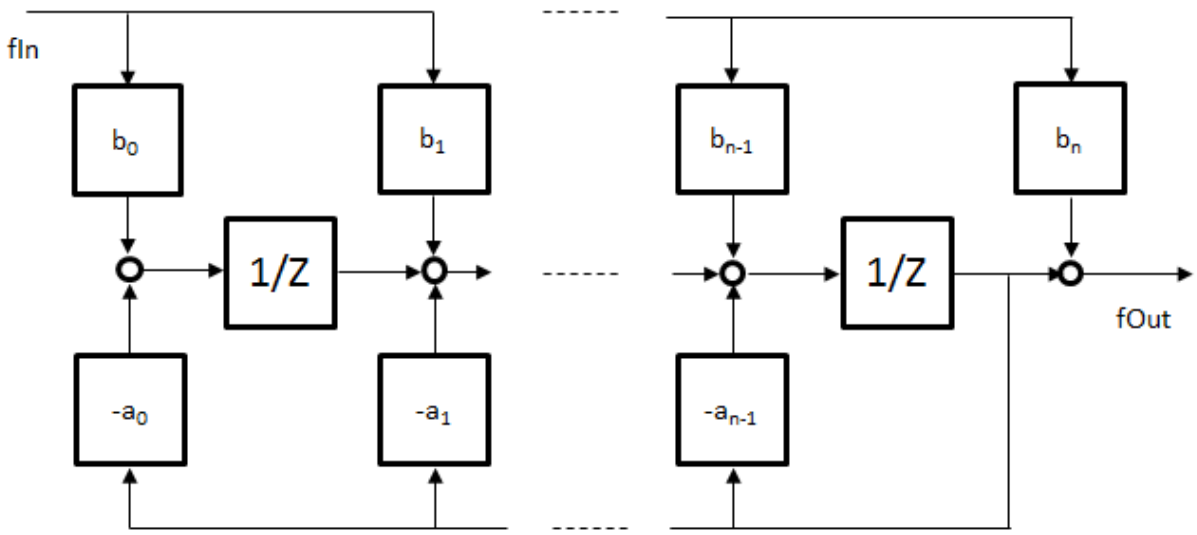
| 名称 | 类型 | 描述 |
|---|---|---|
| tCtrlCycleTime | TIME | 处理控制回路的循环时间。该时间必须大于或等于 **TaskCycleTime**。功能块使用该输入值在内部计算是否需要在当前周期内更新状态和输出值。 |

| 名称 | 类型 | 描述 |
|---|---|---|
| tTaskCycleTime | TIME | 调用功能块的循环时间。如果在每个循环中都调用该功能块，这相当于调用任务的任务循环时间。 |
| nSamplesTo Filter | UINT | 用于生成中位数的数值个数“n”。 |
| pWorkArray_ ADR | POINTER TO FLOAT | 暂时存储输入值的数组的地址。 |
| nWorkArray_ SIZEOF | UINT | `WorkArrays` 的大小 |

### 4.2.1.4.4 FB_CTRL_DIGITAL_FILTER


该功能块使用下述结构计算离散传递函数。该结构允许实现 FIR 滤波器（有限脉冲响应，**FIR**）或 IIR 滤波器（无限脉冲响应，**IIR**）。此处的传递函数可以是任意阶次 n。

以下传递结构的系数存储在参数数组中：




$$G(z)=\frac{Y(z)}{X(z)}=\frac{b_1+b_2z^{-1}+b_3z^{-2}+...+b_nz^{-(n-1)}}{1+a_2z^{-1}+a_3z^{-2}+...+a_nz^{-(n-1)}}$$

如果要使用该功能块，程序员必须在 PLC 中创建以下数组：

```
aCoefficientsArray_a        : ARRAY[1..nFilterOrder+1] OF FLOAT;
aCoefficientsArray_b        : ARRAY[1..nFilterOrder+1] OF FLOAT;
aStDigitalFilterData        : ARRAY[1..nFilterOrder] OF
ST_CTRL_DIGITAL_FILTER_DATA;
```

系数“$b_1$”至“$b_n$”存储在数组 `aCoefficientsArray_b` 中。其结构必须按以下方式组织：

```
aCoefficientsArray_b[1] := b1;
aCoefficientsArray_b[2] := b2;
…
aCoefficientsArray_b[n-1] := bn-1;
aCoefficientsArray_b[n]   := bn;
```

系数“a1”至“an”存储在数组 aCoefficientsArray_a 中。
其结构必须按以下方式组织：

```
aCoefficientsArray_a[1] := xxx; (* not being evaluated *)
aCoefficientsArray_a[2] := a2;
...
aCoefficientsArray_a[n-1] := an-1;
aCoefficientsArray_a[n]   := an;
```

功能块所需的内部数据存储在数组 aStDigitalFilterData 中。绝不能在 PLC 程序内部修改此数据。此流程也在下方列出的示例程序中予以说明。

### 输入

```
VAR_INPUT
    fIn         : FLOAT;
    fManValue   : FLOAT;
    eMode       : E_CTRL_MODE;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| fIn | FLOAT | 数字滤波器的输入 |
| fManValue | FLOAT | 在手动模式下，其数值直接出现在输出端的输入。 |
| eMode | E_CTRL_MODE | 指定功能块的运行模式的输入。 |

### 输出

```
VAR_OUTPUT
    fOut       : FLOAT;
    eState     : E_CTRL_STATE;
    bError     : BOOL;
    eErrorId   : E_CTRL_ERRORCODES;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| fOut | FLOAT | 数字滤波器的输出 |
| eState | E_CTRL_STATE | 功能块的状态 |
| bError | BOOL | 一旦发生错误，则变为 TRUE。 |
| eErrorId | E_CTRL_ERRORCODES | 在设置输出 bError 时，提供错误编号 [▶ 157]。 |

### 输入/输出

```
VAR_IN_OUT
    stParams        : ST_CTRL_DIGITAL_FILTER_PARAMS;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| stParams | ST_CTRL_DIGITAL_FILTER_PARAMS | 功能块的参数结构 |

stParams 由以下元素组成：

```
TYPE
ST_CTRL_DIGITAL_FILTER_PARAMS :
STRUCT
    tTaskCycleTime                  : TIME;
    tCtrlCycleTime                  : TIME := T#0ms;
    nFilterOrder                    : USINT;
    pCoefficientsArray_a_ADR        : POINTER TO FLOAT := 0;
    nCoefficientsArray_a_SIZEOF     : UINT;
    pCoefficientsArray_b_ADR        : POINTER TO FLOAT := 0;
    nCoefficientsArray_b_SIZEOF     : UINT;
    pDigitalFilterData_ADR          : POINTER TO ST_CTRL_DIGITAL_FILTER_DATA;
    nDigitalFilterData_SIZEOF       : UINT;
END_STRUCT
END_TYPE
```

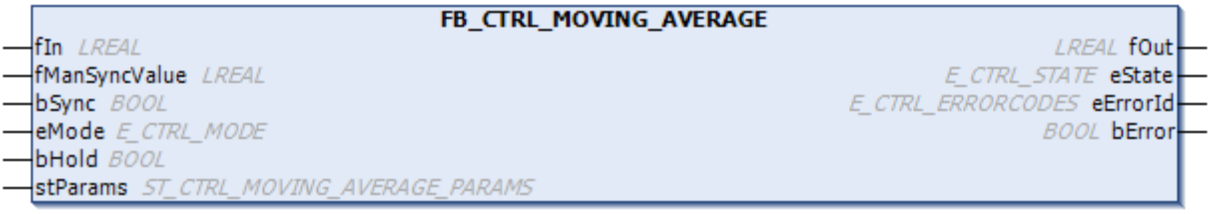
| 名称 | 类型 | 描述 |
|---|---|---|
| tTaskCycleTime | TIME | 调用功能块的循环时间。如果在每个循环中都调用该功能块，这相当于调用任务的任务循环时间。 |
| tCtrlCycleTime | TIME | 处理控制回路的循环时间。该时间必须大于或等于 **TaskCycleTime**。功能块使用该输入值在内部计算是否需要在当前周期内更新状态和输出值。 |
| nFilterOrder | USINT | 数字滤波器的阶次 [0... ] |
| pCoefficientsArray_a_ADR | POINTER TO FLOAT | 包含 a 系数的数组的地址 |
| nCoefficientsArray_a_SIZEOF | UINT | 包含 a 系数的数组的大小，以字节为单位 |
| pCoefficientsArray_b_ADR | POINTER TO FLOAT | 包含 b 系数的数组的地址 |
| nCoefficientsArray_b_SIZEOF | UINT | 包含 b 系数的数组的大小，以字节为单位 |
| pDigitalFilterData_ADR | POINTER TO ST_CTRL_DIGITAL_FILTER_DATA | 数据数组的地址 |
| nDigitalFilterData_SIZEOF | UINT | 数据数组的大小，以字节为单位 |

```
TYPE
ST_CTRL_DIGITAL_FILTER_DATA
STRUCT
 Internal structure. This must not be written to.
END_STRUCT
END_TYPE
```

**示例：**

```
PROGRAM PRG_DIGITAL_FILTER_TEST
VAR
    fbDigitalFilter         : FB_CTRL_DIGITAL_FILTER;
    aCoefficientsArray_a    : ARRAY[1..3] OF FLOAT;
    aCoefficientsArray_b    : ARRAY[1..3] OF FLOAT;
    aStDigitalFilterData    : ARRAY[1..2] OF ST_CTRL_DIGITAL_FILTER_DATA;
    eMode                 : E_CTRL_MODE;
    stParams             : ST_CTRL_DIGITAL_FILTER_PARAMS;
    eErrorId             : E_CTRL_ERRORCODES;
    bError             : BOOL;
    fIn                 : FLOAT := 0;
    fOut                : FLOAT;
    bInit                : BOOL := TRUE;
END_VAR

IF bInit THEN
    aCoefficientsArray_a[1] := 0.0; (* not used *)
    aCoefficientsArray_a[2] := 0.2;
    aCoefficientsArray_a[3] := 0.1;
    aCoefficientsArray_b[1] := 0.6;
    aCoefficientsArray_b[2] := 0.4;
    aCoefficientsArray_b[3] := 0.2;

    stParams.tTaskCycleTime := T#2ms;
    stParams.tCtrlCycleTime := T#2ms;
    stParams.nFilterOrder := 2;

    eMode := eCTRL_MODE_ACTIVE;
    bInit := FALSE;
END_IF

stParams.pCoefficientsArray_a_ADR := ADR(aCoefficientsArray_a);
stParams.nCoefficientsArray_a_SIZEOF := SIZEOF(aCoefficientsArray_a);
stParams.pCoefficientsArray_b_ADR := ADR(aCoefficientsArray_b);
stParams.nCoefficientsArray_b_SIZEOF := SIZEOF(aCoefficientsArray_b);
stParams.pDigitalFilterData_ADR := ADR(astDigitalFilterData);
stParams.nDigitalFilterData_SIZEOF := SIZEOF(aStDigitalFilterData);
fbDigitalFilter ( fIn := fIn,
                  eMode := eMode,
                  stParams := stParams,
                  fOut => fOut,
                  eErrorId => eErrorId,
                  bError => bError);
```

## 4.2.1.4.5 FB_CTRL_MOVING_AVERAGE




该功能块在功能图中提供了一个滑动平均滤波器。

计算最后“n”个值的算术平均值。

$$\bar{x}_k^n = \frac{1}{n}\sum_{i=k-n} x_i$$

程序员必须在 PLC 中创建一个数组“ARRAY [1...n] of FLOAT”，功能块可以在其中存储内部所需数据。

### 输入

```
VAR_INPUT
    fIn             : FLOAT;
    fManSyncValue    : FLOAT;
    bSync           : BOOL;
    eMode           : E_CTRL_MODE;
    bHold           : BOOL;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| fIn | FLOAT | 滑动平均滤波器的输入值 |
| fManSyncValue | FLOAT | 可用于设置滑动平均滤波器的输入值，或在手动模式下从输出端发出的输入值。 |
| bSync | BOOL | 该输入端的上升沿将滑动平均滤波器设置为值“fManSyncValue”。 |
| eMode | E_CTRL_MODE | 指定功能块的运行模式 [▶ 157]的输入。 |
| bHold | BOOL | 该输入为 TRUE 时，内部状态（以及输出）将保持当前值不变，且不受输入值影响。 |

### 输出

```
VAR_OUTPUT
    fOut        : FLOAT;
    eState     : E_CTRL_STATE;
    eErrorId    : E_CTRL_ERRORCODES;
    bError     : BOOL;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| fOut | FLOAT | 滑动平均滤波器的输出 |
| eState | E_CTRL_STATE | 功能块的状态 |
| eErrorId | E_CTRL_ERRORCODES | 在设置输出 bError 时，提供错误编号 [▶ 157]。 |
| bError | BOOL | 一旦发生错误，则变为 TRUE。 |

### 输入/输出

```
VAR_IN_OUT
    stParams                : ST_CTRL_MOVING_AVERAGE_PARAMS;
END_VAR
```

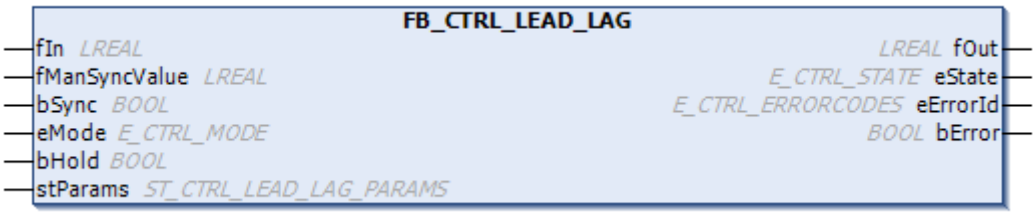
| 名称 | 类型 | 描述 |
|---|---|---|
| stParams | ST_CTRL_MOVING_AVERAGE_PARAMS | 滑动平均滤波器的参数结构 |

`stParams` 由以下元素组成：

```
TYPE
ST_CTRL_MOVING_AVERAGE_PARAMS:
STRUCT
    tCtrlCycleTime        : TIME := T#0ms;
    tTaskCycleTime        : TIME := T#0ms;
    nSamplesToFilter      : UINT;
    pWorkArray_ADR        : POINTER TO FLOAT := 0;
    nWorkArray_SIZEOF     : UINT     := 0;
END_STRUCT
END_TYPE
```

| 名称 | 类型 | 描述 |
|---|---|---|
| tCtrlCycleTime | TIME | 处理控制回路的循环时间。该时间必须大于或等于 **TaskCycleTime**。功能块使用该输入值在内部计算是否需要在当前周期内更新状态和输出值。 |
| tTaskCycleTime | TIME | 调用功能块的循环时间。如果在每个循环中都调用该功能块，这相当于调用任务的任务循环时间。 |
| nSamplesTo Filter | UINT | 计算算术平均值的数值个数“n”。 |
| pWorkArray_ ADR | POINTER TO FLOAT | 暂时存储输入值的数组的地址。 |
| nWorkArray_ SIZEOF | UINT | `WorkArrays` 的大小 |

## 4.2.1.4.6 FB_CTRL_LEAD_LAG

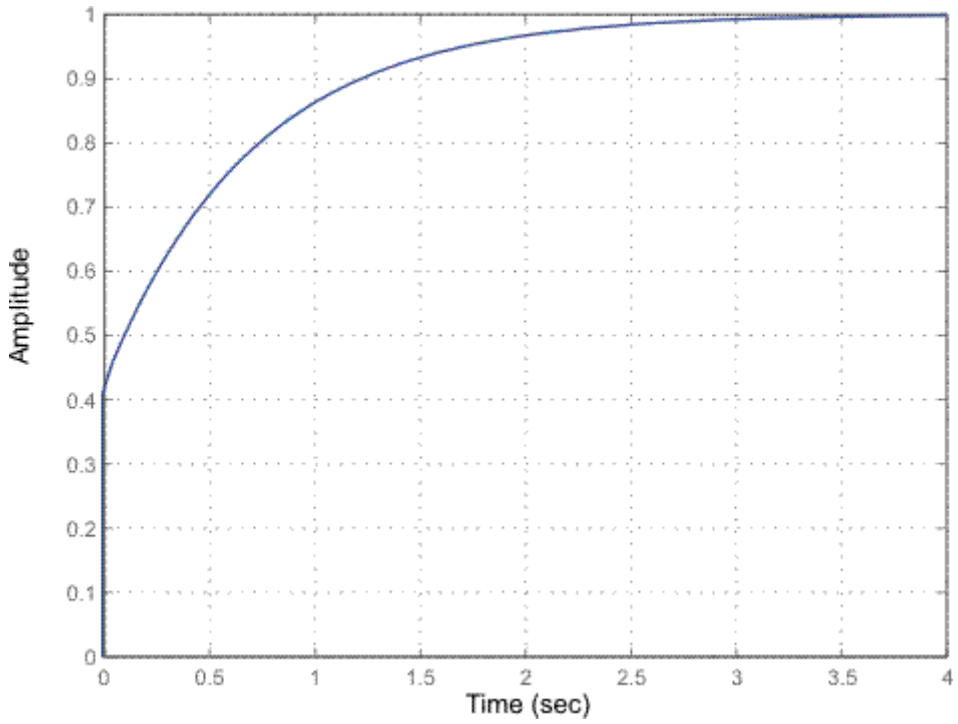



该功能块代表一个数字式超前/滞后滤波器。

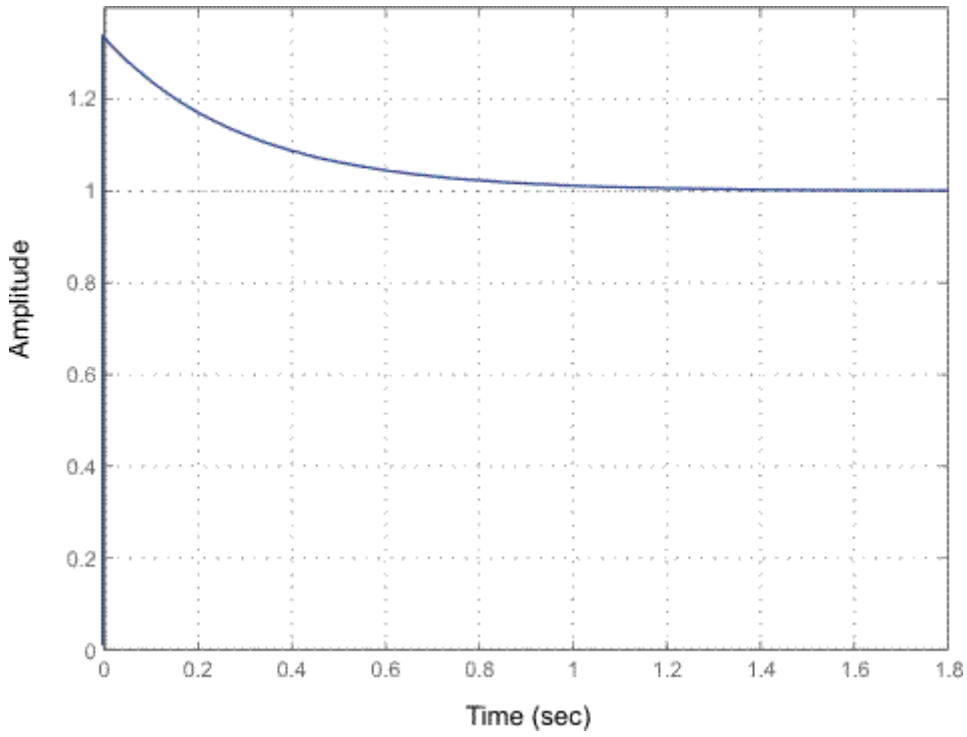
**传递函数**

$$G(s) = \frac{1 + T_1 s}{1 + T_2 s}$$

**阶跃响应，T2 > T1**




**阶跃响应，T1 > T2**




在 T=0 时，阶跃响应为 T1 / T2。

### 输入

```
VAR_INPUT
    fIn                 : FLOAT;
    fManSyncValue       : FLOAT;
    bSync               : FLOAT;
    fManValue           : FLOAT;
    eMode               : E_CTRL_MODE;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| fIn | FLOAT | 陷波滤波器的输入值 |
| fManSyncValue | FLOAT | 在手动模式下从输出端输出的输入值。 |
| bSync | FLOAT | 备用 |
| fManValue | FLOAT | 指定功能块的运行模式 [▶ 157]的输入。 |
| eMode | E_CTRL_MODE | 该输入为 TRUE 时，内部状态（以及输出）将保持当前值不变，且不受输入值影响。 |

### 输出

```
VAR_OUTPUT
    fOut        : FLOAT;
    eState      : E_CTRL_STATE;
    eErrorId    : E_CTRL_ERRORCODES;
    bError      : BOOL;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| fOut | FLOAT | 超前/滞后滤波器输出 |
| eState | E_CTRL_STATE | 功能块的状态 |
| eErrorId | E_CTRL_ERRORCODES | 在设置输出 bError 时，提供错误编号 [▶ 157]。(p>) |
| bError | BOOL | 一旦发生错误，则变为 TRUE。 |

### 输入/输出

```
VAR_IN_OUT
    stParams    : ST_CTRL_LEAD_LAG_FILTER_PARAMS;
END_VAR
```

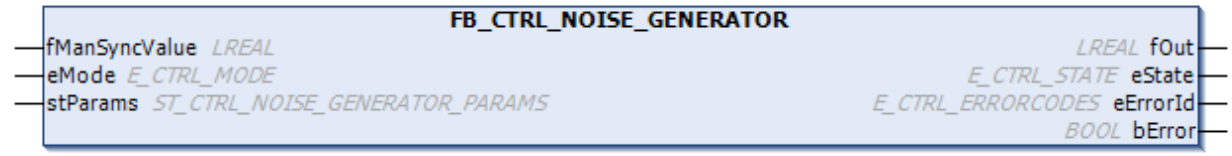
| 名称 | 类型 | 描述 |
|---|---|---|
| stParams | ST_CTRL_LEAD_LAG_FILTER_PARAMS | 超前/滞后滤波器的参数结构 |

stParams 由以下元素组成：

```
TYPE
ST_CTRL_LEAD_LAG_FILTER_PARAMS:
STRUCT
    tCtrlCycleTime    : TIME := T#0ms;
    tTaskCycleTime    : TIME := T#0ms;
    tT1               : TIME := T#0ms; (* T1 *)
    tT2               : TIME := T#0ms; (* T2 *)
END_STRUCT
END_TYPE
```

| 名称 | 类型 | 描述 |
|---|---|---|
| tCtrlCycleTime | TIME | 处理控制回路的循环时间。该时间必须大于或等于 **TaskCycleTime**。功能块使用该输入值在内部计算是否需要在当前周期内更新状态和输出值。 |
| tTaskCycleTime | TIME | 调用功能块的循环时间。如果在每个循环中都调用该功能块，这相当于调用任务的任务循环时间。 |
| tT1 | TIME | T1，请参见 G(s)。 |
| tT2 | TIME | T2，请参见 G(s)。 |

## 4.2.1.4.7 FB_CTRL_NOISE_GENERATOR




该功能块根据范围 [-fAmplitude ... fAmplitude] 内的伪随机数生成噪声信号。

### 输出信号

振幅为 5.0 的输出信号。

### 输入

```
VAR_INPUT
    fManSyncValue   : FLOAT;
    eMode           : E_CTRL_MODE;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| fManSyncValue | FLOAT | 在手动模式下从输出端输出的输入值。 |
| eMode | E_CTRL_MODE | 指定功能块的运行模式 [▶ 157]的输入。 |

### 输出

```
VAR_OUTPUT
    fOut       : FLOAT;
    eState     : E_CTRL_STATE;
    eErrorId   : E_CTRL_ERRORCODES;
    bError     : BOOL;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| fOut | FLOAT | 噪声发生器的输出 |
| eState | E_CTRL_STATE | 功能块的状态 |
| eErrorId | E_CTRL_ERRORCODES | 在设置输出 bError 时，提供错误编号 [▶ 157]。 |
| bError | BOOL | 一旦发生错误，则变为 TRUE。 |

### 输入/输出

```
VAR_IN_OUT
    stParams        : ST_CTRL_NOISE_GENERATOR_PARAMS;
END_VAR
```

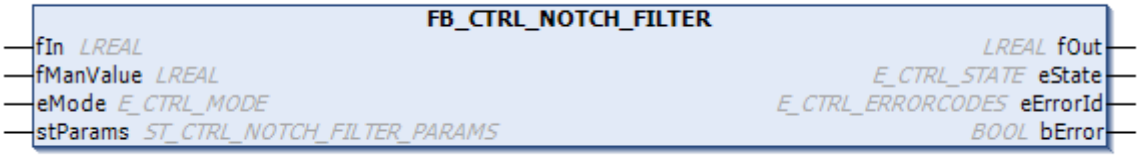
| 名称 | 类型 | 描述 |
|---|---|---|
| stParams | ST_CTRL_NOISE_GENERATOR_PARAMS | 噪声发生器的参数结构 |

stParams 由以下元素组成：

```
TYPE
ST_CTRL_NOISE_GENERATOR_PARAMS:
STRUCT
    tCtrlCycleTime : TIME := T#0ms;
    tTaskCycleTime : TIME := T#0ms;
    fAmplitude     : FLOAT := 0;
END_TYPE
```

| 名称 | 类型 | 描述 |
|---|---|---|
| tCtrlCycleTime | TIME | 处理控制回路的循环时间。该时间必须大于或等于 **TaskCycleTime**。功能块使用该输入值在内部计算是否需要在当前周期内更新状态和输出值。 |
| tTaskCycleTime | TIME | 调用功能块的循环时间。如果在每个循环中都调用该功能块，这相当于调用任务的任务循环时间。 |
| fAmplitude | FLOAT | 输出信号的振幅：在功能块的输出端创建一个扩展范围为 [-fAmplitude/2.0 ... fAmplitude/2.0] 的噪声信号。 |

### 4.2.1.4.8 FB_CTRL_NOTCH_FILTER

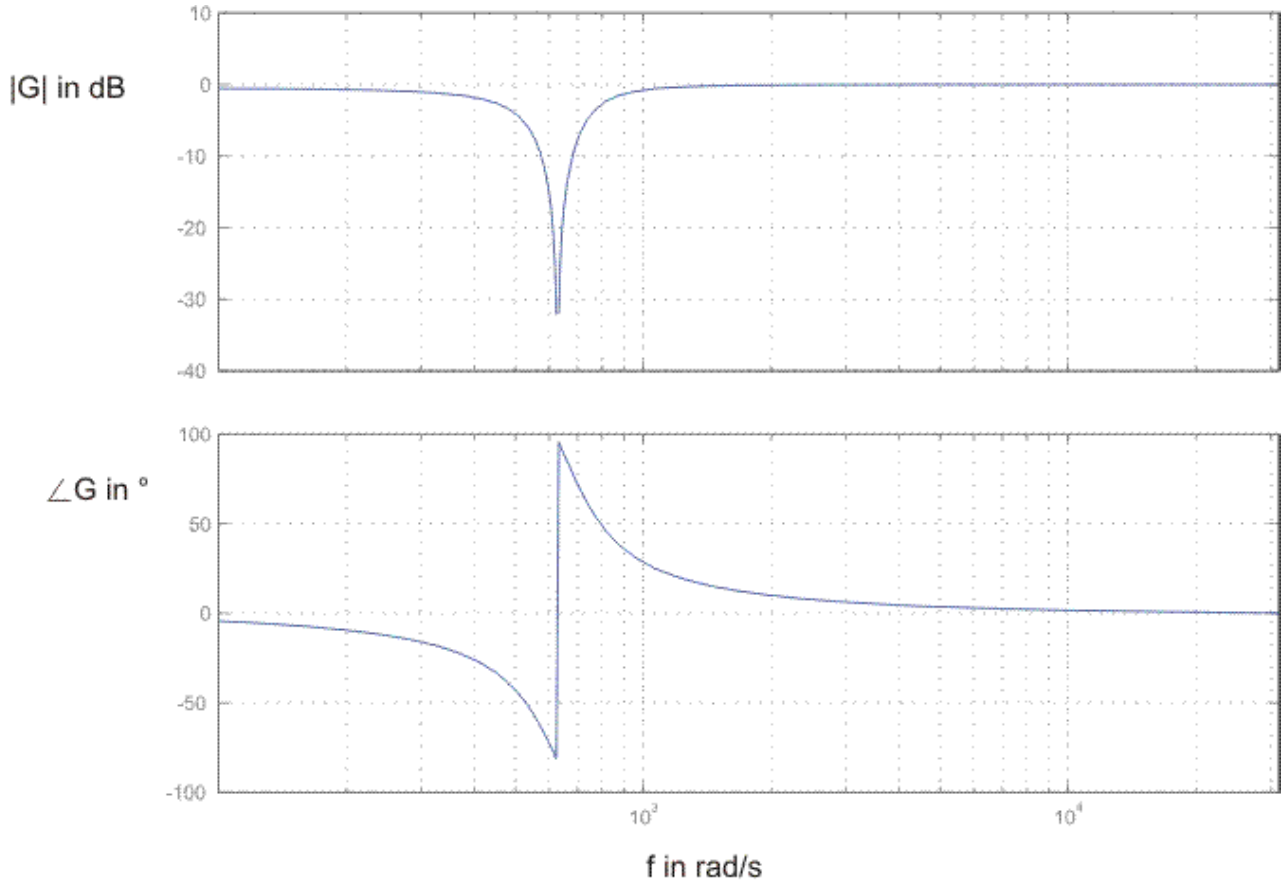



该功能块代表一个数字式陷波滤波器。

**传送指令**

$$G(s) = \frac{s^2 + w_0^2}{s^2 + ew_0 s + w_0^2}$$

**伯德图**



及：

$f_0 = 100Hz$

$e = 0.25$

### 输入

```
VAR_INPUT
    fIn         : FLOAT;
    fManValue   : FLOAT;
    eMode       : E_CTRL_MODE;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| fIn | FLOAT | 陷波滤波器的输入值 |
| fManValue | FLOAT | 在手动模式下从输出端输出的输入值。 |
| eMode | E_CTRL_MODE | 指定功能块的运行模式 [▶ 157]的输入。 |

### 输出

```
VAR_OUTPUT
    fOut        : FLOAT;
    eState      : E_CTRL_STATE;
    eErrorId    : E_CTRL_ERRORCODES;
    bError      : BOOL;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| fOut | FLOAT | 陷波滤波器输出 |
| eState | E_CTRL_STATE | 功能块的状态 |
| eErrorId | E_CTRL_ERRORCODES | 在设置输出 bError 时，提供错误编号 [▶ 157]。 |
| bError | BOOL | 一旦发生错误，则变为 TRUE。 |

### 输入/输出

```
VAR_IN_OUT
    stParams        : ST_CTRL_NOTCH_FILTER_PARAMS;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| stParams | ST_CTRL_NOTCH_FILTER_PARAMS | 陷波滤波器的参数结构 |

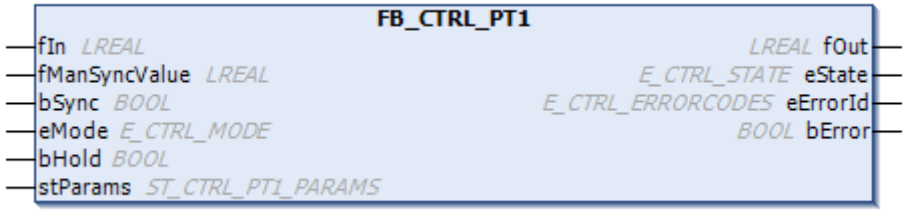
stParams 由以下元素组成：

```
TYPE
ST_CTRL_NOTCH_FILTER_PARAMS:
STRUCT
    tCtrlCycleTime  : TIME := T#0ms;
    tTaskCycleTime  : TIME := T#0ms;
    fNotchFreq      : FLOAT := 0;
    fBandwidth      : FLOAT := 0;
END_STRUCT
END_TYPE
```

| 名称 | 类型 | 描述 |
|---|---|---|
| tCtrlCycleTime | TIME | 处理控制回路的循环时间。该时间必须大于或等于 **TaskCycleTime**。功能块使用该输入值在内部计算是否需要在当前周期内更新状态和输出值。 |
| tTaskCycleTime | TIME | 调用功能块的循环时间。如果在每个循环中都调用该功能块，这相当于调用任务的任务循环时间。 |
| fNotchFreq | FLOAT | 陷波频率，以 Hz 为单位 |

| 名称 | 类型 | 描述 |
|---|---|---|
| fBandwidth | FLOAT | 相对于陷波频率的带宽：<br>以 Hz 为单位的带宽 = fNotchFreq * fBandwidth |

### 4.2.1.4.9 FB_CTRL_PT1

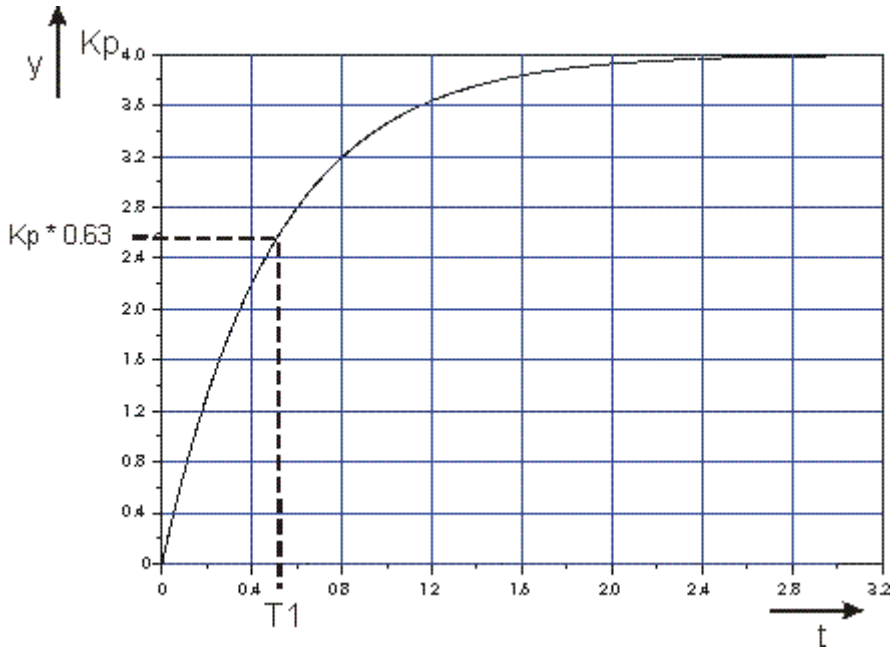



该功能块在功能图中提供了一个 PT1 传递单元。

**传递函数**

$$G(s) = K_p \frac{1}{1+T_1 s}$$

**阶跃响应**




**输入**

```
VAR_INPUT
    fIn             : FLOAT;
    fManSyncValue   : FLOAT;
    bSync           : BOOL;
    eMode           : E_CTRL_MODE;
    bHold           : BOOL;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| fIn | FLOAT | PT1 单元的输入值 |
| fManSyncValue | FLOAT | 可用于设置 PT1 单元的输入值，或在手动模式下从输出端发出的输入值。 |
| bSync | BOOL | 该输入端的上升沿将 PT1 单元设置为值“fManSyncValue”。 |
| eMode | E_CTRL_MODE | 指定功能块的运行模式 [▶ 157]的输入。 |
| bHold | BOOL | 该输入为 TRUE 时，内部状态（以及输出）将保持当前值不变，且不受输入值影响。 |

### 输出

```
VAR_OUTPUT
    fOut       : FLOAT;
    eState     : E_CTRL_STATE;
    eErrorId   : E_CTRL_ERRORCODES;
    bError     : BOOL;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| fOut | FLOAT | PT1 单元的输出 |
| eState | E_CTRL_STATE | 功能块的状态 |
| eErrorId | E_CTRL_ERRORCODES | 在设置输出 bError 时，提供错误编号 [▶ 157]。 |
| bError | BOOL | 一旦发生错误，则变为 TRUE。 |

### 输入/输出

```
VAR_IN_OUT
    stParams         : ST_CTRL_PT1_PARAMS;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| stParams | ST_CTRL_PT1_PARAMS | PT1 单元的参数结构 |

stParams 由以下元素组成：

```
TYPE
ST_CTRL_PT1_PARAMS :
STRUCT
    tCtrlCycleTime   : TIME := T#0ms;
    tTaskCycleTime   : TIME := T#0ms;
    fKp              : FLOAT := 0;
    tT1              : TIME := T#0ms;
END_STRUCT
END_TYPE
```

| 名称 | 类型 | 描述 |
|---|---|---|
| tCtrlCycleTime | TIME | 处理控制回路的循环时间。该时间必须大于或等于 **TaskCycleTime**。功能块使用该输入值在内部计算是否需要在当前周期内更新状态和输出值。 |
| tTaskCycleTime | TIME | 调用功能块的循环时间。如果在每个循环中都调用该功能块，这相当于调用任务的任务循环时间。 |
| fKp | FLOAT | 控制器放大系数/传递系数 |
| tT1 | TIME | 时间常数 |

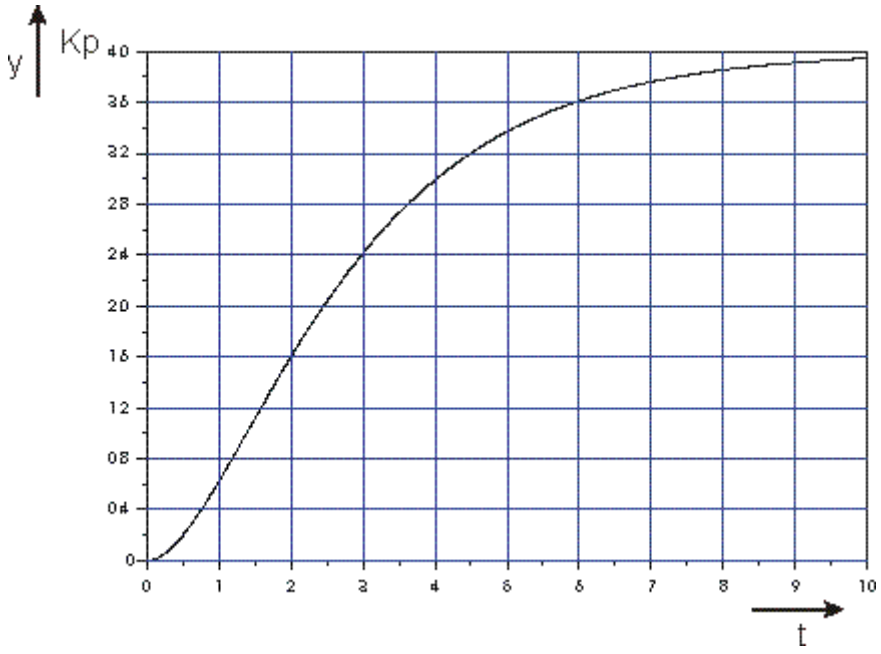
## 4.2.1.4.10 FB_CTRL_PT2


该功能块在功能图中提供了一个非振荡型 PT2 传递单元。

**传递函数**

$$G(s) = K_p \frac{1}{1+T_1 s} \frac{1}{1+T_2 s}$$

**阶跃响应**




### 输入

```
VAR_INPUT
    fIn         : FLOAT;
    fManValue   : FLOAT;
    eMode       : E_CTRL_MODE;
    bHold       : BOOL;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| fIn | FLOAT | PT2 单元的输入值 |
| fManValue | FLOAT | 在手动模式下输出的输入值。 |
| eMode | E_CTRL_MODE | 指定功能块的运行模式 [▶ 157]的输入。 |
| bHold | BOOL | 该输入为 TRUE 时，内部状态（以及输出）将保持当前值不变，且不受输入值影响。 |

### 输出

```
VAR_OUTPUT
    fOut      : FLOAT;
    eState    : E_CTRL_STATE;
    eErrorId  : E_CTRL_ERRORCODES;
    bError    : BOOL;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| fOut | FLOAT | PT2 单元的输出 |
| eState | E_CTRL_STATE | 功能块的状态 |
| eErrorId | E_CTRL_ERRORCODES | 在设置输出 bError 时，提供错误编号 [▶ 157]。 |
| bError | BOOL | 一旦发生错误，则变为 TRUE。 |

### 输入/输出

```
VAR_IN_OUT
    stParams        : ST_CTRL_PT2_PARAMS;
END_VAR
```

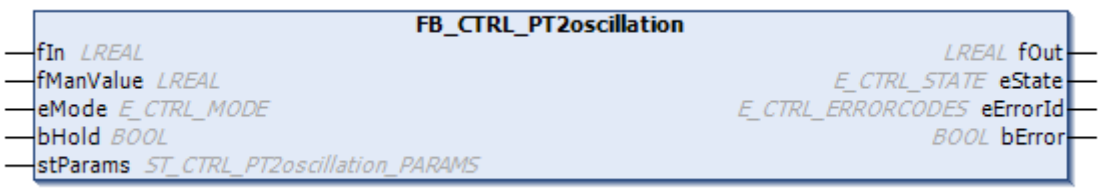
| 名称 | 类型 | 描述 |
|---|---|---|
| stParams | ST_CTRL_PT2_PARAMS | PT2 单元的参数结构 |

stParams 由以下元素组成：

```
TYPE
ST_CTRL_PT2_PARAMS :
STRUCT
    tCtrlCycleTime  : TIME := T#0ms;
    tTaskCycleTime  : TIME := T#0ms;
    fKp             : FLOAT := 0;
    tT1             : TIME := T#0ms;
    tT2             : TIME := T#0ms;
END_STRUCT
END_TYPE
```

| 名称 | 类型 | 描述 |
|---|---|---|
| tCtrlCycleTime | TIME | 处理控制回路的循环时间。该时间必须大于或等于 **TaskCycleTime**。功能块使用该输入值在内部计算是否需要在当前周期内更新状态和输出值。 |
| tTaskCycleTime | TIME | 调用功能块的循环时间。如果在每个循环中都调用该功能块，这相当于调用任务的任务循环时间。 |
| fKp | FLOAT | 控制器放大系数/传递系数 |
| tT1 | TIME | 时间常数 T1 |
| tT2 | TIME | 时间常数 T2 |

## 4.2.1.4.11 FB_CTRL_PT2oscillation

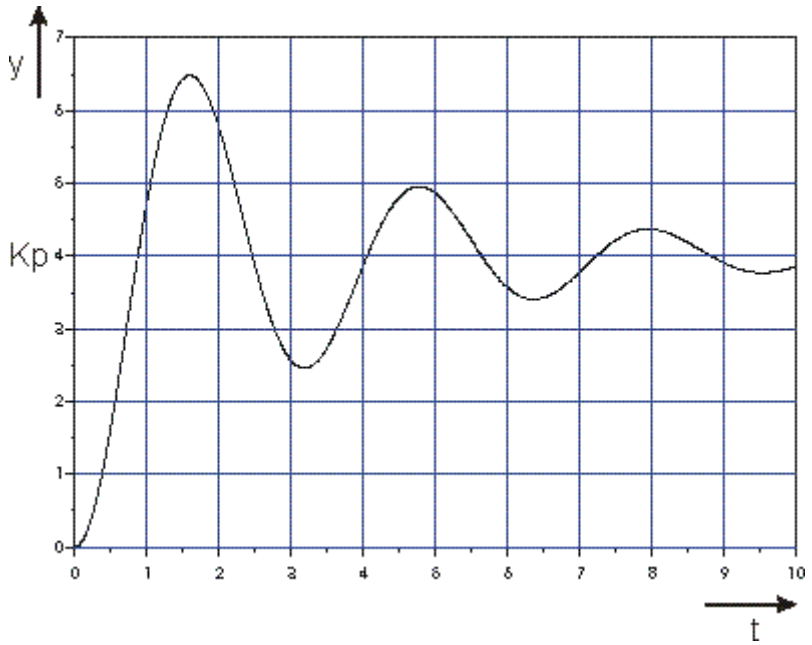



该功能块在功能图中提供了一个振荡型 PT2 传递单元。

### 传递函数

$$G(s) = K_p \frac{1}{1 + 2\vartheta T_0 s + T_0^2 s^2}$$

### 阶跃响应




### 输入

```
VAR_INPUT
    fIn         : FLOAT;
    fManValue   : FLOAT;
    eMode       : E_CTRL_MODE;
    bHold       : BOOL;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| fIn | FLOAT | 振荡型 PT2 单元的输入值 |
| fManValue | FLOAT | 在手动模式下输出的输入值。 |
| eMode | E_CTRL_MODE | 指定功能块的运行模式 [▶ 157]的输入。 |
| bHold | BOOL | 该输入为 TRUE 时，内部状态（以及输出）将保持当前值不变，且不受输入值影响。 |

### 输出

```
VAR_OUTPUT
    fOut        : FLOAT;
    eState      : E_CTRL_STATE;
    eErrorId    : E_CTRL_ERRORCODES;
    bError      : BOOL;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| fOut | FLOAT | PT2 单元的输出 |
| eState | E_CTRL_STATE | 功能块的状态 |
| eErrorId | E_CTRL_ERRORCODES | 在设置输出 bError 时，提供错误编号 [▶ 157]。 |
| bError | BOOL | 一旦发生错误，则变为 TRUE。 |

### 输入/输出

```
VAR_IN_OUT
    stParams        : ST_CTRL_PT2oscillation_PARAMS;
END_VAR
```

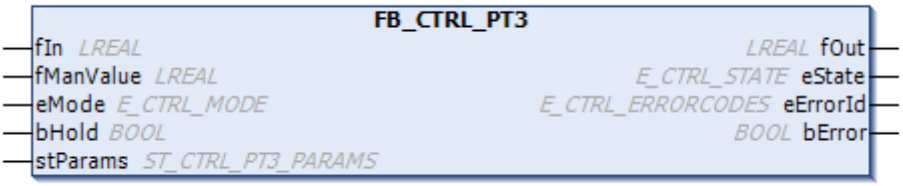
| 名称 | 类型 | 描述 |
|---|---|---|
| stParams | ST_CTRL_PT2 oscillation_ PARAMS | 振荡型 PT2 单元的参数结构 |

`stParams` 由以下元素组成：

```
TYPE
ST_CTRL_PT2oscillation_PARAMS :
STRUCT
    tCtrlCycleTime  : TIME := T#0ms;
    tTaskCycleTime  : TIME := T#0ms;
    fKp             : FLOAT := 0;
    fTheta          : FLOAT := 0;
    tT0             : TIME := T#0ms;
END_STRUCT
END_TYPE
```

| 名称 | 类型 | 描述 |
|---|---|---|
| tCtrlCycleTime | TIME | 处理控制回路的循环时间。该时间必须大于或等于 **TaskCycleTime**。功能块使用该输入值在内部计算是否需要在当前周期内更新状态和输出值。 |
| tTaskCycleTime | TIME | 调用功能块的循环时间。如果在每个循环中都调用该功能块，这相当于调用任务的任务循环时间。 |
| fKp | FLOAT | 比例增益 |
| fTheta | FLOAT | 阻尼比 |
| tT0 | TIME | 特征时间 |

## 4.2.1.4.12 FB_CTRL_PT3

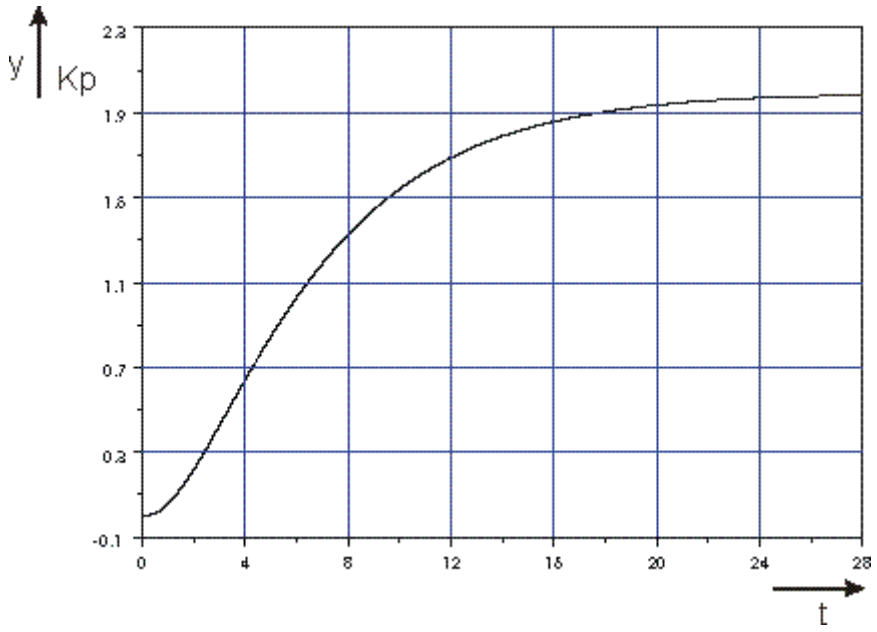



该功能块在功能图中提供了一个非振荡型 PT3 传递单元。

**传递函数**

$$G(s) = K_p \frac{1}{1+T_1 s} \frac{1}{1+T_2 s} \frac{1}{1+T_3 s}$$

### 阶跃响应




### 输入

```
VAR_INPUT
    fIn         : FLOAT;
    fManValue   : FLOAT;
    eMode       : E_CTRL_MODE;
    bHold       : BOOL;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| fIn | FLOAT | PT3 单元的输入值 |
| fManValue | FLOAT | 在手动模式下输出的输入值。 |
| eMode | E_CTRL_MODE | 指定功能块的运行模式 [▶ 157]的输入。 |
| bHold | BOOL | 该输入为 TRUE 时，内部状态（以及输出）将保持当前值不变，且不受输入值影响。 |

### 输出

```
VAR_OUTPUT
    fOut        : FLOAT;
    eState      : E_CTRL_STATE;
    eErrorId    : E_CTRL_ERRORCODES;
    bError      : BOOL;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| fOut | FLOAT | PT3 单元的输出 |
| eState | E_CTRL_STATE | 功能块的状态 |
| eErrorId | E_CTRL_ERRORCODES | 在设置输出 bError 时，提供错误编号 [▶ 157]。 |
| bError | BOOL | 一旦发生错误，则变为 TRUE。 |

### 输入/输出

```
VAR_IN_OUT
    stParams        : ST_CTRL_PT3_PARAMS;
END_VAR
```

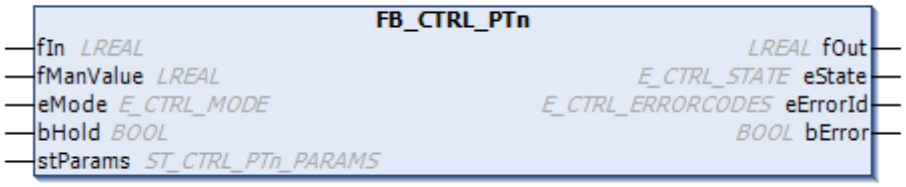
| 名称 | 类型 | 描述 |
|---|---|---|
| stParams | ST_CTRL_PT3_PARAMS | PT3 单元的参数结构 |

`stParams` 由以下元素组成：

```
TYPE
ST_CTRL_PT3_PARAMS :
STRUCT
    tCtrlCycleTime  : TIME := T#0ms;
    tTaskCycleTime  : TIME := T#0ms;
    fKp             : FLOAT := 0;
    tT1             : TIME := T#0ms;
    tT2             : TIME := T#0ms;
    tT3             : TIME := T#0ms;
END_STRUCT
END_TYPE>
```

| 名称 | 类型 | 描述 |
|---|---|---|
| tCtrlCycleTime | TIME | 处理控制回路的循环时间。该时间必须大于或等于 **TaskCycleTime**。功能块使用该输入值在内部计算是否需要在当前周期内更新状态和输出值。 |
| tTaskCycleTime | TIME | 调用功能块的循环时间。如果在每个循环中都调用该功能块，这相当于调用任务的任务循环时间。 |
| fKp | FLOAT | 控制器放大系数/传递系数 |
| tT1 | TIME | 时间常数 T1 |
| tT2 | TIME | 时间常数 T2p |
| tT3 | TIME | 时间常数 T3 |

### 4.2.1.4.13 FB_CTRL_PTn

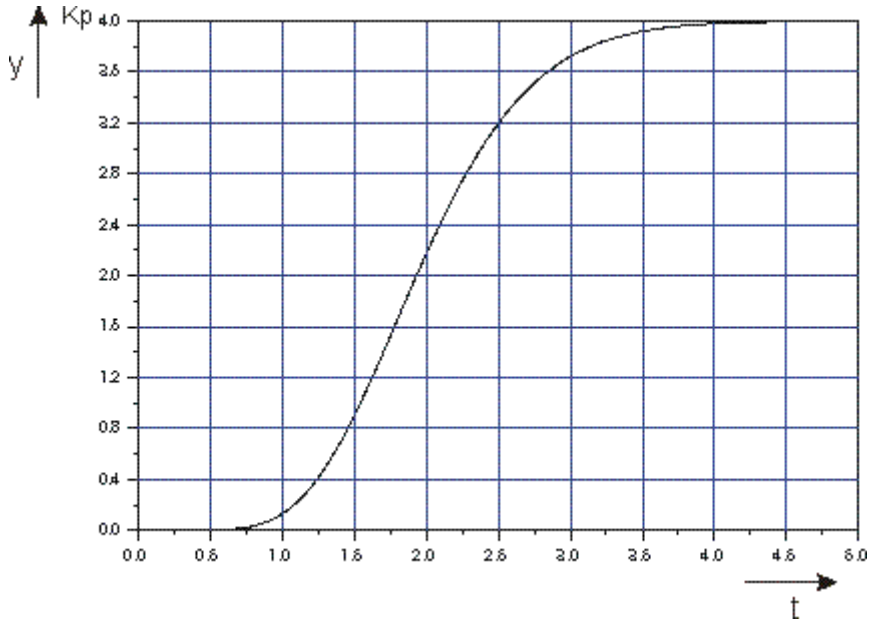



该功能块提供了一个非振荡型 PTn 传递单元，其在功能图中的时间常数相等且“n<= 10”。

**传递函数**

$$G(s) = K_p \frac{1}{(1+T_1 s)^n}$$

**阶跃响应，n=10**




### 输入

```
VAR_INPUT
    fIn         : FLOAT;
    fManValue   : FLOAT;
    eMode       : E_CTRL_MODE;
    bHold       : BOOL;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| fIn | FLOAT | PTn 单元的输入值 |
| fManValue | FLOAT | 在手动模式下输出的输入值。 |
| eMode | E_CTRL_MODE | 指定功能块的运行模式 [▶ 157]的输入。 |
| bHold | BOOL | 该输入为 TRUE 时，内部状态（以及输出）将保持当前值不变，且不受输入值影响。 |

### 输出

```
VAR_OUTPUT
    fOut        : FLOAT;
    eState      : E_CTRL_STATE;
    eErrorId    : E_CTRL_ERRORCODES;
    bError      : BOOL;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| fOut | FLOAT | PTn 单元的输出 |
| eState | E_CTRL_STATE | 功能块的状态 |
| eErrorId | E_CTRL_ERRORCODES | 在设置输出 bError 时，提供错误编号 [▶ 157]。 |
| bError | BOOL | 一旦发生错误，则变为 TRUE。 |

### 输入/输出

```
VAR_IN_OUT
    stParams        : ST_CTRL_PTn_PARAMS;
END_VAR
```

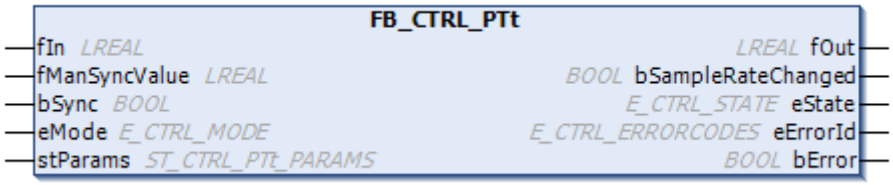
| 名称 | 类型 | 描述 |
|---|---|---|
| stParams | ST_CTRL_PTn_PARAMS | PTn 单元的参数结构 |

`stParams` 由以下元素组成：

```
TYPE
ST_CTRL_PTn_PARAMS :
STRUCT
    tCtrlCycleTime    : TIME := T#0ms;
    tTaskCycleTime    : TIME := T#0ms;
    fKp               : FLOAT := 0;
    tT1               : TIME := T#0ms;
END_STRUCT
END_TYPE
```

| 名称 | 类型 | 描述 |
|---|---|---|
| tCtrlCycleTime | TIME | 处理控制回路的循环时间。该时间必须大于或等于 **TaskCycleTime**。功能块使用该输入值在内部计算是否需要在当前周期内更新状态和输出值。 |
| tTaskCycleTime | TIME | 调用功能块的循环时间。如果在每个循环中都调用该功能块，这相当于调用任务的任务循环时间。 |
| fKp | FLOAT | 控制器放大系数/传递系数 |
| tT1 | TIME | 时间常数 T1 |

## 4.2.1.4.14 FB_CTRL_PTt




该功能块在功能图中提供了一个 PTt 传递单元。

**传递函数**

$$G(s) = K_p \cdot e^{-T_t s}$$

该功能块内部包含一个 500 个元素的数组，用于对输入值进行延迟。在使用 `tCtrlCycleTime` 时，这会导致最大延迟时间为“500 * tCtrlCycleTime”。如果此最大延迟时间仍不足，系统将在内部延长采样时间，以达到所请求的死区时间。但需要注意的是，该过程会增大离散化步长之间的时间间隔。如果已经计算出新的采样时间，这将通过 `bSampleRateChanged` 输出为 TRUE 来指示。

### 输入

```
VAR_INPUT
    fIn             : FLOAT;
    fManSyncValue   : FLOAT;
    bSync           : BOOL;
    eMode           : E_CTRL_MODE;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| fIn | FLOAT | PTt 单元的输入值 |
| fManSyncValue | FLOAT | 可用于设置 PTt 单元的输入值，或在手动模式下从输出端发出的输入值。 |
| bSync | BOOL | 该输入端的上升沿将 PTt 单元设置为值“fManSyncValue”。 |
| eMode | E_CTRL_MODE | 指定功能块的运行模式 [▶ 157]的输入。 |

### 输出

```
VAR_OUTPUT
    fOut                : FLOAT;
    bSampleRateChanged  : BOOL;
    eErrorId            : E_CTRL_ERRORCODES;
    bError              : BOOL;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| fOut | FLOAT | PTt 单元的输出 |
| bSampleRate Changed | BOOL | 表示功能块是否因为用于延迟输入信号的数组没有提供足够的空间而内部降低了采样率的输出。 |
| eErrorId | E_CTRL_ ERRORCODES | 在设置输出 bError 时，提供错误编号 [▶ 157]。 |
| bError | BOOL | 一旦发生错误，则变为 TRUE。 |

### 输入/输出

```
VAR_IN_OUT
    stParams      : ST_CTRL_PTt_PARAMS;
END_VAR
```

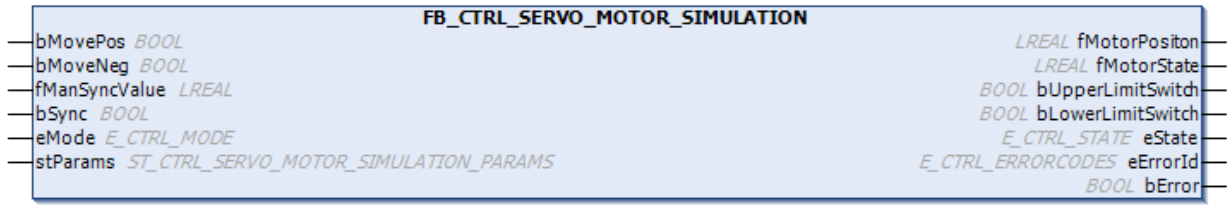
| 名称 | 类型 | 描述 |
|---|---|---|
| stParams | ST_CTRL_PTt_ PARAMS | PTt 单元的参数结构 |

stParams 由以下元素组成：

```
TYPE ST_CTRL_PTt_PARAMS :
STRUCT
    tCtrlCycleTime  : TIME := T#0ms;
    tTaskCycleTime  : TIME := T#0ms;
    fKp             : FLOAT := 0;
    tTt             : TIME := T#0ms;
END_STRUCT
END_TYPE
```

| 名称 | 类型 | 描述 |
|---|---|---|
| tCtrlCycleTime | TIME | 处理控制回路的循环时间。该时间必须大于或等于 **TaskCycleTime**。功能块使用该输入值在内部计算是否需要在当前周期内更新状态和输出值。 |
| tTaskCycleTime | TIME | 调用功能块的循环时间。如果在每个循环中都调用该功能块，这相当于调用任务的任务循环时间。 |
| fKp | FLOAT | 控制器放大系数/传递系数 |
| tTt | TIME | 死区时间 |

## 4.2.1.4.15 FB_CTRL_SERVO_MOTOR_SIMULATION




通过该功能块可以模拟执行器的行为。

### 输出的行为


### 输入

```
VAR_INPUT
    bMovePos        : BOOL;
    bMoveNeg        : BOOL;
    fManSyncValue   : FLOAT;
    bSync           : BOOL;
    eMode           : E_CTRL_MODE;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| bMovePos | BOOL | 使模拟执行器向正方向移动的输入。 |
| bMoveNeg | BOOL | 使模拟执行器向负方向移动的输入。 |
| fManSyncValue | FLOAT | 可用于设置模拟电机位置的输入，或在手动模式下移动的目标值。 |
| bSync | BOOL | 该输入端的上升沿将模拟电机位置设置为值“fManSyncValue”。 |
| eMode | E_CTRL_MODE | 指定功能块的运行模式 [▶ 157]的输入。 |

### 输出

```
VAR_OUTPUT
    fMotorPositon       : FLOAT;
    fMotorState         : FLOAT;
    bUpperLimitSwitch   : BOOL;
    bLowerLimitSwitch   : BOOL;
    eState              : E_CTRL_STATE;
    eErrorId            : E_CTRL_ERRORCODES;
    bError              : BOOL;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| fMotorPositon | FLOAT | 范围 [ fMovingRangeMin ... fMovingRangeMax ] 内的模拟电机位置 |
| fMotorState | FLOAT | 范围 [ 0 … 100.0 ] 内的模拟电机位置 |
| bUpperLimit Switch | BOOL | 执行器正限位点的模拟限位开关 |

| 名称 | 类型 | 描述 |
|---|---|---|
| bLowerLimit Switch | BOOL | 执行器负限位点的模拟限位开关。 |
| eState | E_CTRL_ STATE | 功能块的状态 |
| eErrorId | E_CTRL_ERRORC ODES | 在设置输出 bError 时，提供错误编号 [▶ 157]。 |
| bError | BOOL | 一旦发生错误，则变为 TRUE。 |

### 输入/输出

```
VAR_IN_OUT
    stParams        : ST_CTRL_SERVO_MOTOR_SIMULATION_PARAMS;
END_VAR
```

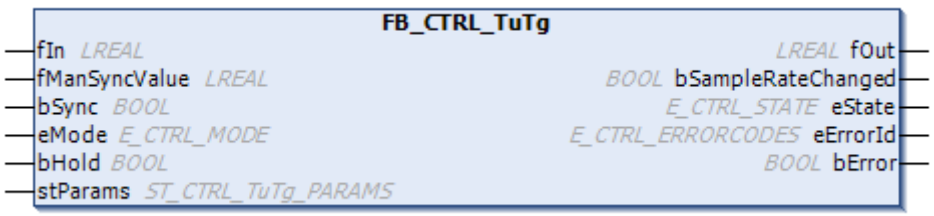
| 名称 | 类型 | 描述 |
|---|---|---|
| stParams | ST_CTRL_ SERVO_MOTOR_ SIMULATION_ PARAMS | 功能块的参数结构 |

stParams 由以下元素组成：

```
TYPE ST_CTRL_SERVO_MOTOR_SIMULATION_PARAMS:
STRUCT
    tCtrlCycleTime    : TIME := T#0ms;
    tTaskCycleTime    : TIME := T#0ms;
    fMovingRangeMin   : FLOAT := 0;
    fMovingRangeMax   : FLOAT := 0;
    tMovingTime       : TIME := T#0ms;
    tDeadTime         : TIME := T#0ms;
END_STRUCT
END_TYPE
```

| 名称 | 类型 | 描述 |
|---|---|---|
| tCtrlCycleTime | TIME | 处理控制回路的循环时间。该时间必须大于或等于 **TaskCycleTime**。功能块使用该输入值在内部计算是否需要在当前周期内更新状态和输出值。 |
| tTaskCycleTime | TIME | 调用功能块的循环时间。如果在每个循环中都调用该功能块，这相当于调用任务的任务循环时间。 |
| fMovingRange Min | FLOAT | 模拟执行器的最小位置 |
| fMovingRange Max | FLOAT | 模拟执行器的最大位置 |
| tMovingTime | TIME | 模拟执行器从一个限位点移动到另一个限位点所需的时间。 |
| tDeadTime | TIME | 模拟执行器的死区时间 |

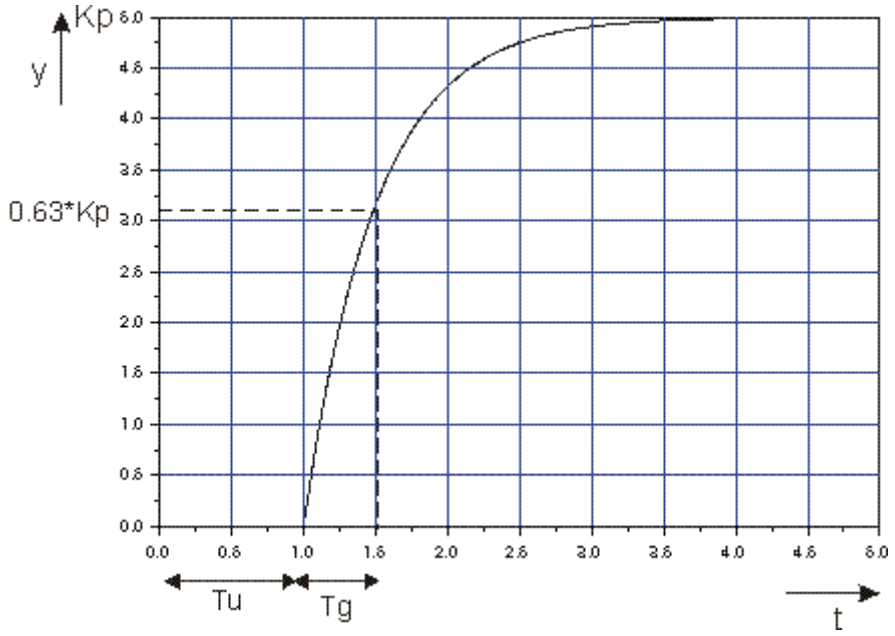
## 4.2.1.4.16 FB_CTRL_TuTg




该功能块在功能图中提供了一个 TuTg 传递单元（死区时间延迟单元）。

**传递函数**

$$G(s) = K_p \frac{1}{1+T_g s} \cdot e^{-T_u s}$$




### 输入

```
VAR_INPUT
    fIn              : FLOAT;
    fManSyncValue    : FLOAT;
    bSync            : BOOL;
    eMode            : E_CTRL_MODE;
    bHold            : BOOL;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| fIn | FLOAT | TuTg 单元的输入值 |
| fManSyncValue | FLOAT | 可用于设置 TuTg 单元的输入值，或在手动模式下从输出端发出的输入值。 |
| bSync | BOOL | 该输入端的上升沿将 TuTg 单元设置为值“fManSyncValue”。 |
| eMode | E_CTRL_MODE | 指定功能块的运行模式 [▶ 157]的输入。 |
| bHold | FLOAT | 该输入为 TRUE 时，内部状态（以及输出）将保持当前值不变，且不受输入值影响。 |

### 输出

```
VAR_OUTPUT
    fOut                 : FLOAT;
    bSampleRateChanged   : BOOL;
    eState               : E_CTRL_STATE;
    eErrorId             : E_CTRL_ERRORCODES;
    bError               : BOOL;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| fOut | FLOAT | TuTg 单元的输出 |
| bSampleRate Changed | BOOL | 表示功能块是否因为用于延迟输入信号的数组没有提供足够的空间而内部降低了采样率的输出。 |
| eState | E_CTRL_STATE | 功能块的状态 |
| eErrorId | E_CTRL_ ERRORCODES | 在设置输出 bError 时，提供错误编号 [▶ 157]。 |

| 名称 | 类型 | 描述 |
|---|---|---|
| bError | BOOL | 一旦发生错误，则变为 TRUE。 |

### 输入/输出

```
VAR_IN_OUT
    stParams       : ST_CTRL_TuTg_PARAMS;
END_VAR
```

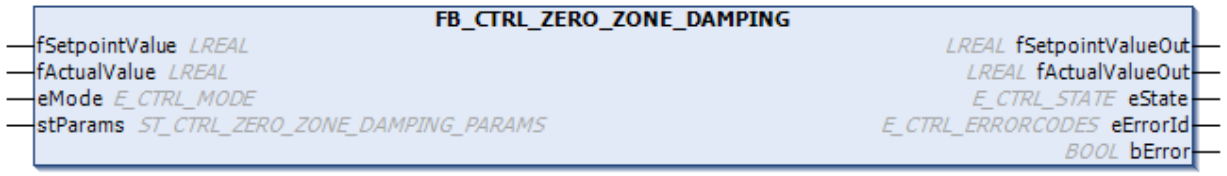
| 名称 | 类型 | 描述 |
|---|---|---|
| stParams | ST_CTRL_TuTg_PARAMS | TuTg 单元的参数结构 |

stParams 由以下元素组成：

```
TYPE ST_CTRL_TuTg_PARAMS :
STRUCT
    tCtrlCycleTime     : TIME  := T#0ms;
    tTaskCycleTime     : TIME  := T#0ms;
    fKp                : FLOAT := 0;
    tTu                : TIME  := T#0ms;
    tTg                : TIME  := T#0ms;
END_STRUCT
END_TYPE
```

| 名称 | 类型 | 描述 |
|---|---|---|
| tCtrlCycleTime | TIME | 处理控制回路的循环时间。该时间必须大于或等于 **TaskCycleTime**。功能块使用该输入值在内部计算是否需要在当前周期内更新状态和输出值。 |
| tTaskCycleTime | TIME | 调用功能块的循环时间。如果在每个循环中都调用该功能块，这相当于调用任务的任务循环时间。 |
| fKp | FLOAT | 控制器放大系数/传递系数 |
| tTu | TIME | 死区时间 |
| tTg | TIME | 时间常数 |

## 4.2.1.4.17 FB_CTRL_ZERO_ZONE_DAMPING

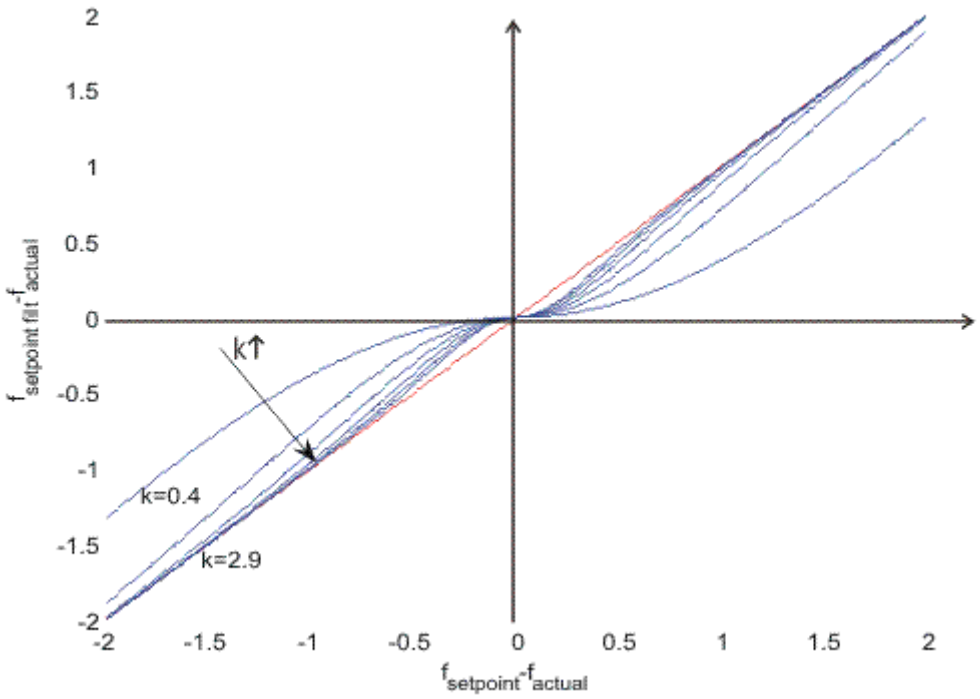



该功能块可实现零点阻尼，以便在 | 实际值 - 设定值 | < ε 的范围内最大限度地减少控制干预。

**时间范围内的传递行为**

$$f_{setpoint\_out} = (f_{setpoint\_in} - f_{actual\_in}) \cdot \tanh(|f_{setpoint\_in} - f_{actual\_in}| \cdot k_{damping}) + f_{actual\_in}$$

$$f_{actual\_out} = f_{setpoint\_in}$$




### 输入

```
VAR_INPUT
    fSetpointValue     : FLOAT;
    fActualValue       : FLOAT;
    eMode              : E_CTRL_MODE;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| fSetpointValue | FLOAT | 控制变量的设定点 |
| fActualValue | FLOAT | 控制变量的实际值 |
| eMode | E_CTRL_MODE | 指定功能块的运行模式 [▶ 157]的输入。 |

### 输出

```
VAR_OUTPUT
    fSetpointValueOut     : FLOAT;
    fActualValueOut       : FLOAT;
    eState                : E_CTRL_STATE;
    eErrorId              : E_CTRL_ERRORCODES;
    bError                : BOOL;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| fSetpointValue Out | FLOAT | 送至控制器的滤波设定点 |
| fActualValueOut | FLOAT | 送至控制器的实际值 |
| eState | E_CTRL_STATE | 功能块的状态 |
| eErrorId | E_CTRL_ERRORC ODES | 在设置输出 bError 时，提供错误编号 [▶ 157]。 |
| bError | BOOL | 一旦发生错误，则变为 TRUE。 |

### 输入/输出

```
VAR_IN_OUT
    stParams          : ST_CTRL_ZERO_ZONE_DAMPING_PARAMS;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| stParams | ST_CTRL_ZERO_ZONE_DAMPING_PARAMS | 传递单元的参数结构 |

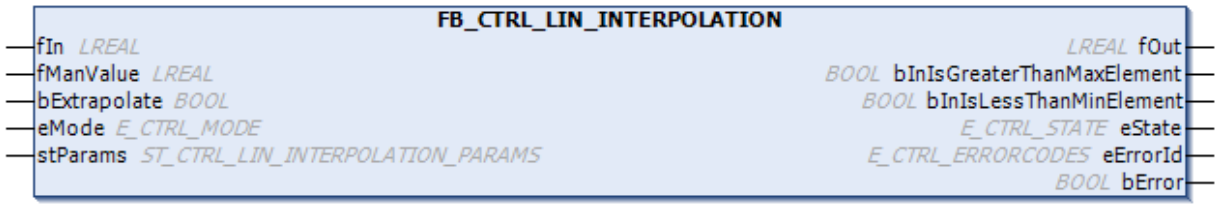
stParams 由以下元素组成：

```
TYPE
ST_CTRL_PI_PST_CTRL_ZERO_ZONE_DAMPING_PARAMS :
STRUCT
    tCtrlCycleTime          : TIME  := T#0ms;
    tTaskCycleTime          : TIME  := T#0ms;
    fDampingCoefficient     : FLOAT := 0.0;
END_STRUCT
END_TYPE
```

| 名称 | 类型 | 描述 |
|---|---|---|
| tCtrlCycleTime | TIME | 处理控制回路的循环时间。该时间必须大于或等于 **TaskCycleTime**。功能块使用该输入值在内部计算是否需要在当前周期内更新状态和输出值。 |
| tTaskCycleTime | TIME | 调用功能块的循环时间。如果在每个循环中都调用该功能块，这相当于调用任务的任务循环时间。 |
| fDampingCoefficient | FLOAT | 该参数相当于传递函数中的$k_{阻尼}$。 |

### 4.2.1.5 插补

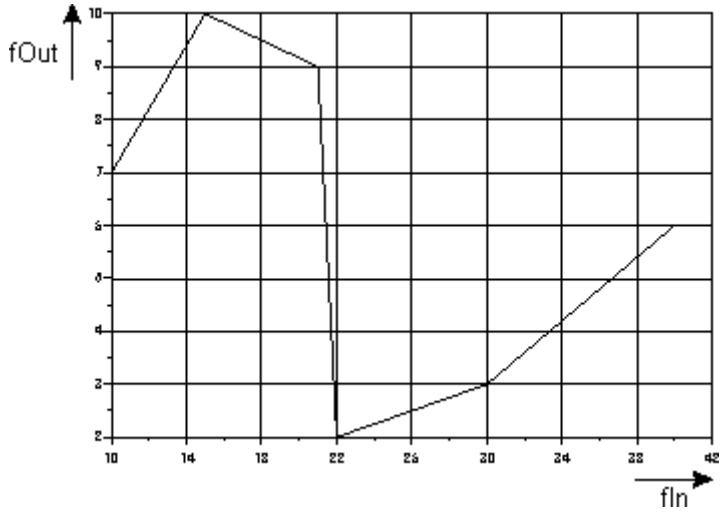
#### 4.2.1.5.1 FB_CTRL_LIN_INTERPOLATION




该功能块根据采样点表，通过线性插值来获得对应的数值。

**输出的行为**

| fIn | fOut |
|---|---|
| arrTable[1,1] := 10; | arrTable[1,2] := 7; |
| arrTable[2,1] := 15; | arrTable[2,2] := 10; |
| arrTable[3,1] := 21; | arrTable[3,2] := 9; |
| arrTable[4,1] := 22; | arrTable[4,2] := 2; |
| arrTable[5,1] := 30; | arrTable[5,2] := 3; |
| arrTable[6,1] := 40; | arrTable[6,2] := 6; |




### 输入

```
VAR_INPUT
    fIn             : FLOAT;
    fManValue       : FLOAT;
    bExtrapolate    : BOOL;
    eMode           : E_CTRL_MODE;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| fIn | FLOAT | 插值功能块的输入值 |
| fManValue | FLOAT | 在手动模式下输出的输入值。 |
| bExtrapolate | BOOL | 如果该输入为 FALSE，则在超出表格限值时，将输出最后一个插值点的值。然而，如果为 TRUE，则将根据最后两个插值点执行外推。 |
| eMode | E_CTRL_MODE | 指定功能块的运行模式 [▶ 157]的输入。 |

### 输出

```
VAR_OUTPUT
    fOut                            : FLOAT;
    bInIsGreaterThanMaxElement      : BOOL;
    bInIsLessThanMinElement         : BOOL;
    eState                          : E_CTRL_STATE;
    eErrorId                        : E_CTRL_ERRORCODES;
    bError                          : BOOL;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| fOut | FLOAT | 线性插值的表值 |
| bInIsGreaterThanMaxElement | BOOL | 该输出为 TRUE 时，表示输入值大于最大插值点。 |
| bInIsLessThanMinElement | BOOL | 该输出为 TRUE 时，表示输入值小于最小插值点。 |
| eState | E_CTRL_STATE | 功能块的状态 |
| eErrorId | E_CTRL_ERRORCODES | 在设置输出 bError 时，提供错误编号 [▶ 157]。 |
| bError | BOOL | 一旦发生错误，则变为 TRUE。 |

### 输入/输出

```
VAR_IN_OUT
    stParams        : ST_CTRL_LIN_INTERPOLATION_PARAMS;
END_VAR
```

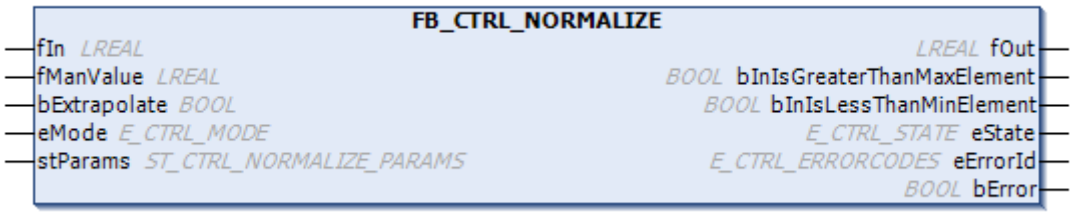
| 名称 | 类型 | 描述 |
|---|---|---|
| stParams | ST_CTRL_LIN_INTERPOLATION_PARAMS | 插值单元的参数结构 |

`stParams` 由以下元素组成：

```
TYPE ST_CTRL_2POINT_PARAMS :
STRUCT
    tCtrlCycleTime          : TIME := T#0ms;
    tTaskCycleTime          : TIME := T#0ms;
    pDataTable_ADR          : POINTER TO FLOAT := 0;
    nDataTable_SIZEOF       : UINT     := 0;
    nDataTable_NumberOfRows : UINT     := 0;
END_STRUCT
END_TYPE
```

| 名称 | 类型 | 描述 |
|---|---|---|
| tCtrlCycleTime | TIME | 处理控制回路的循环时间。该时间必须大于或等于 **TaskCycleTime**。功能块使用该输入值在内部计算是否需要在当前周期内更新状态和输出值。 |
| tTaskCycleTime | TIME | 调用功能块的循环时间。如果在每个循环中都调用该功能块，这相当于调用任务的任务循环时间。 |
| pDataTable_ADR | POINTER TO FLOAT | 要进行线性插值的 n x 2 数组的地址。 |
| nDataTable_SIZEOF | UINT | n x 2 数组的大小 |
| nDataTable_NumberOfRows | UINT | 数组中的行数 |

## 4.2.1.5.2 FB_CTRL_NORMALIZE

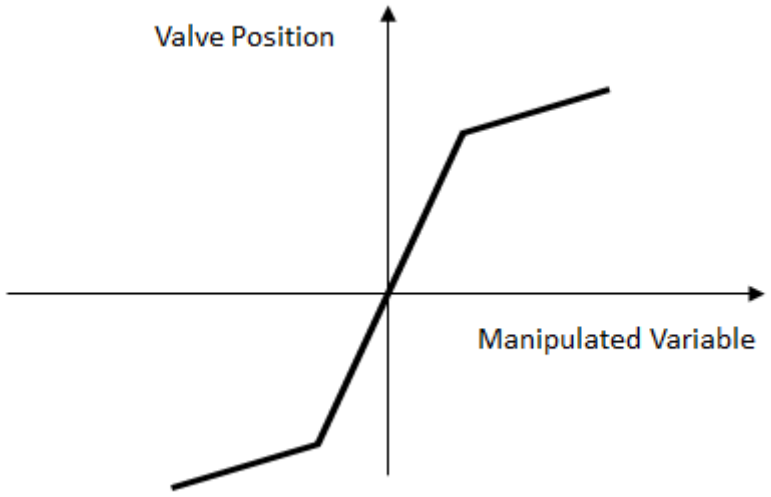



该功能块可用于借助反向特性曲线对非线性传递单元进行线性化处理。

要进行线性化处理的传递单元的特性曲线存储在与该功能块相关的表中。该功能块以此计算反向特性曲线，并以此进行线性化处理。

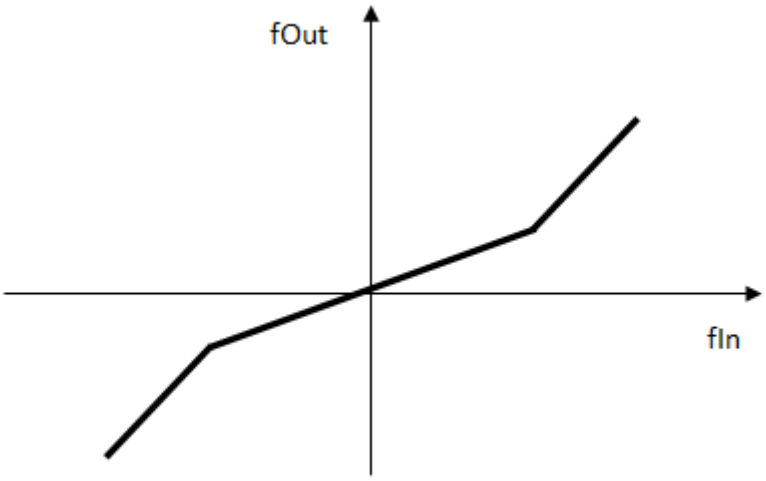
### 例程

该表中存储了以下阀门特性曲线，其中包含 4 个插值点。

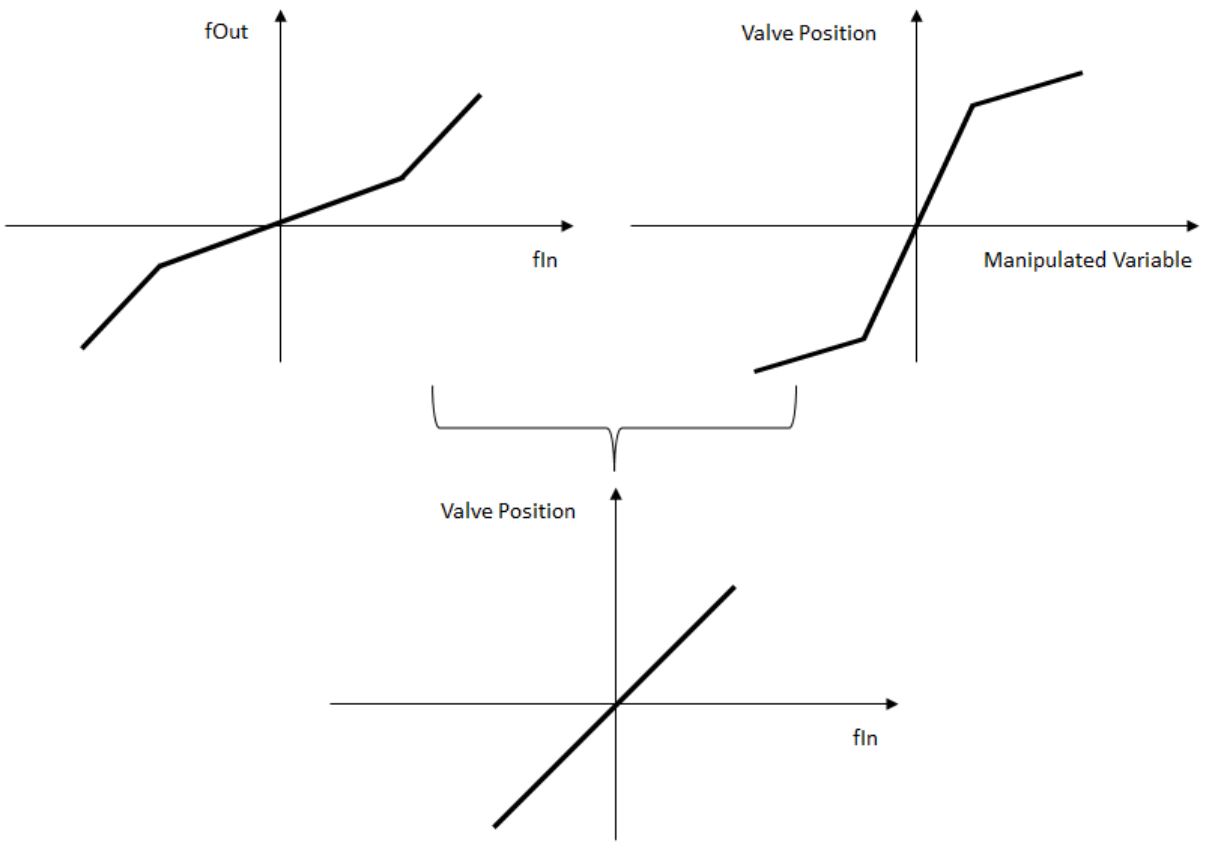



| 控制值 | 阀门位置 |
|---|---|
| arrTable[1,1] := -6; | arrTable[1,2] := -100; |
| arrTable[2,1] := -1; | arrTable[2,2] := -70; |
| arrTable[3,1] := 1; | arrTable[3,2] := 70; |
| arrTable[4,1] := 6; | arrTable[4,2] := 100; |

根据该特性曲线计算反向特性曲线：




在理想情况下，将这两条特性曲线串联起来会产生线性传递行为。




### 输入

```
VAR_INPUT
    fIn           : FLOAT;
    fManValue     : FLOAT;
    bExtrapolate  : BOOL;
    eMode         : E_CTRL_MODE;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| fIn | FLOAT | 输入值 |
| fManValue | FLOAT | 在手动模式下输出的输入值。 |
| bExtrapolate | BOOL | 如果该输入为 FALSE，则在超出表格限值时，将输出最后一个插值点的值。然而，如果为 TRUE，则将根据最后两个插值点执行外推。 |
| eMode | E_CTRL_MODE | 指定功能块的运行模式 [▶ 157]的输入。 |

### 输出

```
VAR_OUTPUT
    fOut                          : FLOAT;
    bInIsGreaterThanMaxElement    : BOOL;
    bInIsLessThanMinElement       : BOOL;
    eState                        : E_CTRL_STATE;
    eErrorId                      : E_CTRL_ERRORCODES;
    bError                        : BOOL;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| fOut | FLOAT | 线性插值的表值 |
| bInIsGreaterThanMaxElement | BOOL | 该输出为 TRUE 时，表示输入值大于最大插值点。 |

| 名称 | 类型 | 描述 |
|---|---|---|
| bInIsLessThanMinElement | BOOL | 该输出为 TRUE 时，表示输入值小于最小插值点。 |
| eState | E_CTRL_STATE | 功能块的状态 |
| eErrorId | E_CTRL_ERRORCODES | 在设置输出 bError 时，提供错误编号 [▶ 157]。 |
| bError | BOOL | 一旦发生错误，则变为 TRUE。 |

### 输入/输出

```
VAR_IN_OUT
    stParams        : ST_CTRL_NORMALIZE_PARAMS;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| stParams | ST_CTRL_NORMALIZE_PARAMS | 功能块的参数结构 |

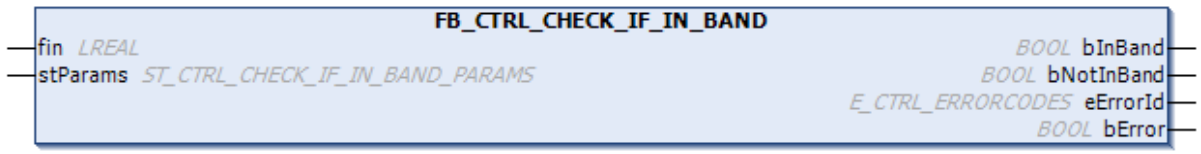
stParams 由以下元素组成：

```
TYPE ST_CTRL_NORMALIZE_PARAMS:
STRUCT
    tCtrlCycleTime           : TIME := T#0ms;
    tTaskCycleTime           : TIME := T#0ms;
    pDataTable_ADR           : POINTER TO FLOAT := 0;
    nDataTable_SIZEOF        : UINT     := 0;
    nDataTable_NumberOfRows  : UINT     := 0;
END_STRUCT
END_TYPE
```

| 名称 | 类型 | 描述 |
|---|---|---|
| tCtrlCycleTime | TIME | 处理控制回路的循环时间。该时间必须大于或等于 **TaskCycleTime**。功能块使用该输入值在内部计算是否需要在当前周期内更新状态和输出值。 |
| tTaskCycleTime | TIME | 调用功能块的循环时间。如果在每个循环中都调用该功能块，这相当于调用任务的任务循环时间。 |
| pDataTable_ADR | POINTER TO FLOAT | 要进行线性插值的 n x 2 数组的地址。 |
| nDataTable_SIZEOF | UINT | n x 2 数组的大小 |
| nDataTable_NumberOfRows | UINT | 数组中的行数 |

## 4.2.1.6 监控/报警

### 4.2.1.6.1 FB_CTRL_CHECK_IF_IN_BAND




该功能块监控输入值是否在范围 [fMin ... fMax] 之内，即是否满足

$$fMin \leq fIn \leq fMax$$

该不等式。

### 输入

```
VAR_INPUT
    fIn     : FLOAT;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| fIn | FLOAT | 受监控的输入值。 |

### 输出

```
VAR_OUTPUT
    bInBand     : BOOL;
    bNotInBand  : BOOL;
    eErrorId    : E_CTRL_ERRORCODES;
    bError      : BOOL;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| bInBand | BOOL | 该输出为 TRUE 时，表示输入值在指定范围内。 |
| bNotInBand | BOOL | 该输出为 TRUE 时，表示输入值**不**在指定范围内。 |
| eErrorId | E_CTRL_ERRORCODES | 在设置输出 `bError` 时，提供错误编号 [▶ 157]。 |
| bError | BOOL | 一旦发生错误，则变为 TRUE。 |

### 输入/输出

```
VAR_IN_OUT
    stParams        : ST_CTRL_CHECK_IF_IN_BAND_PARAMS;
END_VAR
```

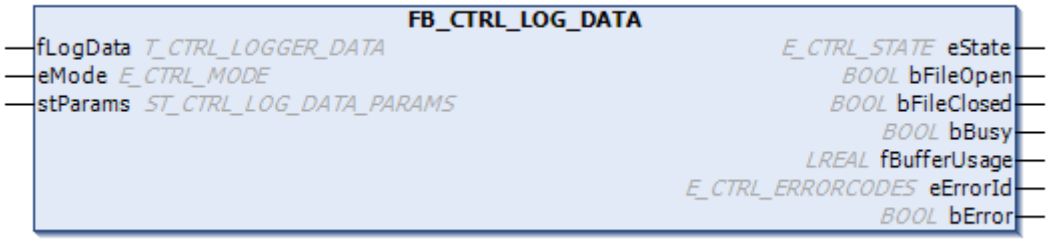
| 名称 | 类型 | 描述 |
|---|---|---|
| stParams | ST_CTRL_CHECK_IF_IN_BAND_PARAMS | 功能块的参数结构 |

**stParams** 由以下元素组成：

```
TYPE ST_CTRL_CHECK_IF_IN_BAND_PARAMS:
STRUCT
    tCtrlCycleTime  : TIME := T#0ms;
    tTaskCycleTime  : TIME := T#0ms;
    fMin            : FLOAT;
    fMax            : FLOAT;
END_STRUCT
END_TYPE
```

| 名称 | 类型 | 描述 |
|---|---|---|
| tCtrlCycleTime | TIME | 处理控制回路的循环时间。该时间必须大于或等于 **TaskCycleTime**。功能块使用该输入值在内部计算是否需要在当前周期内更新状态和输出值。 |
| tTaskCycleTime | TIME | 调用功能块的循环时间。如果在每个循环中都调用该功能块，这相当于调用任务的任务循环时间。 |
| fMin | FLOAT | 范围的下限 |
| fMax | FLOAT | 范围的上限 |

## 4.2.1.6.2 FB_CTRL_LOG_DATA




该功能块允许创建 *.csv 格式（逗号分隔值）的日志文件，其中最多可记录 10 个变量。用户指定的列标题将写入该文件的第一行。输入数据将以相等的时间间隔写入后续行中。各个条目之间以逗号分隔。在 tLogCycleTime 参数中指定条目之间的时间间隔。例如，如果选择 tLogCycleTime := T\#2s，那么在文件中每 2s 就会记录一个条目。生成的文件可以通过电子表格程序等进行分析。

日志条目的时间戳（以 s 为单位）存储在文件的第一列中。其他列包含功能块输入 fLogData 的数据。

当模式被设置为 eCTRL_MODE_ACTIVE 时，系统将打开日志文件并向文件中写入条目。在功能块的模式被设置为 eCTRL_MODE_PASSIVE 之前，文件一直处于打开状态。

在尝试分析日志文件之前，必须先通过切换到 eCTRL_MODE_PASSIVE 来关闭该文件。如果不执行此操作，可能会导致部分条目无法写入文件。

该功能块可在使用或不使用外部缓冲区的情况下运行。

| | |
|---|---|
| **在不使用外部缓冲区的情况下运行：** | 当开始记录行数据时，bBusy 输出为 TRUE。在 bBusy 输出再次变为 FALSE 之前，系统将不会记录后续数据集。fBufferUsage 输出表示内部缓冲区的满载程度。 |
| **在使用外部缓冲区的情况下运行：** | 用户必须创建一个大于 255 字节且类型为 ARRAY OF BYTES 的缓冲区。<br>各条消息将暂时存储在外部缓冲区中，该缓冲区将以最快速度写入文件。<br>fBufferUsage 输出表示缓冲区的满载程度。如果缓冲区发生溢出，则将停止该功能块并输出错误信息。 |

如果进行参数设置的缓冲区地址和缓冲区大小大于零，则使用外部缓冲区。如果没有外部缓冲区，则使用大小为 255 字节的内部缓冲区。

### 输入

```
VAR_INPUT
    fLogData   : T_CTRL_LOGGER_DATA;
    eMode      : E_CTRL_MODE;
END_VAR

VAR_GLOBAL CONSTANT
    nCTRL_LOGGER_DATA_ARRAY_SIZE  : UINT := 10;
END_VAR

TYPE T_CTRL_LOGGER_DATA :
    ARRAY [1..nCTRL_LOGGER_DATA_ARRAY_SIZE] OF FLOAT;
END_TYPE
```

| 名称 | 类型 | 描述 |
|---|---|---|
| fLogData | T_CTRL_LOGGER_DATA | 包含要写入日志文件的值的数组。 |
| eMode | E_CTRL_MODE | 指定功能块的运行模式 [▶ 157]的输入。 |

### 输出

```
VAR_OUTPUT
    eState          : E_CTRL_STATE;
    bFileOpen       : BOOL
    bFileClosed     : BOOL
    fBufferUsage    : FLOAT
    bBusy           : BOOL
    eErrorId        : E_CTRL_ERRORCODES;
    bError          : BOOL;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| eState | E_CTRL_STATE | 功能块的状态 |
| bFileOpen | BOOL | 该输出为 TRUE 时，表示文件已成功打开。 |
| bFileClosed | BOOL | 该输出为 TRUE 时，表示文件已成功关闭。 |
| fBufferUsage | FLOAT | 外部缓冲区的当前填充水平（以百分比表示） |
| bBusy | BOOL | 该输出为 TRUE 时，表示正在记录一行数据。 |
| eErrorId | E_CTRL_ERRORCODES | 在设置输出 bError 时，提供错误编号 [▶ 157]。 |
| bError | BOOL | 一旦发生错误，则变为 TRUE。 |

### 输入/输出

```
VAR_IN_OUT
    stParams        : ST_CTRL_LOG_DATA_PARAMS;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| stParams | ST_CTRL_LOG_DATA_PARAMS | 日志记录功能块的参数结构 |

stParams 由以下元素组成：

```
TYPE ST_CTRL_LOG_DATA_PARAMS:
STRUCT
    tLogCycleTime             : TIME := T#0ms;
    tTaskCycleTime            : TIME := T#0ms;
    sFileName                 : STRING;
    sNetId                    : T_AmsNetId := '';
    tFileOperationTimeou      : TIME := T#3s;
    nNumberOfColumn           : INT(1..10);
    arColumnHeadings          : ARRAY [1..10] OF STRING;
    bAppendData               : BOOL := FALSE;
    bWriteTimeStamps          : BOOL := TRUE;
    bWriteColumnHeadings      : BOOL := TRUE;
    bWriteAbsoluteTimeStamps  : BOOL := FALSE
    pLogBuffer_ADR            : POINTER TO BYTE;
    nLogBuffer_SIZEOF         : UDINT;
END_STRUCT
END_TYPE
```

| 名称 | 类型 | 描述 |
|---|---|---|
| tLogCycleTime | TIME | 将条目写入日志文件的循环时间。该时间必须大于或等于 **TaskCycleTime**。 |
| tTaskCycleTime | TIME | 调用功能块的循环时间。如果在每个循环中都调用该功能块，这相当于调用任务的任务循环时间。 |
| sFileName | STRING | 日志文件的名称和路径，例如：d:\Logfile.csv |
| sNetId | T_AmsNetId | 日志文件将写入到此处指定的 NetId 的系统中。 |
| tFileOperation Timeou | TIME | 所有文件操作的超时时间 |
| nNumberOf Column | INT | 写入文件的列数（最多 10 列） |
| arColumn Headings | ARRAY | 包含列标题的字符串数组。 |

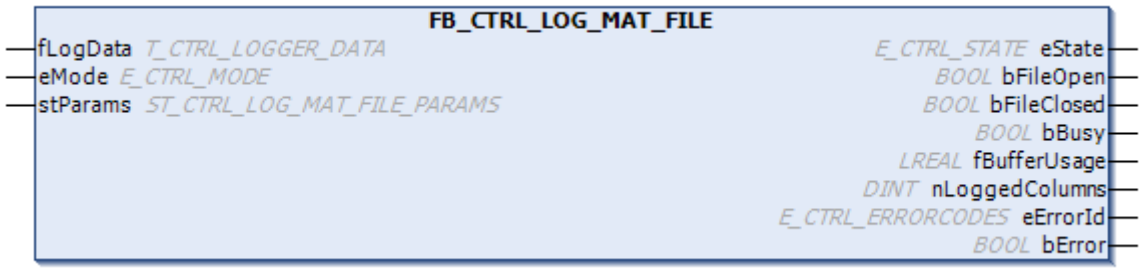
| 名称 | 类型 | 描述 |
|---|---|---|
| bAppendData | BOOL | 如果该参数为 TRUE，则再次打开文件时会添加新的数据集。否则，文件将被覆盖而**不会进行查询**，这将删除任何已经存在的内容。 |
| bWriteTime Stamps | BOOL | 如果该参数被设置为 TRUE，则测量的时间戳将被写入文件的第一列中。 |
| bWriteColumn Headings | BOOL | 如果该参数被设置为 TRUE，则列标题将被写入文件的第一行中。 |
| bWriteAbsolute TimeStamps | BOOL | 如果该参数被设置为 TRUE，则会将本地 NT 时间用作时间戳。在这种情况下，最小 `LogCycleTime` 为 5s! |
| pLogBuffer_ADR | POINTER TO BYTE | 外部 `LogBuffer` 的地址。要检测缓冲区，地址不得为 0。 |
| nLogBuffer_ SIZEOF | UDINT | `LogBuffer` 的大小。缓冲区必须是 ARRAY OF BYTE 类型，并且至少包含 256 个元素。借助于 `fBufferUsage` 输出可以对缓冲区的大小进行优化。 |

**注意**

**文件处理期间出错**

只有在文件关闭（bFileClosed = TRUE）时，才能更改参数集。否则，在文件处理期间可能会出错。

### 4.2.1.6.3 FB_CTRL_LOG_MAT_FILE




该功能块允许创建 Matlab 5（*.mat）格式的日志文件，其中最多可记录 10 个量级。

在文件中创建了两个变量，一个双精度数组，一个单元数组。记录的量级逐行记录在双精度数组中。各行的标识符存储在单元数组中。用户可以在功能块的参数结构中指定双精度数组所用的名称。单元数组的名称源自双精度数组的名称，方法为在变量名称后附加“_Info”。

输入数据将以相等的时间间隔写入数据数组的列中。相关条目的时间戳（以 s 为单位）可以存储在第一列中。在 `tLogCycleTime` 参数中指定条目之间的时间间隔。例如，如果选择“tLogCycleTime := T\#2s”，那么在文件中每 2s 就会记录一个条目。

当模式被设置为 `eCTRL_MODE_ACTIVE` 时，系统将打开日志文件并向文件中写入条目。在功能块的模式被设置为 `eCTRL_MODE_PASSIVE` 之前，文件一直处于打开状态。

在尝试分析日志文件之前，必须先通过切换到 `eCTRL_MODE_PASSIVE` 来关闭该文件。如果不执行此操作，可能会导致部分条目无法写入文件，从而导致文件内容不一致。

该功能块可在使用或不使用外部缓冲区的情况下运行。

| | |
|---|---|
| **在不使用外部缓冲区的情况下运行：** | 当开始记录行数据时，bBusy 输出为 TRUE。在 bBusy 输出再次变为 FALSE 之前，系统将不会记录后续数据集。`fBufferUsage` 输出表示内部缓冲区的满载程度。 |

| | |
|---|---|
| **在使用外部缓冲区的情况下运行：** | 用户必须创建一个大于 1500 字节且类型为 ARRAY OF BYTES 的缓冲区。<br>各条消息将暂时存储在外部缓冲区中，该缓冲区将以最快速度写入文件。<br>fBufferUsage 输出表示缓冲区的满载程度。如果发生缓冲区溢出，则将停止该功能块并输出错误信息。 |

如果进行参数设置的缓冲区地址和缓冲区大小大于零，则使用外部缓冲区。如果没有外部缓冲区，则使用大小为 1500 字节的内部缓冲区。

### 输入

```
VAR_INPUT
    fLogData   : T_CTRL_LOGGER_DATA;
    eMode      : E_CTRL_MODE;
END_VAR
VAR_GLOBAL CONSTANT
    nCTRL_LOGGER_DATA_ARRAY_SIZE :UINT := 10;
END_VAR
TYPE
    T_CTRL_LOGGER_DATA :ARRAY [1..nCTRL_LOGGER_DATA_ARRAY_SIZE]
OF FLOAT;
END_TYPE
```

| 名称 | 类型 | 描述 |
|---|---|---|
| fLogData | T_CTRL_LOGGER_DATA | 包含要写入日志文件的值的数组。 |
| eMode | E_CTRL_MODE | 指定功能块的运行模式 [▶ 157]的输入。 |

### 输出

```
VAR_OUTPUT
    eState          : E_CTRL_STATE;
    bFileOpen       : BOOL
    bFileClosed     : BOOL
    fBufferUsage    : FLOAT
    nLoggedColumns  : DINT;
    bBusy           : BOOL
    eErrorId        : E_CTRL_ERRORCODES;
    bError          : BOOL;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| eState | E_CTRL_STATE | 功能块的状态 |
| bFileOpen | BOOL | 该输出为 TRUE 时，表示文件已成功打开。 |
| bFileClosed | BOOL | 该输出为 TRUE 时，表示文件已成功关闭。 |
| fBufferUsage | FLOAT | 外部缓冲区的当前填充水平（以百分比表示） |
| nLogged Columns | DINT | 写入文件的列数 |
| bBusy | BOOL | 该输出为 TRUE 时，表示正在记录一行数据。 |
| eErrorId | E_CTRL_ERRORCODES | 在设置输出 bError 时，提供错误编号 [▶ 157]。 |
| bError | BOOL | 一旦发生错误，则变为 TRUE。 |

### 输入/输出

```
VAR_IN_OUT
    stParams        : ST_CTRL_LOG_MAT_FILE_PARAMS;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| stParams | ST_CTRL_LOG_MAT_FILE_PARAMS | 日志记录功能块的参数结构 |

`stParams` 由以下元素组成：

```
TYPE ST_CTRL_LOG_MAT_FILE_PARAMS:
STRUCT
    tLogCycleTime          : TIME := T#0ms;
    tTaskCycleTime         : TIME := T#0ms;
    sFileName              : STRING;
    nNumberOfRows          : INT(1..10);
    sMatrixName            : STRING;
    arRowDescription       : ARRAY [1..10] OF STRING(25);
    bWriteTimeStamps       : BOOL := TRUE;
    bWriteRowDescription   : BOOL := TRUE;
    pLogBuffer_ADR         : POINTER TO
     ST_CTRL_MULTIPLE_PWM_OUT_DATA;
    nLogBuffer_SIZEOF        : UDINT;
END_STRUCT
END_TYPE
```

| 名称 | 类型 | 描述 |
|---|---|---|
| tLogCycleTime | TIME | 将条目写入日志文件的循环时间。该时间必须大于或等于 **TaskCycleTime**。 |
| tTaskCycleTime | TIME | 调用功能块的循环时间。如果在每个循环中都调用该功能块，这相当于调用任务的任务循环时间。 |
| sFileName | STRING | 日志文件的名称和路径，例如：d:\Logfile.mat |
| nNumberOfRows | INT | 写入文件的列数（最多 10 列）。 |
| sMatrixName | STRING | 数据数组的名称 |
| arRow Description | ARRAY | 包含行标题的字符串数组。 |
| bWriteTime Stamps | BOOL | 如果该参数被设置为 TRUE，则测量的时间戳将被写入数据数组的第一行中。 |
| bWriteRow Description | BOOL | 如果该参数被设置为 TRUE，则会创建一个单元数组，行描述将被写入其中。 |
| pLogBuffer_ADR | POINTER TO ST_CTRL_ MULTIPLE_PWM_ OUT_DATA | 外部 `LogBuffer` 的地址。要检测缓冲区，地址不得为 0。 |
| nLogBuffer_ SIZEOF | UDINT | `LogBuffer` 的大小。缓冲区必须是 `ARRAY OF BYTE` 类型，并且至少包含 1501 个元素。借助于 `fBufferUsage` 输出可以对缓冲区的大小进行优化。 |

**注意**

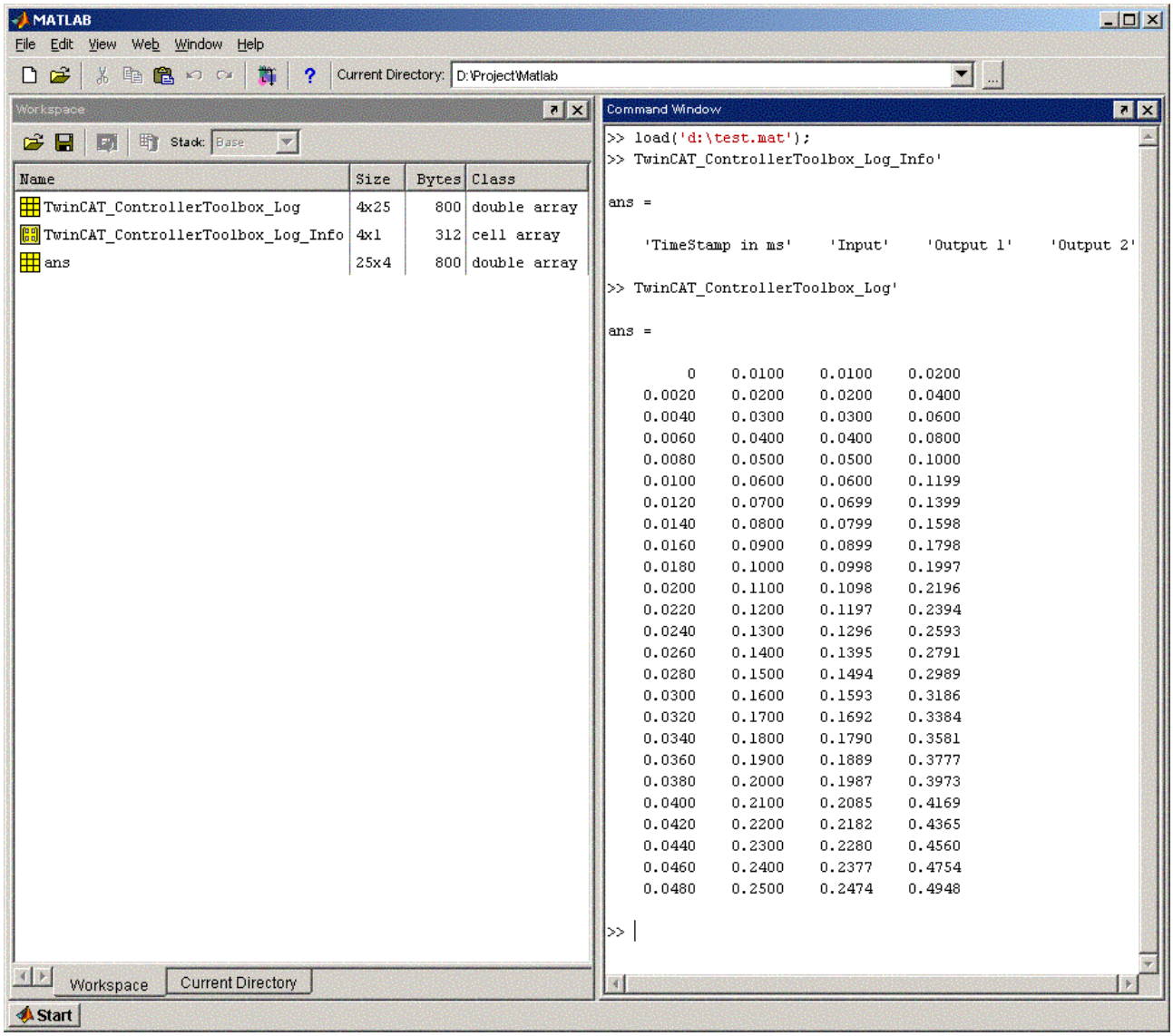
**文件处理期间出错**

只有在文件关闭（bFileClosed = TRUE）时，才能更改参数集。否则，在文件处理期间可能会出错。

### 例程

```
PROGRAM PRG_LOG_MAT_FILE_TEST_BUFFERED
VAR
    eMode              : E_CTRL_MODE;
    stParams           : ST_CTRL_LOG_MAT_FILE_PARAMS;
    LoggerData         : T_CTRL_LOGGER_DATA;
    eErrorId           : E_CTRL_ERRORCODES;
    bError             : BOOL;
    fbCtrlLogMatFile   : FB_CTRL_LOG_MAT_FILE;
    LogBuffer          : ARRAY[0..2000] OF BYTE;
    bInit              : BOOL := TRUE;
    fIn                : LREAL;
    fOut               : LREAL;
    fMaxBufferUsage    : LREAL;
END_VAR

IF bInit THEN
    stCtrl_GLOBAL_CycleTimeInterpretation.bInterpretCycleTimeAsTicks := FALSE;
    stCtrl_GLOBAL_CycleTimeInterpretation.fBaseTime := 0;
    stParams.tLogCycleTime     := T#2ms;
    stParams.tTaskCycleTime          := T#2ms;
    stParams.nNumberOfRows     := 3;
    stParams.sFileName     := 'D:\test.mat';
    stParams.sMatrixName     := 'TwinCAT_ControllerToolbox_Log';
    stParams.arRowDescription[1] := 'Input';
```

```
    stParams.arRowDescription[2] := 'Output 1';
    stParams.arRowDescription[3] := 'Output 2';
    stParams.bWriteRowDescription := TRUE;
    stParams.bWriteTimeStamps     := TRUE;
    eMode := eCTRL_MODE_ACTIVE;
    bInit := FALSE;
END_IF

stParams.nLogBuffer_SIZEOF := SIZEOF( LogBuffer );
stParams.pLogBuffer_ADR := ADR( LogBuffer );

fIn := fIn + 0.01;
fOut := SIN(fIn);

LoggerData[1]:= fIn;
LoggerData[2]:= fOut;
LoggerData[3]:= fOut * 2;

IF fbCtrlLogMatFile.nLoggedColumns >= 25 THEN
    eMode := eCTRL_MODE_Passive;
END_IF

fbCtrlLogMatFile( fLogData := LoggerData,
             eMode := eMode,
             stParams :=stParams,
             eErrorId => eErrorId,
             bError => bError);

fMaxBufferUsage := MAX(fbCtrlLogMatFile.fBufferUsage, fMaxBufferUsage);
```



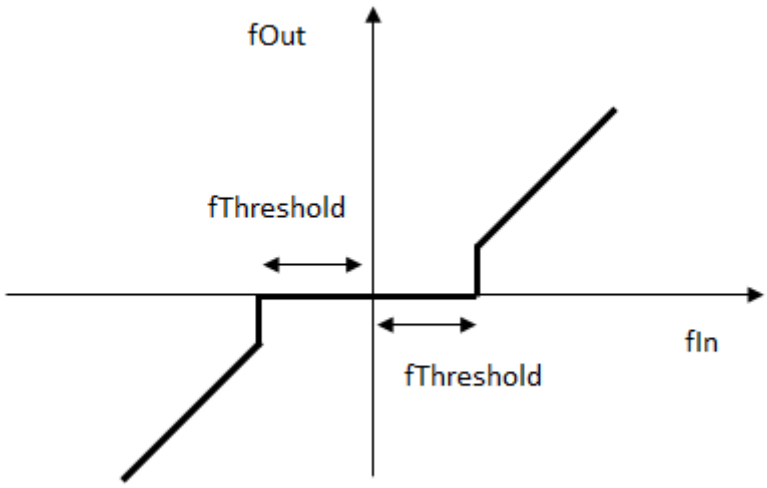
## 4.2.1.7 输出至控制设备

### 4.2.1.7.1 FB_CTRL_DEADBAND


该功能块为输入信号提供死区。如果输入信号在死区范围内，则由 `bInIsUnderThreshold` 输出表示此情况。

**输出行为的描述**

$$fout = \begin{cases} 0.0 : |\ fin\ | \leq fThreshold \\ fin : else \end{cases}$$




**输入**

```
VAR_INPUT
    fIn     : FLOAT;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| fIn | FLOAT | 输入值 |

**输出**

```
VAR_OUTPUT
    fOut                : FLOAT;
    bInIsUnderThreshold : BOOL;
    eState              : E_CTRL_STATE;
    eErrorId            : E_CTRL_ERRORCODES;
    bError              : BOOL;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| fOut | FLOAT | 功能块的输出 |

| 名称 | 类型 | 描述 |
|---|---|---|
| bInIsUnder Threshold | BOOL | 该输出为 TRUE 时，表示输入值在死区范围内。 |
| eState | E_CTRL_STATE | 功能块的状态 |
| eErrorId | E_CTRL_ERRORCODES | 在设置输出 bError 时，提供错误编号 [▶ 157]。 |
| bError | BOOL | 一旦发生错误，则变为 TRUE。 |

### 输入/输出

```
VAR_IN_OUT
    stParams        : ST_CTRL_DEADBAND_PARAMS;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| stParams | ST_CTRL_DEADBAND_PARAMS | 功能块的参数结构 |

stParams 由以下元素组成：

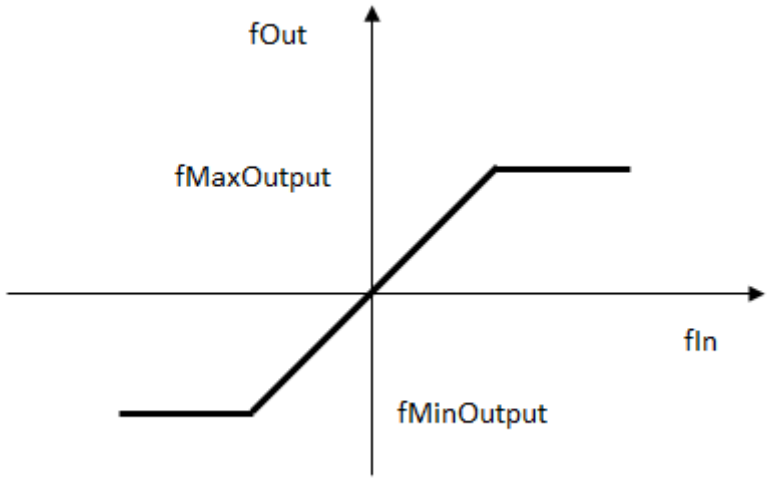
```
TYPE
ST_CTRL_DEADBAND_PARAMS:
STRUCT
    tCtrlCycleTime  : TIME := T#0ms;
    tTaskCycleTime  : TIME := T#0ms;
    fThreshold      : FLOAT;
END_STRUCT
END_TYPE
```

| 名称 | 类型 | 描述 |
|---|---|---|
| tCtrlCycleTime | TIME | 处理控制回路的循环时间。该时间必须大于或等于 **TaskCycleTime**。功能块使用该输入值在内部计算是否需要在当前周期内更新状态和输出值。 |
| tTaskCycleTime | TIME | 调用功能块的循环时间。如果在每个循环中都调用该功能块，这相当于调用任务的任务循环时间。 |
| fThreshold | FLOAT | 功能块的死区，请参见图示。 |

## 4.2.1.7.2 FB_CTRL_LIMITER


该功能块将输入信号限制在一个可参数化的区间内。

### 输出行为的描述




### 输入

```
VAR_INPUT
    fIn         : FLOAT;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| fIn | FLOAT | 功能块的输入值 |

### 输出

```
AR_OUTPUT
    fOut        : FLOAT;
    bMinLimit   : BOOL;
    bMaxLimit   : BOOL;
    eErrorId    : E_CTRL_ERRORCODES;
    bError      : BOOL;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| fOut | FLOAT | 功能块的输出 |
| bMinLimit | BOOL | 该输出为 TRUE 时，表示输出已达到下限。 |
| bMaxLimit | BOOL | 该输出为 TRUE 时，表示输出已达到上限。 |
| eErrorId | E_CTRL_ERRORCODES | 在设置输出 bError 时，提供错误编号 [▶ 157]。 |
| bError | BOOL | 一旦发生错误，则变为 TRUE。 |

### 输入/输出

```
VAR_IN_OUT
    stParams        : ST_CTRL_LIMITER_PARAMS;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| stParams | ST_CTRL_LIMITER_PARAMS | 功能块的参数结构 |

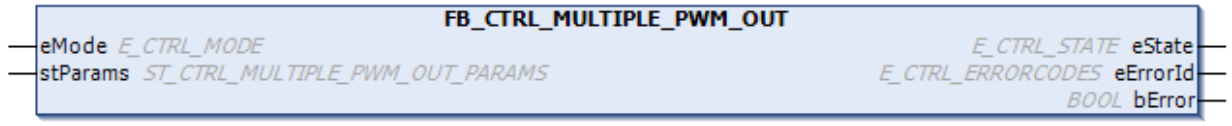
stParams 由以下元素组成：

```
TYPE ST_CTRL_LIMITER_PARAMS:
STRUCT
    tCtrlCycleTime  : TIME := T#0ms;
    tTaskCycleTime  : TIME := T#0ms;
```

```
    fMinOutput      : FLOAT;
    fMaxOutput      : FLOAT;
END_STRUCT
END_TYPE
```

| 名称 | 类型 | 描述 |
|---|---|---|
| tCtrlCycleTime | TIME | 处理控制回路的循环时间。该时间必须大于或等于 **TaskCycleTime**。功能块使用该输入值在内部计算是否需要在当前周期内更新状态和输出值。 |
| tTaskCycleTime | TIME | 调用功能块的循环时间。如果在每个循环中都调用该功能块，这相当于调用任务的任务循环时间。 |
| fMinOutput | FLOAT | 输出的下限。 |
| fMaxOutput | FLOAT | 输出的上限。 |

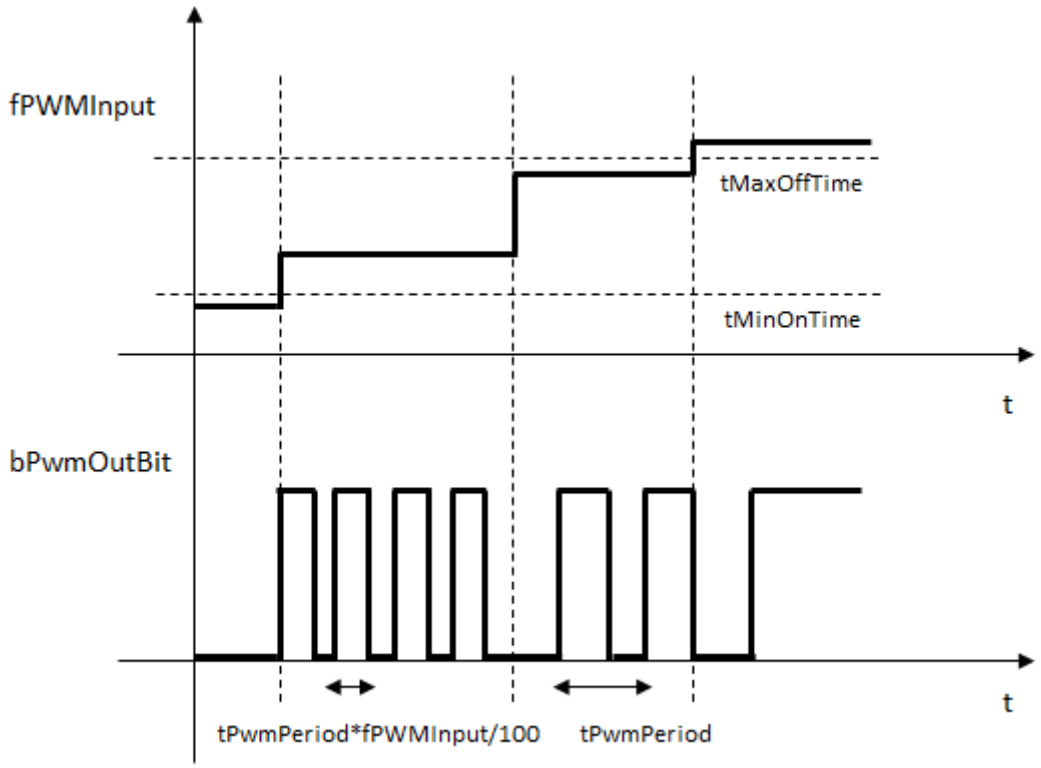
### 4.2.1.7.3 FB_CTRL_MULTIPLE_PWM_OUT



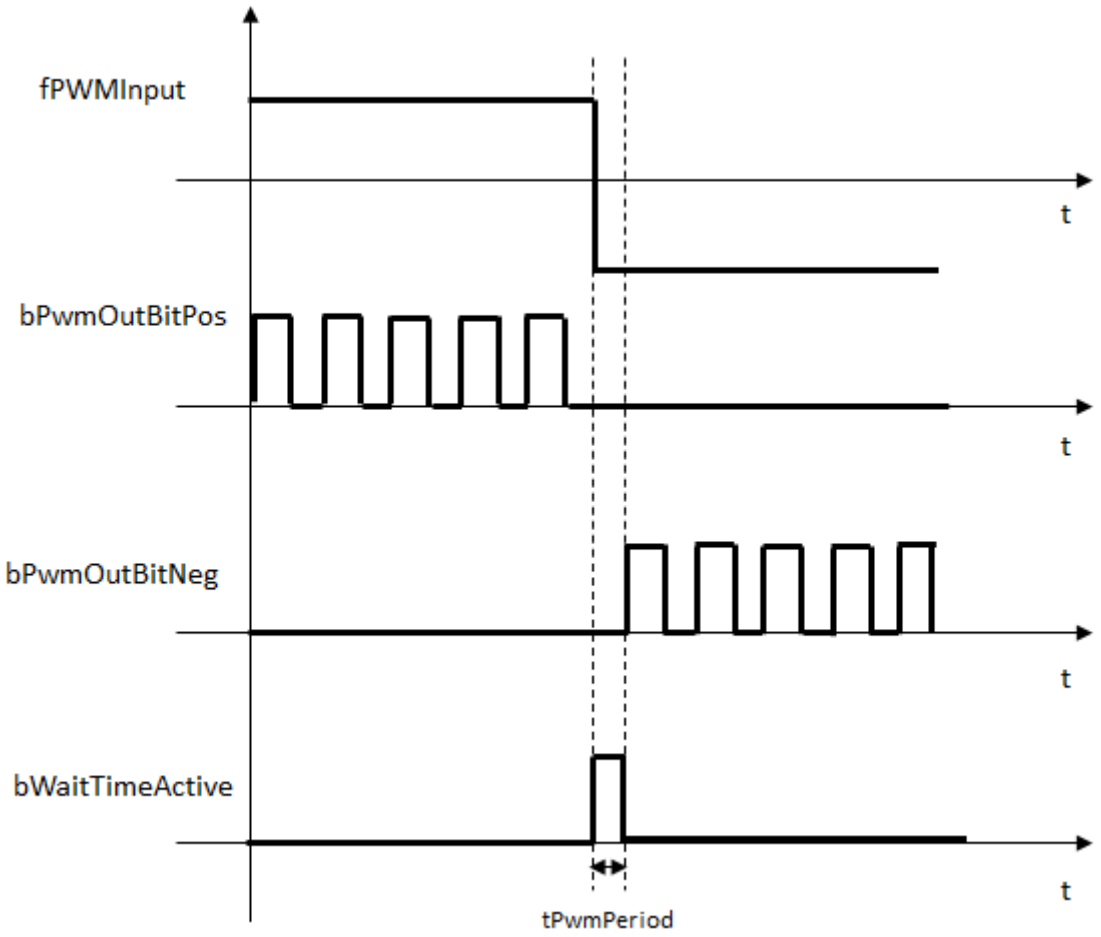

该功能块依据多个输入信号生成 PWM 调制的输出信号，同时通过优化输出之间的时序关系，使在任一时刻被激活的输出数量最小化。这种时序安排降低了执行器所需的最大功率。

在该扩展型功能块中，除了脉宽占空比之外，还可以对最短接通时间和最短关断时间进行参数设置。

**输出行为 1 的描述**



**输出行为 2 的描述**




如果要使用该功能块，程序员必须在 PLC 中创建以下数组：

```
aPwmInput           : ARRAY[1..nNumPwmOut] OF FLOAT;
aStWaitTimesConfig  : ARRAY[1..nNumPwmOut] OF ST_CTRL_MULTIPLE_PWM_OUT_TIMES;
aStPwmOutputs       : ARRAY[1.. nNumPwmOut] OF ST_CTRL_MULTIPLE_PWM_OUT_OUTPUTS;
aStPwmDataVars      : ARRAY[1.. nNumPwmOut] OF ST_CTRL_MULTIPLE_PWM_OUT_DATA;
```

PWM 功能块各个通道的输入值被写入 **ar_fPwmInput** 数组中。程序员可以在 **ar_stWaitTimesConfig** 数组中指定各个通道的可参数化时间。该功能块将输出位写入 **ar_stPwmOutputs** 数组中。功能块所需的内部数据存储在 **ar_stPwmDataVars** 数组中。在任何情况下都不应在 PLC 程序中更改数组 **ar_stPwmDataVars** 所包含的结构。此流程也在下方列出的示例程序中予以说明。

### 输入

```
VAR_INPUT
    eMode    : E_CTRL_MODE;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| eMode | E_CTRL_MODE | 指定功能块的运行模式的输入。 |

### 输出

```
VAR_OUTPUT
    eState    : E_CTRL_STATE;
    bError    : BOOL;
    eErrorId  : E_CTRL_ERRORCODES;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| eState | E_CTRL_STATE | 功能块的状态 |
| bError | BOOL | 在设置输出 bError 时，提供错误编号 [▶ 157]。 |
| eErrorId | E_CTRL_ERRORCODES | 一旦发生错误，则变为 TRUE。 |

### 输入/输出

```
VAR_IN_OUT
    stParams     : ST_CTRL_PWM_OUT_EXT_PARAMS;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| stParams | ST_CTRL_PWM_OUT_EXT_PARAMS | PWM 单元的参数结构 |

stParams 由以下元素组成：

```
TYPE ST_CTRL_MULTIPLE_PWM_OUT_PARAMS :
STRUCT
    tTaskCycleTime             : TIME;
    tPWMPeriod                 : TIME;
    nNumberOfPwmOutputs        : USINT;
    pPwmInputArray_ADR         : POINTER TO FLOAT := 0;
    nPwmInputArray_SIZEOF      : UINT;
    pPwmTimesConfig_ADR        : POINTER TO ST_CTRL_MULTIPLE_PWM_OUT_TIMES;
    nPwmTimesConfig_SIZEOF     : UINT;
    pPwmOutputArray_ADR        : POINTER TO ST_CTRL_MULTIPLE_PWM_OUT_OUTPUTS;
    nPwmOutputArray_SIZEOF     : UINT;
    pPwmData_ADR               : POINTER TO ST_CTRL_MULTIPLE_PWM_OUT_DATA;
    nPwmData_SIZEOF            : UINT;
END_STRUCT
END_TYPE
```

| 名称 | 类型 | 描述 |
|---|---|---|
| tTaskCycleTime | TIME | 调用功能块的循环时间。如果在每个循环中都调用该功能块，这相当于调用任务的任务循环时间。 |
| tPWMPeriod | TIME | PWM 信号的周期 |
| nNumberOfPwmOutputs | USINT | PWM 输出数量。[1...n] |
| pPwmInputArray_ADR | POINTER TO FLOAT | PWM 输入数组的地址 |
| nPwmInputArray_SIZEOF | UINT | PWM 输入数组的大小，以字节为单位 |
| pPwmTimesConfig_ADR | POINTER TO ST_CTRL_MULTIPLE_PWM_OUT_TIMES | PWM 时间数组的地址 |
| nPwmTimesConfig_SIZEOF | UINT | PWM 时间数组的大小，以字节为单位 |
| pPwmOutputArray_ADR | POINTER TO ST_CTRL_MULTIPLE_PWM_OUT_OUTPUTS | PWM 输出数组的地址 |
| nPwmOutputArray_SIZEOF | UINT | PWM 输出数组的大小，以字节为单位 |
| pPwmData_ADR | POINTER TO ST_CTRL_MULTIPLE_PWM_OUT_DATA | 内部 PWM 数据数组的地址 |

| 名称 | 类型 | 描述 |
|---|---|---|
| nPwmData_SIZEOF | UINT | 内部 PWM 数据数组的大小，以字节为单位 |

```
TYPE ST_CTRL_MULTIPLE_PWM_OUT_TIMES :
STRUCT
    tMinOnTime     : TIME;
    tMinOffTime    : TIME;
    tMinWaitTime   : TIME;
END_STRUCT
END_TYPE
```

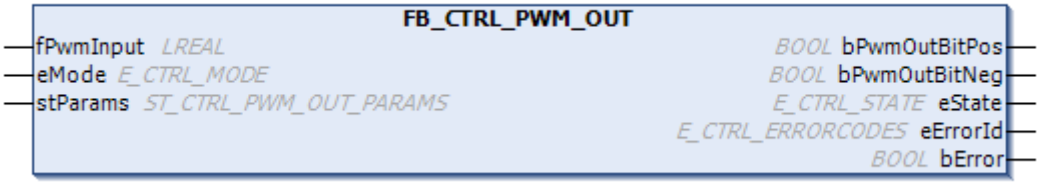
| 名称 | 类型 | 描述 |
|---|---|---|
| tMinOnTime | TIME | PWM 输出的最短接通时间 |
| tMinOffTime | TIME | PWM 输出的最短关断时间 |
| tMinWaitTime | TIME | 正负输出信号切换操作之间的等待时间 |

```
TYPE ST_CTRL_MULTIPLE_PWM_OUT_OUTPUTS :
STRUCT
    bPwmOutBitPos     : BOOL;
    bPwmOutBitNeg     : BOOL;
    bWaitTimeActive   : BOOL;
END_STRUCT
END_TYPE
```

| 名称 | 类型 | 描述 |
|---|---|---|
| bPwmOutBitPos | BOOL | PWM 信号，当 fPwmInput > 0.0 时 |
| bPwmOutBitNeg | BOOL | PWM 信号，当 fPwmInput < 0.0 时 |
| bWaitTimeActive | BOOL | 该输出为 TRUE 时，表示输出信号切换操作之间的等待时间已激活。在此期间，该输出可用于保持前置控制器中可能存在的 I 分量不变。 |

```
TYPE ST_CTRL_MULTIPLE_PWM_OUT_DATA :
STRUCT
Internal structure. This must not be written to.
END_STRUCT
END_TYPE
```

## 4.2.1.7.4 FB_CTRL_PWM_OUT

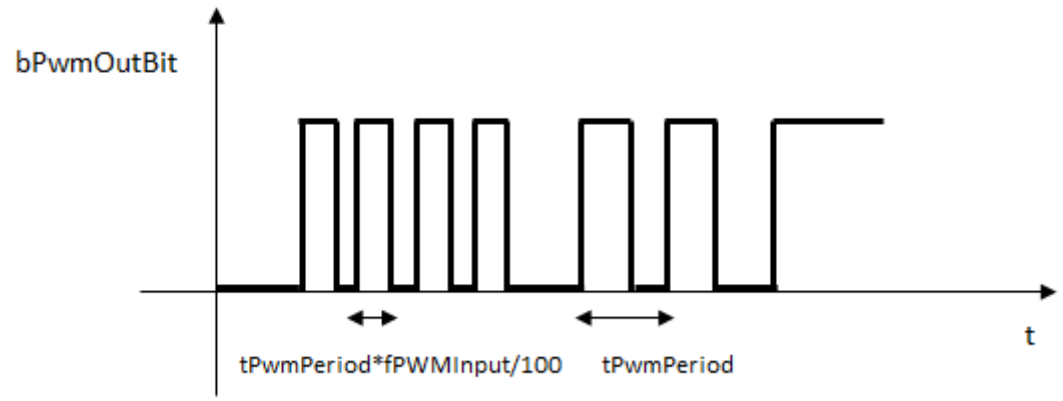



该功能块根据输入信号创建 PWM 调制信号。

**输出行为的描述**

该功能块在输出端创建 PWM 信号，其脉宽占空比与 **fPwmInput** 输入相对应。在输入端以 % 为单位指定脉宽占空比，可用范围为 -100% 至 100%。如果指定的是正值，则在 **bPwmOutBitPos** 输出端提供脉宽调制信号。如果指定的脉宽占空比为负值，则在 **bPwmOutBitNeg** 输出信号。因此，根据算术符号的不同，这两个信号可以控制两个不同的执行器。

如果 **bInstantPWMUpdate** 被设置为 TRUE，则可以立即采用新的输入值。换句话说，即使在当前 PWM 周期中，新的输入值也是有效值。如果该参数为 FALSE，则只有在新的 PWM 周期开始时才会采用新的输入值。




### 输入

```
VAR_INPUT
    fPwmInput   : FLOAT;
    eMode       : E_CTRL_MODE;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| fPwmInput | FLOAT | 输入值 |
| eMode | E_CTRL_MODE | 指定功能块的运行模式的输入。 |

### 输出

```
VAR_OUTPUT
    bPwmOutBitPos   : BOOL;
    bPwmOutBitNeg   : BOOL;
    eState          : E_CTRL_STATE;
    bError          : BOOL;
    eErrorId        : E_CTRL_ERRORCODES;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| bPwmOutBitPos | BOOL | PWM 信号，当 fPwmInput > 0.0 时。 |
| bPwmOutBitNeg | BOOL | PWM 信号，当 fPwmInput < 0.0 时。 |
| eState | E_CTRL_STATE | 功能块的状态 |
| bError | BOOL | 在设置输出 bError 时，提供错误编号 [▶ 157]。 |
| eErrorId | E_CTRL_ERRORCODES | 一旦发生错误，则变为 TRUE。 |

### 输入/输出

```
VAR_IN_OUT
    stParams    : ST_CTRL_PWM_OUT_PARAMS;
END_VAR
```

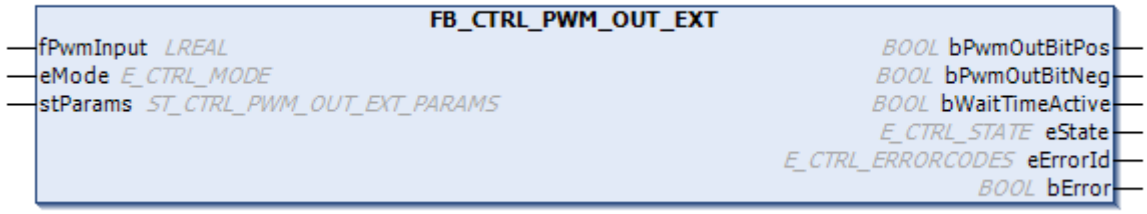
| 名称 | 类型 | 描述 |
|---|---|---|
| stParams | ST_CTRL_PWM_OUT_PARAMS | PWM 单元的参数结构 |

stParams 由以下元素组成：

```
TYPE ST_CTRL_PWM_OUT_PARAMS:
STRUCT
    tTaskCycleTime      : TIME
    tPWMPeriod          : TIME;
    bInstantPWMUpdate   : BOOL;
END_STRUCT
END_TYPE
```

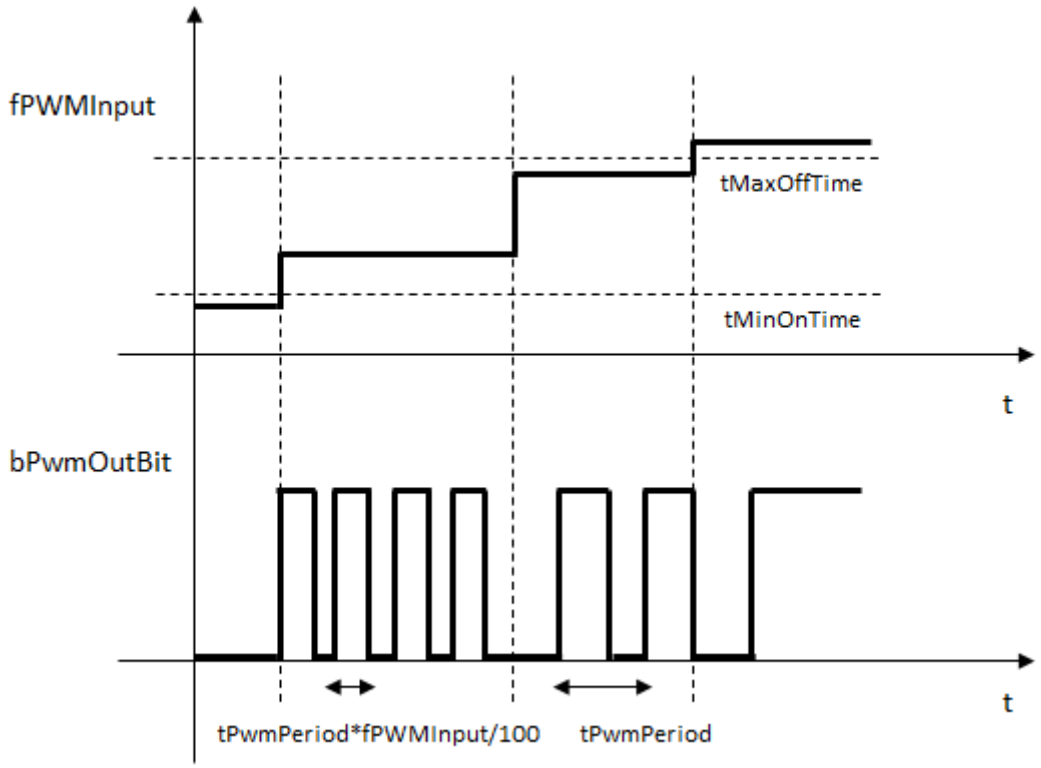
| 名称 | 类型 | 描述 |
|---|---|---|
| tTaskCycleTime | TIME | 调用功能块的循环时间。如果在每个循环中都调用该功能块，这相当于调用任务的任务循环时间。 |
| tPWMPeriod | TIME | PWM 信号的周期 |
| bInstantPWMUpdate | BOOL | 如果该标志为 TRUE，则立即采用新的输入值，即使在当前 PWM 周期中也是如此。 |

### 4.2.1.7.5 FB_CTRL_PWM_OUT_EXT

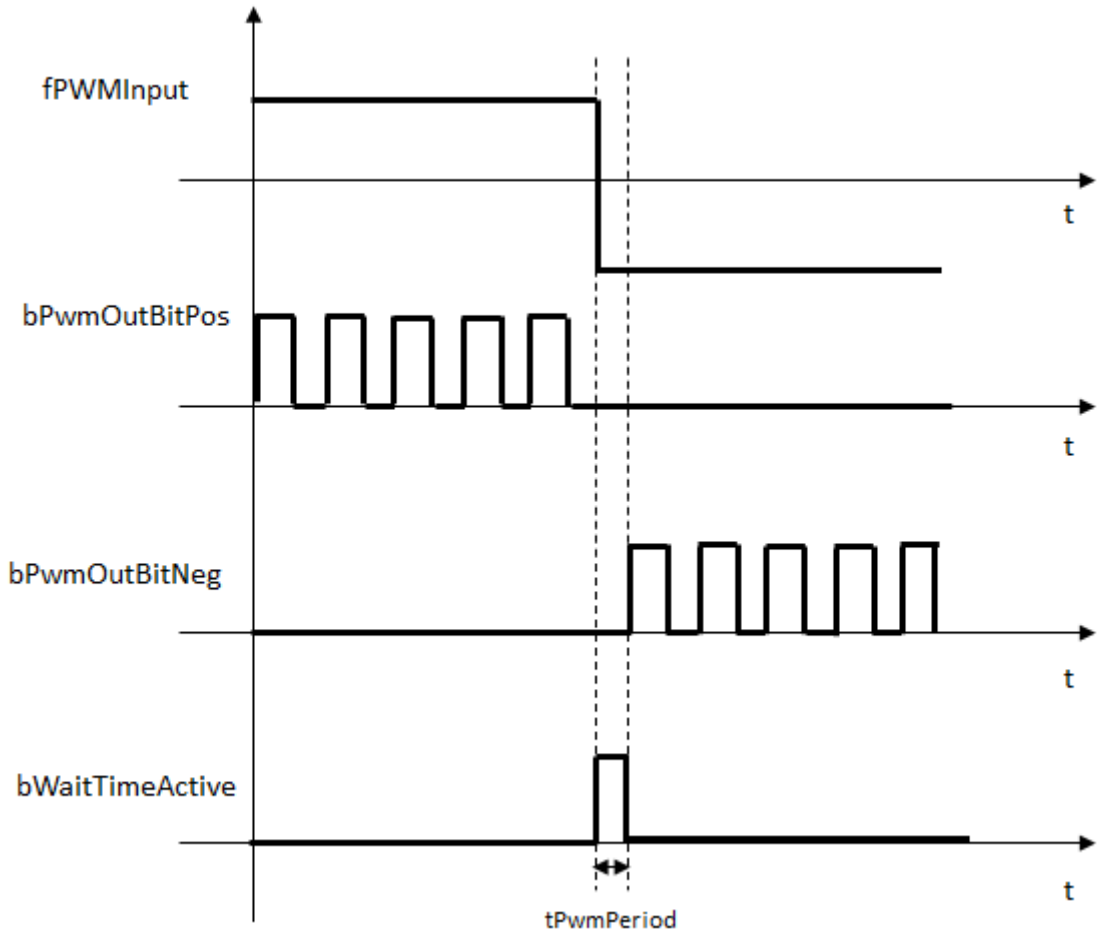



该功能块根据输入信号创建 PWM 调制信号。在该扩展型功能块中，除了脉宽占空比之外，还可以对最短接通时间和最短关断时间进行参数设置。

**输出行为 1 的描述**



**输出行为 2 的描述**




### 输入

```
VAR_INPUT
    fPwmInput   : FLOAT;
    eMode       : E_CTRL_MODE;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| fPwmInput | FLOAT | 功能块的输入值 |
| eMode | E_CTRL_MODE | 指定功能块的运行模式的输入。 |

### 输出

```
VAR_OUTPUT
    bPwmOutBitPos   : BOOL;
    bPwmOutBitNeg   : BOOL;
    eState          : E_CTRL_STATE;
    bError          : BOOL;
    eErrorId        : E_CTRL_ERRORCODES;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| bPwmOutBitPos | BOOL | PWM 信号，当 fPwmInput > 0.0 时 |
| bPwmOutBitNeg | BOOL | PWM 信号，当 fPwmInput < 0.0 时 |
| eState | E_CTRL_STATE | 功能块的状态 |
| bError | BOOL | 在设置输出 bError 时，提供错误编号 [▶ 157]。 |

| 名称 | 类型 | 描述 |
|---|---|---|
| eErrorId | E_CTRL_ERRORCODES | 一旦发生错误，则变为 TRUE。 |

### 输入/输出

```
VAR_IN_OUT
    stParams    : ST_CTRL_PWM_OUT_EXT_PARAMS;
END_VAR
```

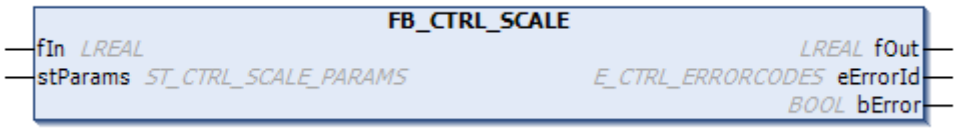
| 名称 | 类型 | 描述 |
|---|---|---|
| stParams | ST_CTRL_PWM_OUT_EXT_PARAMS | PWM 单元的参数结构 |

`stParams` 由以下元素组成：

```
TYPE ST_CTRL_PWM_OUT_EXT_PARAMS :
STRUCT
    tTaskCycleTime  : TIME;
    tPWMPeriod      : TIME;
    tMinOnTime      : TIME;
    tMinOffTime     : TIME;
    tMinWaitTime    : TIME;
END_STRUCT
END_TYPE
```

| 名称 | 类型 | 描述 |
|---|---|---|
| tTaskCycleTime | TIME | 调用功能块的循环时间。如果在每个循环中都调用该功能块，这相当于调用任务的任务循环时间。 |
| tPWMPeriod | TIME | PWM 信号的周期 |
| tMinOnTime | TIME | 最短接通时间 |
| tMinOffTime | TIME | 最短关断时间 |
| tMinWaitTime | TIME | 正负输出信号切换操作之间的等待时间 |

## 4.2.1.7.6 FB_CTRL_SCALE




该功能块可以在两个值范围之间以线性方式调节信号。

### 输入

```
VAR_INPUT
    fIn     : FLOAT;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| fIn | FLOAT | 功能块的输入值 |

### 输出

```
VAR_OUTPUT
    fOut      : FLOAT;
    eErrorId  : E_CTRL_ERRORCODES;
    bError    : BOOL;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| fOut | FLOAT | 缩放后的输出值 |
| eErrorId | E_CTRL_ERRORCODES | 在设置输出 bError 时，提供错误编号 [▶ 157]。 |
| bError | BOOL | 一旦发生错误，则变为 TRUE。 |

### 输入/输出

```
VAR_IN_OUT
    stParams    : ST_CTRL_SCALE_PARAMS;
END_VAR
```

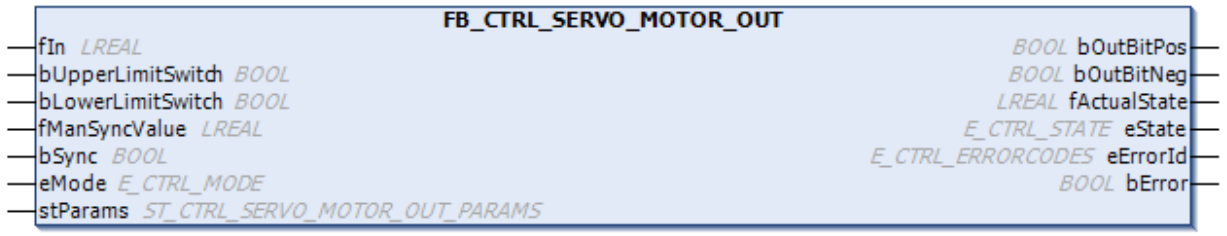
| 名称 | 类型 | 描述 |
|---|---|---|
| stParams | ST_CTRL_SCALE_PARAMS | 功能块的参数结构 |

stParams 由以下元素组成：

```
TYPE ST_CTRL_SCALE_PARAMS:
STRUCT
    tCtrlCycleTime  : TIME := T#0ms;
    tTaskCycleTime  : TIME := T#0ms;
    fInMin          : FLOAT;
    fInMax          : FLOAT;
    fOutMin         : FLOAT;
    fOutMax         : FLOAT;
END_STRUCT
END_TYPE
```

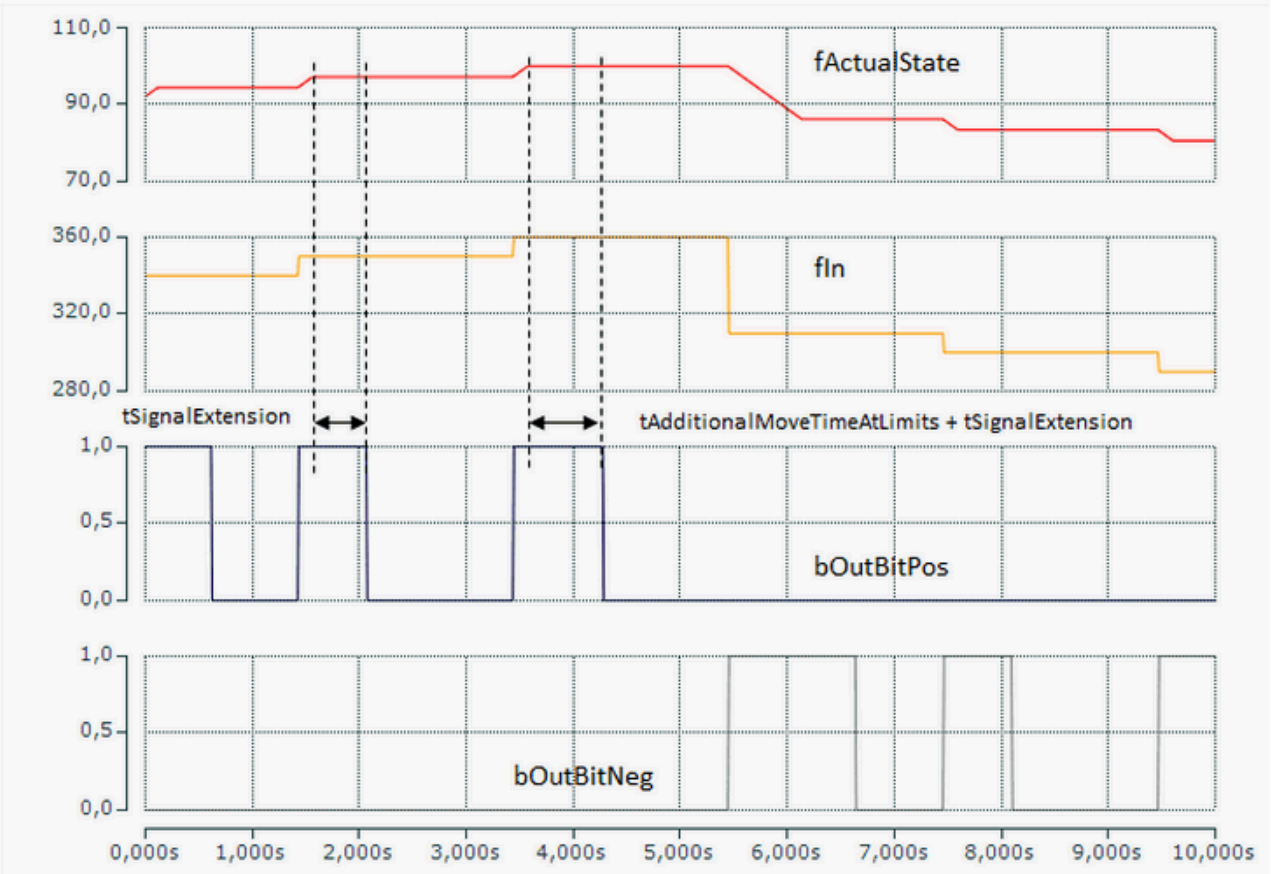
| 名称 | 类型 | 描述 |
|---|---|---|
| tCtrlCycleTime | TIME | 处理控制回路的循环时间。该时间必须大于或等于 **TaskCycleTime**。功能块使用该输入值在内部计算是否需要在当前周期内更新状态和输出值。 |
| tTaskCycleTime | TIME | 调用功能块的循环时间。如果在每个循环中都调用该功能块，这相当于调用任务的任务循环时间。 |
| fInMin | FLOAT | 输入值的最小值 |
| fInMax | FLOAT | 输入值的最大值 |
| fOutMin | FLOAT | 输出值的最小值 |
| fOutMax | FLOAT | 输出值的最大值 |

## 4.2.1.7.7 FB_CTRL_SERVO_MOTOR_OUT




该功能块产生的脉冲可将伺服电机驱动到规定的位置。

### 输出的行为




### 输入

```
VAR_INPUT
    fIn                 : FLOAT;
    bUpperLimitSwitch   : BOOL;
    bLowerLimitSwitch   : BOOL;
    fManSyncValue       : FLOAT;
    bSync               : BOOL;
    eMode               : E_CTRL_MODE;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| fIn | FLOAT | 控制器在范围 fCtrlOutMin ... fCtrlOutMax]（控制器输出）内的控制值。 |
| bUpperLimit Switch | BOOL | 限位开关：如果达到上限位点，则为 TRUE。 |
| bLowerLimit Switch | BOOL | 限位开关：如果达到下限位点，则为 TRUE。 |
| fManSyncValue | FLOAT | 可调整当前电机设置的内部状态的输入，或在手动模式下采用其值的输入。 |
| bSync | BOOL | 该输入端的上升沿将内部电机位置（必须与实际电机位置一致）设置为值“fManSyncValue”。 |
| eMode | E_CTRL_ MODE | 指定功能块的运行模式 [▶ 157]的输入。 |

为了与当前阀门位置同步，该功能块具有模式 **eCTRL_MODE_SYNC_MOVEMENT**。在此模式下，用于关闭阀门的输出已被设置，直至阀门安全关闭。然后，功能块与该阀门位置同步。

同步过程完成后，**eCTRL_STATE_ACTIVE** 模式将被自动激活。

### 输出

```
VAR_OUTPUT
    bOutBitPos      : BOOL;
    bOutBitNeg      : BOOL;
    fActualState    : FLOAT;
    eState          : E_CTRL_STATE;
    eErrorId        : E_CTRL_ERRORCODES;
    bError          : BOOL;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| bOutBitPos | BOOL | 正向驱动电机所需的输出。 |
| bOutBitNeg | BOOL | 反向驱动电机所需的输出。 |
| fActualState | FLOAT | 当前电机在范围 [fCtrlOutMin ... fCtrlOutMax] 内，这是电器当前所在的位置。 |
| eState | E_CTRL_STATE | 功能块的状态 |
| eErrorId | E_CTRL_ERRORCODES | 在设置输出 `bError` 时，提供错误编号 [▶ 157]。 |
| bError | BOOL | 一旦发生错误，则变为 TRUE。 |

### 输入/输出

```
VAR_IN_OUT
    stParams    : ST_CTRL_SERVO_MOTOR_OUT_PARAMS;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| stParams | ST_CTRL_SERVO_MOTOR_OUT_PARAMS | 功能块的参数结构 |

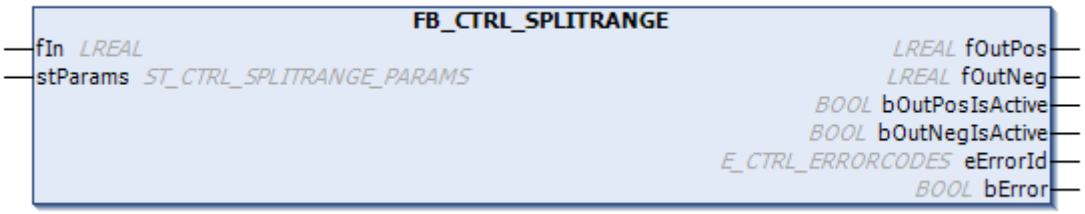
`stParams` 由以下元素组成：

```
TYPE ST_CTRL_SERVO_MOTOR_OUT_PARAMS:
STRUCT
    tCtrlCycleTime                              : TIME := T#0ms;
    tTaskCycleTime                              : TIME := T#0ms;
    tMovingTime                                 : TIME;
    tSignalExtension                            : TIME;
    tAdditionalMoveTimeAtLimits                 : TIME;
    tMinWaitTimeBetweenDirectionChange          : TIME;
    tMinimumPulseTime                           : TIME;
    bMoveOnLimitSwitch                          : BOOL;
    bStopAdditionalMoveTimeIfInputValueIsChanged : BOOL;
    fCtrlOutMax                                 : FLOAT := 100.0;
    fCtrlOutMin                                 : FLOAT := 0.0;
END_STRUCT
END_TYPE
```

| 名称 | 类型 | 描述 |
|---|---|---|
| tCtrlCycleTime | TIME | 处理控制回路的循环时间。该时间必须大于或等于 **TaskCycleTime**。功能块使用该输入值在内部计算是否需要在当前周期内更新状态和输出值。 |
| tTaskCycleTime | TIME | 调用功能块的循环时间。如果在每个循环中都调用该功能块，这相当于调用任务的任务循环时间。 |
| tMovingTime | TIME | 执行器从一个限位点移动到另一个限位点所需的时间。 |
| tSignalExtension | TIME | 信号扩展，用于延长每个输出脉冲，以补偿死区时间。 |

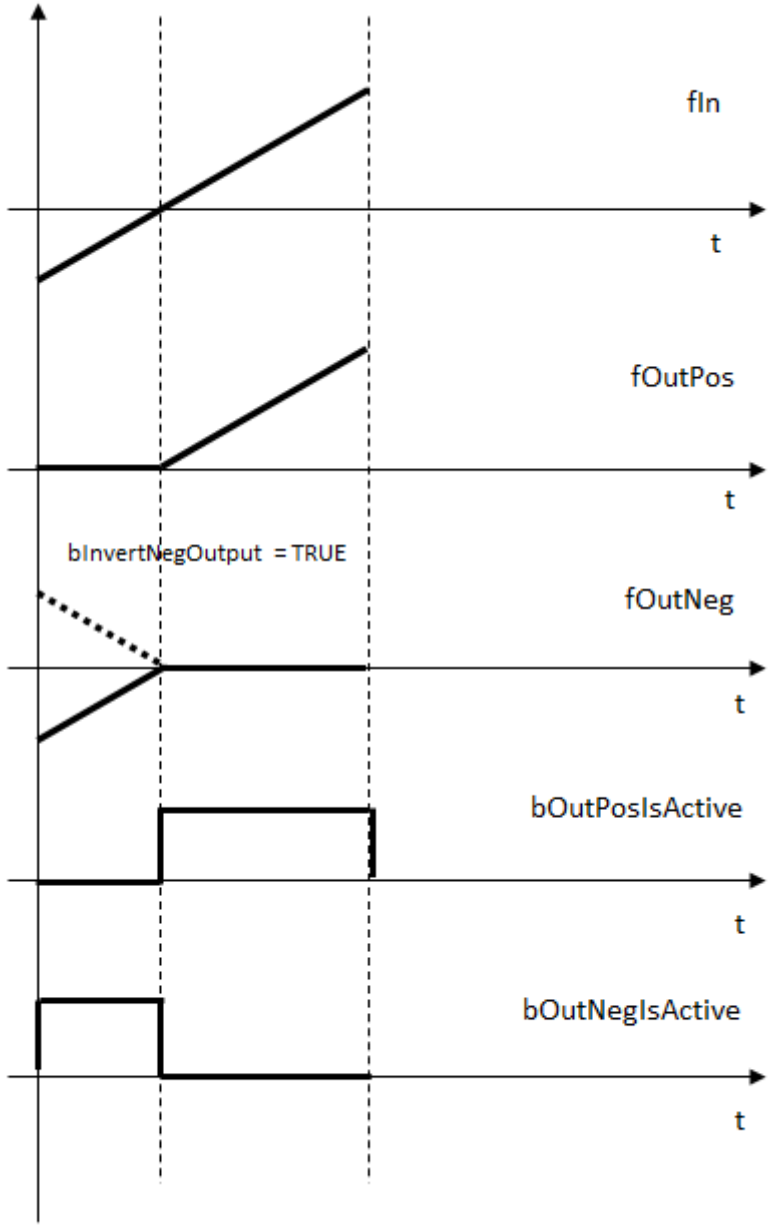
| 名称 | 类型 | 描述 |
|---|---|---|
| tAdditionalMoveTimeAtLimits | TIME | 补充信号扩展，当执行器被驱动到 +/-100% 时，输出能够可靠地达到限值。仅当 bMoveOnLimitSwitch 为 FALSE 时有效。 |
| tMinWaitTimeBetweenDirectionChange | TIME | 正负输出脉冲之间的最短等待时间 |
| tMinimumPulseTime | TIME | 输出脉冲的最小长度 |
| bMoveOnLimitSwitch | BOOL | 如果为 TRUE，则当控制值为 fCtrlOutMin 或 fCtrlOutMax 时，将继续输出信号，直至达到相应的限位开关。 |
| bStopAdditionalMoveTimeIfInputValueIsChanged | BOOL | 如果该标志为 TRUE，则在指定的输入值与终点位置不匹配的情况下，停止由“限位附加移动时间”触发的阀门向终点位置的移动。如果该标志为 FALSE 且输入值与终点位置匹配，则阀门始终首先安全移动到终点位置，然后才能移动到其他阀门位置。 |
| fCtrlOutMax | FLOAT | 将阀门驱动至 100% 的控制值。 |
| fCtrlOutMin | FLOAT | 将阀门驱动至 0% 的控制值。 |

### 4.2.1.7.8 FB_CTRL_SPLITRANGE




该功能块将输入信号分为正负分量。参数 bDisablePosOut 和 bDisableNegOut 可用于停用正输出或负输出 → 仅在冬季采用加热模式，仅在夏季采用冷却模式。bInvertNegOutput 参数允许对负输出进行反转。

**输出行为的描述**




### 输入

```
VAR_INPUT
    fIn     : FLOAT;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| fIn | FLOAT | 功能块的输入值 |

### 输出

```
VAR_OUTPUT
    fOutPos           : FLOAT;
    fOutNeg           : FLOAT;
    bOutPosIsActive   : BOOL;
    bOutNegIsActive   : BOOL;
    eErrorId          : E_CTRL_ERRORCODES;
    bError            : BOOL;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| fOutPos | FLOAT | `fIn` 的正分量 |
| fOutNeg | FLOAT | `fIn` 的负分量 |
| bOutPosIsActive | BOOL | TRUE 表示“fIn > 0.0” |
| bOutNegIsActive | BOOL | TRUE 表示“fIn < 0.0” |
| eErrorId | E_CTRL_ERRORCODES | 在设置输出 `bError` 时，提供错误编号 [▶ 157]。 |
| bError | BOOL | 一旦发生错误，则变为 TRUE。 |

### 输入/输出

```
VAR_IN_OUT
    stParams        : ST_CTRL_SPLITRANGE_PARAMS;
END_VAR
```

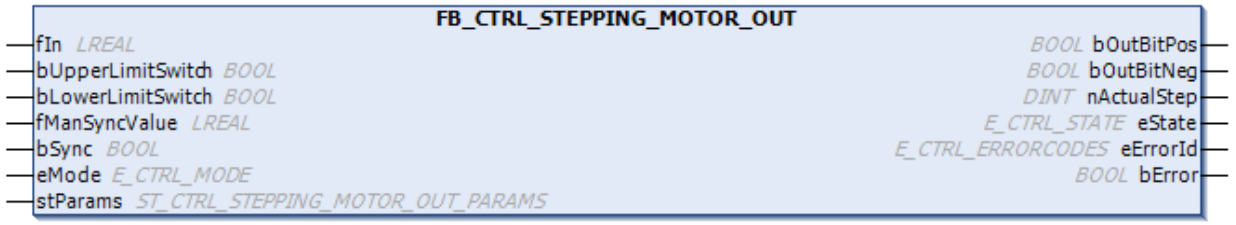
| 名称 | 类型 | 描述 |
|---|---|---|
| stParams | ST_CTRL_SPLITRANGE_PARAMS | 功能块的参数结构 |

`stParams` 由以下元素组成：

```
TYPE
ST_CTRL_SPLITRANGE_PARAMS:
STRUCT
    tCtrlCycleTime    : TIME := T#0ms;
    tTaskCycleTime    : TIME := T#0ms;
    bInvertNegOutput  : BOOL;
    bDisablePosOut    : BOOL;
    bDisableNegOut    : BOOL;
END_STRUCT
END_TYPE
```

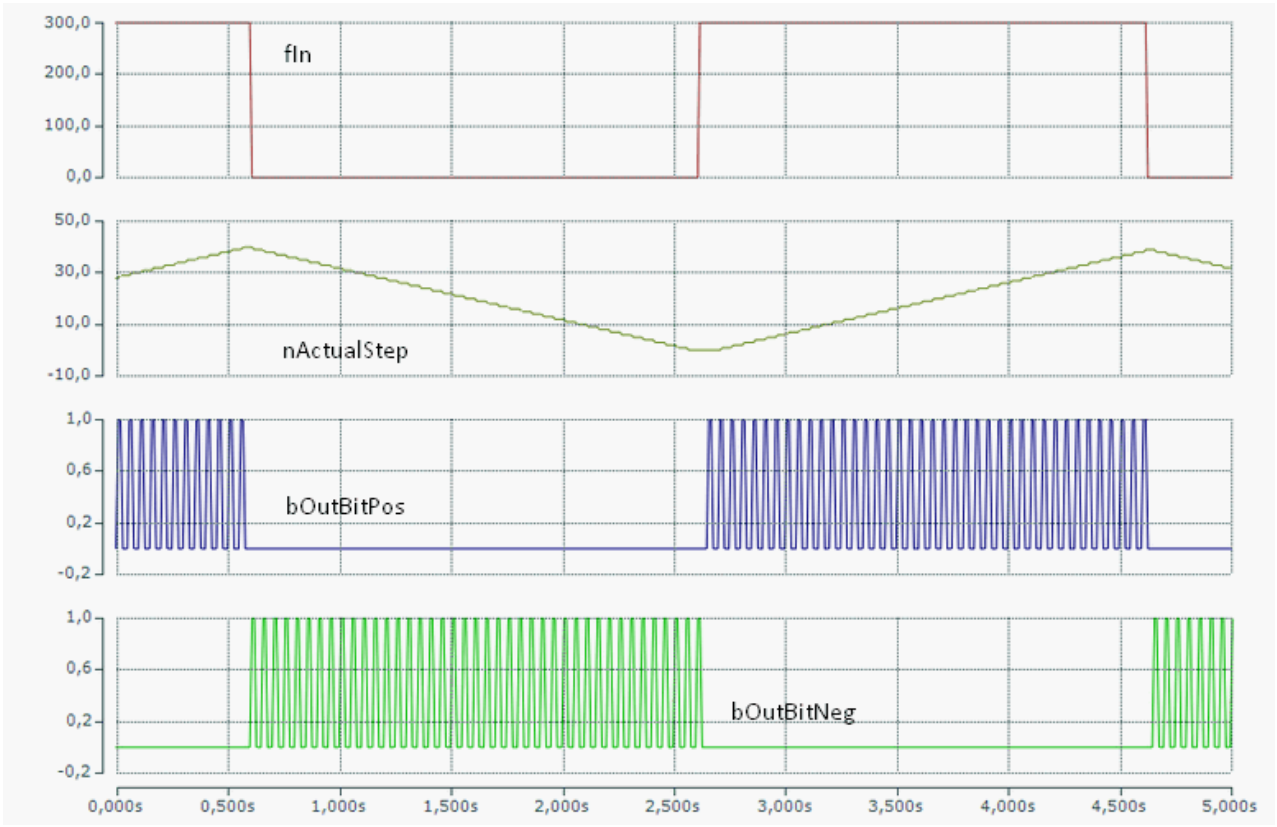
| 名称 | 类型 | 描述 |
|---|---|---|
| tCtrlCycleTime | TIME | 处理控制回路的循环时间。该时间必须大于或等于 **TaskCycleTime**。功能块使用该输入值在内部计算是否需要在当前周期内更新状态和输出值。 |
| tTaskCycleTime | TIME | 调用功能块的循环时间。如果在每个循环中都调用该功能块，这相当于调用任务的任务循环时间。 |
| bInvertNegOutput | BOOL | 当该参数为 TRUE 时，`fOutNeg` 将进行反转。 |
| bDisablePosOut | BOOL | 输出 `fOutPos` 已被禁用，始终为“0.0”。 |
| bDisableNegOut | BOOL | 输出 `fOutNeg` 已被禁用，始终为“0.0”。 |

## 4.2.1.7.9 FB_CTRL_STEPPING_MOTOR_OUT




该功能块可为步进电机生成一个控制值。

### 输出的行为




### 输入

```
VAR_INPUT
    fIn                 : FLOAT;
    bUpperLimitSwitch   : BOOL;
    bLowerLimitSwitch   : BOOL;
    fManSyncValue       : FLOAT;
    bSync               : BOOL;
    eMode               : E_CTRL_MODE;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| fIn | FLOAT | 控制器的控制值（控制器输出） |
| bUpperLimit Switch | BOOL | 限位开关，在达到上限位时变为 TRUE。 |
| bLowerLimit Switch | BOOL | 限位开关，在达到下限位时变为 TRUE。 |
| fManSyncValue | FLOAT | 可调整电机设置的内部状态的输入，或在手动模式下采用其值的输入。 |
| bSync | BOOL | 该输入端的上升沿将内部步进计数器设置为与值“fManSyncValue”相对应的步进。 |
| eMode | E_CTRL_MODE | 指定功能块的运行模式 [▶ 157]的输入。 |

为了与当前阀门位置同步，该功能块具有模式 **eCTRL_MODE_SYNC_MOVEMENT**。在此模式下，用于关闭阀门的脉冲已被输出，直至阀门安全关闭。然后，功能块与该阀门位置同步。

同步过程完成后，**eCTRL_STATE_ACTIVE** 模式将被自动激活。

### 输出

```
VAR_OUTPUT
    bOutBitPos    : BOOL;
    bOutBitNeg    : BOOL;
    nActualStep   : DINT;
    eState        : E_CTRL_STATE;
    eErrorId      : E_CTRL_ERRORCODES;
    bError        : BOOL;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| bOutBitPos | BOOL | 正向驱动电机所需的输出。 |
| bOutBitNeg | BOOL | 反向驱动电机所需的输出。 |
| nActualStep | DINT | 电机定位的实际步进。 |
| eState | E_CTRL_STATE | 功能块的状态 |
| eErrorId | E_CTRL_ERRORCODES | 在设置输出 bError 时，提供错误编号 [▶ 157]。 |
| bError | BOOL | 一旦发生错误，则变为 TRUE。 |

### 输入/输出

```
VAR_IN_OUT
    stParams     : ST_CTRL_STEPPING_MOTOR_OUT_PARAMS;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| stParams | ST_CTRL_STEPPING_MOTOR_OUT_PARAMS | 功能块的参数结构 |

stParams 由以下元素组成：

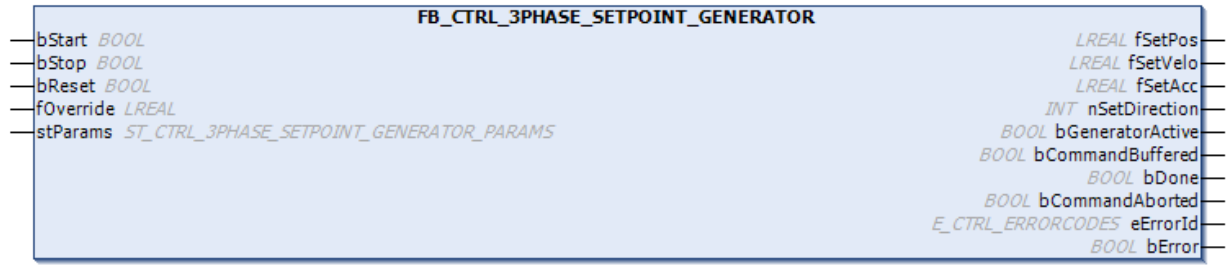
```
TYPE
ST_CTRL_STEPPING_MOTOR_OUT_PARAMS:
STRUCT
    tCtrlCycleTime              : TIME := T#0ms;
    tTaskCycleTime              : TIME := T#0ms;
    tOnTime                     : TIME;
    tOffTime                    : TIME;
    nMaxMovingPulses            : DINT;
    nMinMovingPulses            : UINT := 0;
    bHoldWithOutputOn           : BOOL;
    nAdditionalPulsesAtLimits   : DINT;
    bMoveOnLimitSwitch          : BOOL;
    fCtrlOutMax                 : FLOAT := 100.0;
    fCtrlOutMin                 : FLOAT := 0.0;
END_STRUCT
END_TYPE
```

| 名称 | 类型 | 描述 |
|---|---|---|
| tCtrlCycleTime | TIME | 处理控制回路的循环时间。该时间必须大于或等于 **TaskCycleTime**。功能块使用该输入值在内部计算是否需要在当前周期内更新状态和输出值。 |
| tTaskCycleTime | TIME | 调用功能块的循环时间。如果在每个循环中都调用该功能块，这相当于调用任务的任务循环时间。 |
| tOnTime | TIME | 脉冲长度 |
| tOffTime | TIME | 暂停长度 |
| nMaxMoving Pulses | DINT | 从一个限值移动到另一个限值所需的最大脉冲数。 |
| nMinMoving Pulses | UINT | 从一个限值移动到另一个限值所需的最小脉冲数。 |
| bHoldWith OutputOn | BOOL | 如果该参数被设置为 TRUE，则在驱动器静止时，输出将保持设置状态。这将导致制动。 |
| nAdditionalPulsesAtLimits | DINT | 为可靠确保达到限值而输出的补充脉冲数。 |

| 名称 | 类型 | 描述 |
|---|---|---|
| bMoveOnLimit Switch | BOOL | 如果此值为 TRUE，则当控制值为 0% 或 100% 时，将输出脉冲，直至达到限位开关。 |
| fCtrlOutMax | FLOAT | 将阀门驱动至 100% 的控制值。 |
| fCtrlOutMin | FLOAT | 将阀门驱动至 0% 的控制值。 |

### 4.2.1.8 Setpointgeneration

#### 4.2.1.8.1 FB_CTRL_3PHASE_SETPOINT_GENERATOR




该功能块代表一个三相设定值发生器。

**描述**

该功能块可生成一个三相设定值曲线，其中加速度呈矩形曲线。

在发生器工作时可以指定新的参数集。根据指定的参数集类型，可以立即激活它：`eNewParameterType := eCTRL_NEW_PARAMETER_TYPE_Instant`

或者，首先完成当前移动，然后使用新的参数集开始新的移动：
`eNewPosType := eCTRL_NEW_PARAMETER_TYPE_NotInstant`

**注意**

**超越终点位置**

指定一个新的参数集可能意味着超越先前的终点位置。请参见示例。
通常建议在设定值发生器之后加入终点位置监控。

**输入**

```
VAR_INPUT
    bStart        : BOOL;
    bStop         : BOOL;
    bReset        : BOOL;
    fOverride     : LREAL;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| bStart | BOOL | 当 `bStart` 输入端出现上升沿时，开始生成设定点，前提是发生器未启用且 `bStop` 和 `bReset` 输入为 FALSE。 |
| bStop | BOOL | `bStop` 输入端的上升沿将停止生成设定值。使用在当前参数集中指定的减速度进行制动，并删除任何可能待处理的后续任务。 |
| bReset | BOOL | 重置设定值发生器；任何可能正在进行的定位操作立即终止，输出 `fSetVelo` 和 `fSetAcc` 变为 0.0，设置位置被设置为起始位置，内部状态被删除。 |

| 名称 | 类型 | 描述 |
|---|---|---|
| fOverride | LREAL | 设置速度可以在此范围内通过覆写值进行缩放：[ 0 .. 100.0 % ]。如果覆写值为 100%，则生成的曲线使用在参数集中指定的设定速度。在覆写值发生更改时，此处实现的覆写不会对当前的运行时表进行缩放。相反，系统将生成一个具有不同设置速度的内部重启指令。覆写值的分辨率为 0.1%。 |

### 输出

```
VAR_OUTPUT
    fSetPos           : LREAL;
    fSetVelo          : LREAL;
    fSetAcc           : LREAL;
    nSetDirection     : INT;
    bCommandBuffered  : BOOL;
    bDone             : BOOL;
    bCommandAborted   : BOOL;
    eErrorId          : E_CTRL_ERRORCODES;
    bError            : BOOL;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| fSetPos | LREAL | 设定位置 |
| fSetVelo | LREAL | 设定速度 |
| fSetAcc | LREAL | 设定加速度 |
| nSetDirection | INT | 移动方向 [-1, 0, 1]，<br>1 --> 移动方向为正<br>0 --> 发生器不工作<br>1 --> 移动方向为负 |
| bCommandBuffered | BOOL | 当该输出为 TRUE 时，这表示已存储一个运动命令，该命令将在当前指令完成后开始执行。<br>如果将以下特殊情况指定为一个参数集，则已保存的指令将被删除。<br>`fAcceleration := 0.0;`<br>`fDeceleration := 0.0;`<br>`fStartPos := 0.0;`<br>`fStartVelo := 0.0;`<br>`fTargetPos := 0.0;`<br>`fTargetVelo := 0.0;`<br>`fVelocity := 0.0;`<br>`tCtrlCycleTime := T#0s;`<br>`tTaskCycleTime := T#0s;`<br>`eNewParameterType := eCTRL_NEW_PARAMETER_TYPE_NotInstant;` |
| bDone | BOOL | 当完成移动并到达目标位置时，该输出将变为 TRUE。 |
| bCommandAborted | BOOL | 在当前移动中断时，该输出将变为 TRUE。例如，`bStop` 输入端的上升沿可能导致这种情况。 |
| eErrorId | E_CTRL_ERRORCODES | 在设置输出 `bError` 时，提供错误编号 [▶ 157]。 |
| bError | BOOL | 一旦出现情况，则变为 TRUE。 |

### 输入/输出

```
VAR_IN_OUT
    stParams : ST_CTRL_3PHASE_SETPOINT_GENERATOR_PARAMS;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| stParams | ST_CTRL_3PHASE_SETPOINT_GENERATOR_PARAMS | 设定值发生器的参数结构 |

`stParams` 由以下元素组成：

```
TYPE ST_CTRL_RAMP_GENERATOR_PARAMS :
STRUCT
    tTaskCycleTime       : TIME;
    tCtrlCycleTime       : TIME;
    fStartPos            : LREAL;
    fStartVelo           : LREAL;
    fVelocity            : LREAL; (* >= 0.0 *)
    fTargetPos           : LREAL;
    fTargetVelo          : LREAL;
    fAcceleration        : LREAL; (* > 0.0 *)
    fDeceleration        : LREAL; (* > 0.0 *)
    eNewParameterType    : E_CTRL_NEW_PARAMETER_TYPE;
EN_STRUCT
END_TYPE
```

| 名称 | 类型 | 描述 |
|---|---|---|
| tTaskCycleTime | TIME | 处理控制回路的循环时间。该时间必须大于或等于 **TaskCycleTime**。功能块使用该输入值在内部计算是否需要在当前周期内更新状态和输出值。 |
| tCtrlCycleTime | TIME | 调用功能块的循环时间。若功能块在每个任务周期内均被调用，则其周期时间等同于调用任务的任务周期时间。 |
| fStartPos | LREAL | 运动轨迹的起始位置 |
| fStartVelo | LREAL | 运动轨迹的起始速度 |
| fVelocity | LREAL | 速度（单位/秒） |
| fTargetPos | LREAL | 运动轨迹的目标位置 |
| fTargetVelo | LREAL | 运动轨迹的目标速度。<br>**注意：**在到达目标位置时（bDone 标志已设置），目标速度会被保留，但此后系统将停止位置计算（速度 ≠ 0.0 时，位置不变）。 |
| fAcceleration | LREAL | 加速度（单位/秒$^2$） |
| fDeceleration | LREAL | 减速度（单位/秒$^2$） |
| eNewParameter Type | E_CTRL_NEW_ PARAMETER_ TYPE | `eCTRL_NEW_PARAMETER_TYPE_Instant`<br>和<br>`eCTRL_NEW_PARAMETER_TYPE_NotInstant` |

### eNewParameterType 的特殊特性

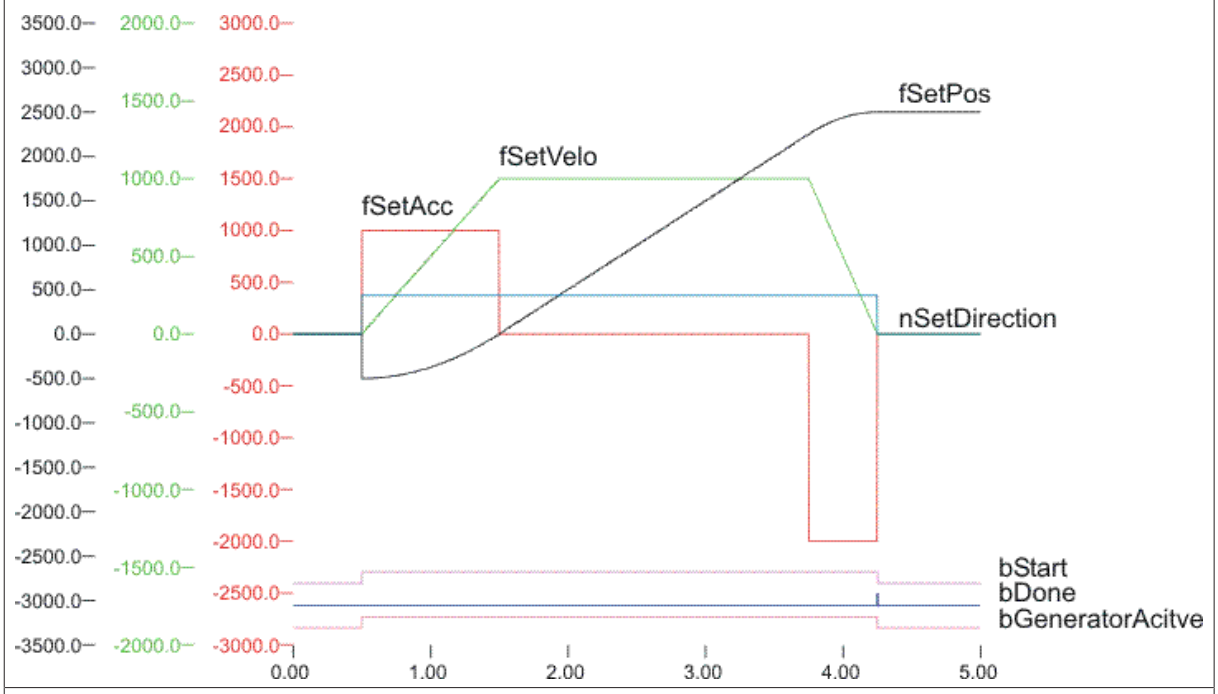
```
TYPE E_CTRL_NEW_PARAMETER_TYPE :
(    eCTRL_NEW_PARAMETER_TYPE_NotInstant := 0,
    eCTRL_NEW_PARAMETER_TYPE_Instant := 1);
END_TYPE
```

**eCTRL_NEW_PARAMETER_TYPE_Instant**：当发出带有新参数集的重启指令时，系统将立即采用该参数集。换句话说，系统将计算从当前移动状态到新参数集所代表的数据的过渡，并丢弃旧参数。

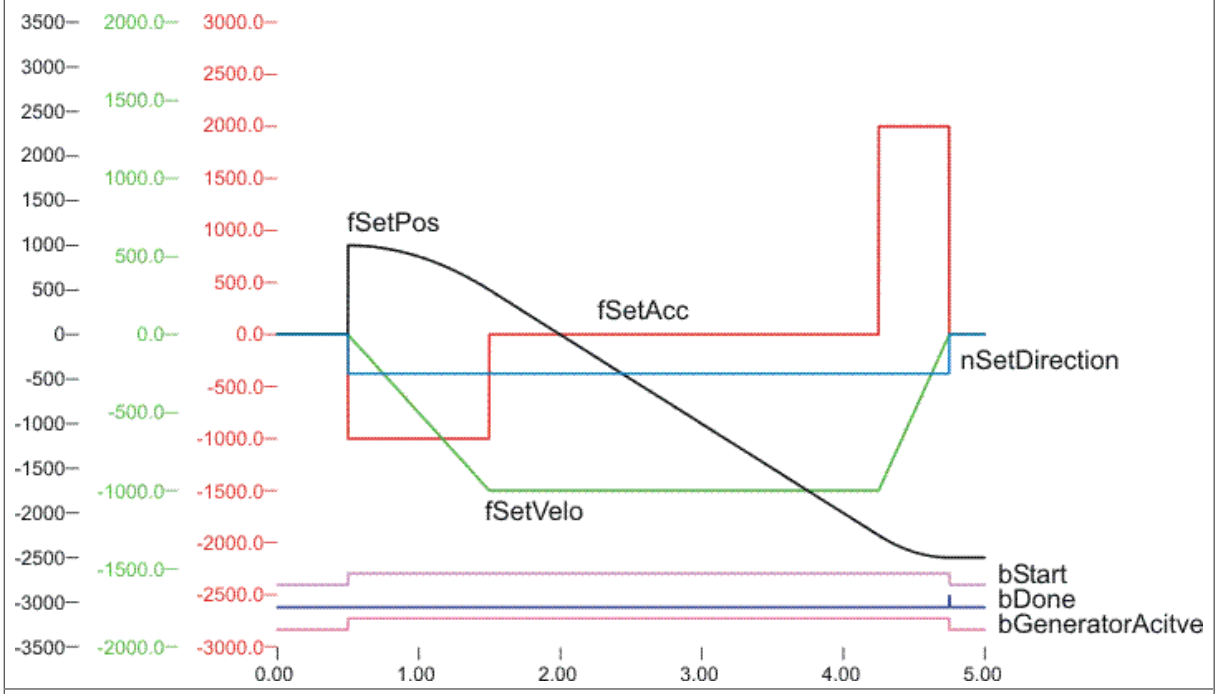
**eCTRL_NEW_PARAMETER_TYPE_NotInstant**：当发出带有新参数集的重启指令时，系统不会立即采用新参数集。换句话说，系统将首先执行当前移动至完成，然后使用新参数执行对新目标的定位。
**bCommandBuffered** 输出为 TRUE 时，表示已存储一个非瞬时重启指令。一个新的非瞬时参数集可以覆盖或删除一个已存储的指令。

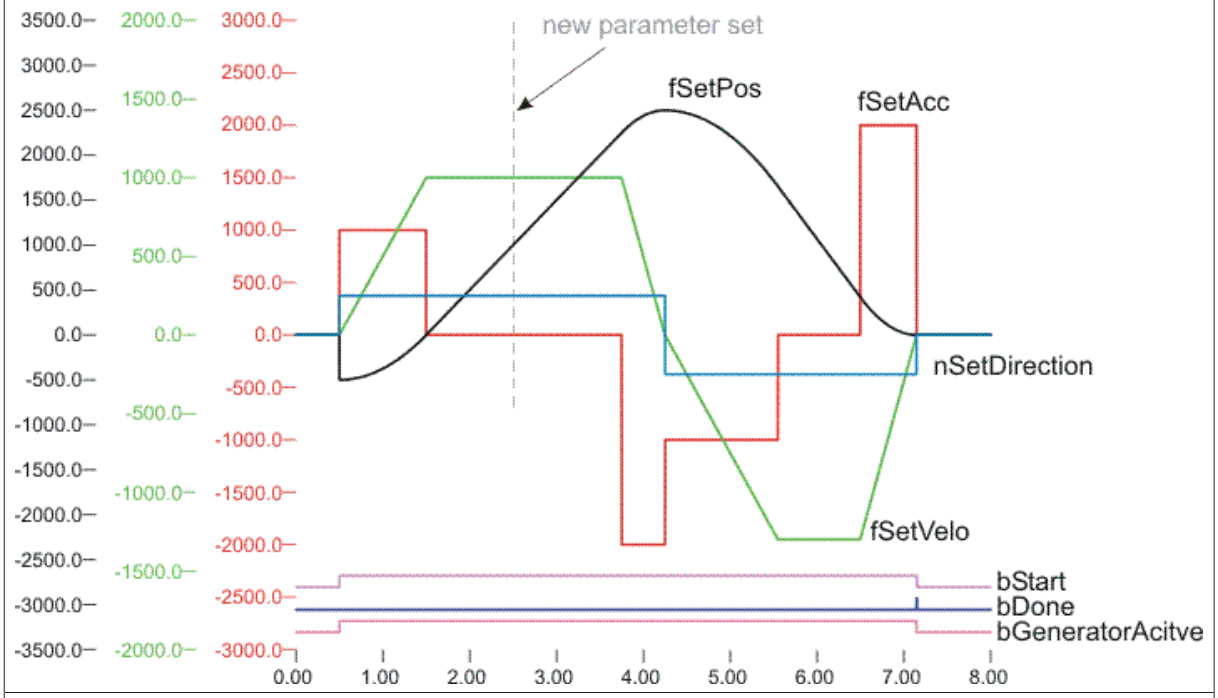
### 定位示例

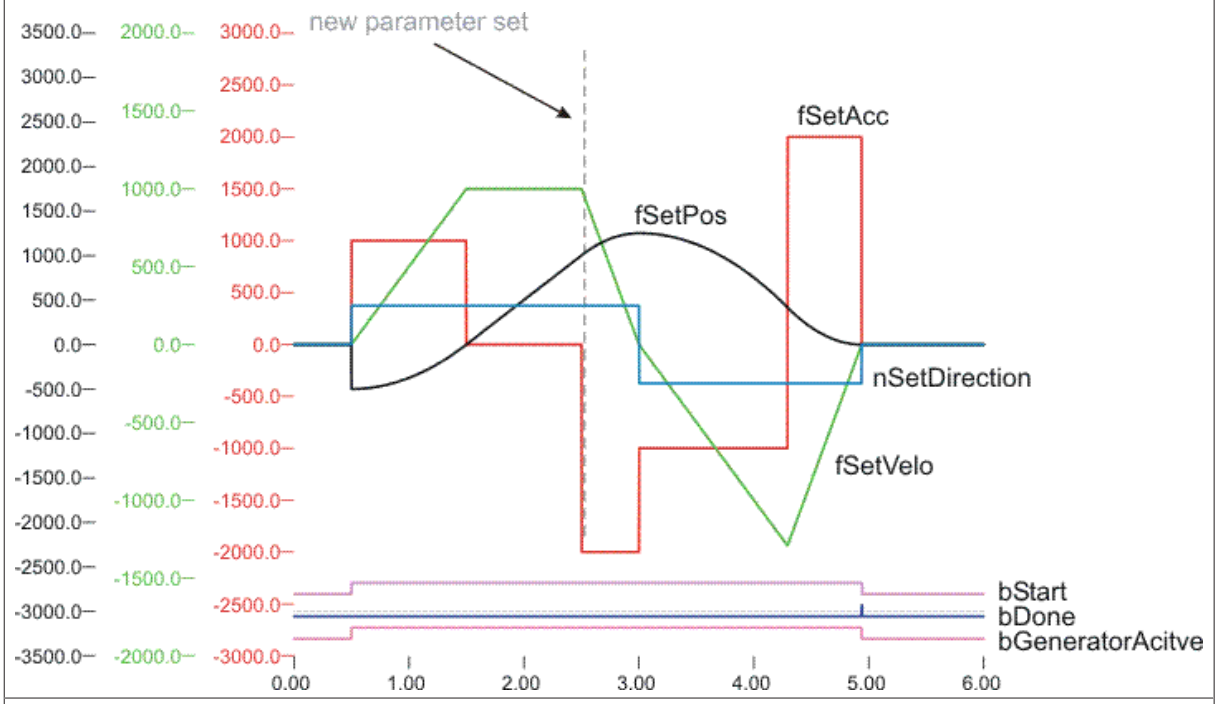
| | |
|---|---|
| **定位示例 1：** | stParams.fStartPos := -500.0;<br>stParams.fTargetPos := 2500.0;<br>stParams.fStartVelo := 0.0; stParams.fVelocity := 1000.0; stParams.fTargetVelo := 0.0;<br>stParams.fAcceleration := 1000.0;<br>stParams.fDeceleration := 2000.0; fOverride := 100.0; |



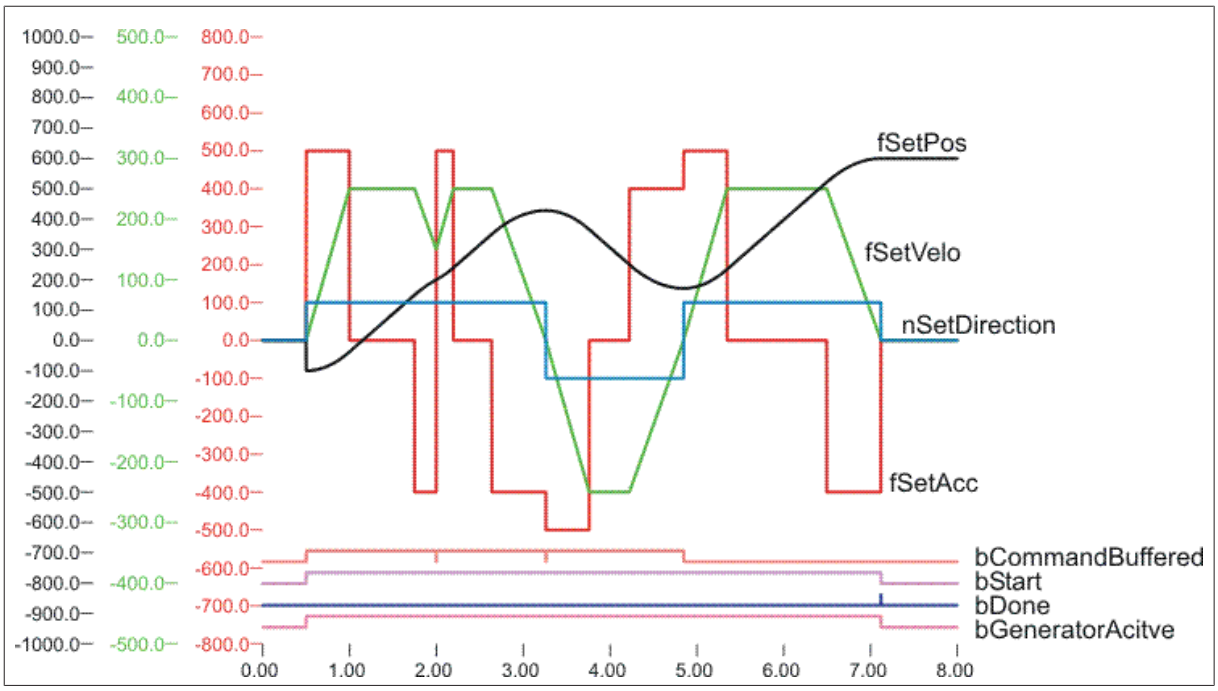

| | |
|---|---|
| **定位示例 2：** | stParams.fStartPos := 1000.0;<br>stParams.fTargetPos := -2500.0;<br>stParams.fStartVelo := 0.0; stParams.fVelocity := 1000.0; stParams.fTargetVelo := 0.0;<br>stParams.fAcceleration := 1000.0;<br>stParams.fDeceleration := 2000.0; fOverride := 100.0; |



| | |
|---|---|
| **定位示例 3：** | stParams.fStartPos := -500.0;<br>stParams.fTargetPos := 2500.0;<br>stParams.fStartVelo := 0.0; stParams.fVelocity := 1000.0; stParams.fTargetVelo := 0.0;<br>stParams.fAcceleration := 1000.0;<br>stParams.fDeceleration := 2000.0;<br>stParams.eNewParameterType := eCTRL_NEW_PARAMETER_TYPE_NotInstant;<br>fOverride := 100.0; **如果 fSetPos > 1000.0，eNewPosType := eCTRL_NEW_POS_TYPE_NotInstant，则参数更改**stParams.fTargetPos := 0.0;<br>stParams.fStartVelo := 0.0; stParams.fVelocity := 1300.0; stParams.fTargetVelo := 0.0;<br>stParams.fAcceleration := 1000.0;<br>stParams.fDeceleration := 2000.0; fOverride := 100.0; |
|  | |
| | |
| **定位示例 4：** | stParams.fStartPos := -500.0;<br>stParams.fTargetPos := 2500.0;<br>stParams.fStartVelo := 0.0; stParams.fVelocity := 1000.0; stParams.fTargetVelo := 0.0;<br>stParams.fAcceleration := 1000.0;<br>stParams.fDeceleration := 2000.0;<br>stParams.eNewParameterType := eCTRL_NEW_PARAMETER_TYPE_NotInstant;<br>fOverride := 100.0;**如果 fSetPos > 1000.0，eNewPosType := eCTRL_NEW_POS_TYPE_Instant，则参数更改**<br>stParams.fTargetPos := 0.0;<br>stParams.fStartVelo := 0.0; stParams.fVelocity := 1300.0; stParams.fTargetVelo := 0.0;<br>stParams.fAcceleration := 1000.0;<br>stParams.fDeceleration := 2000.0; fOverride := 100.0; |




| | |
|---|---|
| **定位示例 5：** | **在点 1 开始：** stParams.fStartPos := -100.0;<br>stParams.fTargetPos := 200.0;<br>stParams.fStartVelo := 0.0; stParams.fVelocity := 250.0; stParams.fTargetVelo := 150.0;<br>stParams.fAcceleration := 500.0;<br>stParams.fDeceleration := 400.0;<br>stParams.eNewParameterType := eCTRL_NEW_PARAMETER_TYPE_Instant**在点 2 重启：**<br>stParams.fTargetPos := 400.0;<br>stParams.eNewParameterType := eCTRL_NEW_PARAMETER_TYPE_NotInstant;**在点 3 重启：**<br>stParams.fTargetPos := 200.0;<br>stParams.eNewParameterType := eCTRL_NEW_PARAMETER_TYPE_NotInstant;**在点 4 重启：**<br>stParams.fTargetPos := 600.0;<br>stParams.fTargetVelo := 0.0;<br>stParams.eNewParameterType := eCTRL_NEW_PARAMETER_TYPE_NotInstant; |




**注意**

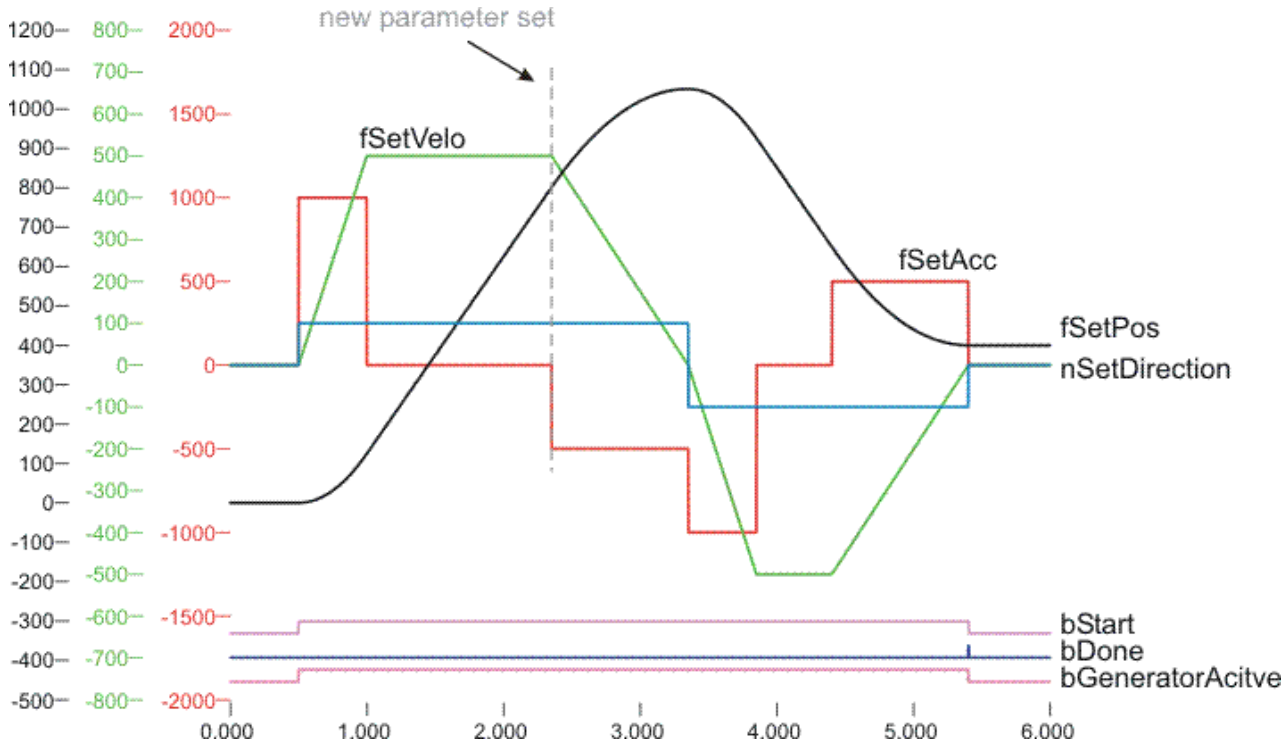
**目标位置超调!**

如果向功能块提供一个类型为“eCTRL_NEW_POS_TYPE_Instant”且减速度降低的新参数集，则可能会超越旧的目标位置。

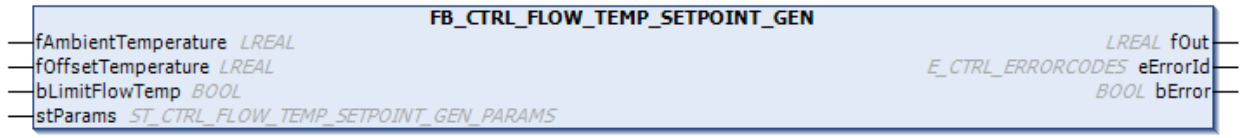
**示例:**

```
stParams.fTargetPos := 1000.0;
stParams.fStartPos := 0.0;
stParams.fVelocity := 500.0;
stParams.fAcceleration := 1000.0;
stParams.fDeceleration := 1000.0;
IF fSetPos > 800.0 THEN
stParams.fTargetPos := 400.0;
stParams.fVelocity := 500.0;
stParams.fAcceleration := 1_000.0;
stParams.fDeceleration := 500.0;
stParams.eNewPosType := eCTRL_NEW_POS_TYPE_Instant;
END_IF
```

从下面的范围追踪中可以清楚地看到，原始的 1000 mm 目标位置已被超越，原因是新参数集的减速度已降低。




## 4.2.1.8.2 FB_CTRL_FLOW_TEMP_SETPOINT_GEN

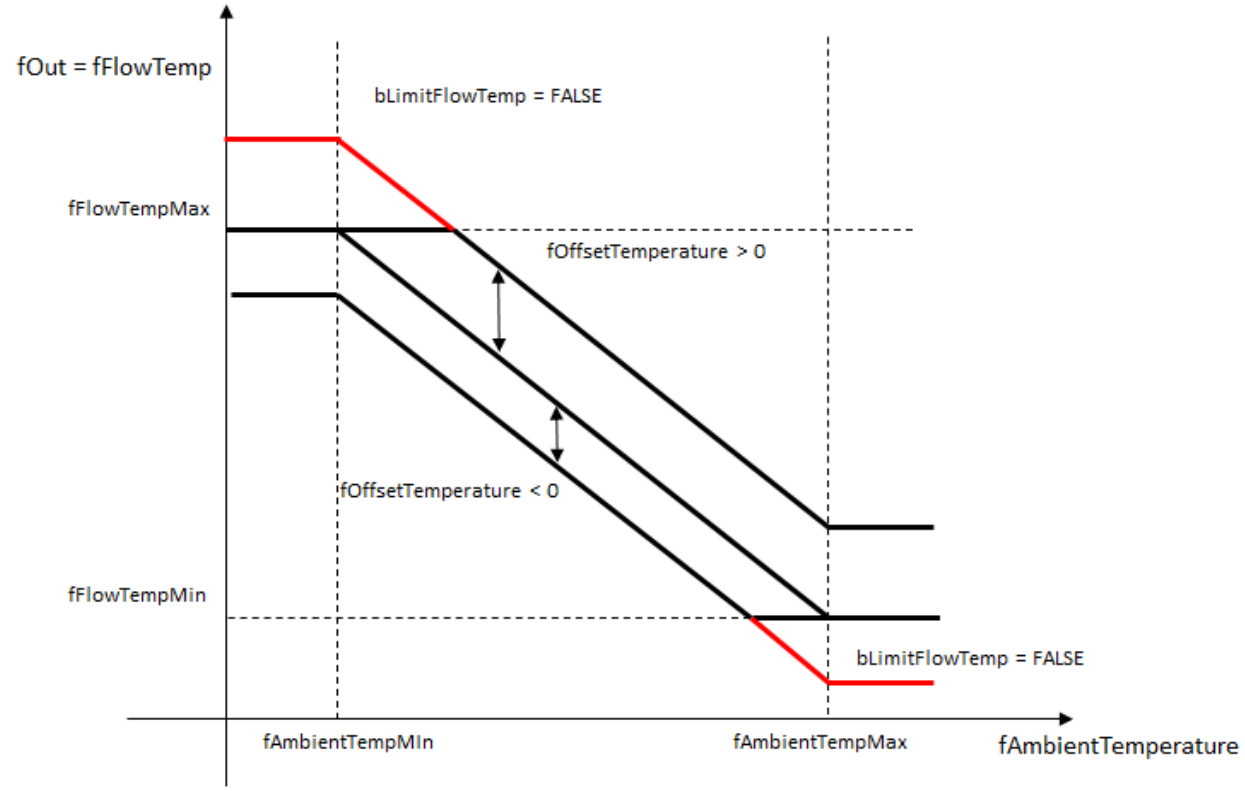



该功能块可根据室外温度设定流量温度。

### 描述

流量温度（`fOut`）的设定值由环境温度（`fAmbientTemperature`）决定。为此，采用了一条可通过偏移量（`fOffsetTemperature`）移动的直线。斜率根据指定的环境温度和流量温度拐点确定。标志（`bLimitFlowTemp`）用于指定流量温度是否受其限值限制。偏移温度可用于实现夜间降温或前馈控制。

### 输出值的行为




### 输入

```
VAR_INPUT
    fAmbientTemperature   : FLOAT;
    fOffsetTemperature    : FLOAT;
    bLimitFlowTemp        : BOOL;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| fAmbient Temperature | FLOAT | 斜坡生成的起点 |
| fOffset Temperature | FLOAT | 斜坡的起始值 |
| bLimitFlowTemp | BOOL | 斜坡的目标值 |

### 输出

```
VAR_OUTPUT
    fOut       : FLOAT;
    eErrorId   : E_CTRL_ERRORCODES;
    bError     : BOOL;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| fOut | FLOAT | 流量温度的设定值 |
| eErrorId | E_CTRL_ERRORCODES | 在设置输出 `bError` 时，提供错误编号 [▶ 157]。 |
| bError | BOOL | 一旦出现情况，则变为 TRUE。 |

### 输入/输出

```
VAR_IN_OUT
    stParams     : ST_CTRL_FLOW_TEMP_SETPOINT_GEN_PARAMS;
END_VAR
```

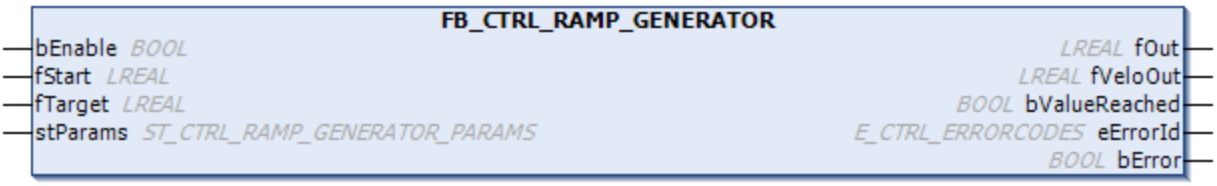
| 名称 | 类型 | 描述 |
|---|---|---|
| stParams | ST_CTRL_FLOW_TEMP_SETPOINT_GEN_PARAMS | 斜坡发生器的参数结构 |

`stParams` 由以下元素组成：

```
TYPE ST_CTRL_FLOW_TEMP_SETPOINT_GEN_PARAMS:
STRUCT
    tTaskCycleTime    : TIME;
    tCtrlCycleTime    : TIME;
    fForeRunTempMax   : FLOAT;
    fForeRunTempMin   : FLOAT;
    fAmbientTempMax   : FLOAT;
    fAmbientTempMin   : FLOAT;
END_STRUCT
END_TYPE
```

| 名称 | 类型 | 描述 |
|---|---|---|
| tTaskCycleTime | TIME | 调用功能块的循环时间。若功能块在每个任务周期内均被调用，则其周期时间等同于调用任务的任务周期时间。 |
| tCtrlCycleTime | TIME | 处理控制回路的循环时间。该时间必须大于或等于 **TaskCycleTime**。功能块使用该输入值在内部计算是否需要在当前周期内更新状态和输出值。 |
| fForeRunTempMax | FLOAT | 最高流量温度（请参见图示） |
| fForeRunTempMin | FLOAT | 最低流量温度（请参见图示） |
| fAmbientTempMax | FLOAT | 指定最低流量温度的室外温度。 |
| fAmbientTempMin | FLOAT | 指定最高流量温度的室外温度。 |

## 4.2.1.8.3 FB_CTRL_RAMP_GENERATOR

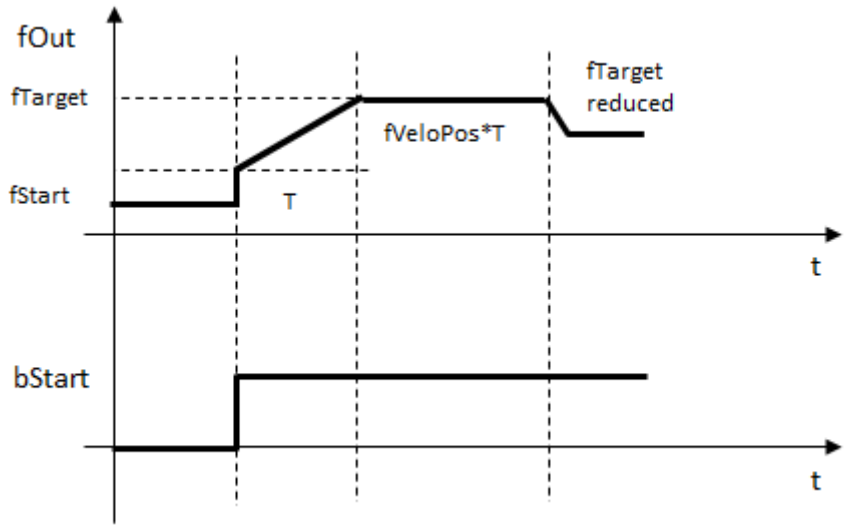



该功能块提供了一个可参数化的斜坡发生器。

### 描述

该功能块生成了一个连接起始值 `fStart` 和目标值 `fTarget` 的斜坡。通过 `fVeloPos` 和 `fVeloNeg` 参数提供斜坡的斜率（即速度），以单位/秒为单位。当 `bEnable` 出现上升沿时，系统将采用起始值；然后开始计算斜坡。只要信号 `bEnable` 保持为 TRUE，即可更改目标值，输出值也会改变，按斜坡形式从当前值移动到当前生效的目标值。

### 输出值的行为




### 输入

```
VAR_INPUT
    bEnable   : BOOL;
    fStart    : FLOAT;
    fTarget   : FLOAT;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| bEnable | BOOL | 斜坡生成的起点 |
| fStart | FLOAT | 斜坡的起始值 |
| fTarget | FLOAT | 斜坡的目标值 |

### 输出

```
VAR_OUTPUT
    fOut            : FLOAT;
    fVeloOut        : FLOAT;
    bValueReached   : BOOL;
    eState          : E_CTRL_STATE;
    eErrorId        : E_CTRL_ERRORCODES;
    bError          : BOOL;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| fOut | FLOAT | 斜坡发生器的输出 |
| fVeloOut | FLOAT | 斜坡发生器的当前速度 |
| bValueReached | BOOL | 该输出为 TRUE 时，表示输出 `fOut` 已达到值 `fTarget`。 |
| eState | E_CTRL_STATE | 功能块的状态 |
| eErrorId | E_CTRL_ERRORCODES | 在设置输出 `bError` 时，提供错误编号 [▶ 157]。 |
| bError | BOOL | 一旦出现情况，则变为 TRUE。 |

### 输入/输出

```
VAR_IN_OUT
    stParams     : ST_CTRL_RAMP_GENERATOR_PARAMS
END_VAR
```

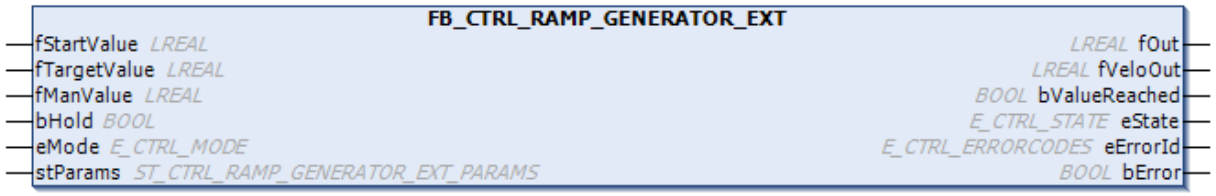
| 名称 | 类型 | 描述 |
|---|---|---|
| stParams | ST_CTRL_RAMP_GENERATOR_PARAMS | 斜坡发生器的参数结构 |

`stParams` 由以下元素组成：

```
TYPE ST_CTRL_RAMP_GENERATOR_PARAMS :
STRUCT
    tTaskCycleTime  : TIME;
    tCtrlCycleTime  : TIME;
    fVeloPos        : FLOAT;
    fVeloNeg        : FLOAT;
END_STRUCT
END_TYPE
```

| 名称 | 类型 | 描述 |
|---|---|---|
| tTaskCycleTime | TIME | 处理控制回路的循环时间。该时间必须大于或等于 **TaskCycleTime**。功能块使用该输入值在内部计算是否需要在当前周期内更新状态和输出值。 |
| tCtrlCycleTime | TIME | 调用功能块的循环时间。若功能块在每个任务周期内均被调用，则其周期时间等同于调用任务的任务周期时间。 |
| fVeloPos | FLOAT | 输出从较低值变为较高值的速度，以单位/秒为单位。 |
| fVeloNeg | FLOAT | 输出从较高值变为较低值的速度，以单位/秒为单位。 |

## 4.2.1.8.4 FB_CTRL_RAMP_GENERATOR_EXT

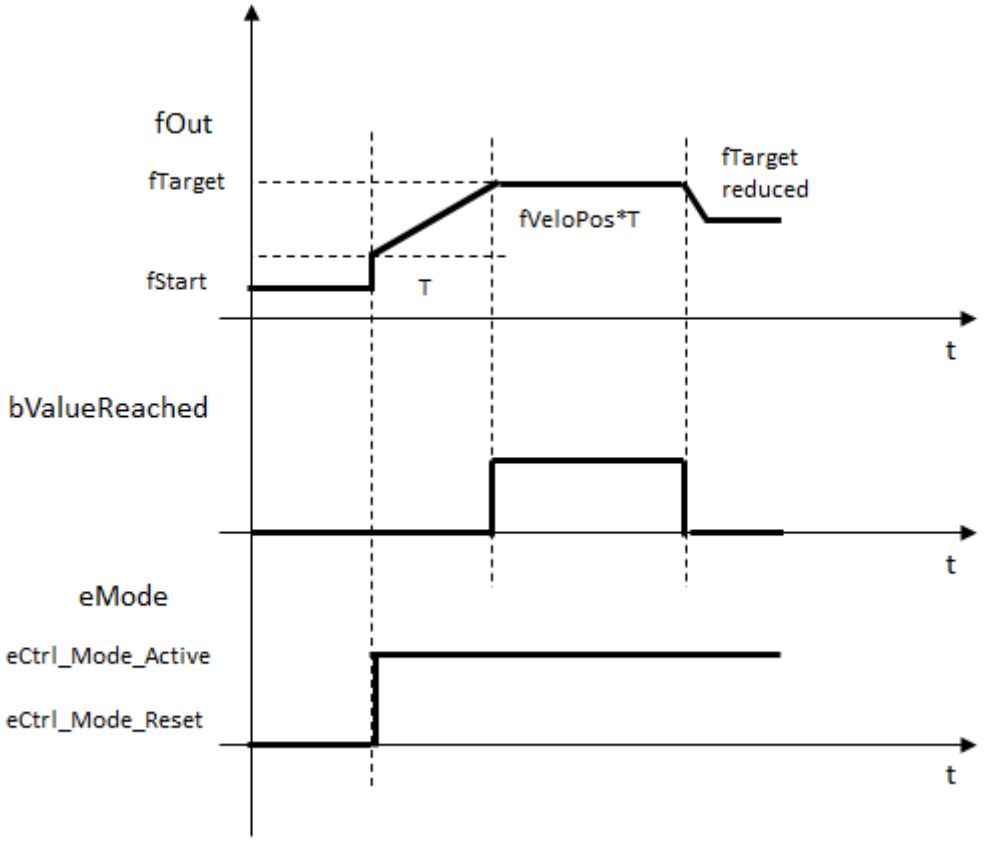



该功能块代表一个可参数化的斜坡发生器。与 **FB_CTRL_RAMP_GENERATOR**不同，它支持 **E_CTRL_MODE**。

**描述：**

该功能块生成了一个连接起始值 `fStartValue` 和目标值 `fTargetValue` 的斜坡。通过 `fVeloPos` 和 `fVeloNeg` 参数提供斜坡的斜率（即速度），以单位/秒为单位。当 **eCTRL_MODE_RESET** 变为 **eCTRL_MODE_ACTIVE** 时，系统将采用起始值，并开始计算斜坡。只要功能块保持在 **eCTRL_MODE_ACTIVE** 模式下，即可更改目标值，输出值也会改变，按斜坡形式从当前值移动到当前生效的目标值。在 `fVeloOut` 输出当前速度。可以将其用于控制回路中的前馈。

### 输出值的行为




### 输入

```
VAR_INPUT
    fStartValue    : FLOAT;
    fTargetValue   : FLOAT;
    fManValue      : FLOAT;
    bHold          : BOOL;
    eMode          : E_CTRL_MODE;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| fStartValue | FLOAT | 斜坡的起始值 |
| fTargetValue | FLOAT | 斜坡的目标值 |
| fManValue | FLOAT | eCTRL_MODE_MANUAL 中的输出所设定的输入值。 |
| bHold | BOOL | 斜坡计算在当前值处停止。 |
| eMode | E_CTRL_MODE | 指定功能块的运行模式 [▶ 157]的输入。 |

### 输出

```
VAR_OUTPUT
    fOut           : FLOAT;
    fVeloOut       : FLOAT;
    bValueReached  : BOOL;
    eState         : E_CTRL_STATE;
    eErrorId       : E_CTRL_ERRORCODES;
    bError         : BOOL;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| fOut | FLOAT | 斜坡发生器的输出 |
| fVeloOut | FLOAT | 斜坡发生器的当前速度 |
| bValueReached | BOOL | 该输出为 TRUE 时，表示输出 fOut 已达到值“fTargetValue”。 |

| 名称 | 类型 | 描述 |
|---|---|---|
| eState | E_CTRL_<br>STATE | 功能块的状态 |
| eErrorId | E_CTRL_<br>ERRORCODES | 在设置输出 bError 时，提供错误编号 [▶ 157]。 |
| bError | BOOL | 一旦出现情况，则变为 TRUE。 |

### 输入/输出

```
VAR_IN_OUT
    stParams    : ST_CTRL_RAMP_GENERATOR_EXT_PARAMS;
END_VAR
```

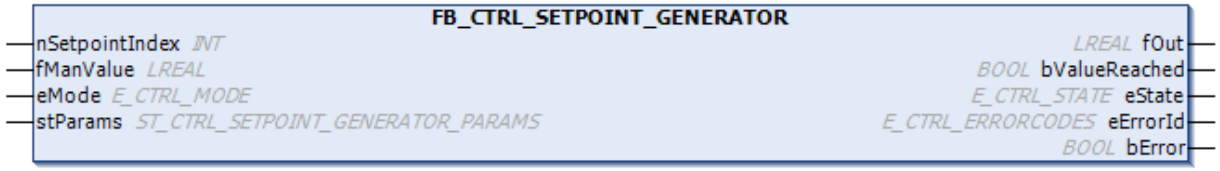
| 名称 | 类型 | 描述 |
|---|---|---|
| stParams | ST_CTRL_RAMP_<br>GENERATOR_EXT<br>_<br>PARAMS | 斜坡发生器的参数结构 |

stParams 由以下元素组成：

```
TYPE ST_CTRL_RAMP_GENERATOR_EXT_PARAMS :
STRUCT
    tTaskCycleTime  : TIME;
    tCtrlCycleTime  : TIME;
    fVeloPos        : FLOAT;
    fVeloNeg        : FLOAT;
END_STRUCT
END_TYPE
```

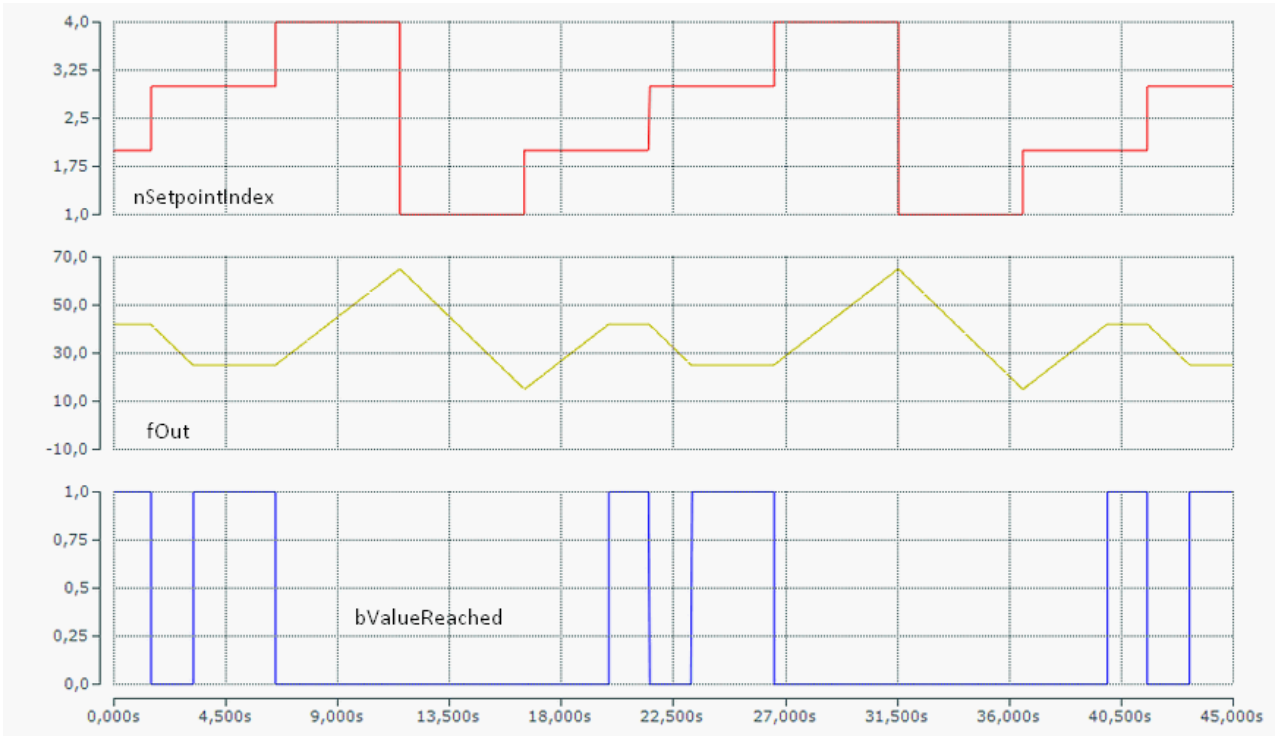
| 名称 | 类型 | 描述 |
|---|---|---|
| tTaskCycleTime | TIME | 处理控制回路的循环时间。该时间必须大于或等于 **TaskCycleTime**。功能块使用该输入值在内部计算是否需要在当前周期内更新状态和输出值。 |
| tCtrlCycleTime | TIME | 调用功能块的循环时间。若功能块在每个任务周期内均被调用，则其周期时间等同于调用任务的任务周期时间。 |
| fVeloPos | FLOAT | 输出从较低值变为较高值的速度（>0.0），以单位/秒为单位。 |
| fVeloNeg | FLOAT | 输出从较高值变为较低值的速度（>0.0），以单位/秒为单位。 |

## 4.2.1.8.5 FB_CTRL_SETPOINT_GENERATOR




该功能块提供了一个设定值发生器，可输出从表中选择的设定值。从一个设定值到另一个设定值的转换可以连续进行，也可以不连续进行。

### 输出值的行为




### 示例表：

| 索引 | |
|---|---|
| 1 | 12 |
| 2 | 42 |
| 3 | 25 |
| … | 73 |

### 描述

各个设定值存储在数组中。通过适当的参数必须向功能块传递该数组。通过 `nSetpointIndex` 输入选择在表中存储的一个设定值。然后，在输出端可以将其用作控制器的设定值。从一个值到另一个值的转换可以是线性的，也可以是跳跃的。由 fVeloPos 和 fVeloNeg 参数指定连续转换的速度。`bValueReached` 输出表示已达到所选设定值。

### 输入

```
VAR_INPUT
    nSetpointIndex  : INT;
    fManValue       : FLOAT;
    eMode           : E_CTRL_MODE;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| nSetpointIndex | INT | 所选设定值的索引 |
| fManValue | FLOAT | 在手动模式下，其数值将被输出的输入。 |
| eMode | E_CTRL_MODE | 指定功能块的运行模式 [▶ 157]的输入。 |

### 输出

```
VAR_OUTPUT
    fOut            : SETPOINT_TABLE_ELEMENT;
    bValueReached   : BOOL;
    eState          : E_CTRL_STATE;
```

```
    eErrorId        : E_CTRL_ERRORCODES;
    bError          : BOOL;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| fOut | SETPOINT_ TABLE_ELEMENT | 设定值发生器的输出 |
| bValueReached | BOOL | 当达到所选设定值时，输出为 TRUE，即通向选定值的斜坡过程已完成。 |
| eState | E_CTRL_STATE | 功能块的状态 |
| eErrorId | E_CTRL_ERRORCODES | 在设置输出 bError 时，提供错误编号 [▶ 157]。 |
| bError | BOOL | 一旦发生错误，则变为 TRUE。 |

### 输入/输出

```
VAR_IN_OUT
    stParams    : ST_CTRL_SETPOINT_GENERATOR_PARAMS;
END_VAR
```

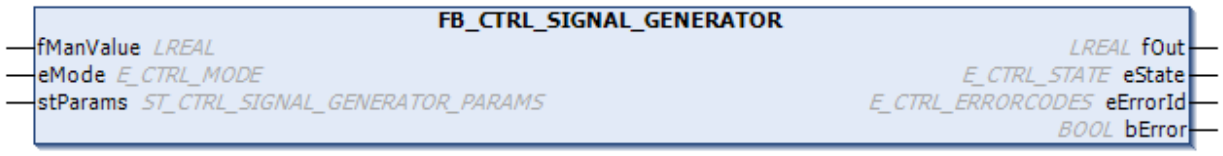
| 名称 | 类型 | 描述 |
|---|---|---|
| stParams | ST_CTRL_SETPOINT_ GENERATOR_ PARAMS | 斜坡发生器的参数结构 |

stParams 由以下元素组成：

```
TYPE ST_CTRL_SETPOINT_GENERATOR_PARAMS:
STRUCT
    tCtrlCycleTime          : TIME := T#0ms;
    tTaskCycleTime          : TIME := T#0ms;
    pDataTable_ADR          : POINTER TO
     INTERPOLATION_TABLE_ELEMENT := 0;
    nDataTable_SIZEOF       : UINT := 0;
    nDataTable_NumberOfRows : UINT := 0;
    fVeloPos                : FLOAT;
    fVeloNeg                : FLOAT;
    bDisableRamping         : BOOL := FALSE;
END_STRUCT
END_TYPE
```

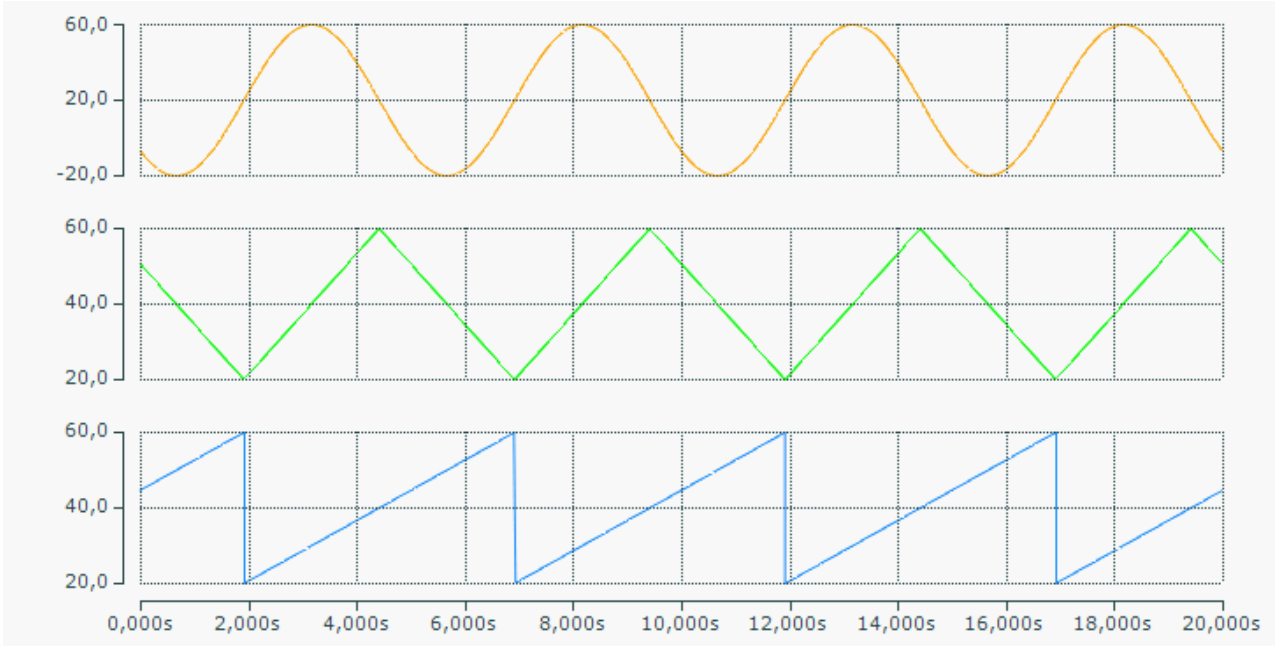
| 名称 | 类型 | 描述 |
|---|---|---|
| tCtrlCycleTime | TIME | 处理控制回路的循环时间。该时间必须大于或等于 **TaskCycleTime**。功能块使用该输入值在内部计算是否需要在当前周期内更新状态和输出值。<br>缩进：-103；左边距：108"> |
| tTaskCycleTime | TIME | 调用功能块的循环时间。如果在每个循环中都调用该功能块，这相当于调用任务的任务循环时间。 |
| pDataTable_ADR | POINTER TO INTERPOLATION_ TABLE_ELEMENT | 数据数组的地址 |
| nDataTable_SIZEOF | UINT | 数据数组的大小 |
| nDataTable_NumberOfRows | UINT | 数据数组中的行数 |
| fVeloPos | FLOAT | 输出从较低值变为较高值的速度，以单位/秒为单位。 |
| fVeloNeg | FLOAT | 输出从较高值变为较低值的速度，以单位/秒为单位。 |
| bDisableRamping | BOOL | 如果该参数为 TRUE，则不计算连续输出值。输出值之间存在阶跃变化。 |

## 4.2.1.8.6 FB_CTRL_SIGNAL_GENERATOR




该功能块提供了一个信号发生器，可生成**三角波**、**正弦波**和**锯齿波**信号。

**输出信号**




### 输入

```
VAR_INPUT
    fManValue   : FLOAT;
    eMode       : E_CTRL_MODE;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| fManValue | FLOAT | 在手动模式下，其数值直接出现在输出端的输入。 |
| eMode | E_CTRL_MODE | 指定功能块的运行模式 [▶ 157]的输入。 |

### 输出

```
VAR_OUTPUT
    fOut        : FLOAT;
    eState      : E_CTRL_STATE;
    eErrorId    : E_CTRL_ERRORCODES;
    bError      : BOOL;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| fOut | FLOAT | 信号发生器的输出 |
| eState | E_CTRL_STATE | 功能块的状态 |
| eErrorId | E_CTRL_ERRORCODES | 在设置输出 bError 时，提供错误编号 [▶ 157]。 |
| bError | BOOL | 一旦发生错误，则变为 TRUE。 |

### 输入/输出

```
VAR_IN_OUT
    stParams    : ST_CTRL_SIGNAL_GENERATOR_PARAMS;
END_VAR
```

| 名称 | 类型 | 描述 |
|---|---|---|
| stParams | ST_CTRL_SIGNAL_GENERATOR_PARAMS | 功能块的参数结构 |

`stParams` 由以下元素组成：

```
TYPE ST_CTRL_SIGNAL_GENERATOR_PARAMS:
STRUCT
    tCtrlCycleTime  : TIME := T#0ms;
    tTaskCycleTime  : TIME := T#0ms;
    eSignalType     : E_CTRL_SIGNAL_TYPE;
    tCylceDuration  : TIME;
    fAmplitude      : FLOAT;
    fOffset         : FLOAT := 0.0;
    tStart          : TIME := T#0s;
END_STRUCT
END_TYPE
```

| 名称 | 类型 | 描述 |
|---|---|---|
| tCtrlCycleTime | TIME | 处理控制回路的循环时间。该时间必须大于或等于 **TaskCycleTime**。功能块使用该输入值在内部计算是否需要在当前周期内更新状态和输出值。 |
| tTaskCycleTime | TIME | 调用功能块的循环时间。如果在每个循环中都调用该功能块，这相当于调用任务的任务循环时间。 |
| eSignalType | E_CTRL_SIGNAL_TYPE | 选择信号类型。<br>`TYPE E_CTRL_SIGNAL_TYPE :`<br>`( eCTRL_TRIANGLE := 0,`<br>`eCTRL_SINUS := 1,`<br>`eCTRL_SAWTOOTH := 2 );`<br>`END_TYPE` |
| tCylceDuration | TIME | 生成信号曲线的周期 |
| fAmplitude | FLOAT | 生成信号曲线的振幅 |
| fOffset | FLOAT | 信号曲线的附加偏移量 |
| tStart | TIME | 当切换到 **eCTRL_MODE_ACTIVE** 时，系统将开始跟踪信号曲线的周期时刻。 |

## 4.2.2 全局常量

### 4.2.2.1 库版本

所有库都有特定版本。此版本也在 PLC 库仓库中显示。
全局常量包含库版本信息：

**Global_Version**

```
VAR_GLOBAL CONSTANT
    stLibVersion_Tc2_ControllerToolbox : ST_LibVersion;
END_VAR
```

ST_LibVersion

为了对比现有版本与所需版本，系统提供了函数 F_CmpLibVersion（在 Tc2_System 库中定义）。

**弃用函数**
所有其他从 TwinCAT2 库中查询库版本的已知方法均已弃用！

## 4.2.3 数据结构

### 4.2.3.1 结构和枚举的定义

本附录介绍了 TwinCAT Controller Toolbox 中使用的所有结构和枚举。

**FLOAT：**

库的功能块仅支持数据类型 FLOAT。该数据类型在补充库中被定义为 LREAL 或 REAL。

在 **PC 系统**中，可自动集成附加库“**TcFloatPC.lib**”。

```
TYPE
FLOAT : LREAL;
END_TYPE
```

**E_CTRL_MODE**

```
TYPE E_CTRL_MODE :
(
    eCTRL_MODE_IDLE     := 0,
    eCTRL_MODE_PASSIVE     := 1,
    eCTRL_MODE_ACTIVE     := 2,
    eCTRL_MODE_RESET     := 3,
    eCTRL_MODE_MANUAL     := 4,
    eCTRL_MODE_TUNE     := 5,
    eCTRL_MODE_SELFTEST     := 6,
    eCTRL_MODE_SYNC_MOVEMENT := 7
)
END_TYPE
```

**E_CTRL_STATE**

```
TYPE E_CTRL_STATE :
(
    eCTRL_STATE_IDLE        := 0,
    eCTRL_STATE_PASSIVE := 1,
    eCTRL_STATE_ACTIVE := 2,
    eCTRL_STATE_RESET     := 3,
    eCTRL_STATE_MANUAL := 4,
    eCTRL_STATE_TUNING := 5,
    eCTRL_STATE_TUNED     := 6,
    eCTRL_STATE_SELFTEST := 7,
    eCTRL_STATE_ERROR     := 8,
    eCTRL_STATE_SYNC_MOVEMENT := 9
);
END_TYPE
```

**E_CTRL_ERRORCODES：**

```
TYPE E_CTRL_ERRORCODES :
(
    eCTRL_ERROR_NOERROR                    := 0, (* no error *)
    eCTRL_ERROR_INVALIDTASKCYCLETIME                    := 1, (* invalid task cycle time *)
    eCTRL_ERROR_INVALIDCTRLCYCLETIME                    := 2, (* invalid ctrl cycle time *)
    eCTRL_ERROR_INVALIDPARAM                        := 3, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_Tv                := 4, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_Td                := 5, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_Tn                := 6, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_Ti                := 7, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_fHystereisisRange          := 8, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_fPosOutOn_Off          := 9, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_fNegOutOn_Off          := 10, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_TableDescription          := 11, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_TableData                := 12, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_DataTableADR          := 13, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_T0                    := 14, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_T1                    := 15, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_T2                := 16, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_T3                := 17, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_Theta                := 18, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_nOrder                    := 19, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_Tt                := 20, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_Tu                := 21, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_Tg                := 22, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_infinite_slope          := 23, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_fMaxIsLessThanfMin          := 24, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_fOutMaxLimitIsLessThanfOutMinLimit := 25, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_fOuterWindow          := 26, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_fInnerWindow          := 27, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_fOuterWindowIsLessThanfInnerWindow := 28, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_fDeadBandInput                := 29, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_fDeadBandOutput                := 30, (* invalid parameter *)
```

```
    eCTRL_ERROR_INVALIDPARAM_PWM_Cycletime          := 31, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_no_Parameterset              := 32, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_fOutOn                    := 33, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_fOutOff                    := 34, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_fGain                 := 35, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_fOffset                    := 36, (* invalid parameter *)
    eCTRL_ERROR_MODE_NOT_SUPPORTED                 := 37, (* invalid mode: mode not supported *)
    eCTRL_ERROR_INVALIDPARAM_Tv_heating                 := 38, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_Td_heating                 := 39, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_Tn_heating                 := 40, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_Tv_cooling                 := 41, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_Td_cooling                 := 42, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_Tn_cooling                 := 43, (* invalid parameter *)
    eCTRL_ERROR_RANGE_NOT_SUPPORTED                      := 44, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_nParameterChangeCycleTicks     := 45, (* invalid parameter *)
    eCTRL_ERROR_ParameterEstimationFailed          := 46, (* invalid parameter *)
    eCTRL_ERROR_NoiseLevelToHigh               := 47, (* invalid parameter *)
    eCTRL_ERROR_INTERNAL_ERROR_0               := 48, (* internal error *)
    eCTRL_ERROR_INTERNAL_ERROR_1               := 49, (* internal error *)
    eCTRL_ERROR_INTERNAL_ERROR_2               := 50, (* internal error *)
    eCTRL_ERROR_INTERNAL_ERROR_3               := 51, (* internal error *)
    eCTRL_ERROR_INTERNAL_ERROR_4               := 52, (* internal error *)
    eCTRL_ERROR_INTERNAL_ERROR_5               := 53, (* internal error *)
    eCTRL_ERROR_INTERNAL_ERROR_6               := 54, (* internal error *)
    eCTRL_ERROR_INTERNAL_ERROR_7               := 55, (* internal error *)
    eCTRL_ERROR_INTERNAL_ERROR_8               := 56, (* internal error *)
    eCTRL_ERROR_INTERNAL_ERROR_9               := 57, (* internal error *)
    eCTRL_ERROR_INVALIDPARAM_WorkArrayADR          := 58, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_tOnTiime                := 59, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_tOffTiime                := 60, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_nMaxMovingPulses          := 61, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_nAdditionalPulsesAtLimits     := 62, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_fCtrlOutMax_Min               := 63, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_fDeltaMax              := 64, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_tMovingTime          := 65, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_tDeadTime              := 66, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_tAdditionalMoveTimeAtLimits     := 67, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_fThreshold               := 68, (* invalid parameter *)
    eCTRL_ERROR_MEMCPY                     := 69, (* MEMCPY failed *)
    eCTRL_ERROR_MEMSET                     := 70, (* MEMSET failed *)
    eCTRL_ERROR_INVALIDPARAM_nNumberOfColumns          := 71, (* invalid parameter *)
    eCTRL_ERROR_FileClose                   := 72, (* File Close failed *)
    eCTRL_ERROR_FileOpen                   := 73, (* File Open failed *)
    eCTRL_ERROR_FileWrite                   := 74, (* File Write failed *)
    eCTRL_ERROR_INVALIDPARAM_fVeloNeg               := 75, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_fVeloPos               := 76, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_DeadBandInput          := 77, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_DeadBandOutput          := 78, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_CycleDuration          := 79, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_tStart                    := 80, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_StepHeigthTuningToLow     := 81, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_fMinLimitIsLessThanZero          := 82, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_fMaxLimitIsGreaterThan100     := 83, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_fStepSize               := 84, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_fOkRangeIsLessOrEqualZero     := 85, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_fForceRangeIsLessOrEqualfOkRange := 86, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPWMPeriod               := 87, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_tMinimumPulseTime          := 88, (* invalid parameter *)
    eCTRL_ERROR_FileDelete                  := 89, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_nNumberOfPwmOutputs          := 90, (* File Delete failed *)
    eCTRL_ERROR_INVALIDPARAM_nPwmInputArray_SIZEOF     := 91, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_nPwmOutputArray_SIZEOF          := 92, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_nPwmWaitTimesConfig_SIZEOF     := 93, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_nPwmInternalData_SIZEOF          := 94, (* invalid parameter *)
    eCTRL_ERROR_SIZEOF                     := 95, (* SIZEOF failed *)
    eCTRL_ERROR_INVALIDPARAM_nOrderOfTheTransferfunction     := 96, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_nNumeratorArray_SIZEOF          := 97, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_nDenominatorArray_SIZEOF     := 98, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_a_n_IsZero               := 99, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_WorkArraySIZEOF              := 100, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_MOVINGRANGE_MIN_MAX          := 101, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_MOVINGTIME              := 102, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_DEADTIME               := 103, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_fMinLimitIsGreaterThanfMaxLimit     := 104, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_DataTableSIZEOF               := 105, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_OutputVectorDescription          := 106, (* invalid parameter *)
    eCTRL_ERROR_TaskCycleTimeChanged                    := 107, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_nMinMovingPulses          := 108, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_fAcceleration          := 109, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_fDeceleration          := 110, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_StartAndTargetPos          := 111, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_fVelocity               := 112, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_fTargetPos               := 113, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_fStartPos               := 114, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_fMovingLength          := 115, (* invalid parameter *)
    eCTRL_ERROR_NT_GetTime                  := 116, (* internal error NT_GetTime *)
    eCTRL_ERROR_INVALIDPARAM_No3PhaseSolutionPossible     := 117, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_fStartVelo               := 118, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_fTargetVelo          := 119, (* invalid parameter *)
    eCTRL_ERROR_INVALID_NEW_PARAMETER_TYPE          := 120 (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_fBaseTime               := 121, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_nOrderOfTheTransferfunction_SIZEOF := 122, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_nFilterOrder          := 124, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_nCoefficientsArray_a_SIZEOF     := 125, (* invalid parameter *)
```

```
    eCTRL_ERROR_INVALIDPARAM_nCoefficientsArray_b_SIZEOF     := 126, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_nDigitalFiterData_SIZEOF     := 127, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_nLogBuffer_SIZEOF          := 128, (* invalid parameter *)
    eCTRL_ERROR_LogBufferOverflow                 := 129, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_nLogBuffer_ADR               := 130, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_nCoefficientsArray_a_ADR     := 131, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_nCoefficientsArray_b_ADR     := 132, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_nPwmInputArray_ADR          := 133, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_nPwmOutputArray_ADR          := 134, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_nPwmWaitTimesConfig_ADR          := 135, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_nPwmInternalData_ADR     := 136, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_nDigitalFiterData_ADR     := 137, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_nNumeratorArray_ADR          := 138, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_nDenominatorArray_ADR     := 139, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_nTransferfunction1Data_ADR     := 140, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_nTransferfunction2Data_ADR     := 141, (* invalid parameter *)
    eCTRL_ERROR_FileSeek                    := 142, (* internal error FB_FileSeek *)
    eCTRL_ERROR_INVALIDPARAM_AmbientTempMaxIsLessThanAmbientTempMin := 143, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_ForerunTempMaxIsLessThanForerunTempMin := 144, (* invalid parameter *)
    eCTRL_ERROR_INVALIDLOGCYCLETIME                     := 145, (* invalid parameter *)
    eCTRL_ERROR_INVALIDVERSION_TcControllerToolbox     := 146,
    eCTRL_ERROR_INVALIDPARAM_Bandwidth               := 147, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_NotchFrequency              := 148, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_DampingCoefficient          := 149, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_fKpIsLessThanZero          := 150 (* invalid parameter *)
);
END_TYPE
```

## E_CTRL_SIGNAL_TYPE

```
TYPE E_CTRL_SIGNAL_TYPE :
(
eCTRL_TRIANGLE := 0,
eCTRL_SINUS := 1,
eCTRL_SAWTOOTH := 2
);
END_TYPE
```

```
TYPE E_CTRL_STEP_SENSORTYPE
(
eSENSOR_NONE := 0,
eSENSOR_PT100 := 1,
eSENSOR_THERMO_J := 2,
eSENSOR_THERMO_K := 3
);
END_TYPE
```

# 5 示例项目

TF4100 Controller Toolbox 功能块的行为通过示例进行介绍。其中包括基础功能。程序的所有闭合回路受控系统均采用仿真实现，项目无需硬件支持。

## 5.1 示例安装

TwinCAT Controller Toolbox 为程序员提供了各种传递单元，通过这些单元可以实现多种不同类型的控制器。

1. 保存示例程序 https://infosys.beckhoff.com/content/1033/TF4100_TC3_Controller/Resources/1461955979.zip 并解压。
   ⇨ 将示例程序 TcControllerToolbox_Examples.sln 加载到 TwinCAT XAE 中，进行编译并启动。
2. 加载解决方案文件。
3. 通过菜单选项 Project - Rebuild All（项目 - 全部重建），编译 PLC 项目。
4. 通过菜单选项 Online - Login（在线 - 登录），将 PLC 项目加载到运行时系统中。
5. 通过菜单选项 Online - Run（在线 - 运行），启动程序。

#### 使用 TwinCAT ScopeView 示例

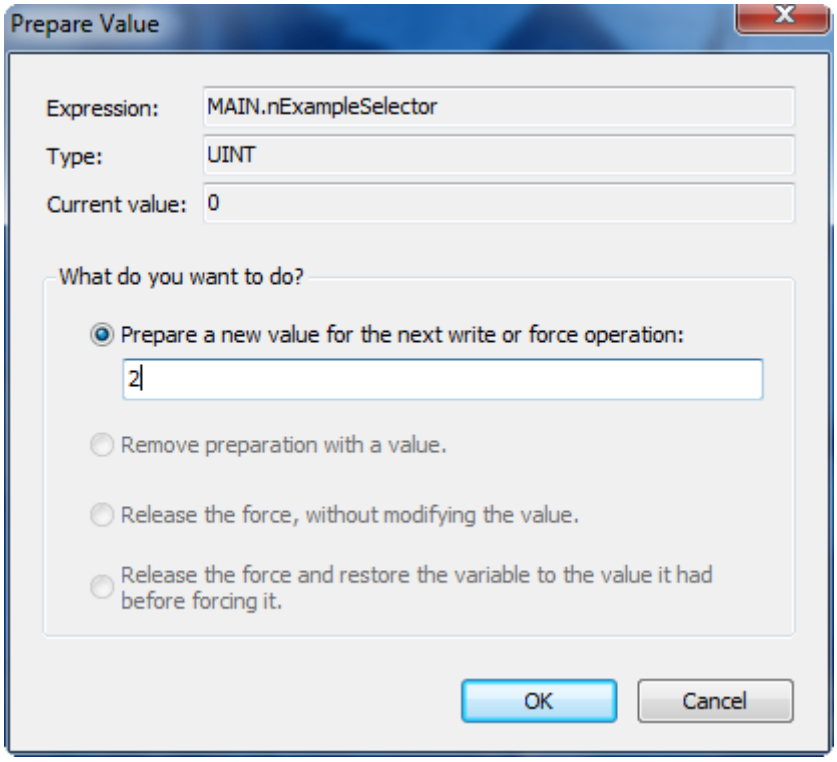
TwinCAT ScopeView 允许以图形方式显示各个示例的信号曲线。
TcControllerToolboxExamples_Scope_x.sv 中提供的设置可用于此目的。在项目的 **MAIN** 程序的注释中可以找到针对每个特定示例所使用的 Scope 文件的说明。

6. 将 Scope 配置文件加载到 TwinCAT ScopeView 中。
7. 通过菜单或按 **Record**（记录）键开始记录。

#### 选择所需的示例项目

由于示例项目包含许多不同的示例，因此有必要在 **MAIN** 程序中将变量 `nExampleSelector` 设置为相应的示例编号。

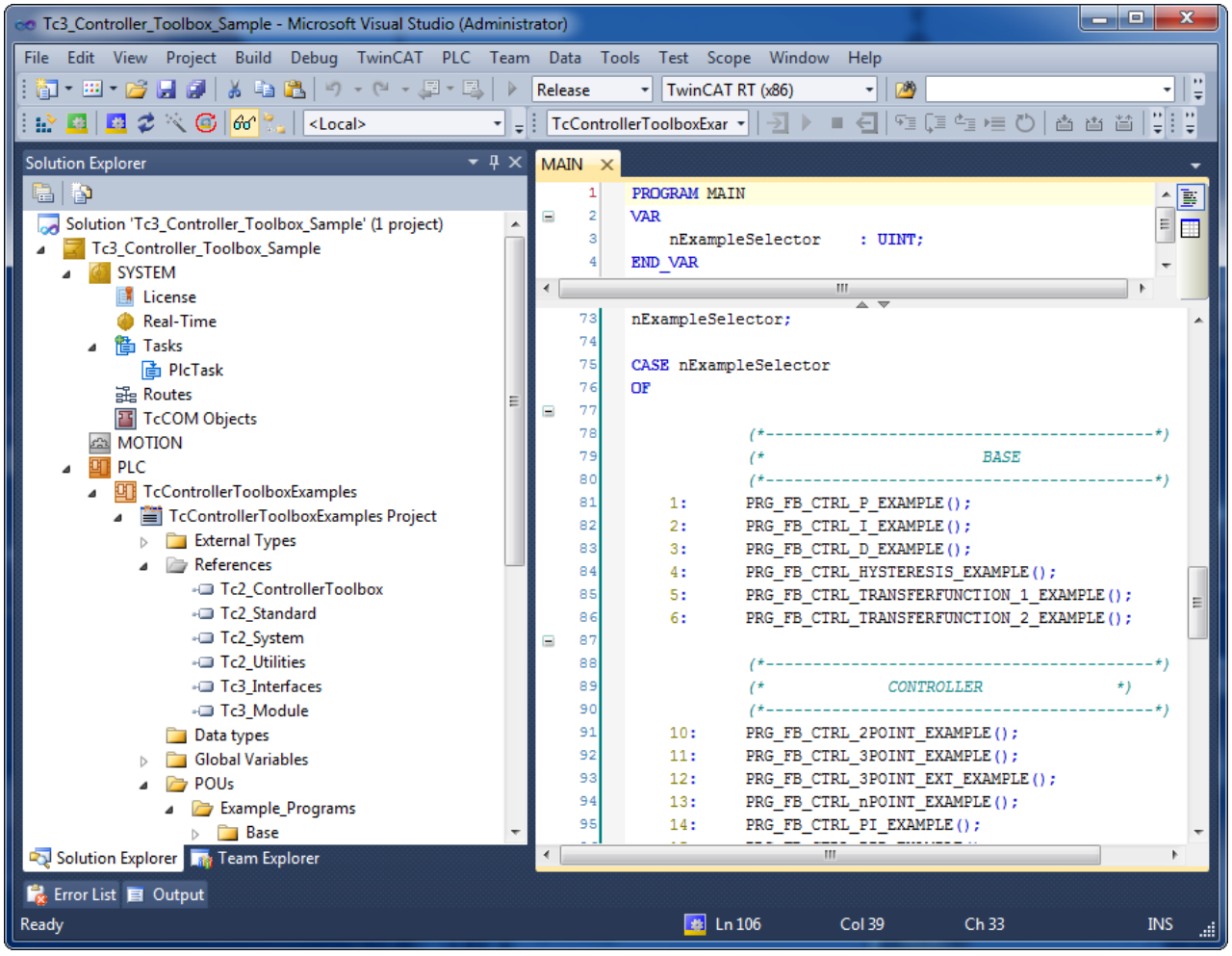
8. 双击变量 `nExampleSelector`。




9. 输入示例的编号。
10. 点击 **OK（确定）**。
11. 按 **[F7]** 键或点击 **Online**（在线）菜单中的 **Force Values**（强制值）。

## 5.2 示例结构

**MAIN** 模块根据变量 `nExampleSelector` 调用相应的示例程序。




各个示例程序均包含注释，以确保清晰性与可追溯性。

# 6 附录

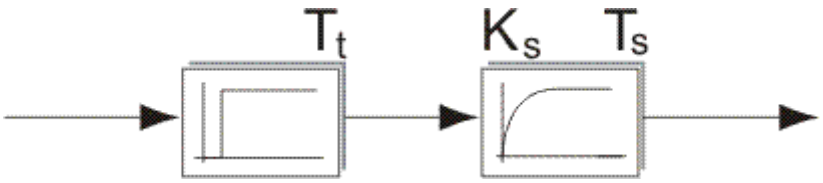
## 6.1 P、PI 和 PID 控制器的设置规则

本页总结了相关文献中的一些设置规则。务必根据受控系统来确定在特定情况下使用的设置规则。

**齐格勒-尼科尔斯**

如果受控系统可以近似为死区时间单元和一阶延迟单元的组合，则可以使用齐格勒-尼科尔斯设置规则。

$$G_s(s) = K_s \cdot \frac{e^{-T_t s}}{1 + s \cdot T_s}$$




$T_t$ = 受控系统的死区时间

$K_s$ = 受控系统的增益系数

$T_s$ = 受控系统的时间常数

| Regler | $K_r$ | $T_n$ | $T_v$ |
|---|---|---|---|
| P-Regler | $\frac{T_s}{K_s \cdot T_t}$ | - | - |
| PI-Regler | $0.9 \cdot \frac{T_s}{K_s \cdot T_t}$ | $3.33 \cdot T_t$ | - |
| PID-Regler | $1.2 \cdot \frac{T_s}{K_s \cdot T_t}$ | $2.0 \cdot T_t$ | $0.5 \cdot T_t$ |

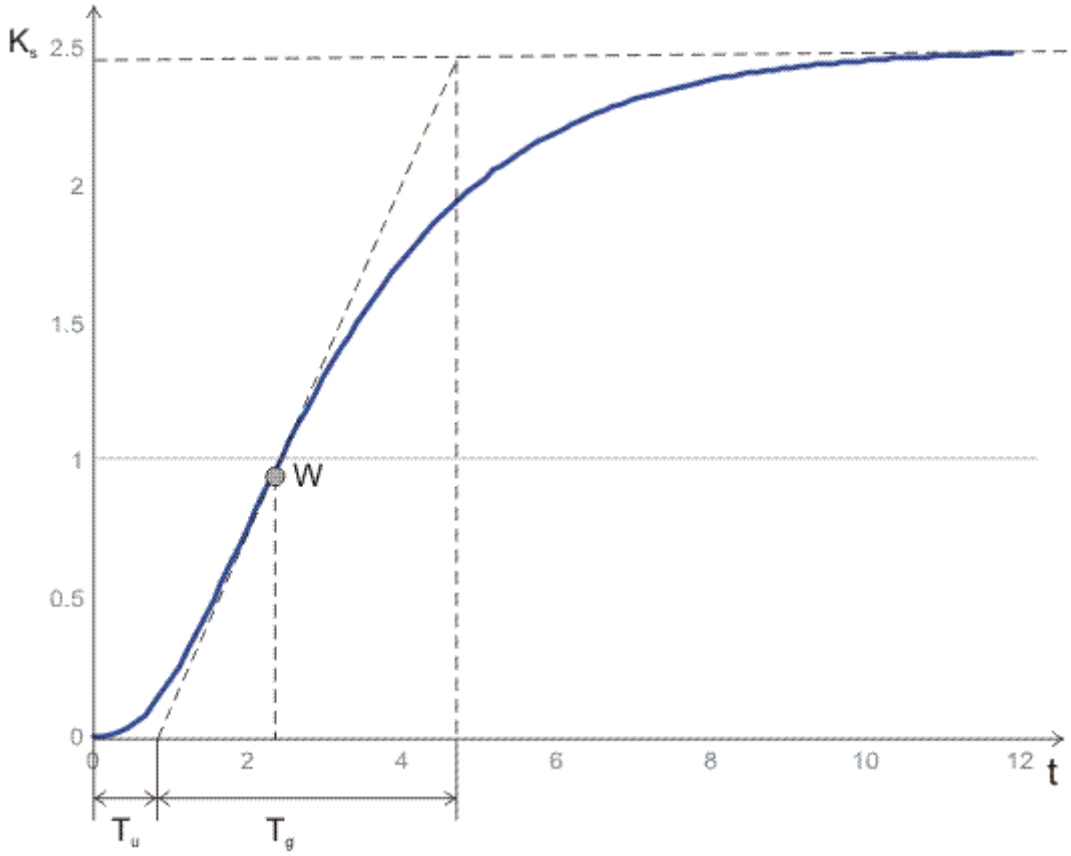
**陈-赫朗尼斯-雷斯维克**

如果要采用此整定方法，系统的阶跃响应必须呈现延迟特性且无超调现象。

根据阶跃响应，确定延迟时间 $T_U$、补偿时间 $T_g$ 和系统放大系数 $K_s$。

系统放大系数根据以下商值计算得出：

$$K_s = \frac{\text{Step Response Height}}{\text{Controller Output Value}}$$




**干扰行为优化：**

| Regler | | Aperiodischer Regelverlauf | 20% Überschwingen |
|---|---|---|---|
| P-Regler | $K_r$ | $0.30 \cdot \frac{T_g}{T_u \cdot K_s}$ | $0.70 \cdot \frac{T_g}{T_u \cdot K_s}$ |
| PI-Regler | $K_r$ | $0.60 \cdot \frac{T_g}{T_u \cdot K_s}$ | $0.70 \cdot \frac{T_g}{T_u \cdot K_s}$ |
| | $T_N$ | $4.00 \cdot T_u$ | $2.3 \cdot T_u$ |
| PID-Regler | $K_r$ | $0.95 \cdot \frac{T_g}{T_u \cdot K_s}$ | $1.20 \cdot \frac{T_g}{T_u \cdot K_s}$ |
| | $T_N$ | $2.40 \cdot T_u$ | $2.00 \cdot T_u$ |
| | $T_V$ | $0.42 \cdot T_u$ | $0.42 \cdot T_u$ |

**控制行为优化**

| Regler | | Aperiodischer Regelverlauf | 20% Überschwingen |
|---|---|---|---|
| P-Regler | $K_r$ | $0.30 \cdot \frac{T_g}{T_u \cdot K_s}$ | $0.70 \cdot \frac{T_g}{T_u \cdot K_s}$ |
| PI-Regler | $K_r$ | $0.35 \cdot \frac{T_g}{T_u \cdot K_s}$ | $0.60 \cdot \frac{T_g}{T_u \cdot K_s}$ |
| | $T_N$ | $1.20 \cdot T_g$ | $1.0 \cdot T_g$ |
| PID-Regler | $K_r$ | $0.60 \cdot \frac{T_g}{T_u \cdot K_s}$ | $0.95 \cdot \frac{T_g}{T_u \cdot K_s}$ |
| | $T_N$ | $1.00 \cdot T_g$ | $1.35 \cdot T_g$ |
| | $T_V$ | $0.50 \cdot T_u$ | $0.47 \cdot T_u$ |

## 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

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

更多信息:
**www.beckhoff.com/tf4100**

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