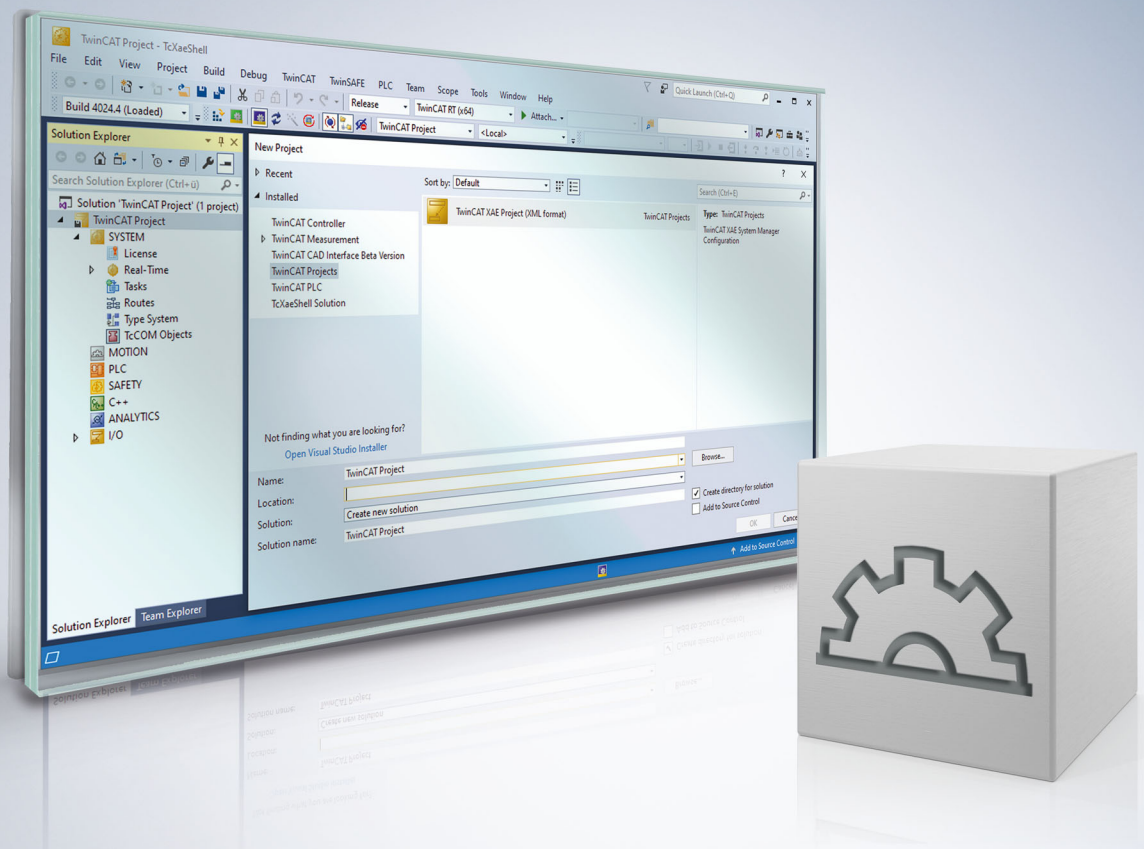
BECKHOFF New Automation Technology

手册 | ZH

TF5050

TwinCAT 3 | NC Camming



2024-09-27 | 版本： 1.8.0

目录

1 前言

1.1 文档说明

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

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

免责声明

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

商标

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

正在申请的专利

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

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

版权所有

© 德国倍福自动化有限公司。
未经明确授权，不得复制、分发、使用和传播本文档内容。
违者将被追究赔偿责任。德国倍福自动化有限公司保留所有发明、实用新型和外观设计专利权。

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

在许多应用场合，需要使两个或多个轴同步运动。TwinCAT NC PTP 可以将轴耦合在一起。然后，主轴可以自主运动，而一个或多个耦合从轴的位置则由 NC 根据主轴位置和速度进行同步控制。

最简单的耦合类型是具有固定传动比的线性耦合（电子齿轮）。

有些应用需要主轴和从轴进行更复杂的耦合，这种耦合不能用简单的数学公式来描述。但可以通过一个表格来描述这种对应关系，该表格为每个主轴位置指定一个对应的从轴位置。

通过这个表格（电子凸轮表）TwinCAT NC PTP 就可以实现从轴与主轴的耦合。这里所说的表格包含若干预定义类型的插补点，NC 在这些插补点之间进行位置和速度插补。

处理凸轮表的功能块包含于Tc2_MC2_Camming 库。该库可处理两种类型的凸轮表：

一种是 2 列形式的凸轮表，包含主轴和从轴位置（Postion Table）。主轴列通过主轴的移动路径定义插补点，位置值从最小值上升到最大值。插补点的对应从轴位置由表格第二列确定。这些插补点之间的主从轴位置对应关系通过线性插值运算来确定。

另一种类型的凸轮表相邻插补点之间的插值类型可以修改，即所谓的可修改的凸轮表motion function。一个可修改的凸轮表motion function只包含一列数据，每个数据都表示一个插补点， 而每个插补点不仅包含位置，而且包含凸轮表在这一段（分段）内插补曲线的完整描述。除了在分段开始时的主轴和从轴位置外，还以数学函数的形式规定了到下一个插补点的插补路径。通过这种方式，一个motion function的凸轮表只需要非常少的插补点。此外，插补点之间的每个点都是通过数学函数精确定义的，插补路径严格一致。

不同于固定的位置凸轮表，Motion funtion 凸轮表在运行时仍然可以修改插补点。系统确保只有在更改对从轴没有直接影响的情况下，操作才会生效。这样就避免了位置跳变。

3 PLC 库概述

下面是 TF5050 TwinCAT 3 MC Camming（凸轮盘）PLC 库的概述。

从 3.4.0 版起的 Tc2_MC2_Camming

Tc2_MC2_Camming 库至少需要在 3.4.0 版以上的运行系统上使用 TwinCAT 3.1.4024.55。

凸轮表

功能块	描述
MC_CamTableSelect [▶ 10]	指定一个表并将其加载到 NC 中。
MC_CamOut [▶ 11]	主轴-从轴解除耦合。
MC_CamIn [▶ 12]	激活主轴-从轴按某一凸轮表进行耦合。
MC_CamScaling [▶ 17]	对凸轮耦合进行缩放。

多表凸轮

功能块	描述
MC_CamIn_V2 [▶ 19]	几个凸轮表可以叠加。
MC_CamAdd [▶ 21]	在一个多表凸轮上增加一个凸轮表。
MC_Cam_Exchange [▶ 23]	在一个多表凸轮中切换一个凸轮表。
MC_CamRemove [▶ 25]	从一个多表凸轮环境中移除一个凸轮表。
MC_CamScaling_V2 [▶ 27]	对凸轮耦合进行缩放。

可修改的凸轮表

功能块	描述
MC_ReadMotionFunction [▶ 29]	读取一个可修改凸轮表的数据。
MC_ReadMotionFunctionPoint [▶ 30]	读取一个可修改凸轮表插补点的数据。
MC_WriteMotionFunction [▶ 31]	将一个可修改凸轮表的数据写入 NC 中。
MC_WriteMotionFunctionPoint [▶ 32]	写下一个可修改凸轮表插补点的数据。
MC_SetCamOnlineChangeMode [▶ 33]	指定凸轮表写入数据的模式。
MC_ReadMotionFunctionValues [▶ 35]	读取一个可修改凸轮表的插补数据。

状态

功能块	描述
MC_ReadCamTableSlaveDynamics [▶ 37]	确定凸轮表某一点的从轴动态。
MC_CamInfo [▶ 39]	获得与凸轮耦合的当前状态和当前参数化有关的数据。
MC_CamInfo_V2 [▶ 41]	获得与凸轮耦合的当前状态和当前参数化有关的数据。
MC_ReadCamTableCharacteristics [▶ 43]	计算和读取一个可修改凸轮表的特征参数。
MC_ReadCamTableMasterPosition [▶ 44]	根据给定的从轴位置计算主轴位置。

数据类型

数据类型	描述
MC_CAM_ID [▶ 46]	表 ID 的类型定义。
MC_CAM_REF [▶ 47]	用另一个 PLC 变量描述凸轮表的数据存储。
MC_CamActivationMode [▶ 48]	指定凸轮表的切换时间和类型。
MC_CamScalingMode [▶ 51]	包含凸轮耦合的缩放类型和范围，由功能块 MC_CamScaling 使用。

数据类型	描述
MC_CamInfoData [▶ 54]	包含凸轮耦合当前状态的数据。
MC_InterpolationType [▶ 55]	位置表的插补模式。
MC_MotionFunctionPoint [▶ 56]	描述了一个可修改凸轮表的插补点。
MC_MotionFunctionPoint_ID [▶ 57]	类型定义，用于一个可修改凸轮表内插补点的 ID。
MC_MotionFunctionType [▶ 58]	用于可修改凸轮表的类型定义。
MC_MotionPointType [▶ 59]	表中插补点的类型定义。
MC_TableCharacValues [▶ 60]	定义可修改凸轮表的特征参数。
MC_TableErrorCodes [▶ 61]	表错误代码的类型定义。
MC_TableType [▶ 61]	表的类型定义。
MC_ValueSelectType [▶ 61]	用于通过功能块 MC_ReadMotionFunctionValues 访问数值表的类型定义。
MC_StartMode [▶ 62]	定义凸轮表是绝对于轴坐标系的原点还是相对于耦合位置进行插补。
ST_CamInOptions [▶ 63]	ST_CamInOptions 类型的数据作为可选项，可传输到功能块 MC_CamIn。
CamMasterData [▶ 64]	CamMasterData 类型的数据作为可选项，可由功能块 MC_ReadCamTableMasterPosition 进行传输。
MC_CamOperationMode [▶ 64]	用于管理通过功能块 MC_CamIn_V2 耦合的叠加凸轮耦合。
ST_CamScalingData [▶ 65]	包含用于缩放凸轮表的信息，与功能块 MC_CamIn_V2 一起使用。

4 凸轮表

4.1 MC_CamTableSelect

通过功能块 *MC_CamTableSelect*，可以指定一个表格并加载到 NC 中。该功能块创建了一个新的表格，同时用 PLC 提供的数据填充该表格。

如果要使用通过 TwinCAT 凸轮盘编辑器创建的表格，则不必使用 *MC_CamTableSelect*。在这种情况下，通过 MC_CamIn [▶ 12] 的简单耦合就足够了。

输入

```
VAR_INPUT
   Execute        : BOOL;
   Periodic       : BOOL;
   CamTableID     : MC_CAM_ID;
END_VAR
```

名称	类型	描述
执行	BOOL	命令在 *Execute* 输入的上升沿执行。
定期	BOOL	如果凸轮盘周期性重复，则 *Periodic* 为“TRUE”。
CamTableID	MC_CAM_ID	用于耦合的凸轮盘 ID

输入/输出

```
VAR_IN_OUT
   CamTable : MC_CAM_REF;
END_VAR
```

名称	类型	描述
CamTable	MC_CAM_REF	MC_CAM_REF [▶ 47] 类型的数据结构描述了 PLC 中凸轮盘的数据存储情况。

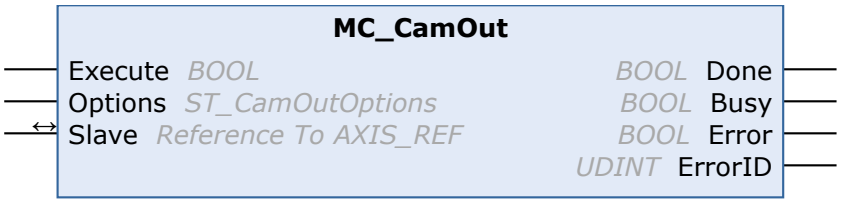
AXIS_REF 类型的轴数据结构，应当在系统中对应到唯一的轴。通过诸多其他参数，轴数据结构包含了轴的当前状态：位置、速度或错误状态等信息。

输出

```
VAR_OUTPUT
   Done    : BOOL;
   Busy    : BOOL;
   Error   : BOOL;
   ErrorID : UDINT;
END_VAR
```

名称	类型	描述
Done	BOOL	如果凸轮盘被成功创建，则变为“TRUE”。
Busy	BOOL	当用 *Execute* 开始执行命令时，*Busy* 输出变为“TRUE”，并且在命令执行过程中保持“TRUE”状态。如果 *Busy* 再次变成“FALSE”，功能块就可以重新执行。同时，其中一个输出被设定： *Done* 或 *Error*。
错误	BOOL	一旦发生错误，则变为“TRUE”。
ErrorID	UDINT	如果错误输出已设定，此参数提供 错误代码。

4.2 MC_CamOut



功能块 *MC_CamOut* 解除主轴-从轴耦合。

注意：如果一个从轴在运动过程中解除耦合，它不会自动停止，而是以解耦时的速度继续无休止地运动。可以用“Stop”命令来停止轴。

注意

在运动期间调用

解除耦合后，从轴切换到无加速状态，并继续以由此产生的恒定速度运动。此时，从轴位置与主轴行进路径及基于耦合系数的计算值无关。相反，该行为与 MC_MoveVelocity 命令后的行为一致。

输入

```
VAR_INPUT
   Execute : BOOL;
   Options : ST_GearOutOptions;
END_VAR
```

名称	类型	描述
执行	BOOL	命令在 *Execute* 输入的上升沿执行。
选项	ST_CamOutOptions	当前未执行

输入/输出

```
VAR_IN_OUT
   Slave    : AXIS_REF;
END_VAR
```

名称	类型	描述
从轴	AXIS_REF	从轴数据结构。

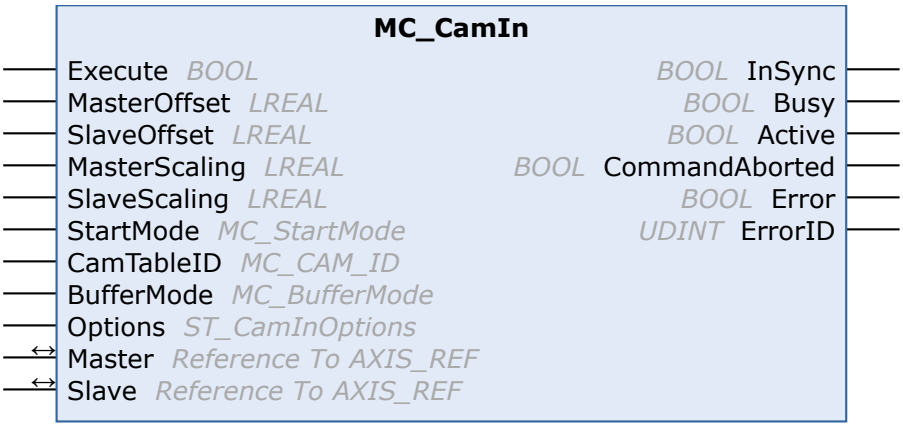
AXIS_REF 类型的轴数据结构，应当在系统中对应到唯一的轴。通过诸多其他参数，轴数据结构包含了轴的当前状态：位置、速度或错误状态等信息。

输出

```
VAR_OUTPUT
   Done    : BOOL;
   Busy    : BOOL;
   Error   : BOOL;
   ErrorID : UDINT;
END_VAR
```

名称	类型	描述
Done	BOOL	如果凸轮盘被成功创建，则变为“TRUE”。
Busy	BOOL	当用 *Execute* 开始执行命令时，*Busy* 输出变为“TRUE”，并且在命令执行过程中保持“TRUE”状态。如果 *Busy* 再次变成“FALSE”，功能块就可以重新执行。同时，其中一个输出被设定： *Done* 或 *Error*。
错误	BOOL	一旦发生错误，则变为“TRUE”。
ErrorID	UDINT	如果错误输出已设定，此参数提供 错误代码。

4.3 MC_CamIn



功能块 *MC_CamIn* 激活主轴-从轴按某个凸轮盘进行耦合。此外，还可以在耦合状态下切换到一个新的凸轮盘。可以指定切换规则，特别是切换的时间或位置。

状态标志 Axis.Status.CamTableQueued（AXIS_REF）可用于检查是否有一个凸轮盘在队列中等待切换。

重要信息：

关于凸轮盘耦合的详细信息 [▶ 14]

ActivationMode [▶ 48]（凸轮盘的耦合或切换）

StartMode [▶ 62]

ScalingMode [▶ 51]

输入

```
VAR_INPUT
   Execute       : BOOL;
   MasterOffset  : LREAL;
   SlaveOffset   : LREAL;
   MasterScaling : LREAL := 1.0;
   SlaveScaling  : LREAL := 1.0;
   StartMode     : MC_StartMode;
   CamTableID    : MC_CAM_ID;
   BufferMode    : MC_BufferMode;
   Options       : ST_CamInOptions;
END_VAR
```

名称	类型	描述	
执行	BOOL	命令在 *Execute* 输入的上升沿执行。	
MasterOffset	LREAL	至凸轮盘主轴位置的偏移量。	
SlaveOffset	LREAL	至凸轮盘从轴位置的偏移量。	
MasterScaling	LREAL	凸轮盘主轴位置的缩放。	
SlaveScaling	LREAL	凸轮盘从轴位置的缩放。	
StartMode	MC_StartMode	StartMode [▶ 62] 决定凸轮盘是根据绝对位置进行插补还是相对于耦合位置进行插补。对于主轴（X 坐标）和从轴（Y 坐标） *StartMode* 可以分别是相对的或绝对的。	
CamTableID	MC_CAM_ID	用于耦合的凸轮盘 ID [▶ 46]。	
BufferMode	MC_BufferMode	当前未执行	
选项	ST_CamInOptions	包含进一步耦合和切换选项的 数据结构 [▶ 63]：	
		ActivationMode	ActivationMode [▶ 48] 用于指定凸轮盘耦合或切换的时间或位置。 *ActivationMode* 也可以在从动装置首次耦合时指定。

名称	类型	描述	
		ActivationPosition	可选的主轴位置，根据 *ActivationMode*。 （第一次耦合时不需要） 如果使用 *ActivationMode* MC_CAMACTIVATION_ATMASTERCAMPOS，该位置是指未标定的凸轮盘。如果应用中的位置指的是缩放后的凸轮盘主轴位置，那么在调用功能块之前，要除以 *MasterScaling*。
		MasterScalingMode	选项 Scaling mode [▶ 51]，用于凸轮盘的主轴位置。
		SlaveScalingMode	选项 Scaling mode [▶ 51]，用于凸轮盘的从轴位置。
		InterpolationType	位置表的 插补类型 [▶ 55]。对于运动功能无需此设置。

输入/输出

```
VAR_IN_OUT
   Master : AXIS_REF;
   Slave  : AXIS_REF;
END_VAR
```

名称	类型	描述
主轴	AXIS_REF	主轴数据结构。
从轴	AXIS_REF	从轴数据结构。

AXIS_REF 类型的轴数据结构，应当在系统中对应到唯一的轴。通过诸多其他参数，轴数据结构包含了轴的当前状态：位置、速度或错误状态等信息。

输出

```
VAR_OUTPUT
   InSync          : BOOL;
   Busy            : BOOL;
   Active          : BOOL;
   CommandAborted  : BOOL;
   Error           : BOOL;
   ErrorID         : UDINT;
END_VAR
```

名称	类型	描述
InSync	BOOL	如果耦合成功并且凸轮盘处于活动状态，则变为“TRUE”。
Busy	BOOL	当用 *Execute* 开始执行命令时，*Busy* 输出变为“TRUE”，并且在命令执行过程中保持“TRUE”状态。如果 *Busy* 再次变成“FALSE”，功能块就可以重新执行。同时，其中一个输出被设定： *Done* 或 *Error*。
启用	BOOL	启用表示该命令已执行。对于凸轮盘切换，如果耦合命令被成功执行，但凸轮盘仍在队列中等待执行，则 *Active* 变为“TRUE”。如果凸轮盘根据 *ActivationMode* 被激活，则 *Active* 变成 FALSE 并设定 *InSync*。
CommandAborted	BOOL	如果命令不能被完全执行，则变为“TRUE”。在耦合过程中，轴可能已解耦（发生了多个命令同时执行的情况）。
错误	BOOL	一旦发生错误，则变为“TRUE”。
ErrorID	UDINT	如果错误输出已设定，此参数提供 错误代码。

4.4 MC_CaminAppendix

凸轮表的耦合

功能块 MC_CamIn [▶ 12] 可用于在主轴和从轴之间建立凸轮耦合。请注意，在耦合之前，从轴必须处于由凸轮表定义的当前主轴位置对应的从轴位置。耦合后，一旦主轴启动，从轴的位置就根据主轴位置按凸轮表直接计算。因此，从轴不会渐进地追上凸轮表。如果它还未到达凸轮表规定的位置，就会发生位置跳变。

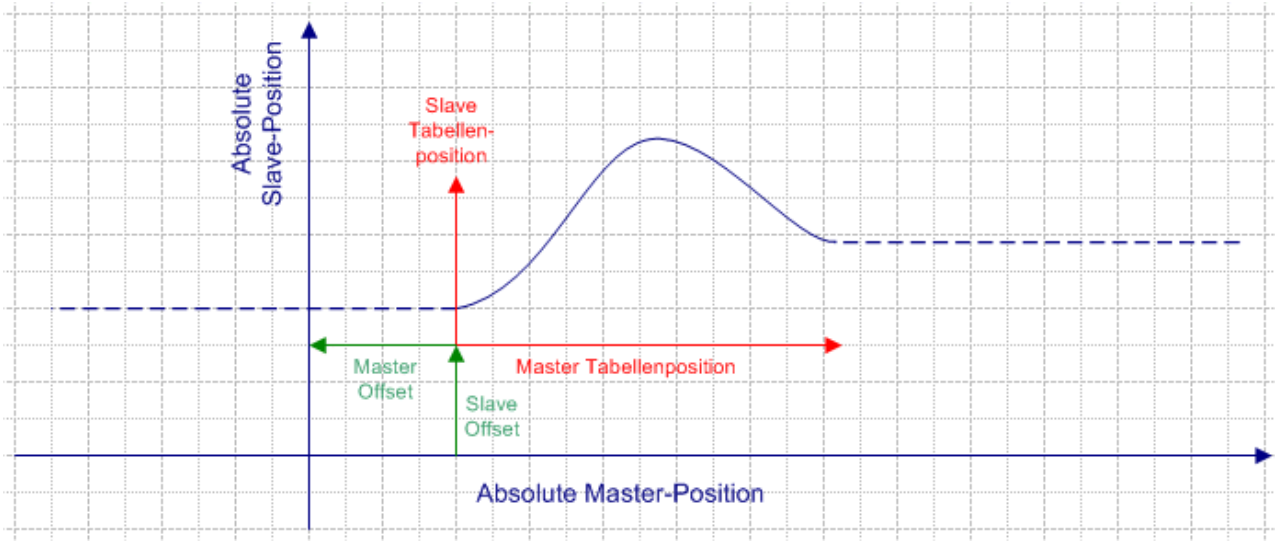
在实践中，出现了这样的问题：在耦合之前，从轴应该处于什么位置，以及如何计算这个位置。下面的数字说明了这个过程。

注意：对于所有后续的计算，只使用轴的设置位置。实际位置不用于计算，因为它们会导致计算错误，特别是循环凸轮表。

只考虑主轴从都是绝对位置模式的凸轮耦合情况。对于相对耦合，主轴或从轴的耦合位置在计算中被认为是一个附加偏移量。

线性凸轮表

线性凸轮表只通过有限的主轴位置范围来定义。在这个范围之外，从轴位置由凸轮表的第一个或最后一个位置来定义。因此，一旦主轴离开定义的范围，从轴就会停止在凸轮曲线的两个端点所定义的位置。



图中显示，绝对轴坐标系（蓝色）不一定要与凸轮表坐标系（红色）完全一致。凸轮表坐标系可以通过master offset 或 slave offset 进行偏移， 也可以进行缩放。

通过功能块 MC_ReadCamTableSlaveDynamics [▶ 37]，可以确定与某一主轴位置对应的从轴位置。这里指的是原始凸轮表数据，这意味着必须通过 PLC 程序本身来计算偏移和缩放系数。最初，master offset 被叠加到当前主轴位置。如果要对凸轮表进行缩放，则要除以这个缩放系数。

```
MasterCamTablePosition := (MasterPosition + MasterOffset) /MasterScaling;
```

凸轮表中的主轴位置被用作功能块 MC_ReadCamTableSlaveDynamics [▶ 37] 的输入参数。如有必要，结果将被转换为带有从轴偏移和缩放的绝对从轴位置。

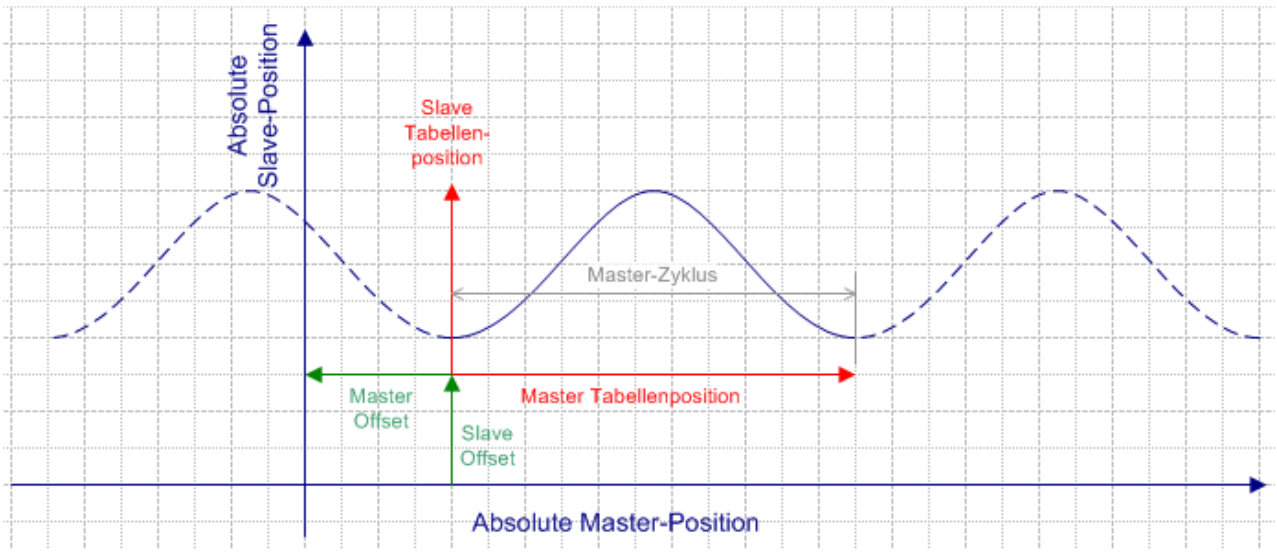
```
SlaveCamTablePosition := ReadSlaveDynamics.SlavePosition;

SlavePosition := (SlaveCamTablePosition * SlaveScaling) + SlaveOffset;
```

在耦合之前，从轴装置移动到此位置。或者，主轴可以移动到与当前从轴位置相对应的位置。然而，一般来说，这个位置不能从凸轮表确定，因为凸轮表可能是不明确的。

注意：由于在第一个公式中加入了master offset，正偏移量会导致凸轮表坐标系向左侧的负向移动。因此，图中的master offset是负值。正的slave offset导致凸轮表坐标系向上方的正向偏移。

不带节距的循环凸轮表

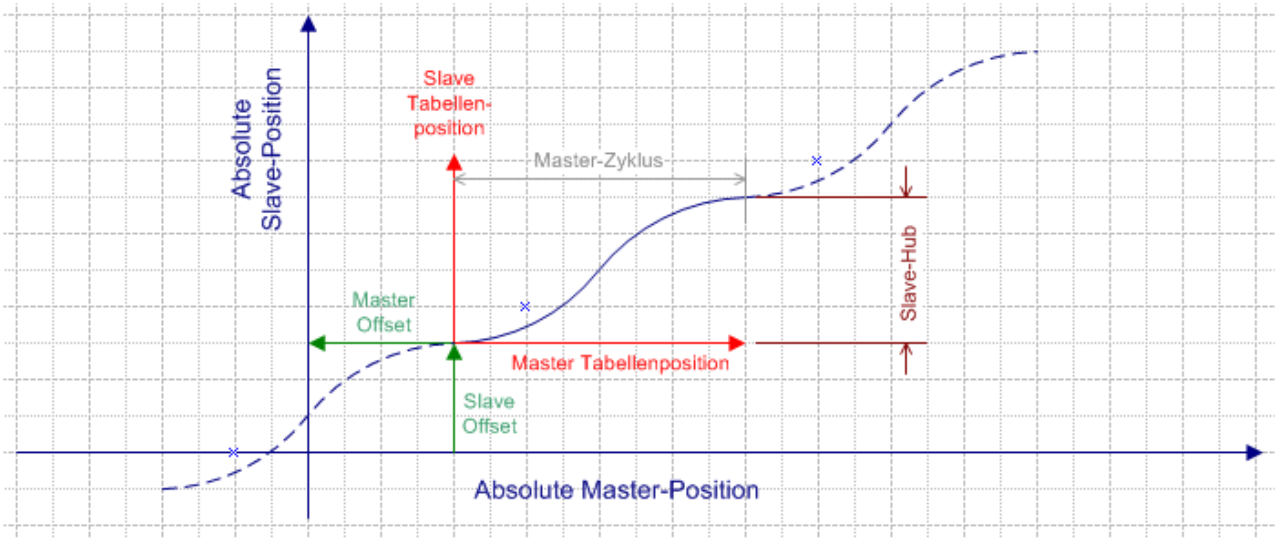
不带节距的循环凸轮表的特点是：表中的从轴起始和结束位置是相同的。因此，从轴在规定的范围内周期性移动，而不在某个特定方向上永久改变其位置。



对于这些凸轮表类型，主/从轴耦合需要与线性凸轮表的准备工作相同。因此，从轴的起始位置可以按上述方法计算。没有必要使用主轴的模长位置进行计算，因为已经通过耦合命令正确计算了主轴的绝对位置。

带节距的循环凸轮表

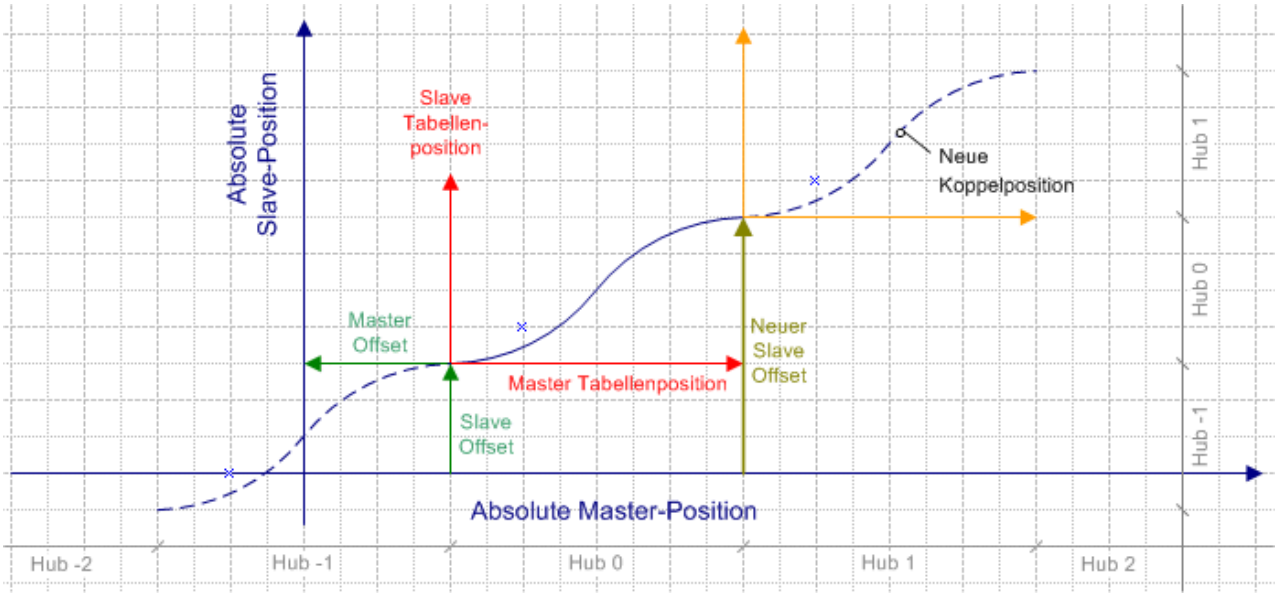
循环凸轮表的节距是从轴最后一个和第一个表位置之间的差异。



这样的凸轮表在结束点周而复始地持续， 从轴位置不会跳回凸轮表中的初始值， 而是连续稳定地向前运动。因此，在每个新的周期中，节距作为一个额外的内部slave offset叠加量，如果运动反向，则减去相应的偏移量。

带节距循环凸轮表的解耦和再耦合

如果一个从轴被耦合到一个带节距的凸轮表上，耦合总是在基本周期（红色坐标系）中进行，即没有增加节距距离。如果从轴装置在几个周期后被解耦，然后再重新耦合，从轴装置就会回到基本周期。如有必要，必须考虑到这种行为，并通过重新计算slave offset来补偿。



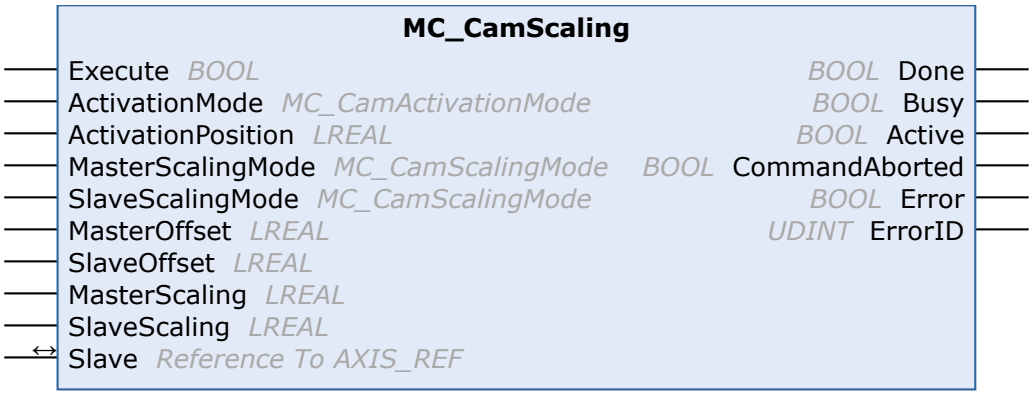

```
MasterCamTablePos := (MasterPosition + MasterOffset) / MasterScaling;
```

凸轮表中的主轴位置被用作功能块 MC_ReadCamTableSlaveDynamics [▶ 37] 的输入参数。如有必要，结果将被转换为带有从轴偏移和缩放的绝对从轴位置。此外，还必须计算出需要偏移的节距数量，并将其叠加到从轴位置。

```
SlaveCamTablePosition := ReadSlaveDynamics.SlavePosition;

LiftNumberl := MODTURNS（（SlavePosition - SlaveOffset），SlaveHub）；

NewSlaveOffset := SlaveOffset + (SlaveHub * lift number);

SlavePosition := (SlaveCamTablePosition * SlaveScaling) + NewSlaveOffset;
```

Autooffset [▶ 51] 功能块可以简化偏移量的计算，特别是对于凸轮表的切换。

4.5 MC_CamScaling



耦合的凸轮盘可以用功能块 *MC_CamScaling* 来缩放。缩放不会影响凸轮盘的原始表格数据，而是影响现有的主/从轴耦合关系。可以修改以下参数：主轴和从轴的缩放系数，以及在坐标系中凸轮盘的偏置量。

可以选择本次修改只从特定的主轴位置开始生效，从而在运动期间实现精确的缩放。在运动期间进行缩放时要注意。缩放瞬间的从轴位置应该只有轻微的变化。

状态标志 Axis.Status.CamScalingPending（AXIS_REF）可用于检查是否有一个缩放程序在队列中等待执行。

输入

```
VAR_INPUT
   Execute            : BOOL;
   ActivationMode     : MC_CamActivationMode;
   ActivationPosition : LREAL;
   MasterScalingMode  : MC_CamScalingMode;
   SlaveScalingMode   : MC_CamScalingMode;
   MasterOffset       : LREAL;
   SlaveOffset        : LREAL;
   MasterScaling      : LREAL := 1.0;
   SlaveScaling       : LREAL := 1.0;
END_VAR
```

名称	类型	描述
执行	BOOL	命令在 *Execute* 输入的上升沿执行。
ActivationMode	MC_CamActivationMode	ActivationMode [▶ 48] 指定缩放时间或位置。
ActivationPosition	LREAL	凸轮盘被缩放的主轴位置，根据 ActivationMode [▶ 48]。 如果使用 *ActivationMode* MC_CAMACTIVATION_ATMASTERCAMPOS，该位置指的是未缩放的凸轮盘。如果应用中的位置指的是缩放后的凸轮盘主轴位置，那么在调用功能块之前，要除以 *MasterScaling*。
MasterScalingMode	MC_CamScalingMode	选项 Scaling mode [▶ 51]，用于凸轮盘的主轴位置。
SlaveScalingMode	MC_CamScalingMode	选项 Scaling mode [▶ 51]，用于凸轮盘的从轴位置。
MasterOffset	LREAL	至凸轮盘主轴位置的偏移量。
SlaveOffset	LREAL	至凸轮盘从轴位置的偏移量。
MasterScaling	LREAL	凸轮盘主轴位置的缩放。
SlaveScaling	LREAL	凸轮盘从轴位置的缩放。

输入/输出

```
VAR_IN_OUT
   Slave    : AXIS_REF;
END_VAR
```

名称	类型	描述
从轴	AXIS_REF	从轴数据结构。

AXIS_REF 类型的轴数据结构，应当在系统中对应到唯一的轴。通过诸多其他参数，轴数据结构包含了轴的当前状态：位置、速度或错误状态等信息。

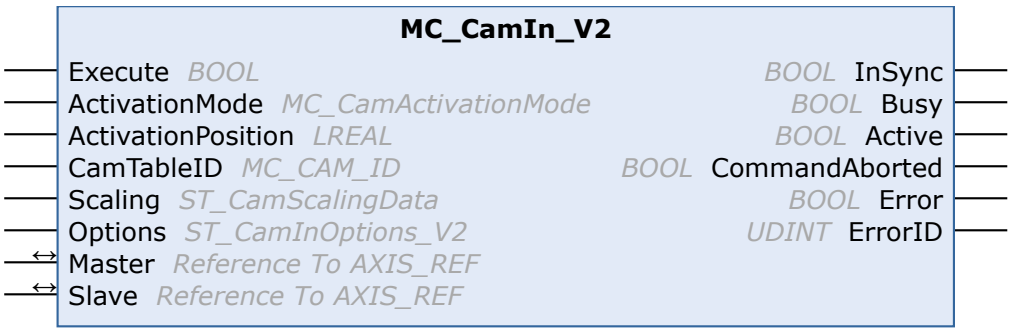
输出

```
VAR_OUTPUT
   Done           : BOOL;
   Busy           : BOOL;
   Active         : BOOL;
   CommandAborted : BOOL;
   Error          : BOOL;
   ErrorID        : UDINT;
END_VAR
```

名称	类型	描述
Done	BOOL	如果凸轮盘被成功创建，则变为“TRUE”。
Busy	BOOL	当用 *Execute* 开始执行命令时，*Busy* 输出变为“TRUE”，并且在命令执行过程中保持“TRUE”状态。如果 *Busy* 再次变成“FALSE”，功能块就可以重新执行。同时，其中一个输出被设定： *Done* 或 *Error*。
启用	BOOL	启用表示该命令已执行。对于凸轮盘切换，如果耦合命令被成功执行，但凸轮盘仍在队列中等待执行，则 Active 变为“TRUE”。如果凸轮盘根据 ActivationMode 被激活，则 Active 变成“FALSE”并设定 InSync。
CommandAborted	BOOL	如果命令不能被完全执行，则变为“TRUE”。在耦合过程中，轴可能已解耦（发生了多个命令同时执行的情况）。
错误	BOOL	一旦发生错误，则变为“TRUE”。
ErrorID	UDINT	如果错误输出已设定，此参数提供 错误代码。

5 多凸轮表

5.1 MC_CamIn_V2



MC_CamIn_V2 是功能块 MC_CamIn [▶ 12] 的扩展版，能够同时耦合多个叠加凸轮盘（多凸轮）。当 MC_CamIn_V2 第一次被调用时，它按凸轮盘建立了一个主/从轴耦合。在凸轮耦合运行起来后再次调用这个功能块，可以为同一从轴叠加另一个凸轮盘或再次将其移除。可以指定切换规则，特别是切换的时间或位置。

MC_CamIn_V2 只能作为 *MC_CamIn* 的替代使用， 两个功能块不能同时作用于同一个从轴。对于凸轮盘的增加、更换和移除，可选用功能块 MC_CamAdd [▶ 21]、MC_CamExchange [▶ 23] 和 MC_CamRemove [▶ 25]。这些操作也可以通过 *MC_CamIn_V2* 来完成。

状态标志 Axis.Status.CamTableQueued（AXIS_REF）可用于检查是否有一个凸轮盘在队列中等待添加或切换。

重要信息：

ActivationMode [▶ 48]（操作生效的时间或位置）

CamOperationMode [▶ 64]（添加、切换或移除叠加凸轮盘）

ScalingMode [▶ 51]

输入

```
VAR_INPUT
   Execute            : BOOL;
   ActivationMode     : MC_CamActivationMode := MC_CAMACTIVATION_INSTANTANEOUS;
   ActivationPosition : LREAL;
   CamTableID         : MC_CAM_ID;
   Scaling            : ST_CamScalingData;
   Options            : ST_CamInOptions_V2;
END_VAR
```

名称	类型	描述	
执行	BOOL	命令在 *Execute* 输入的上升沿执行。	
ActivationMode	MC_CamActivationMode	ActivationMode [▶ 48] 指定缩放时间或位置。	
ActivationPosition	LREAL	凸轮盘被缩放的主轴位置，根据 ActivationMode [▶ 48]。如果使用 *ActivationMode* MC_CAMACTIVATION_ATMASTERCAMPOS，该位置指的是未缩放的凸轮盘。如果应用中的位置指的是缩放后的凸轮盘主轴位置，那么在调用功能块之前，要除以 *MasterScaling*。	
CamTableID	MC_CAM_ID	用于耦合的凸轮盘 ID [▶ 46]。	
缩放	ST_CamScalingData	可选的凸轮盘 缩放参数 [▶ 65]。	
选项	ST_CamInOptions_V2	包含进一步耦合和切换选项的 数据结构 [▶ 63]：	
		插补类型	位置表的 插补类型 [▶ 55]。对于运动功能无需此设置。
		CamOperationMode	CamOperationMode [▶ 64] 定义了指定的凸轮盘（*CamTableID*）在耦合系统中的作用方式。凸轮盘可以添加、切换或移除。

名称	类型	描述	
		ReferenceCamTableID	已经在耦合中处于活动状态的凸轮盘的可选 ID [▶ 46]。此 ID 对于那些不明确的操作特别需要，例如替换多重耦合的某些凸轮盘。在不明确的操作中，该值可能保持为 0。

输入/输出

```
VAR_IN_OUT
   Master : AXIS_REF;
   Slave  : AXIS_REF;
END_VAR
```

名称	类型	描述
主轴	AXIS_REF	主轴数据结构。
从轴	AXIS_REF	从轴数据结构。

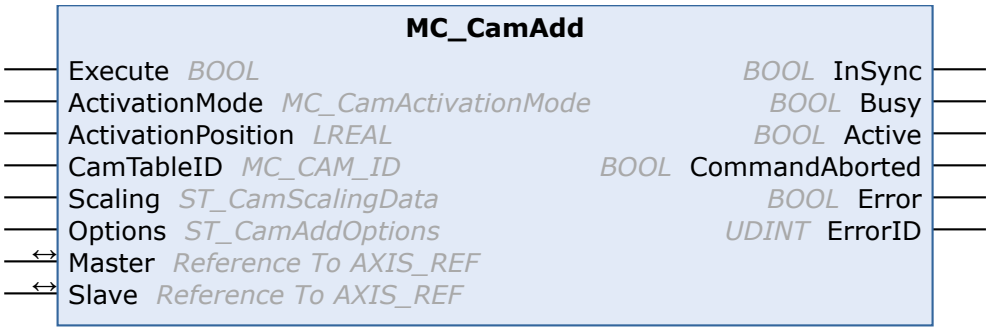
AXIS_REF 类型的轴数据结构，应当在系统中对应到唯一的轴。通过诸多其他参数，轴数据结构包含了轴的当前状态：位置、速度或错误状态等信息。

输出

```
VAR_OUTPUT
   InSync          : BOOL;
   Busy            : BOOL;
   Active          : BOOL;
   CommandAborted : BOOL;
   Error           : BOOL;
   ErrorID         : UDINT;
END_VAR
```

名称	类型	描述
InSync	BOOL	如果凸轮盘操作成功完成，则变为“TRUE”。在有激活位置的操作中，InSync 只有在实际激活后才会变成“TRUE”。
Busy	BOOL	当用 *Execute* 开始执行命令时，*Busy* 输出变为“TRUE”，并且在命令执行过程中保持“TRUE”状态。如果 *Busy* 再次变成“FALSE”，功能块就可以重新执行。同时，其中一个输出被设定： *Done* 或 *Error*。
启用	BOOL	启用表示该命令已执行。对于凸轮盘切换，如果耦合命令被成功执行，但凸轮盘仍在队列中等待执行，则 *Active* 变为“TRUE”。如果凸轮盘根据 *ActivationMode* 被激活，则 *Active* 变成 FALSE 并设定 *InSync*。
CommandAborted	BOOL	如果命令不能被完全执行，则变为“TRUE”。在耦合过程中，轴可能已解耦（发生了多个命令同时执行的情况）。
错误	BOOL	一旦发生错误，则变为“TRUE”。
ErrorID	UDINT	如果错误输出已设定，此参数提供 错误代码。

5.2 MC_CamAdd



MC_CamAdd 在一个多凸轮耦合上增加一个凸轮盘。凸轮盘耦合最初通过 MC_CamIn_V2 [▶ 19] 创建。

另外，可以用 MC_CamIn_V2 添加一个凸轮盘。

状态标志 Axis.Status.CamTableQueued（AXIS_REF）可用于检查是否有一个凸轮盘在队列中等待添加或切换。

重要信息：
ActivationMode [▶ 48]（操作发生的时间或位置）

CamOperationMode [▶ 64]（添加、切换或移除叠加凸轮盘）

ScalingMode [▶ 51]

输入

```
VAR_INPUT
   Execute            : BOOL;
   ActivationMode     : MC_CamActivationMode := MC_CAMACTIVATION_INSTANTANEOUS;
   ActivationPosition : LREAL;
   CamTableID         : MC_CAM_ID;
   Scaling            : ST_CamScalingData;
   Options            : ST_CamAddOptions;
END_VAR
```

名称	类型	描述	
执行	BOOL	命令在 *Execute* 输入的上升沿执行。	
ActivationMode	MC_CamActivationMode	ActivationMode [▶ 48] 用于指定凸轮盘耦合或切换的时间或位置。 ActivationMode [▶ 48] 也可以在从动装置首次耦合时指定。	
ActivationPosition	LREAL	凸轮盘切换时可选的主轴位置，根据 ActivationMode [▶ 48]。 （第一次耦合时不需要。） 如果使用 MC_CAMACTIVATION_ATMASTERCAMPOSActivationMode [▶ 48]，该位置是指未标定的凸轮盘。如果应用中的位置指的是缩放后的凸轮盘主轴位置，那么在调用功能块之前，要除以 *MasterScaling*。	
CamTableID	MC_CAM_ID	用于耦合的凸轮盘 ID [▶ 46]。	
缩放	ST_CamScalingData	可选的凸轮盘 缩放参数 [▶ 65]。	
选项	ST_CamAddOptions	**InterpolationType**	位置表的 插补类型 [▶ 55]。对于运动功能无需此设置。

输入/输出

```
VAR_IN_OUT
   Master : AXIS_REF;
   Slave  : AXIS_REF;
END_VAR
```

名称	类型	描述
主轴	AXIS_REF	主轴数据结构。
从轴	AXIS_REF	从轴数据结构。

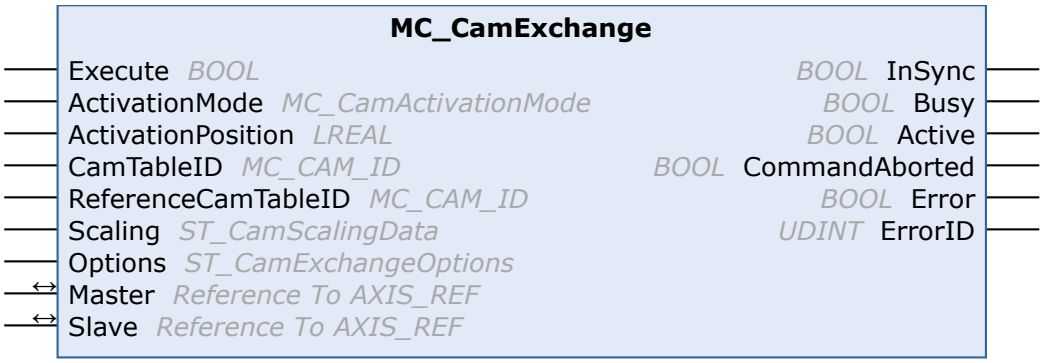
AXIS_REF 类型的轴数据结构，应当在系统中对应到唯一的轴。通过诸多其他参数，轴数据结构包含了轴的当前状态：位置、速度或错误状态等信息。

输出

```
VAR_OUTPUT
   InSync          : BOOL;
   Busy            : BOOL;
   Active          : BOOL;
   CommandAborted : BOOL;
   Error           : BOOL;
   ErrorID         : UDINT;
END_VAR
```

名称	类型	描述
InSync	BOOL	如果凸轮盘操作成功完成，则变为“TRUE”。在有激活位置的操作中，InSync 只有在实际激活后才会变成“TRUE”。
Busy	BOOL	当用 *Execute* 开始执行命令时，*Busy* 输出变为“TRUE”，并且在命令执行过程中保持“TRUE”状态。如果 *Busy* 再次变成“FALSE”，功能块就可以重新执行。同时，其中一个输出被设定： *Done* 或 *Error*。
启用	BOOL	启用表示该命令已执行。对于凸轮盘切换，如果耦合命令被成功执行，但凸轮盘仍在队列中等待执行，则 *Active* 变为“TRUE”。如果凸轮盘根据 *ActivationMode* 被激活，则 *Active* 变成 FALSE 并设定 *InSync*。
CommandAborted	BOOL	如果命令不能被完全执行，则变为“TRUE”。在耦合过程中，轴可能已解耦（发生了多个命令同时执行的情况）。
错误	BOOL	一旦发生错误，则变为“TRUE”。
ErrorID	UDINT	如果错误输出已设定，此参数提供 错误代码。

5.3 MC_Cam Exchange



MC_CamExchange 在一个多凸轮耦合中切换一个凸轮。凸轮盘耦合最初通过 MC_CamIn_V2 [▶ 19] 创建。

另外，可以用 MC_CamIn_V2 切换凸轮盘。

状态标志 Axis.Status.CamTableQueued（AXIS_REF）可用于检查是否有一个凸轮盘在队列中等待添加或切换。

重要信息：

ActivationMode [▶ 48]（操作生效的时间或位置）

CamOperationMode [▶ 64]（添加、切换或移除叠加凸轮盘）

ScalingMode [▶ 51]

输入

```
VAR_INPUT
   Execute              : BOOL;
   ActivationMode       : MC_CamActivationMode := MC_CAMACTIVATION_INSTANTANEOUS;
   ActivationPosition   : LREAL;
   CamTableID           : MC_CAM_ID;
   ReferenceCamTableID  : MC_CAM_ID;
   Scaling              : ST_CamScalingData;
   Options              : ST_CamExchangeOptions;
END_VAR
```

名称	类型	描述	
执行	BOOL	命令在 *Execute* 输入的上升沿执行。	
ActivationMode	MC_CamActivationMode	ActivationMode [▶ 48] 用于指定凸轮盘耦合或切换的时间或位置。 ActivationMode [▶ 48] 也可以在从动装置首次耦合时指定。	
ActivationPosition	LREAL	凸轮盘切换时可选的主轴位置，根据 ActivationMode [▶ 48]。 （第一次耦合时不需要。） 如果使用 MC_CAMACTIVATION_ATMASTERCAMPOSActivationMode [▶ 48]，该位置是指未标定的凸轮盘。如果应用中的位置指的是缩放后的凸轮盘主轴位置，那么在调用功能块之前，要除以 *MasterScaling*。	
CamTableID	MC_CAM_ID	用于耦合的凸轮盘 ID [▶ 46]。	
ReferenceCamTableID	MC_CAM_ID	已经在耦合中处于活动状态的凸轮盘的可选 ID。此 ID 对于那些不明确的操作特别需要，例如替换多重耦合的某些凸轮盘。在不明确的操作中，该值可能保持为 0。	
缩放	ST_CamScalingData	可选的凸轮盘 缩放参数 [▶ 65]。	
选项	ST_CamAddOptions	**InterpolationType**	位置表的 插补类型 [▶ 55]。对于运动功能无需此设置。

输入/输出

```
VAR_IN_OUT
   Master : AXIS_REF;
   Slave  : AXIS_REF;
END_VAR
```

名称	类型	描述
主轴	AXIS_REF	主轴数据结构。
从轴	AXIS_REF	从轴数据结构。

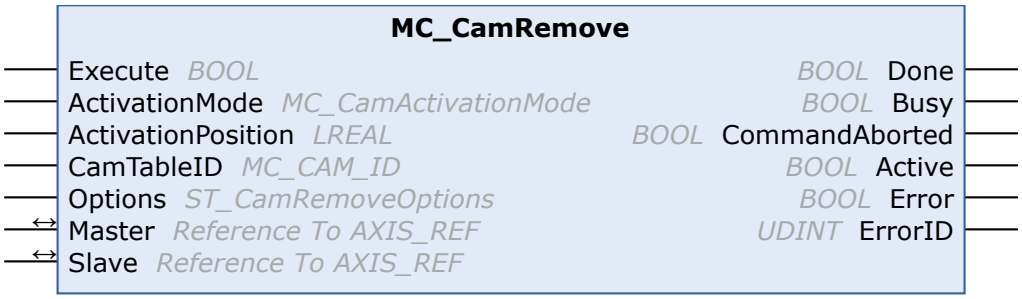
AXIS_REF 类型的轴数据结构，应当在系统中对应到唯一的轴。通过诸多其他参数，轴数据结构包含了轴的当前状态：位置、速度或错误状态等信息。

输出

```
VAR_OUTPUT
   InSync          : BOOL;
   Busy            : BOOL;
   Active          : BOOL;
   CommandAborted : BOOL;
   Error           : BOOL;
   ErrorID         : UDINT;
END_VAR
```

名称	类型	描述
InSync	BOOL	如果凸轮盘操作成功完成，则变为“TRUE”。在有激活位置的操作中，InSync 只有在实际激活后才会变成“TRUE”。
Busy	BOOL	当用 *Execute* 开始执行命令时，*Busy* 输出变为“TRUE”，并且在命令执行过程中保持“TRUE”状态。如果 *Busy* 再次变成“FALSE”，功能块就可以重新执行。同时，其中一个输出被设定： *Done* 或 *Error*。
启用	BOOL	启用表示该命令已执行。对于凸轮盘切换，如果耦合命令被成功执行，但凸轮盘仍在队列中等待执行，则 *Active* 变为“TRUE”。如果凸轮盘根据 *ActivationMode* 被激活，则 *Active* 变成 FALSE 并设定 *InSync*。
CommandAborted	BOOL	如果命令不能被完全执行，则变为“TRUE”。在耦合过程中，轴可能已解耦（发生了多个命令同时执行的情况）。
错误	BOOL	一旦发生错误，则变为“TRUE”。
ErrorID	UDINT	如果错误输出已设定，此参数提供 错误代码。

5.4 MC_CamRemove



MC_CamRemove 从一个多凸轮环境中删除一个凸轮盘。另见 MC_CamIn_V2 [▶ 19]。

重要信息：

ActivationMode [▶ 48]（操作生效的时间或位置）

输入

```
VAR_INPUT
   Execute            : BOOL;
   ActivationMode     : MC_CamActivationMode := MC_CAMACTIVATION_INSTANTANEOUS;
   ActivationPosition : LREAL;
   CamTableID         : MC_CAM_ID;
   Options            : ST_CamRemoveOptions;
END_VAR
```

名称	类型	描述
执行	BOOL	命令在 *Execute* 输入的上升沿执行。
ActivationMode	MC_CamActivationMode	ActivationMode [▶ 48] 用于指定凸轮盘耦合或切换的时间或位置。 ActivationMode [▶ 48] 也可以在从动装置首次耦合时指定。
ActivationPosition	LREAL	凸轮盘切换时可选的主轴位置，根据 ActivationMode [▶ 48]。 （第一次耦合时不需要。） 如果使用 *MC_CAMACTIVATION_ATMASTERCAMPOS* ActivationMode [▶ 48]，该位置是指未标定的凸轮盘。如果应用中的位置指的是缩放后的凸轮盘主轴位置，那么在调用功能块之前，要除以 *MasterScaling*。
CamTableID	MC_CAM_ID	从耦合系统中移除的凸轮盘 ID [▶ 46]。
选项	ST_CamRemoveOptions	目前没有使用。

输入/输出

```
VAR_IN_OUT
   Master : AXIS_REF;
   Slave  : AXIS_REF;
END_VAR
```

名称	类型	描述
主轴	AXIS_REF	主轴数据结构。
从轴	AXIS_REF	从轴数据结构。

AXIS_REF 类型的轴数据结构，应当在系统中对应到唯一的轴。通过诸多其他参数，轴数据结构包含了轴的当前状态：位置、速度或错误状态等信息。

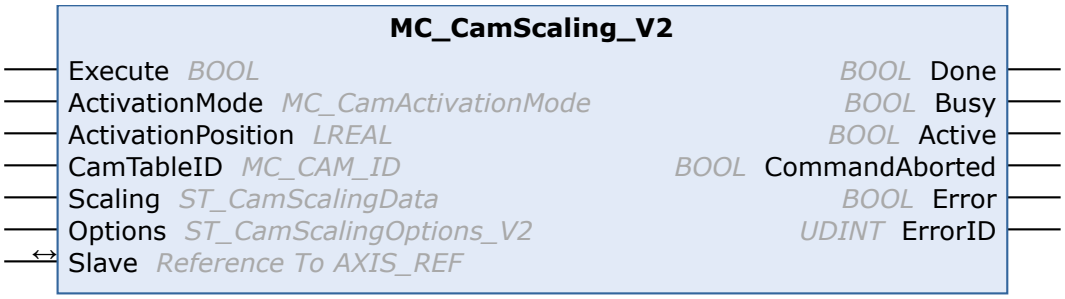
输出

```
VAR_OUTPUT
    Done           : BOOL;
    Busy           : BOOL;
    Active         : BOOL;
```

```
    CommandAborted : BOOL;
    Error          : BOOL;
    ErrorID        : UDINT;
END_VAR
```

名称	类型	描述
Done	BOOL	如果凸轮盘操作成功完成，则变为“TRUE”。在有激活位置的操作中，Done 只有在实际停用后才变为“TRUE”。
Busy	BOOL	当用 *Execute* 开始执行命令时，*Busy* 输出变为“TRUE”，并且在命令执行过程中保持“TRUE”状态。如果 *Busy* 再次变成“FALSE”，功能块就可以重新执行。同时，其中一个输出被设定： *Done* 或 *Error*。
启用	BOOL	启用表示该命令已执行。如果命令成功发出，但操作仍在队列中等待，则 *Active* 变为“TRUE”。如果凸轮盘根据 ActivationMode [▶ 48] 被激活，则 *Active* 变成 FALSE 并设定 *InSync*。
CommandAborted	BOOL	如果命令不能被完全执行，则变为“TRUE”。在耦合过程中，轴可能已解耦（发生了多个命令同时执行的情况）。
错误	BOOL	一旦发生错误，则变为“TRUE”。
ErrorID	UDINT	如果错误输出已设定，此参数提供 错误代码。

5.5 MC_CamScaling_V2



耦合的凸轮盘可以用功能块 *MC_CamScaling_V2* 来缩放。缩放不会影响凸轮盘的原始表格数据，而是影响现有的主/从轴耦合关系。可以修改以下参数：主轴和从轴的缩放系数，以及在坐标系中凸轮盘的偏置量。

可以选择本次修改只从特定的主轴位置开始生效，从而在运动期间实现精确的缩放。在运动期间进行缩放时要注意。缩放瞬间的从轴位置应该只有轻微的变化。

状态标志 Axis.Status.CamScalingPending（AXIS_REF）可用于检查是否有一个缩放程序在队列中等待执行。

输入

```
VAR_INPUT
   Execute            : BOOL;
   ActivationMode     : MC_CamActivationMode;
   ActivationPosition : LREAL;
   CamTableID         : MC_CAM_ID;
   Scaling            : ST_CamScalingData;
   Options            : ST_CamScalingOptions_V2;
END_VAR
```

名称	类型	描述
执行	BOOL	命令在 *Execute* 输入的上升沿执行。
ActivationMode	MC_CamActivationMode	ActivationMode [▶ 48] 指定缩放时间或位置。
ActivationPosition	LREAL	凸轮盘被缩放的主轴位置，根据 ActivationMode [▶ 48]。 如果使用 MC_CAMACTIVATION_ATMASTERCAMPOS ActivationMode [▶ 48]，该位置指的是未缩放的凸轮盘。如果应用中的位置指的是缩放后的凸轮盘主轴位置，那么在调用功能块之前，要除以 *MasterScaling*。
CamTableID	MC_CAM_ID	进行缩放的凸轮盘的 ID [▶ 46]。
缩放	ST_CamScalingData	模式、偏移量和缩放系数等缩放数据
选项	ST_CamScalingOptions_V2	未使用

输入/输出

```
VAR_IN_OUT
   Slave    : AXIS_REF;
END_VAR
```

名称	类型	描述
从轴	AXIS_REF	从轴数据结构。

AXIS_REF 类型的轴数据结构，应当在系统中对应到唯一的轴。通过诸多其他参数，轴数据结构包含了轴的当前状态：位置、速度或错误状态等信息。

输出

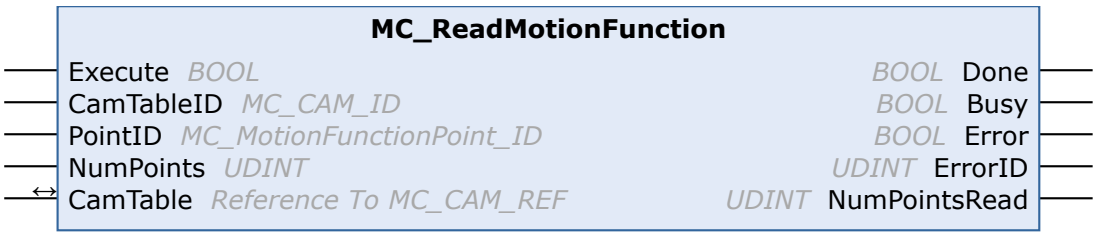
```
VAR_OUTPUT
   Done           : BOOL;
   Busy           : BOOL;
   Active         : BOOL;
   CommandAborted : BOOL;
```

```
   Error          : BOOL;
   ErrorID        : UDINT;
END_VAR
```

名称	类型	描述
Done	BOOL	如果缩放成功，则变为“TRUE”。
Busy	BOOL	当用 *Execute* 开始执行命令时，*Busy* 输出变为“TRUE”，并且在命令执行过程中保持“TRUE”状态。如果 *Busy* 再次变成“FALSE”，功能块就可以重新执行。同时，其中一个输出被设定： *Done* 或 *Error*。
启用	BOOL	启用表示该命令已执行。对于凸轮盘切换，如果耦合命令被成功执行，但凸轮盘仍在队列中等待执行，则 Active 变为“TRUE”。如果凸轮盘根据 ActivationMode 被激活，则 Active 变成“FALSE”并设定 InSync。
CommandAborted	BOOL	如果命令不能被完全执行，则变为“TRUE”。在耦合过程中，轴可能已解耦（发生了多个命令同时执行的情况）。
错误	BOOL	一旦发生错误，则变为“TRUE”。
ErrorID	UDINT	如果错误输出已设定，此参数提供 错误代码。

6 可修改的凸轮表

6.1 MC_ReadMotionFunction



功能块 *MC_ReadMotionFunction* 可用于读取运动功能的数据。既可以读取包含所有插补点的完整功能，也可以只读取一部分。这些数据存储在 PLC 内，其结构由 CamTable [▶ 47] 描述。

输入

```
VAR_INPUT
   Execute    : BOOL;
   CamTableID : MC_CAM_ID;
   PointID    : MC_MotionFunctionPoint_ID;
   NumPoints  : UDINT; (* 0 = fill MFsize *)
END_VAR
```

名称	类型	描述
执行	BOOL	命令在 *Execute* 输入的上升沿执行。
CamTableID	MC_CAM_ID	加载表的 ID [▶ 46]。
PointID	MC_MotionFunctionPoint_ID	要读取的运动功能的第一个点的 ID [▶ 57]。
NumPoints	UDINT	要读取的运动功能点的数量。要读取所有点，可以在这里指定 0，在这种情况下，实际读取的数量会在输出变量 *NumPointsRead* 中返回。

输入/输出

```
VAR_IN_OUT
   CamTable : MC_CAM_REF;
END_VAR
```

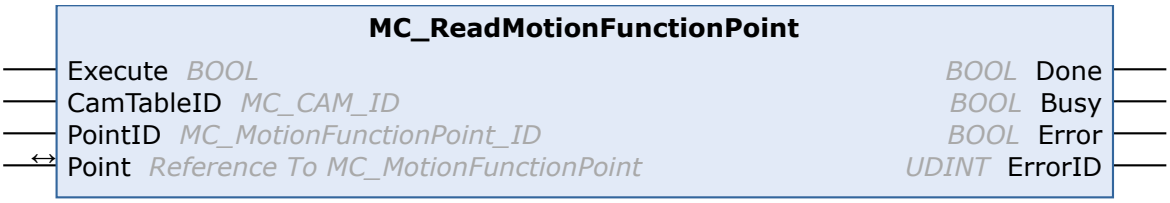
名称	类型	描述
CamTable	MC_CAM_REF	表的 参考 [▶ 47]（结构）。

输出

```
VAR_OUTPUT
   Done          : BOOL;
   Busy          : BOOL;
   Error         : BOOL;
   ErrorID       : UDINT;
   NumPointsRead : UDINT; (* return value <= NumPoints *)
END_VAR
```

名称	类型	描述
Done	BOOL	如果数据被成功读取，则变为“TRUE”。
Busy	BOOL	当用 *Execute* 开始执行命令时，*Busy* 输出变为“TRUE”，并且在命令执行过程中保持“TRUE”状态。如果 *Busy* 再次变成“FALSE”，功能块就可以重新执行。同时，其中一个输出被设定： *Done* 或 *Error*。
错误	BOOL	一旦发生错误，则变为“TRUE”。
ErrorID	UDINT	如果错误输出已设定，此参数提供 错误代码。
NumPointsRead	UDINT	实际读取的点数。这个数字可能小于或等于 *NumPoints*。

6.2 MC_ReadMotionFunctionPoint



功能块 *MC_ReadMotionFunctionPoint* 可用于读取一个运动功能的插补点数据。

输入

```
VAR_INPUT
   Execute    : BOOL;
   CamTableID : MC_CAM_ID;
   PointID    : MC_MotionFunctionPoint_ID;
END_VAR
```

名称	类型	描述
执行	BOOL	命令在 *Execute* 输入的上升沿执行。
CamTableID	MC_CAM_ID	加载表的 ID [▶ 46]。
PointID	MC_MotionFunctionPoint_ID	要读取的运动功能的第一个点的 ID [▶ 57]。

输入/输出

```
VAR_IN_OUT
   Point : MC_MotionFunctionPoint;
END_VAR
```

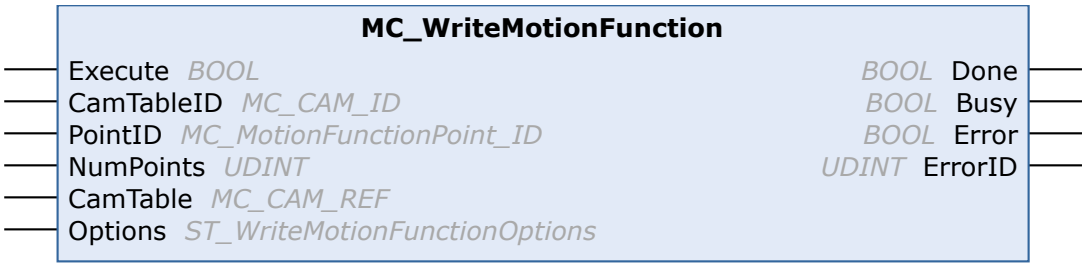
名称	类型	描述
点	MC_MotionFunctionPoint	Data structure [▶ 56] 包含一个运动功能插补点的数据。

输出

```
VAR_OUTPUT
   Done          : BOOL;
   Busy          : BOOL;
   Error         : BOOL;
   ErrorID       : UDINT;
END_VAR
```

名称	类型	描述
Done	BOOL	如果数据被成功读取，则变为“TRUE”。
Busy	BOOL	当用 *Execute* 开始执行命令时，*Busy* 输出变为“TRUE”，并且在命令执行过程中保持“TRUE”状态。如果 *Busy* 再次变成“FALSE”，功能块就可以重新执行。同时，其中一个输出被设定： *Done* 或 *Error*。
错误	BOOL	一旦发生错误，则变为“TRUE”。
ErrorID	UDINT	如果错误输出已设定，此参数提供 错误代码。

6.3 MC_WriteMotionFunction



功能块 *MC_WriteMotionFunction* 可用于将运动功能的数据写入 NC。可以写入全部插补点，或者只写入部分插补点。刚开始，数据存储在 PLC 内，其结构由 CamTable [▶ 47] 描述。

功能块 MC_SetCamOnlineChangeMode [▶ 33] 可用于指定何时将数据传送至凸轮盘。如果数据的激活要延迟到主轴到达某个位置，系统将先把数据写入等待队列，然后在主轴位置将其激活。

状态标志 Axis.Status.CamDataQueued（AXIS_REF）可用于检查数据是否已进入等待队列，即已写入但尚未激活。

输入

```
VAR_INPUT
   Execute     : BOOL;
   CamTableID  : MC_CAM_ID;
   PointID     : MC_MotionFunctionPoint_ID;
   NumPoints   : UDINT;
   CamTable    : MC_CAM_REF;
   Options     : ST_WriteMotionFunctionOptions;
END_VAR
```

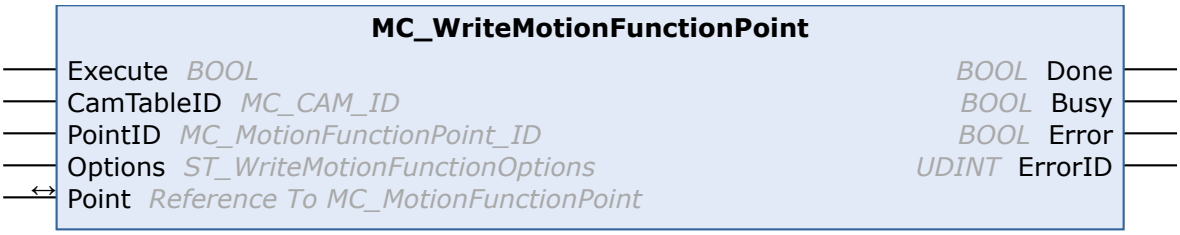
名称	类型	描述	
Execute	BOOL	命令在 *Execute* 输入的上升沿执行。	
CamTableID	MC_CAM_ID	加载表的 ID [▶ 46]。	
PointID	MC_MotionFunctionPoint_ID	要读取的运动函数的第一个点的 ID [▶ 57]。	
NumPoints	UDINT	要写入的运动函数的点的数量。	
CamTable	MC_CAM_REF	表的 参考 [▶ 47]（结构）。表数据结构（CamTable.pArray）的起始地址表示要写入的第一个点。	
Options	ST_WriteMotionFunctionOptions [▶ 66]	具有更多选项的数据结构：	
		SynchronousAccess	写入通过同步访问进行，没有延迟时间。应仅用于时间要求极高的应用。

输出

```
VAR_OUTPUT
   Done        : BOOL;
   Busy        : BOOL;
   Error       : BOOL;
   ErrorID     : UDINT;
END_VAR
```

名称	类型	描述
Done	BOOL	如果数据被成功读取，则变为“TRUE”。
Busy	BOOL	当用 *Execute* 开始执行命令时，*Busy* 输出变为“TRUE”，并且在命令执行过程中保持“TRUE”状态。如果 *Busy* 再次变成“FALSE”，功能块就可以重新执行。同时，其中一个输出被设定： *Done* 或 *Error*。
错误	BOOL	一旦发生错误，则变为“TRUE”。
ErrorID	UDINT	如果错误输出已设定，此参数提供 错误代码。

6.4 MC_WriteMotionFunctionPoint



功能块 *MC_WriteMotionFunctionPoint* 可用来写入一个运动函数插补点的数据。

功能块 MC_SetCamOnlineChangeMode [▶ 33] 可用于指定何时将数据传送至凸轮。如果数据的激活要延迟到主轴到达某个位置，系统将先把数据写入等待队列，然后在主轴位置将其激活。

状态标志 Axis.Status.CamDataQueued（AXIS_REF）可用于检查数据是否已进入等待队列，即已写入但尚未激活。

输入

```
VAR_INPUT
   Execute     : BOOL;
   CamTableID  : MC_CAM_ID;
   PointID     : MC_MotionFunctionPoint_ID;
   Options     : ST_WriteMotionFunctionOptions;
END_VAR
```

名称	类型	描述	
Execute	BOOL	命令在 *Execute* 输入的上升沿执行。	
CamTableID	MC_CAM_ID	加载表的 ID [▶ 46]。	
PointID	MC_MotionFunctionPoint_ID	要读取的运动函数的第一个点的 ID [▶ 57]。	
Options	ST_WriteMotionFunctionOptions [▶ 66]	具有更多选项的数据结构：	
		SynchronousAccess	写入通过同步访问进行，没有延迟时间。应仅用于时间要求极高的应用。

输入/输出

```
VAR_IN_OUT
   Point : MC_MotionFunctionPoint;
END_VAR
```

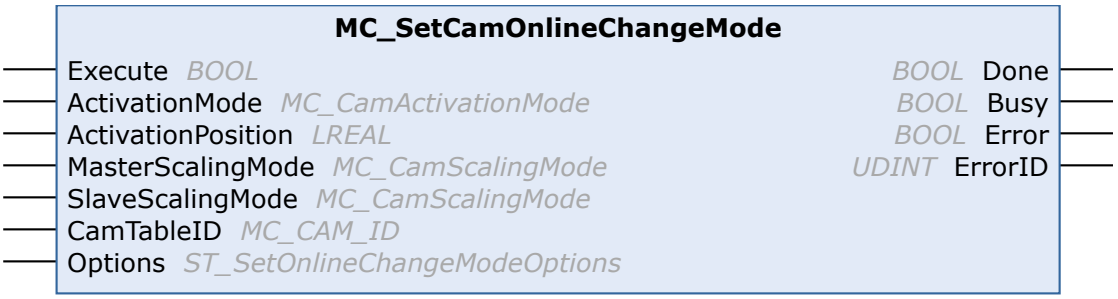
名称	类型	描述
点	MC_MotionFunctionPoint	Data structure [▶ 56] 包含一个运动功能插补点的数据。

输出

```
VAR_OUTPUT
   Done        : BOOL;
   Busy        : BOOL;
   Error       : BOOL;
   ErrorID     : UDINT;
END_VAR
```

名称	类型	描述
Done	BOOL	如果数据被成功读取，则变为“TRUE”。
Busy	BOOL	当用 *Execute* 开始执行命令时，*Busy* 输出变为“TRUE”，并且在命令执行过程中保持“TRUE”状态。如果 *Busy* 再次变成“FALSE”，功能块就可以重新执行。同时，其中一个输出被设定： *Done* 或 *Error*。
错误	BOOL	一旦发生错误，则变为“TRUE”。
ErrorID	UDINT	如果错误输出已设定，此参数提供 错误代码。

6.5 MC_SetCamOnlineChangeMode



功能块 *MC_SetCamOnlineChangeMode* 指定写入凸轮盘数据的模式。

凸轮盘可以在运行时通过 PLC 进行修改（见 MC_WriteMotionFunction [▶ 31]、MC_WriteMotionFunctionPoint [▶ 32]）。功能块 *MC_SetCamOnlineChangeMode* 用于指定这些变化何时发生以及如何生效。这里的模式设置会影响所有后续的写操作。因此，没有必要在每次写访问之前调用该功能块。

此功能块指定了凸轮盘内容修改的激活模式，但不影响凸轮盘的更换或移除。

输入

```
VAR_INPUT
   Execute            : BOOL;
   ActivationMode     : MC_CamActivationMode;
   ActivationPosition : LREAL;
   MasterScalingMode  : MC_CamScalingMode;
   SlaveScalingMode   : MC_CamScalingMode;
   CamTableID         : MC_CAM_ID;
   Options            : ST_SetOnlineChangeModeOptions;
END_VAR
```

名称	类型	描述	
Execute	BOOL	命令在 *Execute* 输入的上升沿执行。	
ActivationMode	MC_CamActivationMode	定义了缩放的时间和方式。(MC_CamActivationMode [▶ 48])	
ActivationPosition	LREAL	可选的主轴位置，在这个位置上进行缩放（根据 ActivationMode [▶ 48]）。 如果使用 ActivationMode [▶ 48] MC_CAMACTIVATION_ATMASTERCAMPOS，则该位置是指未缩放的凸轮。如果应用中的位置指的是缩放后的凸轮主轴位置，那么在调用函数块之前，要除以 *MasterScaling*。	
MasterScalingMode	MC_CamScalingMode	主轴缩放类型。（MC_CamScalingMode [▶ 51]）	
SlaveScalingMode	MC_CamScalingMode	从轴缩放类型。（MC_CamScalingMode [▶ 51]）	
CamTableID	MC_CAM_ID	表 ID [▶ 46]。	
Options	ST_SetOnlineChangeModeOptions [▶ 66]	具有更多选项的数据结构：	
		SynchronousAccess	写入通过同步访问进行，没有延迟时间。应仅用于时间要求极高的应用。

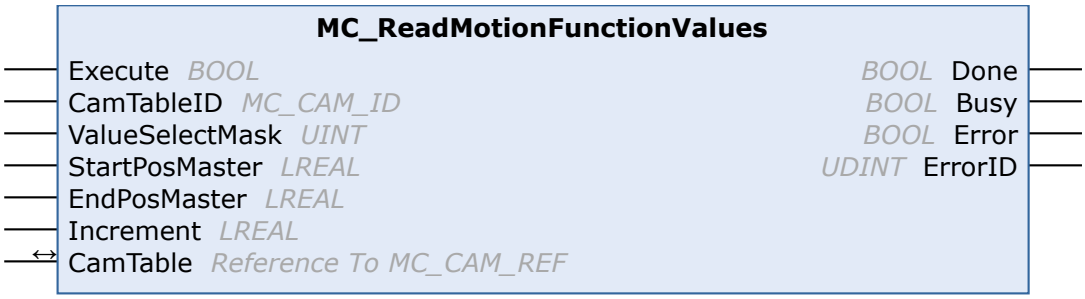
输出

```
VAR_OUTPUT
   Done          : BOOL;
   Busy          : BOOL;
   Error         : BOOL;
   ErrorID       : UDINT;
END_VAR
```

名称	类型	描述
Done	BOOL	当功能块被成功执行时，变为“TRUE”。

名称	类型	描述
Busy	BOOL	当用 *Execute* 开始执行命令时，*Busy* 输出变为“TRUE”，并且在命令执行过程中保持“TRUE”状态。如果 *Busy* 再次变成“FALSE”，功能块就可以重新执行。同时，其中一个输出被设定： *Done* 或 *Error*。
错误	BOOL	一旦发生错误，则变为“TRUE”。
ErrorID	UDINT	如果错误输出已设定，此参数提供 错误代码。

6.6 MC_ReadMotionFunctionValues



功能块 *MC_ReadMotionFunctionValues* 可用于以表格形式读取一个运动功能的插补数据。

例如，该功能可用于显示一个运动功能的曲线。完整的连续曲线按设置的步长进行离散。用这种确定的数据来显示可修改的凸轮盘会更加容易。

输入

```
VAR_INPUT
   Execute          : BOOL;
   CamTableID       : MC_CAM_ID;
   ValueSelectMask  : UINT; (* MC_ValueSelectType; position, velocity, acceleration, jerk… *)
   StartPosMaster   : LREAL; (* master position of first point *)
   EndPosMaster     : LREAL; (* master position of last point *)
   Increment        : LREAL; (* increment of master position *)
END_VAR
```

名称	类型	描述
执行	BOOL	命令在上升沿执行。
CamTableID	MC_CAM_ID	加载表（运动功能类型）的 ID [▶ 46]。
ValueSelectMask	UINT	选择掩码，可用于指定哪些数据将参与插补并返回。该值通过添加数据类型 MC_ValueSelectType [▶ 61] 的单个值创建。 CamTable [▶ 47] 所描述的数据结构的列数必须与 ValueSelectMask [▶ 61] 确定的列数一致。例如，如果只读取位置数据，则列数为 2（主轴和从轴位置）。每多一次求导（速度、加速度、加加速度），列的数量都会增加 1。
StartPosMaster	LREAL	第一个插补点的主轴的位置值。
EndPosMaster	LREAL	最后一个插补点的主轴的位置值。
增量	LREAL	插补步长

输入/输出

```
VAR_IN_OUT
   CamTable : MC_CAM_REF;
END_VAR
```

名称	类型	描述
CamTable	MC_CAM_REF	表的 参考 [▶ 47]（结构）。

输出

```
VAR_OUTPUT
   Done          : BOOL;
   Busy          : BOOL;
   Error         : BOOL;
   ErrorID       : UDINT;
END_VAR
```

名称	类型	描述
Done	BOOL	如果数据被成功读取，则变为“TRUE”。

名称	类型	描述
Busy	BOOL	当用 *Execute* 开始执行命令时，*Busy* 输出变为“TRUE”，并且在命令执行过程中保持“TRUE”状态。如果 *Busy* 再次变成“FALSE”，功能块就可以重新执行。同时，其中一个输出被设定： *Done* 或 *Error*。
错误	BOOL	一旦发生错误，则变为“TRUE”。
ErrorID	UDINT	如果错误输出已设定，此参数提供 错误代码。

7 状态

7.1 MC_ReadCamTableSlaveDynamics

功能块 *MC_ReadCamTableSlaveDynamics* 可用于确定凸轮表某一点的从轴动态特性。该功能块对原始凸轮表数据进行评估， 而不考虑任何缩放。

对于较老的凸轮表类型 [▶ 61]，不是所有的动态参数都可以确定。下面的概述显示了预期结果：

MC_TABLETYPE_MOTIONFUNCTION：确定从轴位置、速度和加速度。

MC_TABLETYPE_EQUIDISTANT：确定从轴位置和速度， 加速度始终为 0。

MC_TABLETYPE_NONEQUIDISTANT：确定从轴位置， 速度和加速度始终为 0。

输入

```
VAR_INPUT
   Execute         : BOOL;
   CamTableID      : MC_CAM_ID;
   MasterPosition  : LREAL;
   Options         : ST_ReadCamTableSlaveDynamicsOptions;
END_VAR
```

名称	类型	描述	
Execute	BOOL	命令在 *Execute* 输入的上升沿执行。	
CamTableID	MC_CAM_ID	加载表的 ID [▶ 46]。	
MasterPosition	LREAL	需要确定从轴动态的位置点，即表内的主轴位置。	
Options	ST_ReadCamTableSlaveDynamicsOptions [▶ 66]	具有更多选项的数据结构：	
		SynchronousAccess	写入通过同步访问进行，没有延迟时间。应仅用于时间要求极高的应用。

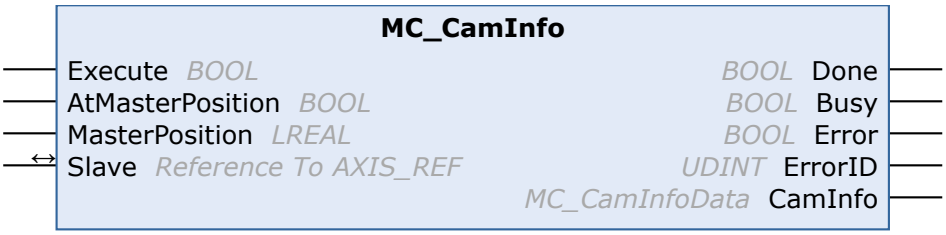
输出

```
VAR_OUTPUT
   Done               : BOOL;
   Busy               : BOOL;
   SlavePosition      : LREAL;
   SlaveVelocity      : LREAL;
   SlaveAcceleration  : LREAL;
   Error              : BOOL;
   ErrorID            : UDINT;
END_VAR
```

名称	类型	描述
Done	BOOL	如果命令被成功执行，则变为“TRUE”。
Busy	BOOL	当用 *Execute* 开始执行命令时，*Busy* 输出变为“TRUE”，并且在命令执行过程中保持“TRUE”状态。如果 *Busy* 再次变成“FALSE”，功能块就可以重新执行。同时，其中一个输出被设定： *Done* 或 *Error*。
SlavePosition	LREAL	凸轮盘表内指定主轴位置的从轴位置。
SlaveVelocity	LREAL	凸轮盘表内指定主轴位置的从轴速度。

名称	类型	描述
SlaveAcceleration	LREAL	凸轮盘表内指定主轴位置的从轴加速度。
错误	BOOL	一旦发生错误，则变为“TRUE”。
ErrorID	UDINT	如果错误输出已设定，此参数提供 错误代码。

7.2 MC_CamInfo



MC_CamInfo 功能块用于确定凸轮盘耦合的当前状态和当前参数化有关的数据。使用该命令的前提是从轴已经通过凸轮盘耦合。如果 *AtMasterPosition* 输入为“TRUE”，则不是确定当前状态，而是确定与指定的主位置相关的状态。获得的数据被放入 *CamInfo* 数据结构中。

注意：如果耦合的轴组进入错误状态（如紧急停止），该功能块将返回耦合的最新有效状态。必须在解耦从轴之前调用该功能块。获得的数据可以用于将耦合恢复到原来的轴位置。

输入

```
VAR_INPUT
   Execute          : BOOL;
   AtMasterPosition : BOOL;
   MasterPosition   : LREAL;
END_VAR
```

名称	类型	描述
执行	BOOL	命令在 *Execute* 输入的上升沿执行。
AtMasterPosition	BOOL	如果 *AtMasterPosition* 为“TRUE”，则确定与指定的 *MasterPosition* 有关的数据。否则，数据指的是当前的主轴位置。
MasterPosition	LREAL	确定的数据所指的主轴位置。如果 *AtMasterPosition* 为“FALSE”，则不需要此输入参数。

输入/输出

```
VAR_IN_OUT
   Slave    : AXIS_REF;
END_VAR
```

名称	类型	描述
从轴	AXIS_REF	从轴数据结构。

AXIS_REF 类型的轴数据结构，应当在系统中对应到唯一的轴。通过诸多其他参数，轴数据结构包含了轴的当前状态：位置、速度或错误状态等信息。

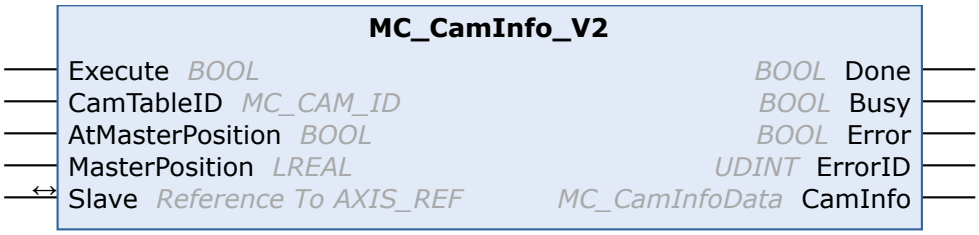
输出

```
VAR_OUTPUT
   Done    : BOOL;
   Busy    : BOOL;
   Error   : BOOL;
   ErrorID : UDINT;
   CamInfo : MC_CamInfoData;
END_VAR
```

名称	类型	描述
Done	BOOL	如果命令被成功执行，则变为“TRUE”。
Busy	BOOL	当用 *Execute* 开始执行命令时，*Busy* 输出变为“TRUE”，并且在命令执行过程中保持“TRUE”状态。如果 *Busy* 再次变成“FALSE”，功能块就可以重新执行。同时，其中一个输出被设定： *Done* 或 *Error*。
错误	BOOL	一旦发生错误，则变为“TRUE”。
ErrorID	UDINT	如果错误输出已设定，此参数提供 错误代码。

名称	类型	描述
CamInfo	MC_CamInfoData	CamInfo [▶ 54] 数据结构包含所有关于凸轮盘耦合的已确定数据

7.3 MC_CamInfo_V2



MC_CamInfo_V2 功能块获得与凸轮盘耦合的当前状态和当前参数化有关的数据。使用该命令的前提是从轴已经通过凸轮盘耦合。如果 *AtMasterPosition* 输入为“TRUE”，则不是确定当前状态，而是确定与指定的主位置相关的状态。获得的数据被放入 *CamInfo* 数据结构中。对于多凸轮盘耦合，*CamTableId* 也必须明确传递。如果在耦合中涉及单个凸轮盘，*CamTableId* 可以设置为 0。

注意：如果耦合的轴组进入错误状态（如紧急停止），该功能块将返回耦合的最新有效状态。必须在解耦从轴之前调用该功能块。获得的数据可以用于将耦合恢复到原来的轴位置。

输入

```
VAR_INPUT
   Execute          : BOOL;
   CamTableId       : BOOL;
   AtMasterPosition : BOOL;
   MasterPosition   : LREAL;
END_VAR
```

名称	类型	描述
执行	BOOL	命令在 *Execute* 输入的上升沿执行。
CamTableId	BOOL	对于多凸轮盘耦合，也必须明确传递，以便在系统中明确识别凸轮。
AtMasterPosition	BOOL	如果 *AtMasterPosition* 为“TRUE”，则确定与指定的 *MasterPosition* 有关的数据。否则，数据指的是当前的主轴位置。
MasterPosition	LREAL	确定的数据所指的主轴位置。如果 *AtMasterPosition* 为“FALSE”，则不需要此输入参数。

输入/输出

```
VAR_IN_OUT
   Slave    : AXIS_REF;
END_VAR
```

名称	类型	描述
从轴	AXIS_REF	从轴数据结构。

AXIS_REF 类型的轴数据结构，应当在系统中对应到唯一的轴。通过诸多其他参数，轴数据结构包含了轴的当前状态：位置、速度或错误状态等信息。

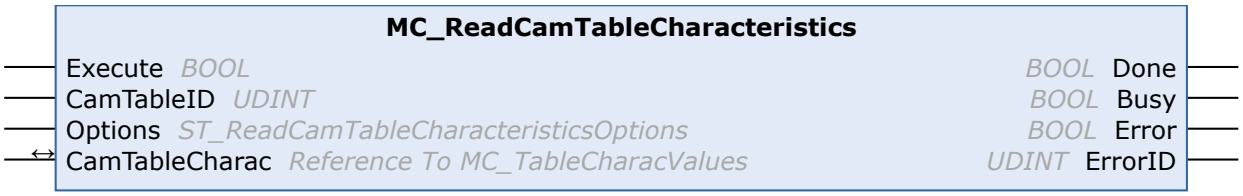
输出

```
VAR_OUTPUT
   Done          : BOOL;
   Busy          : BOOL;
   Error         : BOOL;
   ErrorID       : UDINT;
END_VAR
```

名称	类型	描述
Done	BOOL	当功能块被成功执行时，变为“TRUE”。
Busy	BOOL	当用 *Execute* 开始执行命令时，*Busy* 输出变为“TRUE”，并且在命令执行过程中保持“TRUE”状态。如果 *Busy* 再次变成“FALSE”，功能块就可以重新执行。同时，其中一个输出被设定： *Done* 或 *Error*。

名称	类型	描述
错误	BOOL	一旦发生错误，则变为“TRUE”。
ErrorID	UDINT	如果错误输出已设定，此参数提供 错误代码。

7.4 MC_ReadCamTableCharacteristics



功能块 *MC_ReadCamTableCharacteristics* 用于计算和读取运动功能的特征参数。参数包括位置、速度、加速度和加加速度的最小和最大值。

输入

```
VAR_INPUT
   Execute    : BOOL;
   CamTableID : MC_CAM_ID;
END_VAR
```

名称	类型	描述
执行	BOOL	命令在 *Execute* 输入的上升沿执行。
CamTableId	MC_CAM_ID	表 ID [▶ 46]

输入/输出

```
VAR_IN_OUT
   CamTableCharac : MC_TableCharacValues;
END_VAR
```

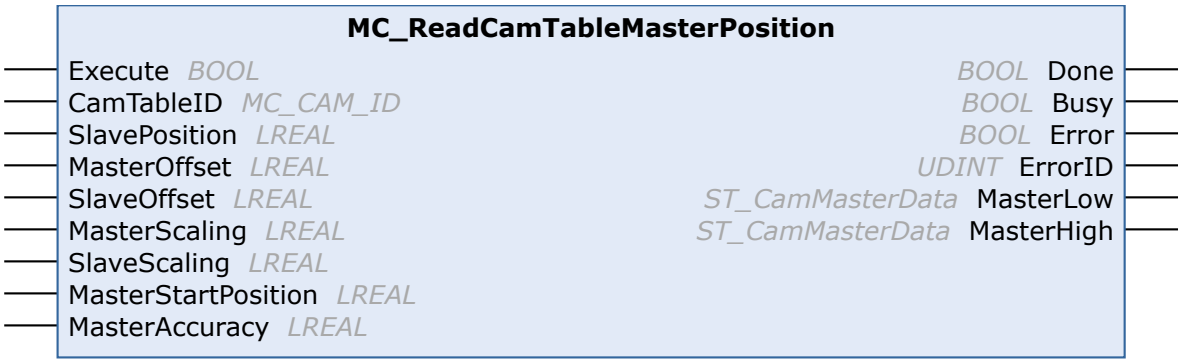
名称	类型	描述
CamTableCharac	MC_TableCharacValues	带有运动功能特征参数的 数据结构 [▶ 60]

输出

```
VAR_OUTPUT
   Done    : BOOL;
   Busy    : BOOL;
   Error   : BOOL;
   ErrorID : UDINT;
END_VAR
```

名称	类型	描述
Done	BOOL	如果计算成功执行，则为“TRUE”。
Busy	BOOL	当用 *Execute* 开始执行命令时，*Busy* 输出变为“TRUE”，并且在命令执行过程中保持“TRUE”状态。如果 *Busy* 再次变成“FALSE”，功能块就可以重新执行。同时，其中一个输出被设定： *Done* 或 *Error*。
错误	BOOL	一旦发生错误，则变为“TRUE”。
ErrorID	UDINT	如果错误输出已设定，此参数提供 错误代码。

7.5 MC_ReadCamTableMasterPosition



功能块 *MC_ReadCamTableMasterPosition* 可用于计算给定从轴位置的主轴位置。虽然一个给定主轴位置的从轴位置肯定是唯一的，但反过来并非如此。为了限制功能块的主轴输出选项的数量，对于一个给定的主轴位置（*MasterStartPosition*），会输出从轴值的较小主轴位置（*MasterLow*）和较大主轴位置（*MasterHigh*）。

例如，对于图 1 中的凸轮盘，当从轴值为 80，主轴启动值为 180 时，*MasterHigh* 的输出值为 225，*MasterLow* 的输出为 135。如果凸轮盘是循环的，对于一个主轴启动值 90，除了计算出 *MasterHigh* 为 135，还计算出 *MasterLow* 为 -135。在线性凸轮盘的情况下（非循环），图 2 中仅数值 *MasterHigh* 显示为有效。

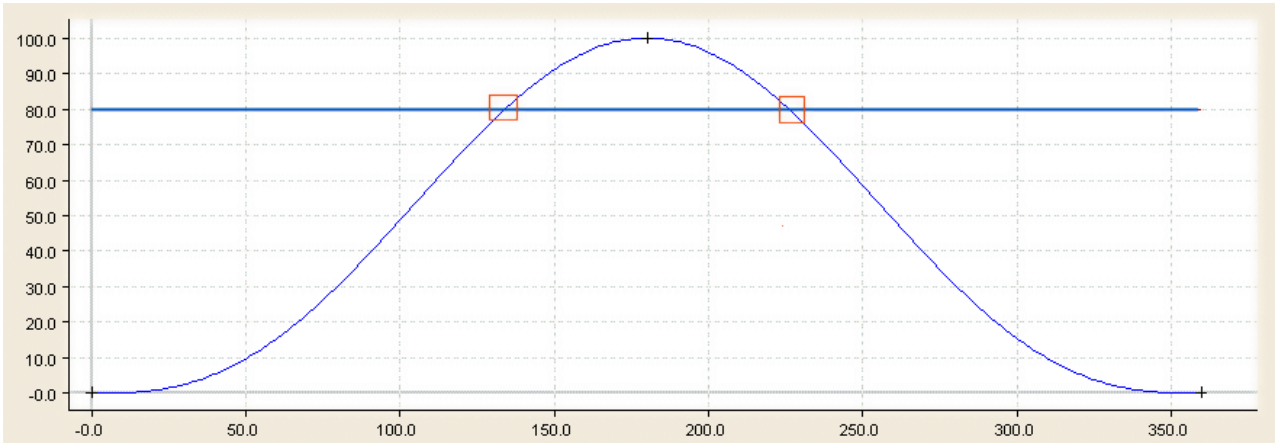


图 1

在带节距的循环凸轮盘中，主轴位置不仅可以位于与 StartMasterpos 相邻的一个周期中，还可以位于更远的几个周期中，这取决于从轴位置。

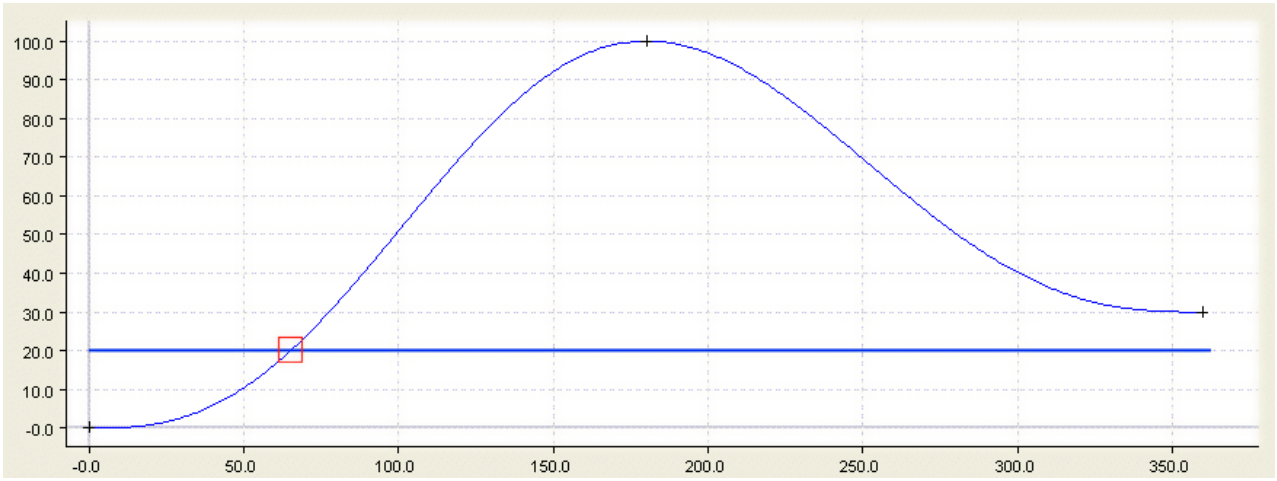


图 2

主轴位置用数学公式进行计算，其精度可以通过变量 *MasterAccuracy* 来设置。

输入

```
VAR_INPUT
   Execute             : BOOL;
   CamTableID          : MC_CAM_ID;
   SlavePosition       : LREAL; (* absolute slave axis position *)
   MasterOffset        : LREAL; (* E *)
   SlaveOffset         : LREAL; (* E *)
   MasterScaling       : LREAL := 1.0; (* E *)
   SlaveScaling        : LREAL := 1.0; (* E *)
   MasterStartPosition : LREAL; (* Master position to start the iteration from *)
   MasterAccuracy      : LREAL; (* Master iteration accuracy *)
END_VAR
```

名称	类型	描述
执行	BOOL	命令在 *Execute* 输入的上升沿执行。
CamTableId	MC_CAM_ID	表 ID [▶ 46]
SlavePosition	LREAL	寻找主轴位置的从轴位置。
MasterOffset	LREAL	至凸轮盘主轴位置的偏移量
SlaveOffset	LREAL	至凸轮盘从轴位置的偏移量
MasterScaling	LREAL	凸轮盘主轴位置的缩放
SlaveScaling	LREAL	凸轮盘从轴位置的缩放
MasterStartPosition	LREAL	主轴的起始位置
MasterAccuracy	LREAL	计算的准确性

输出

```
VAR_OUTPUT
   Execute    : BOOL;
   Done       : BOOL;
   Busy       : BOOL;
   Active     : BOOL;
   Error      : BOOL;
   ErrorID    : UDINT;
   MasterLow  : ST_CamMasterData; (* position information of the lower bound master position *)
   MasterHigh : ST_CamMasterData; (* position information of the upper bound master position *)
END_VAR
```

名称	类型	描述
Done	BOOL	如果耦合成功并且凸轮盘处于活动状态，则变为“TRUE”。
Busy	BOOL	当用 *Execute* 开始执行命令时，*Busy* 输出变为“TRUE”，并且在命令执行过程中保持“TRUE”状态。如果 *Busy* 再次变成“FALSE”，功能块就可以重新执行。同时，其中一个输出被设定： *Done* 或 *Error*。
错误	BOOL	一旦发生错误，则变为“TRUE”。
ErrorID	UDINT	如果错误输出已设定，此参数提供 错误代码。
MasterLow	ST_CamMasterData	主轴位置小于数据结构 ST_CamMasterData [▶ 64] 中的 MasterStartPosition
MasterHigh	ST_CamMasterData	主轴位置大于数据结构 ST_CamMasterData [▶ 64] 中的 MasterStartPosition

8 数据类型

8.1 MC_CAM_ID

```
TYPE
   MC_CAM_ID : UDINT;
END_TYPE
```

表 ID 的类型定义。

8.2 MC_CAM_REF

```
TYPE MC_CAM_REF :
STRUCT
   pArray      : UDINT;
   ArraySize   : UDINT;
   TableType   : MC_TableType;
   NoOfRows    : UDINT;
   NoOfColumns : UDINT;
END_STRUCT
END_TYPE
```

数据结构 *MC_CAM_REF* 在另一个 PLC 变量（数组）中描述了凸轮的数据存储结构。

第一个参数 *pArray* 是一个指向含有凸轮数据的数据结构的指针。数据结构取决于表类型 *nTableType*。行的数量在元素 nNoOfRows 中输入，列的数量在 nNoOfCols 中输入（通常是 1 或 2）。

示例 1：位置表结构描述

名称	类型	描述
pArray	UDINT	一个二维数组的地址。第一列包含一个主轴位置的升序列表。第二列包含对应的从轴位置。该地址可以用 ADR 函数分配。 示例： Table1 : ARRAY[0..360, 0..1] OF LREAL; pArray := ADR（Table1）;
ArraySize	UDINT	二维数组的存储容量，可以用 SIZEOF 函数确定。 示例： ArraySize := SIZEOF（Table1）;
TableType	MC_TableType	如果主轴位置间距均匀，则表类型 [▶ 61] 为 *MC_TABLETYPE_EQUIDISTANT* 或 如果间距不均匀，则为 *MC_TABLETYPE_NONEQUIDISTANT*。
NoOfRows	UDINT	行数，与表点的数量相对应。
NoOfColumns	UDINT	列数，等于 2。

示例 2：运动函数的结构描述

名称	类型	描述
pArray	UDINT	MC_MotionFunctionPoint [▶ 56] 类型的一维数组的地址。每个数组元素包含一个凸轮插补点的描述。 示例： MotionFunction : ARRAY[1..10] OF MC_MotionFunctionPoint; pArray := ADR（MotionFunction）;
ArraySize	UDINT	一维数组的存储容量，可以用 SIZEOF 函数确定。 示例： ArraySize := SIZEOF（MotionFunction）;
TableType	MC_TableType	表类型为 *MC_TABLETYPE_MOTIONFUNCTION*。
NoOfRows	UDINT	行数，与表点的数量相对应。
NoOfColumns	UDINT	列数，等于 1。

8.3 MC_CamActivationMode

```
TYPE MC_CamActivationMode :
(
   (* instantaneous change *)
   MC_CAMACTIVATION_INSTANTANEOUS,

   (* modify the data at a defined master position referring to the cam tables master position *)
   MC_CAMACTIVATION_ATMASTERCAMPOS,

   (* modify the data at a defined master position referring to the absolute master axis position *)
   MC_CAMACTIVATION_ATMASTERAXISPOS

   (* modify the data at the beginning of the next cam table cycle *)
   MC_CAMACTIVATION_NEXTCYCLE,

   (* not yet implemented!
   modify the data at the beginning of the next cam table cycle, activation is valid for one cycle
only *)
   MC_CAMACTIVATION_NEXTCYCLEONCE,

   (* modify the data as soon as the cam table is in a safe state to change its data *)
   MC_CAMACTIVATION_ASSOONASPOSSIBLE,

   (* don't accept any modification *)
   MC_CAMACTIVATION_OFF,

   (* delete all data which was written to modify the cam table but is still not activated *)
   MC_CAMACTIVATION_DELETEQUEUEDDATA,

   (* special mode at a defined master axis position in a defined positive direction *)
   MC_CAMACTIVATION_ATMASTERAXISPOS_POSITVEDIRECTION,

   (* special mode at a defined master axis position in a defined negative direction *)
   MC_CAMACTIVATION_ATMASTERAXISPOS_NEGATIVEDIRECTION
);
END_TYPE
```

MC_CamActivationMode 指定了一个凸轮的变化时间和类型。通过缩放、修改凸轮数据或切换凸轮都可以造成凸轮的变化。

该变化具有下列几种模式：

凸轮表的缩放

凸轮表可以用功能块 MC_CamScaling [▶ 17] 进行缩放。有以下激活模式可供选择。

MC_CAMACTIVATION_INSTANTANEOUS	缩放立即生效。
MC_CAMACTIVATION_ATMASTERCAMPOS	缩放在某一凸轮表位置（凸轮表内的主轴位置）生效。缩放命令必须在这个位置之前发出。 该位置指的是非缩放的凸轮表。如果项目中的位置指的是缩放后的凸轮表主轴位置，那么在调用功能块之前，要除以 *MasterScaling*。
MC_CAMACTIVATION_ATMASTERAXISPOS	缩放在主轴的某个绝对位置上生效。缩放命令必须在这个位置之前发出。
MC_CAMACTIVATION_NEXTCYCLE	对于循环凸轮表，到下一个周期的过渡点缩放才会生效。
MC_CAMACTIVATION_OFF	不进行缩放。例如，这个模式可以用来将缩放限制在一个轴（主轴或从轴）上。

设置在线改变凸轮表的模式（插补点数据的写入）

MC_SetCamOnlineChangeMode [▶ 33] 用于指定修改后的凸轮表数据何时生效（另见 MC_WriteMotionFunction [▶ 31] 和 MC_WriteMotionFunctionPoint [▶ 32]）。

两种情况下均具有下列模式：

MC_CAMACTIVATION_INSTANTANEOUS	变化立即生效。

MC_CAMACTIVATION_ATMASTERCAMPOS	该变化在某一凸轮表位置（凸轮表内的主轴位置）生效。该命令必须在这个位置之前发出。 该位置指的是非缩放的凸轮表。如果项目中的位置指的是缩放后的凸轮表主轴位置，那么在调用功能块之前，要除以 *MasterScaling*。
MC_CAMACTIVATION_ATMASTERAXISPOS	变化在主轴的某个绝对位置生效。该命令必须在这个位置之前发出。
MC_CAMACTIVATION_NEXTCYCLE	对于循环凸轮表，该变化在过渡到下一个周期时生效。
MC_CAMACTIVATION_ASSOONASPOSSIBLE	修改后的凸轮表数据在系统动态特性允许的情况下尽快生效。
MC_CAMACTIVATION_OFF	忽略凸轮表数据的变化
mc_camactivation_deletequeueddata	排队等待生效的凸轮表数据被删除。如果数据修改请求发生在某一主轴位置或周期结束时，则数据会排队等待。

凸轮表的耦合

功能块 MC_CamIn [▶ 12] 可用于轴与凸轮表的耦合。可以选择使用 *ActivationMode* 来指定从轴从哪个位置开始激活。

MC_CAMACTIVATION_INSTANTANEOUS	凸轮耦合立即生效，从轴根据凸轮表数据移动。
MC_CAMACTIVATION_ATMASTERCAMPOS	暂停激活凸轮表。根据凸轮表数据，从轴只按确定的凸轮表位置（凸轮表内的主轴位置）移动。 在耦合模式下，*ActivationMode* 不能与 MC_StartMode [▶ 62] = MC_STARTMODE_RELATIVE 或 MC_STARTMODE_MASTERREL_SLAVEABS 一起使用。 该位置指的是非缩放的凸轮表。如果项目中的位置指的是缩放后的凸轮表主轴位置，那么在调用功能块之前，要除以 *MasterScaling*。
MC_CAMACTIVATION_ATMASTERAXISPOS	暂停激活凸轮表。根据凸轮表数据，从轴只按主轴的定义绝对位置移动。
MC_CAMACTIVATION_NEXTCYCLE	暂停激活凸轮表。从轴从下一个周期转换开始移动（对于循环凸轮表）。 在耦合模式下，*ActivationMode* 不能与 MC_StartMode [▶ 62] = MC_STARTMODE_RELATIVE 或 MC_STARTMODE_MASTERREL_SLAVEABS 一起使用。
MC_CAMACTIVATION_ATMASTERAXISPOS_POSITVEDIRECTION	凸轮耦合在主轴正向通过预定位置时生效。 （该模式是 MC_CAMACTIVATION_ATMASTERAXISPOS 的一个特例，以确保在当前位置附近安全激活，即使是在非常低的速度和短时换向时（由于噪声））
MC_CAMACTIVATION_ATMASTERAXISPOS_NEGATIVEDIRECTION	凸轮耦合在主轴反向通过预定位置时生效。 （该模式是 MC_CAMACTIVATION_ATMASTERAXISPOS 的一个特例，以确保在当前位置附近安全激活，即使是在非常低的速度和短时换向时（由于噪声））

凸轮表的切换

功能块 MC_CamIn [▶ 12] 可用于在耦合状态下的凸轮表之间进行切换。*ActivationMode* 可用于指定从哪个位置开始切换。

MC_CAMACTIVATION_INSTANTANEOUS	凸轮表立即被切换，从轴根据新的凸轮表数据移动。
MC_CAMACTIVATION_ATMASTERCAMPOS	凸轮表的切换发生在一个定义的凸轮表位置（凸轮表内的主轴位置）。 该位置指的是非缩放的凸轮表。如果项目中的位置指的是缩放后的凸轮表主轴位置，那么在调用功能块之前，要除以 *MasterScaling*。
MC_CAMACTIVATION_ATMASTERAXISPOS	凸轮表的切换是在定义的绝对主轴位置进行的。

MC_CAMACTIVATION_NEXTCYCLE	对于循环凸轮表，凸轮表的切换是在下一个循环转换时发生的。对于线性凸轮表，切换发生在一个定义区域的边缘。
mc_camactivation_deletequeueddata	暂停的凸轮表切换过程如果没有激活就会被放弃。
MC_CAMACTIVATION_ATMASTERAXISPOS_POSITVEDIRECTION	凸轮表切换在主轴正向通过预定位置时生效。（该模式是 MC_CAMACTIVATION_ATMASTERAXISPOS 的一个特例，以确保在当前位置附近安全激活，即使是在非常低的速度和短时换向时（由于噪声））
MC_CAMACTIVATION_ATMASTERAXISPOS_NEGATIVEDIRECTION	凸轮表切换在主轴反向通过预定位置时生效。。（该模式是 MC_CAMACTIVATION_ATMASTERAXISPOS 的一个特例，以确保在当前位置附近安全激活，即使是在非常低的速度和短时换向时（由于噪声））

8.4 MC_CamScalingMode

```
TYPE MC_CamScalingMode :
(
   (* user defines scaling parameters -scaling and -offset *)
   MC_CAMSCALING_USERDEFINED,

   (* offset is calculated automatically for best result *)
   MC_CAMSCALING_AUTOOFFSET,

   (* no modification accepted *)
   MC_CAMSCALING_OFF
);
END_TYPE
```

通过函数块 MC_CamScaling [▶ 17] 实现凸轮耦合缩放的类型和范围。

MC_CAMSCALING_USERDEFINED：缩放和偏移保留不变。用户必须计算出缩放和偏移量，从而避免位置的跳跃。

MC_CAMSCALING_AUTOOFFSET：缩放生效，系统调整偏移量，从而避免位置的跳变 。然而，缩放应该在从轴速度为 0 的阶段发生，否则无法避免速度的跳跃。

MC_CAMSCALING_OFF：忽略缩放和偏移。该模式用于只进行从轴缩放（即没有主轴缩放）的情况。

Autooffset

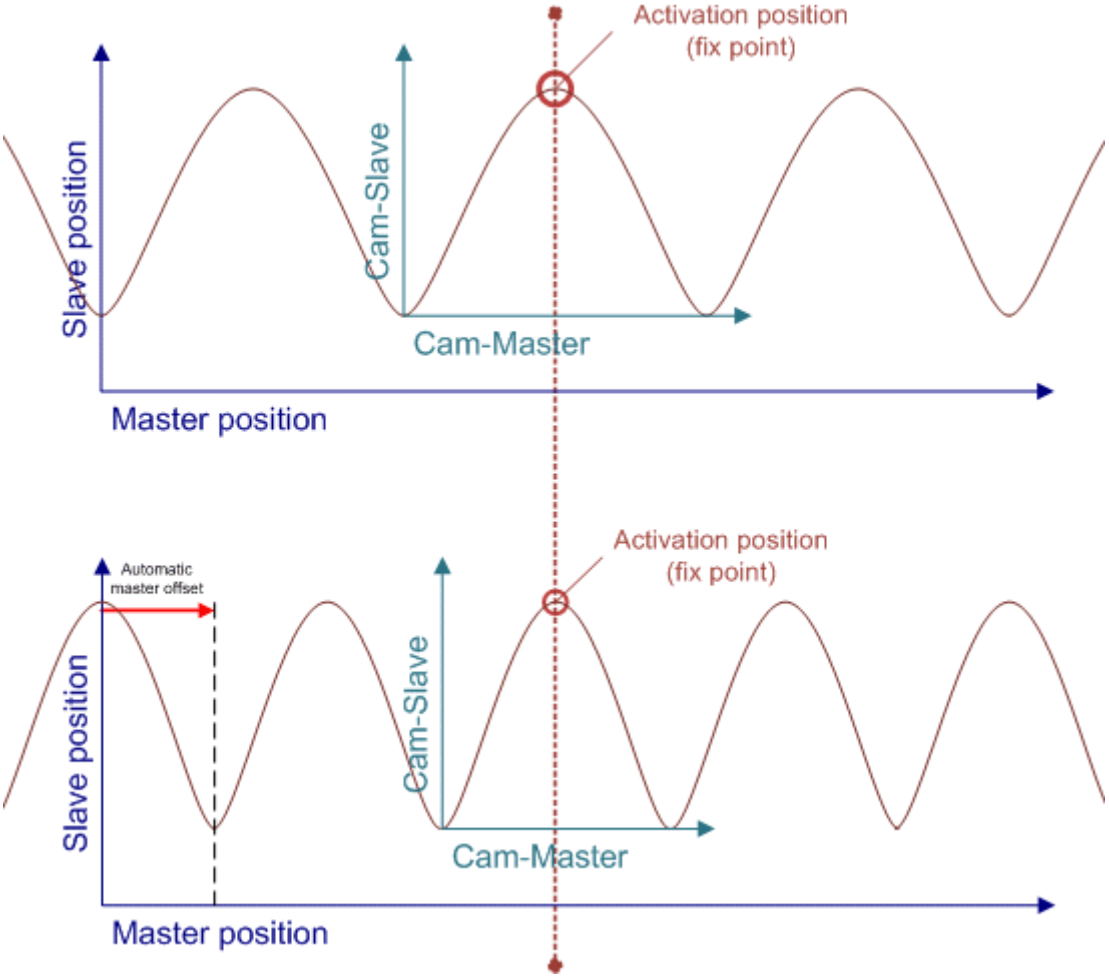
Autooffset 模式确保自动适应凸轮表的偏移。*Autooffset* 可独立用于凸轮表的主轴或从轴，并影响凸轮表的切换和缩放。该功能块基于以下描述的规则运行。

Master-Autooffset

Master-Autooffset 防止在切换不同主轴周期的凸轮表或缩放凸轮表（主轴缩放）时凸轮表在轴坐标系中的主位置不连续。之所以需要这个功能块，是因为凸轮表在轴坐标系中的相对位置取决于主轴周期。如果通过缩放等方式主轴周期改变，则位置也会改变。

作为一项规则，*Master Autooffset* 假设已经存在一个凸轮表作为相应轴耦合的参考。因此，第一次耦合时不能使用这个选项。*Master-Autooffset* 确定凸轮表的主偏移量，从而保持凸轮表内的主轴位置。对于缩放或切换到具有不同主轴周期的凸轮表，这意味着切换之前和之后的相对（百分比）位置是相同的。

示例：一个凸轮表的主轴周期为 360°，以 2 为系数放大后为 720°。缩放发生在凸轮表内的 90° 位置，即在一个周期开始的 25% 处。因此，在缩放之后，凸轮表中 180° 处的相对主轴位置也是一个周期开始时的 25%。



在凸轮表边缘切换期间（见 MC_CamActivationMode [▶ 48] = *MC_CAMACTIVATION_NEXTCYCLE*），Master-Autooffset 确保凸轮表的无缝衔接，循环和线性凸轮表都是如此。

Master-Autooffset 不能用于以相对模式耦合或切换的凸轮表，因为这些功能是相互排斥的。进一步的限制适用于初始耦合。这些情况在下表中显示。

		Coupling with Cam Tables			
	Master-ScalingMode:	**without** Master-AutoOffset		**with** Master-AutoOffset	
	StartMode:	Absolute	Relative	Absolute	Relative
Activation Mode	Instantaneous (default)	√	√	—	—
	AtMasterAxisPos	√	√	—	—
	AtMasterCamPos	√	—	—	—
	NextCycle	√	—	—	—
	DeleteQueuedData	—	—	—	—

Activation Mode	Switching of Cam Tables			
Master-ScalingMode:	**without** Master-AutoOffset		**with** Master-AutoOffset	
StartMode:	Absolute	Relative	Absolute	Relative
Instantaneous (default)	√	√	√	—
AtMasterAxisPos	√	√	√	—
AtMasterCamPos	√	√	√	—
NextCycle	√	√	√	—
DeleteQueuedData	√	√	√	—

Slave-Autooffset

Slave-Autooffset 计算一个slave offset，从而避免在凸轮表切换或缩放时从轴位置不连续。调整slave offset，以确保从轴在动作前后的位置是相同的。

如果 *Master Autooffset* 和 *Slave-Autooffset* 都用于凸轮表切换或缩放，则首先计算master offset，然后计算slave offset。

Slave Autooffset 可与任何 MC_StartMode [▶ 62] 结合使用。它调整凸轮表，使从轴位置没有跳跃。

8.5 MC_CamInfoData

```
TYPE MC_CamInfoData :
STRUCT
   CamTableId                   : MC_CAM_ID;
   TableType                    : MC_TableType;
   Periodic                     : BOOL;
   InterpolationType            : MC_InterpolationType;
   NumberOfRows                 : UDINT; (* number of cam table entries, e. g. number of points *)
   NumberOfColumns              : UDINT; (* number of table columns, typically 1 or 2 *)
   MasterAxisId                 : UDINT; (* AxisId of the master axis *)
   SlaveAxisId                  : UDINT; (* AxisId of the slave axis *)
   MasterCamStartPos            : LREAL; (* Master pos. of the first cam table point (raw, unscaled
cam table pos.) *)
   SlaveCamStartPos             : LREAL; (* Slave pos. of the first cam table point (raw, unscaled
cam table pos.) *)
   RawMasterPeriod              : LREAL; (* raw, unscaled difference between last and first cam
point *)
   RawSlaveStroke               : LREAL; (* raw, unscaled difference between last and first cam
point *)
   MasterAxisCouplingPos        : LREAL; (* Master axis position when slave has been coupled *)
   SlaveAxisCouplingPos         : LREAL; (* Slave axis position when slave has been coupled *)
   MasterAbsolute               : BOOL;  (* raw, unscaled distance from first to last master cam
table position *)
   SlaveAbsolute                : BOOL;  (* raw, unscaled distance from first to last slave cam
table position *)
   MasterOffset                 : LREAL; (* total master offset *)
   SlaveOffset                  : LREAL; (* total slave offset *)
   MasterScaling                : LREAL; (* total master scaling factor *)
   SlaveScaling                 : LREAL; (* total slave scaling factor *)
   SumOfSlaveStrokes            : LREAL; (* sum of the slave strokes up to ActualMasterAxisPos *)
   SumOfSuperpositionDistance   : LREAL; (* sum of additional moves through MC_MoveSuperimposed *)
   ActualMasterAxisPos          : LREAL; (* absolute set position of the master axis *)
   ActualSlaveAxisPos           : LREAL; (* absolute set position of the slave axis *)
   ActualMasterCamPos           : LREAL; (* raw, unscaled cam table position of the master *)
   ActualSlaveCamPos            : LREAL; (* raw, unscaled cam table position of the slave *)

   (* mode for the scaling of cam tables *)
   ScalingPending               : BOOL;  (* a change is currently pending *)
   ScalingActivationMode        : MC_CamActivationMode;
   ScalingActivationPos         : LREAL;
   ScalingMasterScalingMode     : MC_CamScalingMode;
   ScalingSlaveScalingMode      : MC_CamScalingMode;

   (* mode for online changes of cam table data *)
   CamDataQueued                : BOOL;  (* a change is currently pending *)
   OnlineChangeActivationMode   : MC_CamActivationMode;
   OnlineChangeActivationPos    : LREAL;
   OnlineChangeMasterScalingMode: MC_CamScalingMode;
   OnlineChangeSlaveScalingMode : MC_CamScalingMode;

   (* mode for exchanging cam tables with MC_CamIn *)
   CamTableQueued               : BOOL;  (* a change is currently pending *)
   CamExchangeCamTableID        : MC_CAM_ID;
   CamExchangeActivationMode    : MC_CamActivationMode;
   CamExchangeActivationPos     : LREAL;
   CamExchangeMasterScalingMode : MC_CamScalingMode;
   CamExchangeSlaveScalingMode  : MC_CamScalingMode;
END_STRUCT
END_TYPE
```

数据结构*MC_CamInfoData*包含了凸轮耦合的当前状态数据。这些数据通过 MC_CamInfo [▶ 39] 函数块确定。

该结构包含了与主轴或从轴坐标系对应的轴绝对位置， 还包含参考凸轮坐标系的凸轮位置（例如：*ActualMasterCamPos* 和 *ActualSlaveCamPos*）。所有凸轮位置都基于非缩放的凸轮坐标系，如果需要，可以转换为缩放的坐标系。一个主轴凸轮位置可以与 *MasterScaling* 因子相乘，一个从轴凸轮位置可以与 *SlaveScaling* 因子相乘。

激活位置 *ActivationPos* 基于主轴坐标系还是凸轮坐标系，取决于 *ActivationMode*。如果是后者，则定义的是非缩放凸轮位置。

8.6 MC_InterpolationType

位置表（凸轮）的插补模式。位置表由主轴和从轴的位置列表组成，它们之间可以用不同的方式进行插补。

插补类型不用于扩展凸轮（运动函数）。

```
TYPE MC_InterpolationType :
(
   (* linear 2 point interpolation *)
   MC_INTERPOLATIONTYPE_LINEAR             := 0,

   (* no longer supported - 4 point interpolation (for equidistant tables only) *)
   MC_INTERPOLATIONTYPE_4POINT             := 1,

   (* spline interpolation (tangential or cyclic depending on table) *)
   MC_INTERPOLATIONTYPE_SPLINE             := 2,

   (* moving cubic spline interpolation with n sampling points ('local spline') *)
   MC_INTERPOLATIONTYPE_SLIDINGSPLINE      := 3
);
END_TYPE
```

8.7 MC_MotionFunctionPoint

数据结构 *MC_MotionFunctionPoint* 描述了一个运动函数的插补点。运动函数是 *MC_MotionFunctionPoint* 类型的一维列表（数组）。

```
TYPE MC_MotionFunctionPoint :
STRUCT
   PointIndex          : MC_MotionFunctionPoint_ID;
   FunctionType        : MC_MotionFunctionType;
   PointType           : MC_MotionPointType;
   RelIndexNextPoint   : MC_MotionFunctionPoint_ID;
   MasterPos           : LREAL; (* X *)
   SlavePos            : LREAL; (* Y *)
   SlaveVelo           : LREAL; (* Y' *)
   SlaveAcc            : LREAL; (* Y'' *)
   SlaveJerk           : LREAL; (* Y''' *)
END_STRUCT
END_TYPE
```

名称	类型	描述
PointIndex	MC_MotionFunctionPoint_ID	该插补点在运动函数中的绝对索引。所有插补点的索引号必须严格单向递增，必须无间隔且大于 0。
FunctionType	MC_MotionFunctionType	此点与后续插补点之间进行插补的数学函数类型 MC_MotionFunctionType [▶ 58]。
PointType	MC_MotionPointType	此插补点的类型 MC_MotionPointType [▶ 59]
RelIndexNextPoint	MC_MotionFunctionPoint_ID	后续插补点的相对索引（通常为 1）。对于后续的“MOTIONPOINTTYPE_IGNORE”，也必须相应地设置参考。
MasterPos	LREAL	该插补点的主轴位置
SlavePos	LREAL	该插补点的从轴位置
SlaveVelo	LREAL	该插补点的从轴速度
SlaveAcc	LREAL	该插补点的从轴加速度
SlaveJerk	LREAL	该插补点的从轴加加速度

8.8 MC_MotionFunctionPoint_ID

```
TYPE
   MC_MotionFunctionPoint_ID : UDINT;
END_TYPE
```

用于运动函数的插补点 ID 的类型定义。

8.9 MC_MotionFunctionType

```
TYPE MC_MotionFunctionType :
(
   MOTIONFUNCTYPE_NOTDEF,
   MOTIONFUNCTYPE_POLYNOM1             := 1,   (* 1: polynom with order 1 (Synchron)*)
   MOTIONFUNCTYPE_POLYNOM3             := 3,   (* 3: polynom with order 3 (rest <-> rest) *)
   MOTIONFUNCTYPE_POLYNOM5             := 5,   (* 5: polynom with order 5 (rest <-> rest) *)
   MOTIONFUNCTYPE_POLYNOM7             := 7,   (* 7: polynom with order 7 (rest <-> rest) *)
   MOTIONFUNCTYPE_POLYNOM8             := 8,   (* 8: polynom with order 8 (rest <-> rest) *)
   MOTIONFUNCTYPE_POLYNOM9             := 9,   (* 9: polynom with order 9 (rest <-> rest) *)
   MOTIONFUNCTYPE_SINUSLINIE           := 10,
   MOTIONFUNCTYPE_MODSINUSLINIE        := 11,
   MOTIONFUNCTYPE_BESTEHORN            := 12,
   MOTIONFUNCTYPE_BESCHLTRAPEZ         := 13,  (* 13: Beschleunigungstrapez *)
   MOTIONFUNCTYPE_POLYNOM5_MM          := 15,  (* 15: polynom with order 5 (motion <-> motion) *)
   MOTIONFUNCTYPE_HARMONIC_KOMBI_RT    := 17,
   MOTIONFUNCTYPE_HARMONIC_KOMBI_TR    := 18,
   MOTIONFUNCTYPE_HARMONIC_KOMBI_VT    := 19,
   MOTIONFUNCTYPE_HARMONIC_KOMBI_TV    := 20,
   MOTIONFUNCTYPE_BESCHLTRAPEZ_RT      := 21,  (* 21: Beschleunigungstrapez (rest <-> turn) *)
   MOTIONFUNCTYPE_BESCHLTRAPEZ_TR      := 22,  (* 22: Beschleunigungstrapez (turn <-> rest) *)
   MOTIONFUNCTYPE_MODSINUSLINIE_VV     := 23,
   MOTIONFUNCTYPE_POLYNOM7_MM          := 24,  (* 24: polynom with order 7 (motion <-> motion) *)
   MOTIONFUNCTYPE_BESCHLTRAPEZ_RV      := 38,  (* 38: Beschleunigungstrapez (rest <-> velocity) *)
   MOTIONFUNCTYPE_BESCHLTRAPEZ_VR      := 39,  (* 39: Beschleunigungstrapez (velocity <-> rest) *)
   MOTIONFUNCTYPE_POLYNOM3_VV          := 40,  (* 40: polynom with order 3 (Velo <-> velo) *)
   MOTIONFUNCTYPE_POLYNOM11            := 41,  (* 41: polynom with order 11 (rest <-> rest) *)
   MOTIONFUNCTYPE_PEISEKAH11           := 45,  (* 45: Peisekah polynom with order 11 (rest <-> rest)
*)
   MOTIONFUNCTYPE_SPLINE               := 100, (* 100: inner spline interpolation point *)
   MOTIONFUNCTYPE_SPLINE_NATURAL       := 101, (* 101: first or last spline point *)
   MOTIONFUNCTYPE_SPLINE_TANGENTIAL    := 102, (* 102: first or last spline point *)
   MOTIONFUNCTYPE_SPLINE_PERIODIC      := 103, (* 103: first or last spline point *)
   MOTIONFUNCTYPE_SPLNE_LINEAR         := 120 (* 120: first or last spline point *)
);
END_TYPE
```

用于运动函数的类型定义。

类型注意

在 TwinCAT 凸轮设计编辑器中使用的运动函数类型 *Automatic* 对应于 *MOTIONFUNCTYPE_POLYNOM5_MM*，运动函数类型 *Synchronous* 对应于 *MOTIONFUNCTYPE_POLYNOM1*

8.10 MC_MotionPointType

```
TYPE MC_MotionPointType :
(
   MOTIONPOINTTYPE_IGNORE,                  (* Ignore point *)
   MOTIONPOINTTYPE_REST          := 16#0001, (* Restpoint - Rastpunkt *)
   MOTIONPOINTTYPE_VELOCITY      := 16#0002, (* Velocity Point - Geschwindigkeitspunkt *)
   MOTIONPOINTTYPE_TURN          := 16#0004, (* Turn Point - Umkehrpunkt *)
   MOTIONPOINTTYPE_MOTION        := 16#0008, (* Motion Point - Bewegungspunkt *)
   MOTIONPOINTTYPE_ADD           := 16#0F00, (* Addieren von Segmenten *)
   MOTIONPOINTTYPE_ACTIVATION    := 16#2000  (* 1: activation point *)
);
END_TYPE
```

用于数据表点的类型定义。

⚠ 危险

当心由于轴的意外运动而造成的生命危险或严重伤害或财产损失的风险

不允许在数据表定义的第一个和最后一个 MotionFunctionPoint 使用 MotionPOINTTYPE_IGNORE，切勿使用。

8.11 MC_TableCharacValues

```
TYPE MC_TableCharacValues :
STRUCT
   (* Master Velocity*)
   fMasterVeloNom    : LREAL; (* 1. master nominal velocity (normed: => 1.0) *)

   (* characteristic slave data *)
   (*==========================*)
   (* Start of cam table *)
   fMasterPosStart   : LREAL; (* 2. master start position *)
   fSlavePosStart    : LREAL; (* 3. slave start position *)
   fSlaveVeloStart   : LREAL; (* 4. slave start velocity *)
   fSlaveAccStart    : LREAL; (* 5. slave start acceleration *)
   fSlaveJerkStart   : LREAL; (* 6. slave start jerk *)

   (* End of cam table*)
   fMasterPosEnd     : LREAL; (* 7. master end position *)
   fSlavePosEnd      : LREAL; (* 8. slave end position *)
   fSlaveVeloEnd     : LREAL; (* 9. slave end velocity *)
   fSlaveAccEnd      : LREAL; (* 10. slave end acceleration *)
   fSlaveJerkEnd     : LREAL; (* 11. slave end jerk *)

   (* minimum slave position *)
   fMPosAtSPosMin    : LREAL; (* 12. master position AT slave minimum position *)
   fSlavePosMin      : LREAL; (* 13. slave minimum position *)

   (* minimum Slave velocity *)
   fMPosAtSVeloMin   : LREAL; (* 14. master position AT slave minimum velocity *)
   fSlaveVeloMin     : LREAL; (* 15. slave minimum velocity *)

   (* minimum slave acceleration *)
   fMPosAtSAccMin    : LREAL; (* 16. master position AT slave minimum acceleration *)
   fSlaveAccMin      : LREAL; (* 17. slave minimum acceleration *)
   fSVeloAtSAccMin   : LREAL; (* 18. slave velocity AT slave minimum acceleration *)

   (* minimum slave jerk and dynamic momentum *)
   fSlaveJerkMin     : LREAL; (* 19. slave minimum jerk *)
   fSlaveDynMomMin   : LREAL; (* 20. slave minimum dynamic momentum (NOT SUPPORTED YET !) *)

   (* maximum slave position *)
   fMPosAtSPosMax    : LREAL; (* 21. master position AT slave maximum position *)
   fSlavePosMax      : LREAL; (* 22. slave maximum position *)

   (* maximum Slave velocity *)
   fMPosAtSVeloMax   : LREAL; (* 23. master position AT slave maximum velocity *)
   fSlaveVeloMax     : LREAL; (* 24. slave maximum velocity *)

   (* maximum slave acceleration *)
   fMPosAtSAccMax    : LREAL; (* 25. master position AT slave maximum acceleration *)
   fSlaveAccMax      : LREAL; (* 26. slave maximum acceleration *)
   fSVeloAtSAccMax   : LREAL; (* 27. slave velocity AT slave maximum acceleration *)

   (* maximum Slave slave jerk and dynamic momentum *)
   fSlaveJerkMax     : LREAL; (* 28. slave maximum jerk *)
   fSlaveDynMomMax   : LREAL; (* 29. slave maximum dynamic momentum (NOT SUPPORTED YET !) *)

   (* mean and effective values *)
   fSlaveVeloMean    : LREAL; (* 30. slave mean absolute velocity (NOT SUPPORTED YET !) *)
   fSlaveAccEff      : LREAL; (* 31. slave effective acceleration (NOT SUPPORTED YET !) *)

   (* reserved space for future extension *)
   reserved          : ARRAY[32..47] OF LREAL;

   (* organization structure of the cam table *)
   CamTableID        : UDINT;
   NumberOfRows      : UDINT; (* number of cam table entries, e.g. number of points *)
   NumberOfColumns   : UDINT; (* number of table columns, typically 1 or 2 *)
   TableType         : UINT;  (* MC_TableType *)
   Periodic          : BOOL;

   reserved2         : ARRAY[1..121] OF BYTE;
END_STRUCT
END_TYPE
```

用于运动函数特征参数的类型定义。

8.12 MC_TableErrorCodes

```
TYPE MC_TableErrorCodes :
(
   (* Cam Table Error Codes *)
   MC_ERROR_POINTER_INVALID      := 16#4B30, (* invalid pointer (address) value *)
   MC_ERROR_ARRAYSIZE_INVALID    := 16#4B31, (* invalid size of data structure *)
   MC_ERROR_CAMTABLEID_INVALID   := 16#4B32, (* invalid cam table ID (not [1..255]) *)
   MC_ERROR_POINTID_INVALID      := 16#4B33, (* invalid point ID *)
   MC_ERROR_NUMPOINTS_INVALID    := 16#4B34,
   MC_ERROR_MCTABLETYPE_INVALID  := 16#4B35,
   MC_ERROR_NUMROWS_INVALID      := 16#4B36,
   MC_ERROR_NUMCOLUMNS_INVALID   := 16#4B37,
   MC_ERROR_INCREMENT_INVALID    := 16#4B38
);
END_TYPE
```

8.13 MC_TableType

```
TYPE MC_TableType :
(
   (* n*m tabular with equidistant ascending master values *)
   MC_TABLETYPE_EQUIDISTANT      := 10,

   (* n*m tabular with strictly monotone ascending master values (not imperative equidistant) *)
   MC_TABLETYPE_NONEQUIDISTANT   := 11,

   (* motion function calculated in runtime *)
   MC_TABLETYPE_MOTIONFUNCTION   := 22
);
END_TYPE
```

8.14 MC_ValueSelectType

```
TYPE MC_ValueSelectType :
(
   (* a bitmask can be created by adding the following values *)
   MC_VALUETYPE_POSITION      := 1,
   MC_VALUETYPE_VELOCITY      := 2,
   MC_VALUETYPE_ACCELERATION  := 4,
   MC_VALUETYPE_JERK          := 8
);
END_TYPE
```

用于通过功能块 MC_ReadMotionFunctionValues [▶ 35] 访问数值表的类型定义。

8.15 MC_StartMode

```
TYPE MC_StartMode :
(
   MC_STARTMODE_ABSOLUTE            := 1,  (* cam table is absolute for master and slave *)
   MC_STARTMODE_RELATIVE            := 2,  (* cam table is relative for master and slave *)
   MC_STARTMODE_MASTERABS_SLAVEREL  := 3, (* cam table is absolute for master and relative for slave
*)
   MC_STARTMODE_MASTERREL_SLAVEABS  := 4  (* cam table is relative for master and absolute for slave
*)
);
END_TYPE
```

StartMode 用于通过 MC_CamIn [▶ 12] 实现凸轮耦合，并定义凸轮是绝对插补（基于轴坐标系的原点）还是相对于耦合位置进行的插补。对于两个坐标轴，可以分别指定为绝对或相对模式。

使用 *StartModeAbsolute*，则凸轮坐标系与轴坐标系一致。如果需要，可以通过偏移进行移动（主轴偏移或从轴偏移）。

使用 *StartModeRelativ*，则凸轮坐标系的原点在耦合或凸轮切换时各轴（主轴或从轴）的当前位置。凸轮盘还可以通过偏移来移动。

注意： MC_STARTMODE_RELATIVE 和 MC_STARTMODE_MASTERREL_SLAVEABS 模式不能与自动计算主轴偏移（MC_CamScalingMode [▶ 51]）一起使用，因为这将导致冲突。

8.16 ST_CamInOptions

ST_CamInOptions 类型的数据是功能块 MC_CamIn [▶ 12]的可选输入选项。

```
TYPE ST_CamInOptions :
STRUCT
   (* ActivationMode defines when and where the cam table will be activated *)
   (* (only valid if slave is already coupled and cam table will be exchanged) *)
   ActivationMode          : MC_CamActivationMode := MC_CAMACTIVATION_INSTANTANEOUS;
   ActivationPosition      : LREAL;

   (* Scaling Modes enable, disable or define the way of scaling the cam table *)
   MasterScalingMode       : MC_CamScalingMode := MC_CAMSCALING_USERDEFINED;
   SlaveScalingMode        : MC_CamScalingMode := MC_CAMSCALING_USERDEFINED;

   (* InterpolationType is required for position tables only. *)
   (* MotionFunctions don't need an InterpolationType *)
   InterpolationType       : MC_InterpolationType := MC_InterpolationType_Linear;
END_STRUCT
END_TYPE
```

8.17 CamMasterData

CamMasterData 类型的数据是功能块 MC_ReadCamTableMasterPosition [▶ 44] 的可选输入选项。

```
TYPE CamMasterData :
STRUCT
   Valid               : BOOL;  (* position information is valid *)
   MasterAxisPosition  : LREAL; (* absolute master axis position *)
   MasterCamPosition   : LREAL; (* local master cam position *)
   SlaveOffset         : LREAL; (* slave cam offset corresponding to the master position *)
END_STRUCT
END_TYPE
```

8.18 MC_CamOperationMode

CamOperationMode 用于管理通过功能块 MC_CamIn_V2 [▶ 19]（多凸轮）耦合的叠加凸轮。凸轮可以添加、切换或移除。

```
TYPE MC_CamOperationMode :
(
   CAMOPERATIONMODE_DEFAULT,  (* same as additive *)
   CAMOPERATIONMODE_ADDITIVE, (* additive cam in a multi cam scenario *)
   CAMOPERATIONMODE_EXCHANGE, (* exchange existing cam in a multi cam scenario *)
   CAMOPERATIONMODE_REMOVE    (* remove cam from a multi cam scenario *)
);
END_TYPE
```

8.19 ST_CamScalingData

ST_CamScalingData 结构包含凸轮的缩放信息，与功能块 MC_CamIn_V2 [▶ 19] 一起使用。

```
TYPE ST_CamScalingData :
STRUCT
   (* scaling of the X axis of the cam (master scaling) *)
   MasterScalingMode  : MC_CamScalingMode;
   MasterRelative     : BOOL;
   MasterOffset       : LREAL;
   MasterScaling      : LREAL := 1.0;

   (* scaling of the Y axis of the cam (slave scaling) *)
   SlaveScalingMode   : MC_CamScalingMode;
   SlaveRelative      : BOOL;
   SlaveOffset        : LREAL;
   SlaveScaling       : LREAL := 1.0;
END_STRUCT
END_TYPE
```

名称	类型	描述
MasterScalingMode	MC_CamScalingMode	用于凸轮主轴位置的缩放模式 [▶ 51]
MasterRelative	BOOL	如果为 “TRUE”，则凸轮在激活时相对于当前的主轴位置运行。
MasterOffset	LREAL	轴坐标系中用于凸轮定位的主轴偏移。*MasterOffset* 在绝对模式下从主轴位置 0 开始生效，在相对模式下从激活时的当前位置开始生效。
MasterScaling	LREAL	凸轮主轴位置的缩放。默认值为 1.0
SlaveScalingMode	MC_CamScalingMode	用于凸轮从轴位置的缩放模式 [▶ 51]
SlaveRelative	BOOL	如果为 “TRUE”，则凸轮盘在激活时相对于当前的从轴位置运行。
SlaveOffset	LREAL	轴坐标系中用于凸轮定位的从轴偏移。*SlaveOffset* 在绝对模式下从从轴位置 0 开始生效，在相对模式下从激活时的当前位置开始生效。
SlaveScaling	LREAL	凸轮盘从轴位置的缩放。默认值为 1.0

8.20 ST_ReadCamTableSlaveDynamicsOptions

可传输至功能块 MC_ReadCamTableSlaveDynamics [▶ 37] 的可选数据。

```
TYPE ST_ReadCamTableSlaveDynamicsOptions :
STRUCT
    SynchronousAccess          : BOOL;
END_STRUCT
END_TYPE
```

8.21 ST_SetOnlineChangeModeOptions

可传输至功能块 MC_SetCamOnlineChangeMode [▶ 33] 的可选数据。

```
TYPE ST_SetOnlineChangeModeOptions :
STRUCT
    SynchronousAccess          : BOOL;
END_STRUCT
END_TYPE
```

8.22 ST_WriteMotionFunctionOptions

可传输至功能块 MC_WriteMotionFunction [▶ 31] 或 MC_WriteMotionFunctionPoint [▶ 32] 的可选数据。

```
TYPE ST_WriteMotionFunctionOptions :
STRUCT
    SynchronousAccess          : BOOL;
END_STRUCT
END_TYPE
```

9 示例程序

电子凸轮表

该示例程序通过凸轮耦合一个主轴和一个从轴。在耦合运动期间，凸轮表被切换，凸轮表的个别采样点被修改，凸轮表被缩放。

该示例程序需要 Tc2_MC2_Camming 库，并可在仿真模式下完全运行。可以在例程中包含的 TwinCAT Measurement项目中监视运动过程。

点击此处，保存该示例程序：

https://infosys.beckhoff.com/content/1033/TF5050_TC3_NC_Camming/Resources/2570265355.zip

旋切和色标

本例程使用旋切来切割规定长度的板材。为了实现这个功能，在切割板材时，圆周速度必须与卷材同步。旋转刀具必须加速或减速，因为刀具的周长与板材长度不同。色标用来与产品同步。刀具被不断调整，以补偿由温度或拉伸引起的微小差异。

用一个凸轮表来使旋转刀具与板材同步。这是一个标准化的凸轮表，定义长度为 360° 。凸轮表执行一个循环，刀具移动一圈。指定切割位置为 0° 。从 270° 到 30° 是需要圆周速度与板材同步的。从 30° 到 270° 的区间，用于根据色标的实际距离调整刀具的运动。

https://infosys.beckhoff.com/content/1033/TF5050_TC3_NC_Camming/Resources/2570267019.zip

更多信息:
www.beckhoff.com/tf5050

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