ASiM 多轴运动功能
TM441

贝加莱工业自动化
Perfection in Automation
www.br - automation.com

简介

前提

培训模块：	TM440 – ASiM 基本功能
软件：	Automation Studio 2.5 或更高版本 Automation Runtime 2.80或更高版本 ACP10_MC函数库1.170或更高版本
硬件：	无

目录

1、简介

B&R驱动方案(ACOPOS)提供灵活的,高性能的工具来电子连接驱动，可以创建线形或动态(非线形)连接同步电机运动。实际上,有许多应用需要做这个功能比如同步切割程序,动态传输过程和灵活的长度分割。ACP10_MC 库中以通常的形式提供相应的功能块来全面操作这些功能。

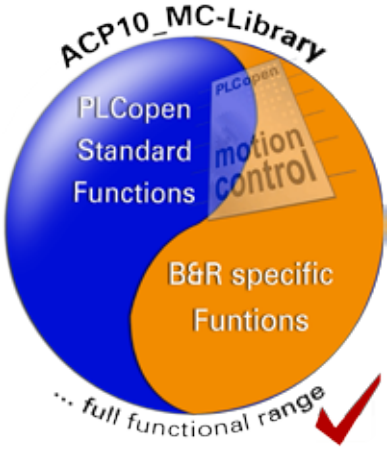


图. 1 ACP10_MC 库

本文档涉及如何使用不同的函数来设置和控制电子连接运动进程。

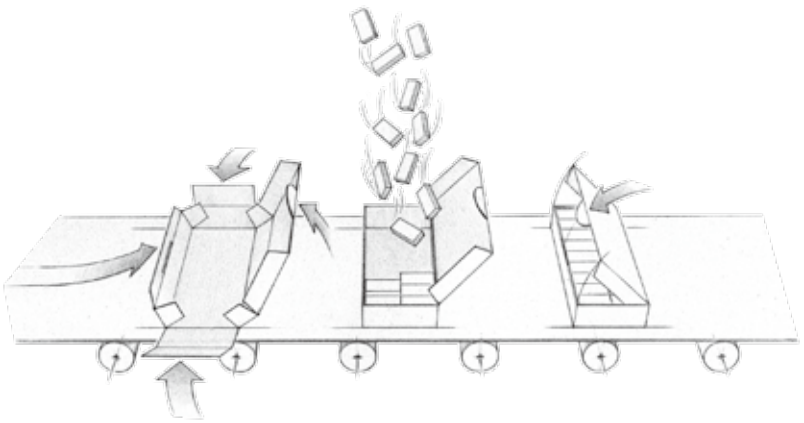


图. 2 纸盒成型

首先我们看下简单的综述来熟悉每个选项，在理论基础和概念的帮助下,我们将会学习如何使用多轴功能。

简介

1.1 目的

熟悉使用运动控制多轴函数(ACP10_MC)。

使用已选的函数来连接驱动，并在驱动连接时执行特定的进程。

学习创建动态凸轮的步骤以及能够使用这些知识来电子连接驱动。

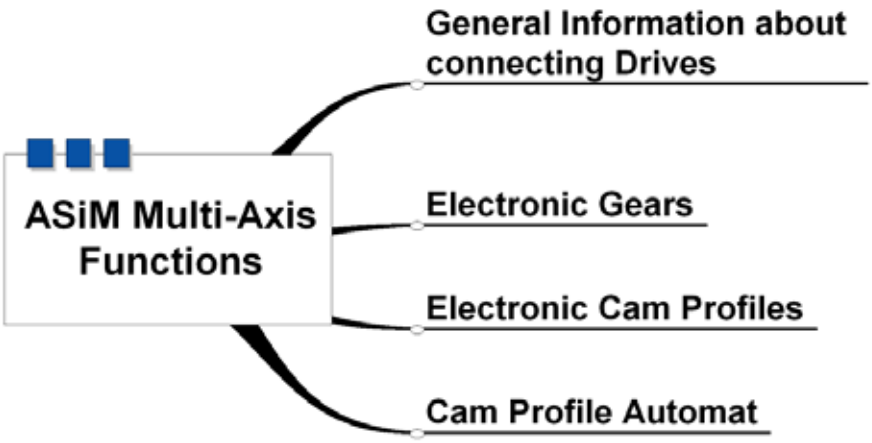


图. 3 综述

2、连接驱动的一般信息

电子连接驱动意味着什么?

电子连接驱动也就是预先定义同步运动。

例子:

驱动A通过位置和驱动B连接，这就意味着当驱动连接时,驱动A必须按指定的方式调节它的位置来对应驱动B的位置。在这种情况下,驱动B是主轴(指定参考位置)驱动A是从轴(位置基于主轴位置)。

图. 4 预定义驱动连接

驱动连接需要一个主轴信号来提供参考(位置,目标)，以及至少一个从轴来按照指定的"规则"跟随此参考值。在做这步骤时,主轴信号并不一定象先前例子中讨论到的从一个实际的驱动得到。原则上,驱动可以连接到不同的合适的参考值(外部编码器,时间等).

> 备注:
>
> 一般来说,主轴并不受连接程序的影响。它仅仅用做期望的连接信号的基础。比如说,驱动的位置值作为主轴信号,当驱动连接激活时，我们仍然可以给主轴发命令。在这种情况下,从轴就完全独立于主轴信号了。

这种形式的位置连接(i.e. 从轴必须跟随主轴信号的"位置规则")可以清晰的在图表中显示来比较主轴和从轴的位置。

下图显示主轴和从轴的线形关系:

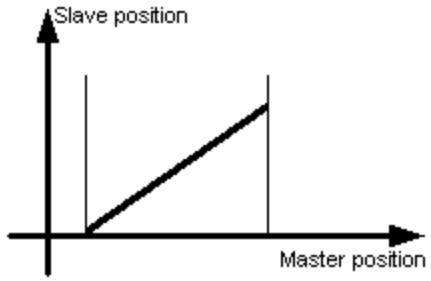


图. 5 线形连接

水平轴显示连接的主轴位置，垂直轴显示连接的从轴位置。

当主轴信号有规律的改变(e.g. 主轴匀速运动), 从轴的速度按照指定也是匀速的(固定的位置改变)。

在这种情况下,我们可以讨论下"电子齿轮",一种经常用到的连接方式。"线形曲线"的斜率决定齿轮比率:

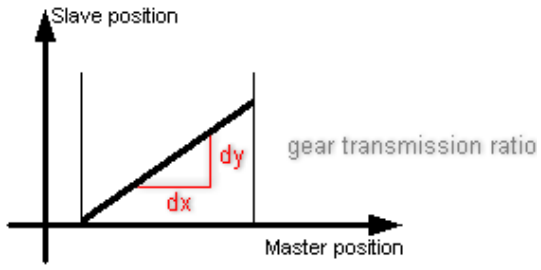


图. 6 齿轮比率

然而,位置关系并不一定是线形的。原则上,可以创建电子凸轮用于任意需要的位置轮廓。

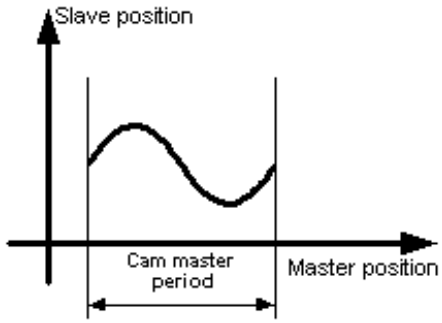


图. 7 动态位置轮廓

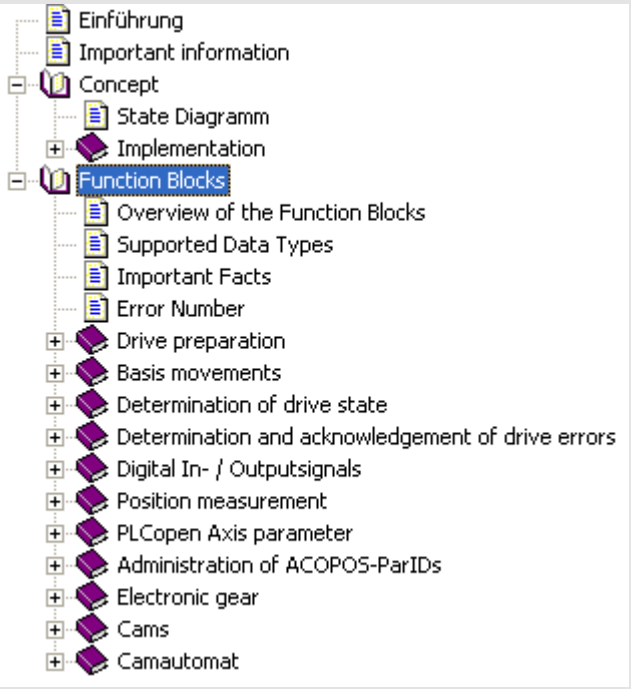
可以很快的在ACOPOS 上执行简单的电子齿轮连接以及通过电子凸轮连接。Automation Studio 中提供凸轮编译器来创建用户指定的凸轮。可以在ACP10_MC库中找到用于设置和控制驱动连接相应的功能块。

Cam profile automat提供大量用于连接多个凸轮的设定。

备注:

ACP10_MC 中多轴函数的操作方法和我们已经熟悉的功能块一样的，这就意味着这些函数同样可以结合到统一应用程序的自动进程中。

可以在Automation Studio 在线帮助中找到对于每个独立功能块的详细信息。



3、电子齿轮

我们已经知道, 电子齿轮用来建立主轴和从轴的特定比率("齿轮比")的线形关系。

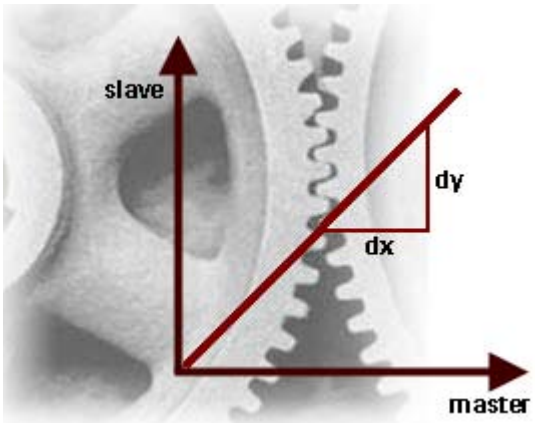


图. 8 电子齿轮

该功能经常用于简单的传送带，想象下产品必须从一个传送带传到另一个。为了传输正常工作,传送带的速度必须同步。

3.1 简单连接

ACP10_MC 库中用来控制电子齿轮的函数使用起来十分方便，所以并不需要更多的解释,让我们直接开始第一步测试。

备注:

连接轴对象的函数同样需要主轴和从轴的轴ID。这样,必须使用"ncaccess"函数来决定轴对象的ID。象以往一样,可以访问在NC 配置表中定义的NC 对象(实轴或虚轴)。

NC Object Name	Network Interface	Node Number	Advanced	NC Object Type
Axis1	SS1.IF2 (Powerlink)	1		ncAXIS
Axis2	SS1.IF2 (Powerlink)	2		ncAXIS
Axis3	SS1.IF2 (Powerlink)	1		ncV_AXIS

当然,您也需要经过已知的步骤使轴准备好运动。

任务: "电子齿轮"

使用电子齿轮连接两个驱动。测试"MC_GearIn"和"MC_GearOut"函数来启动和结束线形连接。

准备工作:

项目中必须要有ACP10_MC库。

基于可行性,可以连接实轴或虚轴。创建独立的任务来控制主轴和从轴是一个很有效的方式。

- 控制主轴任务:

如先前探讨过的一样,主轴一般不受连接影响。它仅仅提供主轴信号(默认为设定位置)。这样,只需要简单的定位例程来控制主轴。例子项目中有"basic"任务。该任务包括所有准备驱动的例程,以及基本定位命令。

为了加快进展,您可以在项目中执行该任务来控制主轴。

- 控制从轴任务:

除了准备驱动的函数外(任何定位函数等)还需要连接的功能块。

必须在初始化子程序中得到连接轴的IDs - e.g.:

```
(* Init Subprogram *)

(* Determine master ID *)
status_ma:= ncaccess(ncACP10MAN,ADR(’Axis1’),ADR(Axis1Obj));

(* Determine slave ID *)
status_sl:= ncaccess(ncACP10MAN,ADR(’Axis2’),ADR(Axis2Obj));
...
```

电子齿轮

本测试所有需要的函数都可以放在循环任务中，轴ID 可以使用通常的步骤在任务中安全的指定 – e.g.:

(* Cyclic program section *)

(* Function Block Calls *)

(* Preparing Drive *)

MC_Power_0.Axis:= Axis2Obj;

MC_Power_0();

..

...

(* Gearing Functions *)

MC_GearIn_0.Master:= Axis1Obj;

MC_GearIn_0.Slave:= Axis2Obj;

MC_GearIn_0();

MC_GearOut.Slave:= Axis2Obj;

MC_GearOut();

The function blocks for status and error monitoring as well as for error acknowledgement must also be added.

执行控制轴对象的预先设置，并下载到项目中。

测试函数:

使用watch 操作任务。

轴必须先通过已知的步骤准备好运动:

- 打开控制器
- 寻找参考点

"MC_ReadAxisError"功能块可以用来确认任何错误，当轴正确设置后应该没有错误。

现在可以测试连接函数。我们可以使用主轴执行运动(e.g. "basic"任务中的持续运动或点动例程)。

设定"MC_GearIn"的输入参数并激活功能块("Execute").

测试"MC_GearIn"和"MC_GearOut"功能块的不同设置。

注意观察驱动状态的变化。

备注:

主轴状态始终不变,不受连接时候激活的影响("离散的运动", "连续的运动", 等).

当连接启动成功后从轴改变到"同步状态"。当使用"MC_GearOut"函数中断连接后驱动仍保持当前速度并变成"连续运动"状态。于是,还要使用"MC_Stop"功能块来停止从轴的运动。(参考状态图表)

用做连接信号(主轴信号)的驱动连接参数(i.e. 主轴参数值) ,可以通过"MasterParID"函数的输入参数定义成适合这种情况或其他的连接功能。默认使用主轴的设定速度。根据应用,可以选用其他合适的值连接(e.g. 实际编码器位置, etc.)。

其他使用的功能块嵌入在ACP10_MC 例子项目"gear"任务的完成功能进程中。该任务可以添加到项目中(使用已知的方法)来控制轴(实轴或虚轴)。该结构("gAxisSlave")中还有额外的变量来操作连接功能。

3.2 基于参考位置的驱动连接

"MC_GearInPos"功能块是"MC_GearIn"函数的扩展，一些应用需要预先定义启动驱动连接的位置。

Fig. 9 接替

"MC_GearInPos"用来定义主轴和从轴启动电子齿轮的位置。这样我们就可以为启动驱动连接定义每个轴的"位置"。

函数可以按以下方式使用:

一个带有产品感受器的传送带将从静止加速到一定的速度,另一个的传送带提供产品。 这样,产品并不能简单的传输到带上的任意位置。"MC_GearInPos"函数保证两个传输带接口点的位置和速度一致。

需要做什么设置?

除了"MC_GearIn"功能块的设定外, 该函数还要定义主轴和从轴启动线形部分(线形驱动连接)的位置。此外,还给主轴指定一段距离，在该段距离里主轴平滑的进入线形部分("补偿从轴运动")。一旦给定"GearInPos"命令和相应的参数,到达预先定义的启动点后系统开始进入齿轮耦合运动。

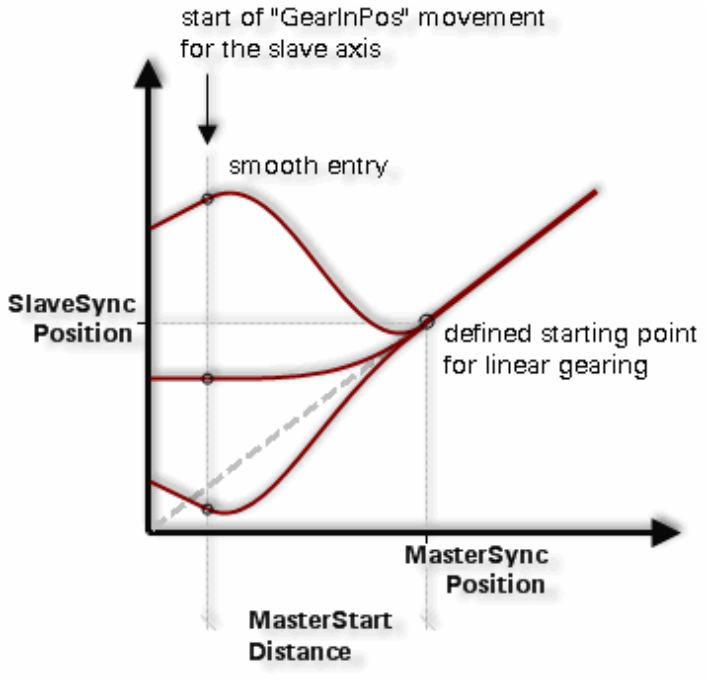


图. 10 按预定义的启动位置连接齿轮

上图显示不同启动位置的步骤(从轴位置任意)。当主轴到达指定位置时从轴驱动进入齿轮耦合运动。这由实际同步位置(启动线形部分)和特定补偿距离(主轴侧)产生。

在这种情况,运动平滑的进入相应的齿轮比。

> 备注:
>
> 主轴不受该动作影响，可以象平常一样执行基本运动。

可以使用一个模式参数来定义位置周期,在该周期内从轴驱动移动，进入齿轮耦合。驱动连接的实际点是由依赖于在位置周期内(当前,前一个或下一个周期)当前从轴位置的这个模式决定的。这就允许从轴在当前周期前一个周期或后一个周期内针对连接点做补偿运动:

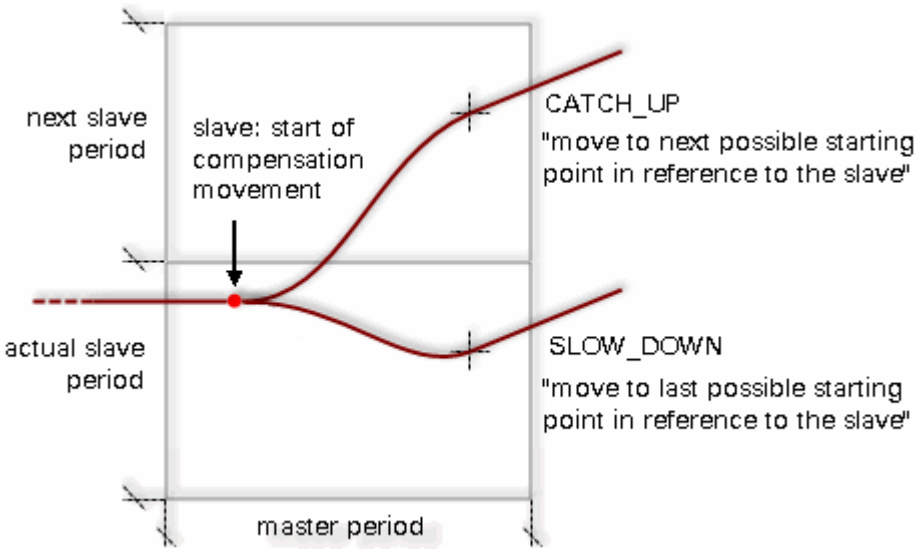


图. 11 CATCH_UP 和 SLOW_DOWN 图表

如上图所示, "CATCH_UP"总是启动运动到下一个驱动连接点。"SLOW_DOWN"总是启动运动到先前驱动连接点。根据从轴的当前位置,"CATCH_UP"模式改变到下一个周期(参考先前)。同样"SLOW_DOWN"模式改变到先前的周期。

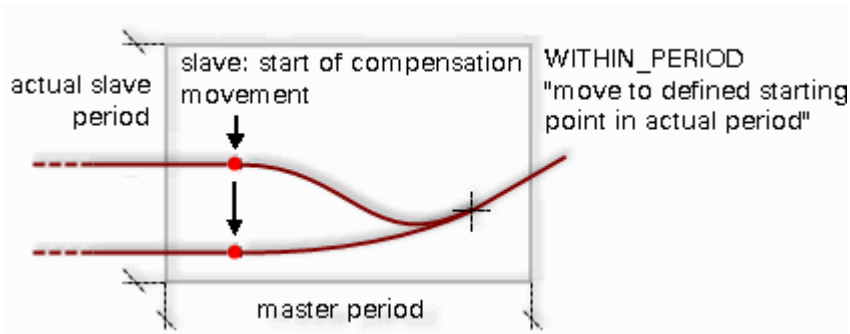


图. 12 WITHIN_PERIOD 图表

上图显示不同启动位置在"WITHIN_PERIOD"模式的运行方式。在这些情况中,从轴始终在当前周期运动到启动点。

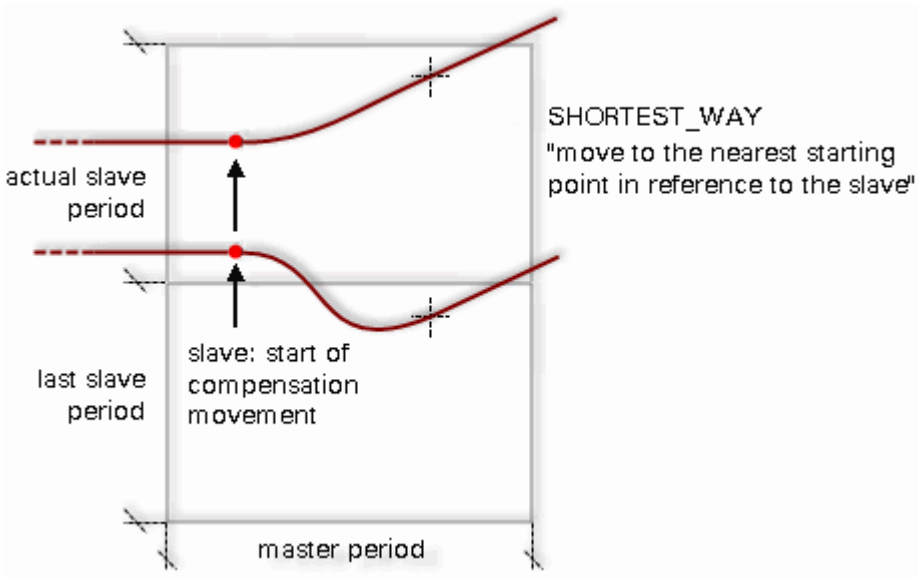


图. 13 SHORTEST_WAY 图

在"SHORTEST_WAY"模式中,从轴总是移动到下一个最接近的驱动连接点。在上图中显示了两个不同的启动位置在该模式下如何运行。基于位置,可以改变到先前周期(如先前所述)或下一个周期。

当超过补偿主轴启动位置时如何处理?

当使用循环的轴,该起始点总是返回到下一个周期。否则,功能块会发出相应的回应("Error")。

任务: "基于参考位置的电子齿轮"

现在可以用先前例子一样的方法来测试"MC_GearInPos"功能块。

准备测试并使用watch 窗口来操作函数。当从轴闲置时启动函数，观察从轴如何动作。

3.3 动态相移

"MC_Phasing"函数用来在驱动连接激活时"虚拟"改变主轴位置。

是如何做的呢?

从轴的位置是基于连接的主轴"位置"以及驱动的连接关系(线形或凸轮):

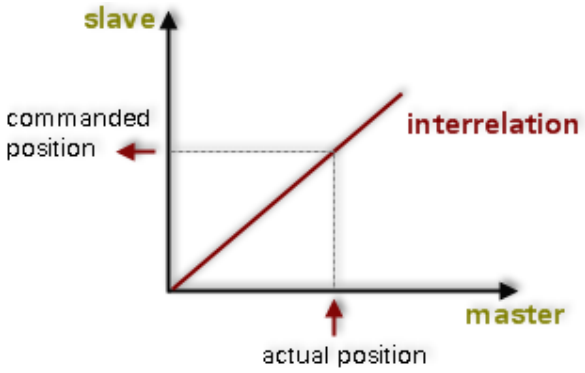


图. 14 主/从 驱动连接关系

"MC_Phasing"功能块于是生成一个附加的元素或附加的主轴值，该元素附加到实际的主轴位置，最终值应用到驱动连接关系的主轴侧:

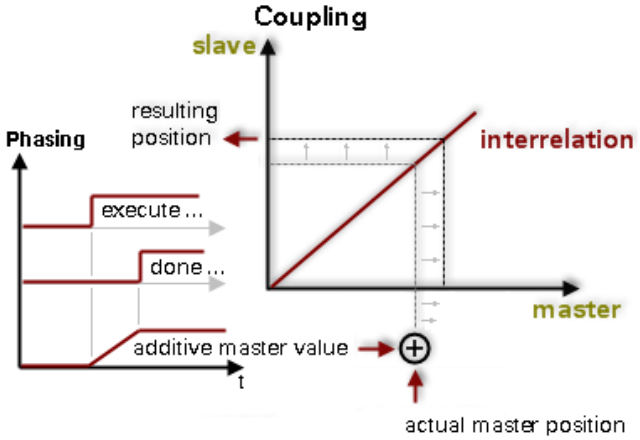


图. 15 主轴 –附加元素, –驱动连接关系, -从轴

这将改变连接从轴的设定位置，这意味着可以执行一个特定的移相，生成的附加主轴值是一个全面的值(期望的目标值)。功能块激活后相应的增加速度创建附加轴的最终值，这样可以在进程中避免(从轴)任何位置跳变的发生，主轴不受该动作影响。

"MC_Phasing"可以设置进行产品分类。切割盒子后,薄片一个接一个在传送带上传输。接着传输到另一个传送带。当一个薄片到达第二个传送带时执行相位转换。可以产生产品间的间隙,可供更进一步的处理。

备注:

从轴的位置基于驱动连接的关系。比如,电子齿轮的齿轮比对从轴位置有以下影响:

齿轮比 = 1:5 (主:从)

主轴侧移相 = 2000 units (附加主轴)

→ 从轴有效移相= 10000 units

任务: "移相"

使用已知的方法测试"MC_Phasing"函数，运行直到激活连接点的所有步骤并执行不同设置下的移相。

备注:

移相附加到当前的运动，当连接的轴闲置时(主轴闲置)也可以测试"MC_Phasing"功能块。

4、电子凸轮

为了实现动态,非线形的运动,ACOPOS提供电子凸轮的选项来连接轴。这些凸轮可以由用户创建。

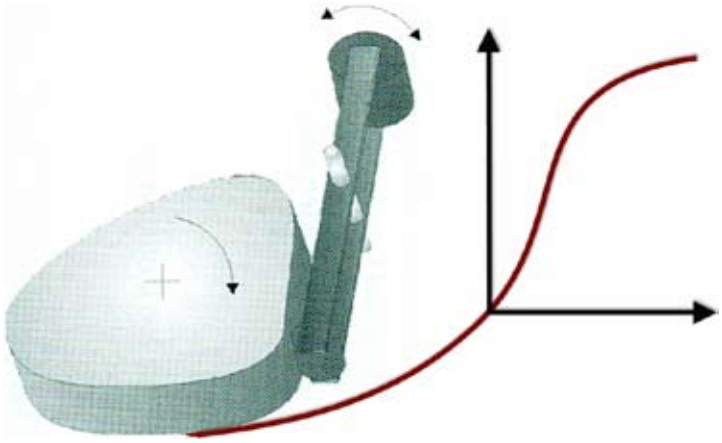


图. 16 电子凸轮

ACP10_MC库中提供适当的功能块来控制必要的任务。

电子凸轮可以有多种使用方式。比如,电子凸轮用与弹簧机械效率相当高。 不同的轴用来控制各自的送料,弯曲和倾斜。这样可以制造任意需要的外型(斜面,圆锥体, etc.)

4.1 介绍

就象我们先前章节看到的,驱动连接的位置关系可以清晰的图表中显示。

在凸轮图中, 我们可以看到水平轴显示主轴位置值，垂直轴显示从轴位置值。凸轮在特定的区域内(凸轮控制周期)针对每个主轴位置有一个独立的从轴位置值。当驱动连接激活时，从轴必须遵循此轮廓。

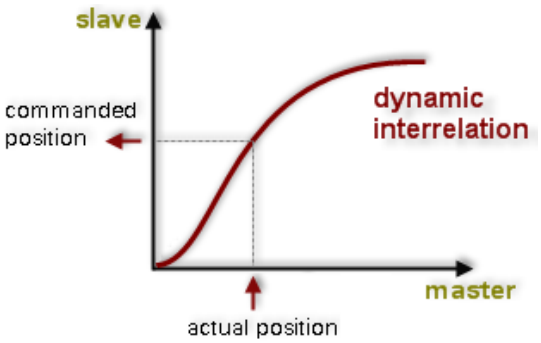


图. 17 遵照位置关系的凸轮

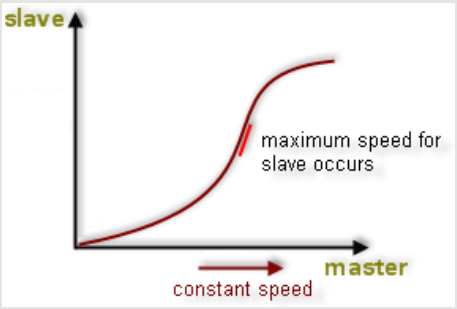
通过凸轮主轴位置转变成相应的从轴位置。主轴正转反转都可以。从轴驱动通过凸轮"连接"到主轴。

这意味着从轴驱动的速度和加速度也是通过曲线特性连接的主轴的速度和加速度得到的。

因此,必须检测整个凸轮的过程来确认从轴驱动能够承受要达到的速度和加速度值。

例子:

让我们假设主轴信号按恒定比率改变(e.g. 不变的主轴运动,主轴时间, etc.).



临界区域(从轴速度和加速度的最大值)在凸轮中分别以最大斜率 (-> 速度即位置的一阶导数)以及决定最大斜率变化(->加速度/减速度即位置的二阶导数)位置对照来显示。

备注:

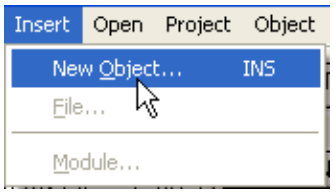
在从轴极限值超过的情况下,会生成此NC对象相应的错误信息。

4.2 创建凸轮

Automation Studio提供一个强有力的凸轮编译器来创建凸轮。在项目中插入凸轮后可以在凸轮编译器中编辑它。

4.2.1 添加凸轮

凸轮在Automation Studio 中作为NC软件对象创建，首先在我们创建一个新的凸轮前必须添加相应的对象到项目中。



在Insert菜单中选择Insert object... 打开对话框并选择Advanced Object。

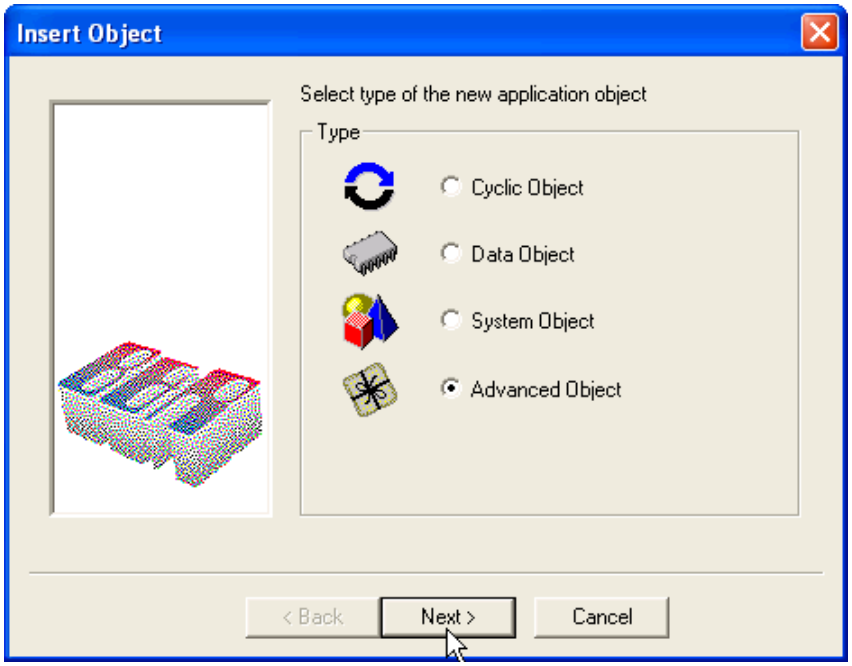


图. 18 添加advanced object

在点击Next>按钮确认选择后,在下一个窗体中可以选择不同的NC数据对象。比如,我们选择类型: NC Cam Profile 来源: ACP10: Cam Profile。

我们可以在Name 栏输入NC 软件对象的名称:

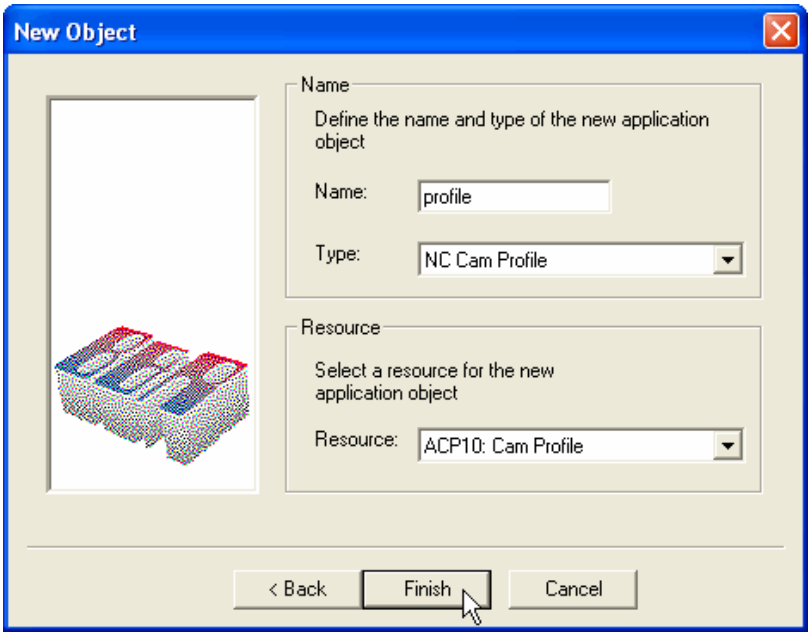


图. 19 添加凸轮

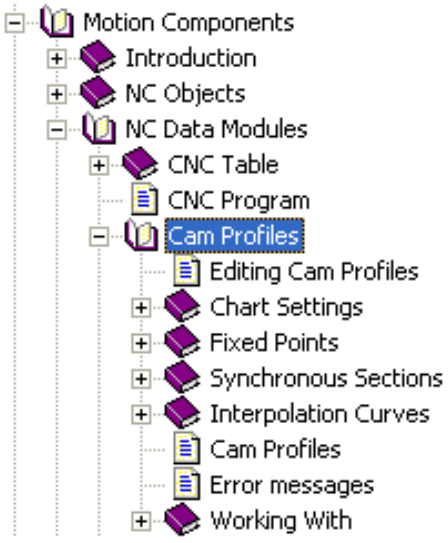
点击Finish 按钮确认输入后我们就在项目中创建了一个新的NC软件对象。

完成该动作后凸轮编译器自动打开，我们可以立即模拟凸轮。以后,可以通过双击软件树上对应的图标打开并编辑。

4.2.2 编辑一个电子凸轮

Automation Studio 中的凸轮编译器是一个全方面的工具可以帮助我们创建和改编针对不同连接需求相当清晰准确的凸轮。这里提供大量设置来完成此步骤。

Automation Studio帮助系统包括插入NC软件对象"cam profile"(如先前章节中提到的一样)的用户详细信息,创建凸轮的信息和不同凸轮格式的信息。



以下章节将介绍编辑凸轮的步骤。

创建了一个新的凸轮对象后在编译器中有一个"空"的区域。编辑窗体分成三部分:

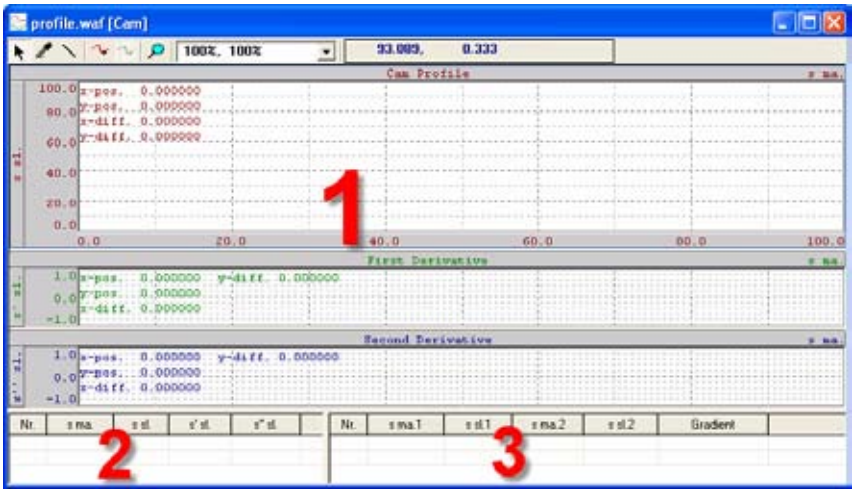
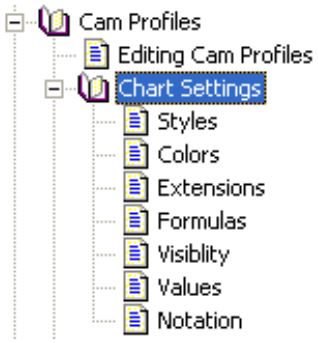


图. 20 凸轮编译器

- 显示作为主轴位置和从轴位置函数的凸轮(1, 上部) 。从曲线特性得到的大量图表(加速度, 抖动, etc.)在窗体的下半部分显示。
- (2, 左下部)定义曲线特性的固定点的列表
- (3, 右下部)定义同步部分列表

建议在插入固定点和同步部分前调节图表属性，可以在Automation Studio 帮助文件中找到属性和设定的详细描述。



设置包括:

- 一般属性
- 颜色设定
- 范围
- 显示选项
- 标签和公式
- 曲线的特性值
- 图表的符号

现在我们可以在曲线上定义固定点和同步部分(曲线特性的线形部分)来创建凸轮。通过插补曲线凸轮编译器可以自动的生成完整凸轮:

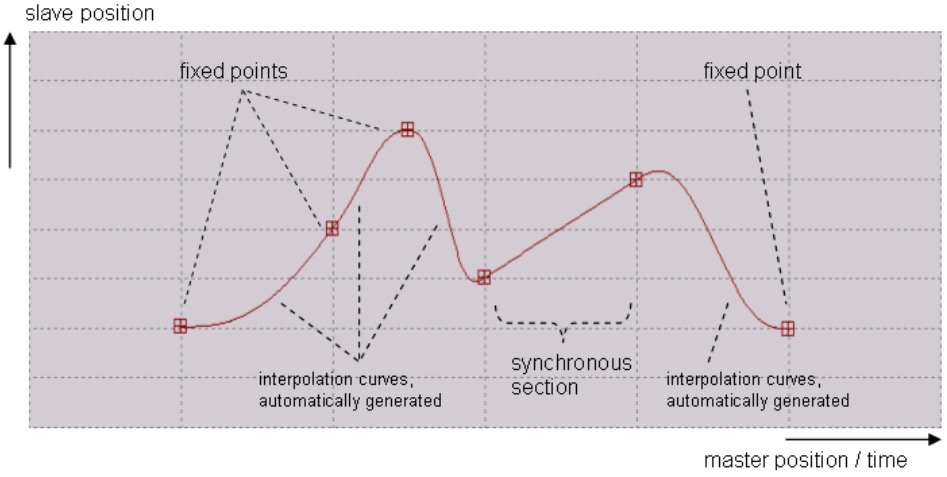


图. 21 凸轮的结构

上图显示了一个凸轮的例子。一共定义了四个固定点和一段同步部分(线形)。凸轮编译器自动连接这些定义来创建完整的凸轮:此时,同样计算和显示插补曲线。后面我们将会看到用户同样可以定义插补曲线的形状。

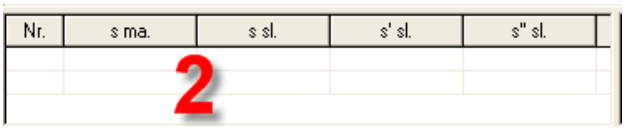
■ 固定点

固定点是凸轮中的一点,在这点上必须预先定义基于主轴位置的从轴位置。这样,通过与指定的主轴位置比较确定每个从轴位置。

有三种方式可以添加指定点:

- 在固定点表中,在工作区的左下方(2)
- 使用输入对话框, 通过"Insert" / "Fixpoint"打开
- 在图表中使用鼠标

让我们使用第一种方法来定义固定点。工作区左下放的区域包括存在固定点的列表。



以下值使用在表不同栏中:

栏标签	意义 (数学符号1)
Nr.	表中固定点的连续号
s ma.	主轴固定点的位置
s sl.	从轴固定点的位置
s' sl.	固定点上凸轮函数的一阶导数(→ 当前齿轮比率)
s" sl.	固定点上凸轮函数的二阶导数

备注:

符号显示图表中横坐标是位置还是时间单位(e.g. 针对主轴)。使用位置视为"数学符号"。使用时间视为"物理符号"。

这样,物理符号固定点的一阶导数为从轴的速度，二阶导数为从轴的加速度。凸轮显示从轴的轨迹-时间图表。

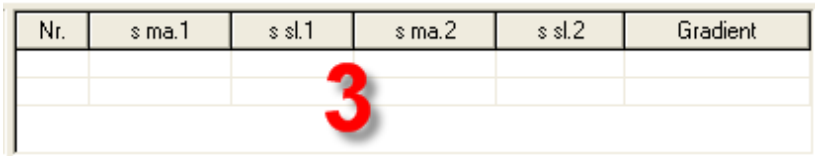
■ 同步部分

同步部分作为凸轮的一部分,在该阶段必须预先定义基于主轴位置的从轴位置。就象我们先前看到的一样,当主轴以匀速运动时，此曲线特性也使从轴匀速运动。也就是说,凸轮包括一段直线(和电子齿轮相似)。

同样有三种方式来添加同步部分:

- 在同步部分表中添加,该表位于工作区的右下方(3)
- 使用输入对话框,可以通过"Insert" / "Synchronous Section"打开
- 在图中使用鼠标

让我们用第一种方式来使用同步部分表格。工作区右下方的表格包括存在的同步部分。



以下值使用在表不同栏中:

栏标签	意义 (数学符号1)
Nr.	表中同步部分的连续号
s ma. 1	主轴上同步部分启动点的位置
s sl. 1	从轴上同步部分启动点的位置
s ma. 2.	主轴上同步部分结束点的位置
s sl. 2	从轴上同步部分结束点的位置
Gradient	同步部分的斜率或齿轮比率(由先前定义得出)

备注:

当使用物理符号时(主轴 = 时间)主轴位置完全根据时间点改变。在此区间内同步部分的斜率相当于从轴的速度。(→ 时间"均匀"的流逝)

■ 插补曲线

位于两个定义段(固定点,同步部分)之间由凸轮编译器计算出来的凸轮函数部分称为插补曲线。

在输入每个新的固定点或同步部分后,以及在定义段间创建，计算和显示插补曲线。凸轮编译器确保插补曲线准确的位于两个定义部分之间。

同样,如果一个固定点或同步部分删除后插补曲线同样删除。

> 备注:
>
> 计算保证凸轮函数以及它在转换点的一阶导数为常数(e.g. 曲线在结束点没有任何跳变)。

每个插补曲线可以有大量的曲线类型提供选择来更精确的设计定义区域(固定点和同步部分)间的曲线特性。针对不同的类型可以有不同的预定义形状。特定的曲线特性可以通过类型-指定设置(转向点, 连接点, etc.)来实现。

相应的对话框可以通过右键点击曲线打开。选择Curve Properties 后可以编辑曲线部分。

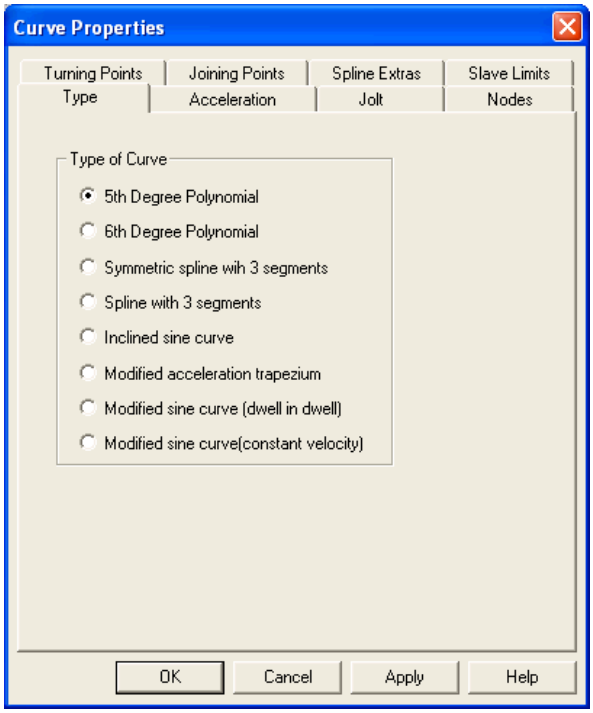


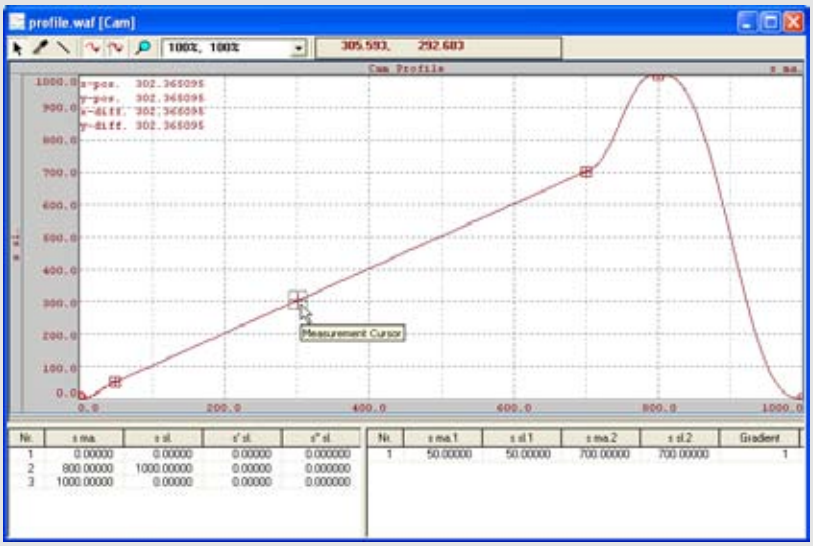
图. 22 曲线属性对话框

关于这里可以做的设置的详细信息可以在Automation Studio 在线帮助 Interpolation Curves 栏下找到。

电子凸轮

任务: "创建一个凸轮"

添加NC 软件对象"cam profile"到您的项目中并在凸轮编译器中进行编辑。使用选项来定义固定点和同步部分。
比如:下图显示一个开关凸轮应用的位置轮廓。从轴依赖于主轴位置(在凸轮右边四分之一处)到达最大位置。



当创建凸轮时,确保轮廓的启始点和结束点有相同的斜率。该特性对于接下来的驱动连接应用十分重要。

补充:
把新的凸轮传输到项目中并激活监控模式。现在凸轮在项目软件树中以数据模块的形式出现在控制器上。针对凸轮的耦合应用程序通过使用"MC_CamTable Select"函数传输到ACOPOS 伺服驱动中。

4.3 连接函数

先前成功创建并传输到控制器的凸轮现在可以用做轴的连接。要做这一步,必须使用一个简单的函数把期望的凸轮对象传输到从轴对象，这样就可以使用该凸轮连接轴对象。

以下部分将阐述需要做的相应进程和步骤。

4.3.1 凸轮准备

需要使用"MC_CamTableSelect"功能块来把凸轮对象传输到连接的从轴。当调用了该函数,相应的凸轮(输入参数 "CamTable")传输到从轴并返回一个ID号以备后续的连接函数使用。

凸轮准备时,功能块可以设置单次或循环处理凸轮。这样,可以连接多个凸轮:

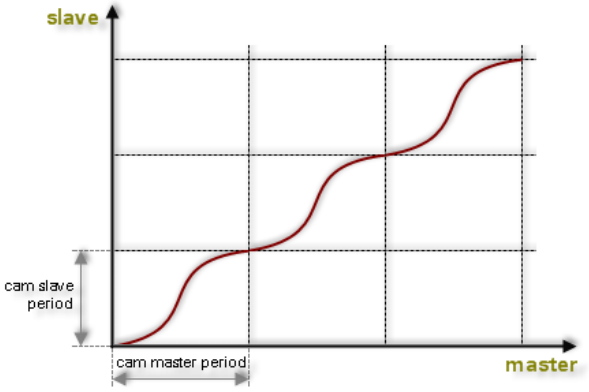


图. 23 循环连接凸轮

这就实现了从轴完整的定位进程。

> 警告:
> 当循环连接凸轮时,后面的凸轮的启始点和先前凸轮的结束点需要无缝的连接。 你必须首先确保转换的速度和加速度恒定(在位置轮廓不弯曲)。
>
> 当启动凸轮时同样必须保证平滑地进入耦合阶段，该步骤的操作会在后续章节中涉及。

4.3.2 连接凸轮

ACOPOS 上的凸轮是通过使用"MC_CamIn"函数连接的。

启始位置

和"MC_GearInPos"功能块相似，在这种情况下同样可以以一个预定义的主轴和从轴位置启动连接。有两种方式定义启动凸轮期望的位置(主和从):

- 绝对,位置周期的零点启动
- 相对于当前位置

实际位置可以通过使用偏移参数定义。

> 备注:
>
> 就象我们先前学到的,可以通过在NC配置表中输入ModPos=“<period>“,“<factor>“来定义位置系统，配置一个周期的位置曲线。

在启动选项中,从轴首先运动到它位置系统中定义的启始位置。这就意味着它寻找下一个最接近的启始点:

- 在"绝对"模式下的"周位置周期的零点" + "偏移", (总是正转)
- 在"相对"模式下的"当前轴位置" + "偏移"

程序等待主轴到达它定义的启始位置，主轴的启始位置周期的返回("<period>"):

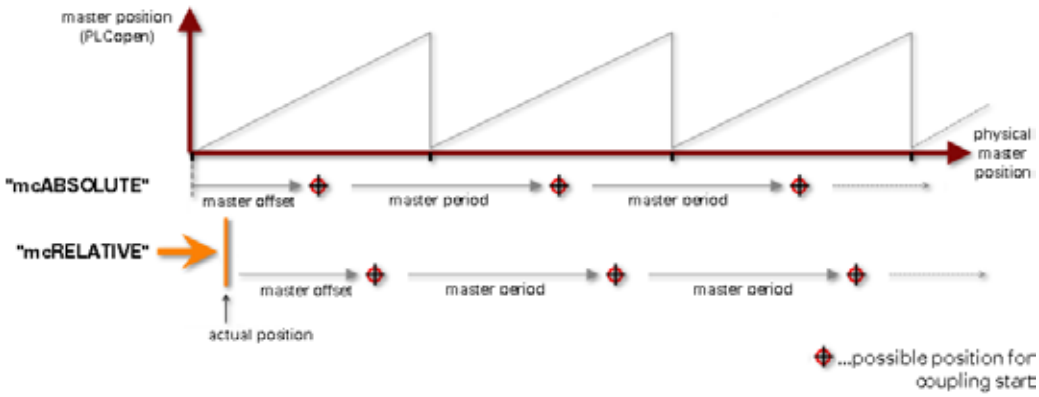


图. 24 启动连接的位置 – 主轴

在这种情况下,主轴通过连接启动位置(根据间隔)多次，直到从轴到达它启始位置后才进行连接时间。如果从轴没有达到启始位置,连接启动点切换到下一周期。

> 备注:
>
> 再次声明, 主轴不受连接步骤的影响。

拉伸凸轮

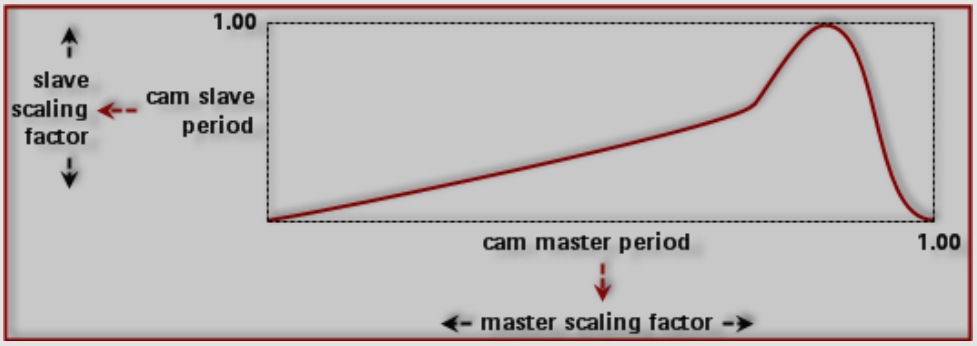
为了连接凸轮可以在主轴和从轴上拉伸，倍增系数的输入参数包括在"MC_CamIn" 功能块中。

不同的凸轮尺寸("凸轮主轴周期" / "凸轮从轴周期")也根据相应的元素扩展。

使用什么设置选项?

例子:

凸轮一般以主轴侧扩充一个单位(凸轮主轴周期 = 1)和从轴侧扩充一个单位(凸轮主轴周期 = 1)创建。这使"拉伸"凸轮满足实际过程更为简单:



打个比方,让我们假设先前提到过的作为开关凸轮功能的凸轮(主轴周期=1,从轴周期=1)需要设置来满足每次主轴旋转发生一次切换。这样,主轴侧的倍增系数需要设置成和主轴旋转数一致。

当使用线形凸轮时(相当于电子齿轮),"齿轮比"可以通过使用倍增系数得到。

电子凸轮

任务:

使用已知的方法测试先前创建的凸轮的连接，凸轮需要循环执行。
准备工作:
把函数"MC_CamTableSelect" (传输凸轮准备), 和"MC_CamIn" (启动凸轮连接)添加到从轴的循环函数部分。给函数做必要的设置 – e.g.:

```
(* Cyclic program section *)

(* Function Block Calls *)
..
...
(* Camming Functions *)
MC_CamTableSelect_0.Master:= Axis1Obj;
MC_CamTableSelect_0.Slave:= Axis2Obj;
MC_CamTableSelect_0.Periodic:= mcPeriodic;
MC_CamTableSelect_0.CamTable:= ’ profile’ ;
MC_CamTableSelect_0();
```

当函数执行后,定义的凸轮已经准备好循环处理了。
成功下载凸轮后可以使用"MC_CamIn"来连接驱动。

```
MC_CamIn_0.Master:= Axis1Obj;
MC_CamIn_0.Slave:= Axis2Obj;
MC_CamIn_0.StartMode:= mcRELATIVE;
MC_CamIn_0.CamTableID:= MC_CamTableSelect_0.CamTableID;
MC_CamIn_0();

MC_CamOut_0.Slave:= Axis2Obj;
MC_CamOut_0();
```

使用"相对"启动模式。这样,偏移值可以用来决定当前位置的实际启动点(主/从)。
使用watch窗体操作函数。

警告:

- 在先前例子程序中"MC_CamTableSelect"函数返回的凸轮ID 立即传输到"MC_CamIn"函数:

MC_CamIn.CamTableID:= MC_CamTableSelect_0.CamTableID;

这样,你需要把"MC_CamTableSelect"函数的"Execute"输入设成"TRUE"否则该值会重新回到零。
在一个"独立"的应用任务中,当多个凸轮准备时，必须确保凸轮的ID准确，连接函数必须总输入正确的ID。

- 比例值(主轴侧和从轴侧拉伸的凸轮)为了连接必须至少设成1 → factors!
  测试连接凸轮和终止凸轮连接的函数。注意轴状态!
  试着使用不同的偏移值连接凸轮。观察效果 –可以通过以一个低的主轴转速观察位置清晰的看到之间的关系。

备注:

其他使用的函数集成在ACP10_MC例子项目的"cam"任务中。该任务可以添加到项目中(使用已知的方法)来控制轴(实轴或虚轴)。结构("gAxisSlave")中提供额外的变量来操作连接函数。当命令给定启动凸轮，连接后凸轮对象传输到伺服中。当得到"CamTableID"后进行实际连接。

4.3.3 改变凸轮

可以通过执行"MC_CamIn"函数再次激活连接来改变凸轮。

当前的凸轮周期结束，并且在定义新的"CamTableID"和"Execute"输入得到上升沿信号后连接新的凸轮。第一个凸轮的结束点为第二个凸轮的启始点。主从轴偏移和启动模式都对凸轮变化不起作用。

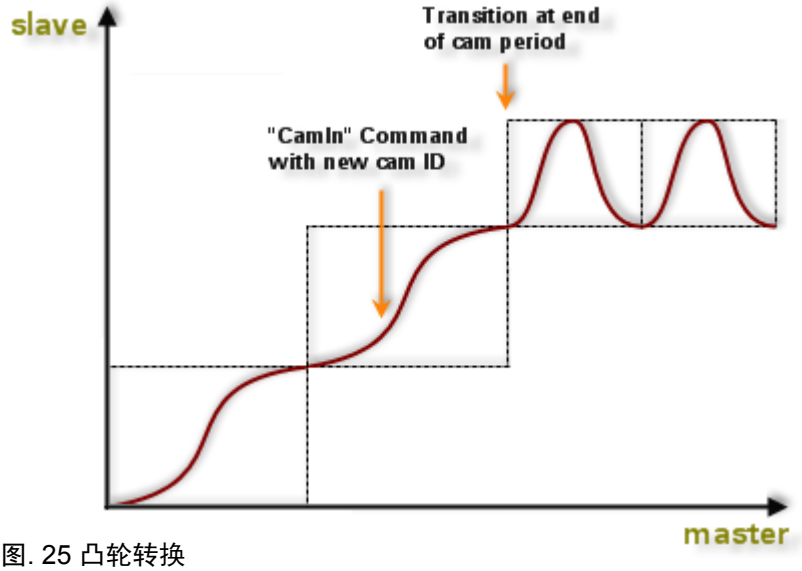


图. 25 凸轮转换

需要的凸轮可以在传输到相应的ACOPOS (一次或循环)后，以一个方便的序列连接到一起。改变凸轮的进程必须在应用程序不同的部分执行。

警告:

您必须确认在这种情况下凸轮间的转折点要稳定，以免发生曲线的弯曲。

任务: "改变凸轮"

现在我们可以在不用做任何额外设置的情况下测试该函数，按先前的例子连接凸轮。

再次执行"MC_CamIn"函数,但这次改变凸轮拉伸的值。原则上,该步骤执行方式和连接一个新的凸轮是一样的。

5、凸轮跟随

凸轮跟随是一个有参数设置的一个连续的机构，它提供给电子凸轮建立有效的连接，补偿功能和事件处理。

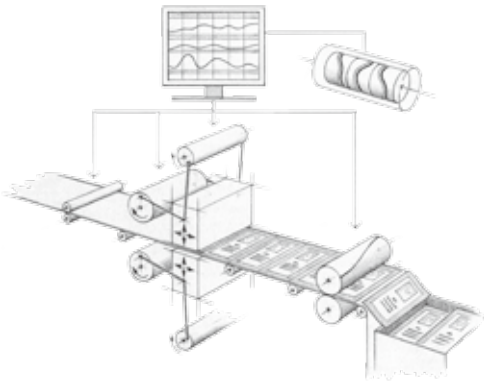


图. 26 凸轮跟随

凸轮跟随在相应的ACOPOS 从轴驱动初始化(设置参数)并独立运行，可以快速反映灵活处理，同样可以有许多可能性在运行进程中介入。

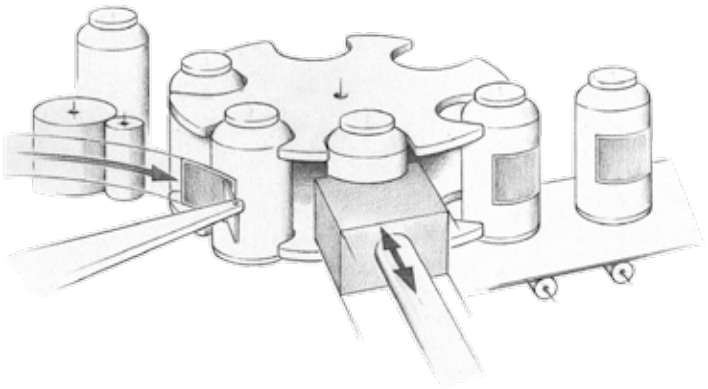


图. 27 贴标

例如贴标机，产品传输机作为主轴，从轴驱动一个带有粘性标签的带子，跟随必须保证标签以正确位置正确的速度贴在产品上，使用一个高速数字输入检测到产品就开始贴标。(参考ACP_10例子项目,"automat"任务)。

在以下章节,我们将一步步涉及结构以及如何操作凸轮跟随。

5.1 结构和功能

不同的凸轮可以以相似的方法排列并连接到步进顺序器的步骤中，我们称为跟随状态:

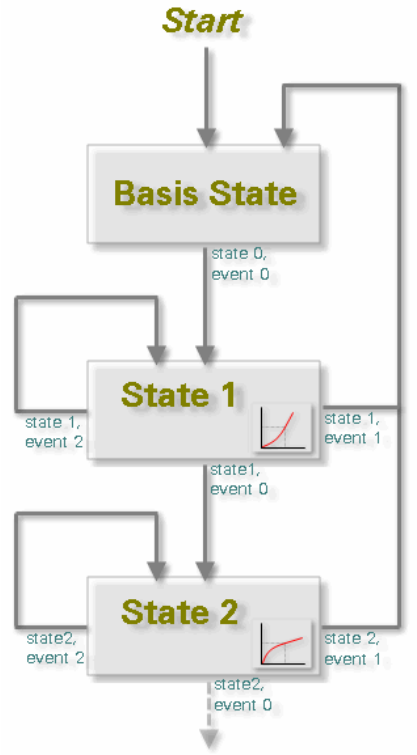


图. 28 结构图 – 凸轮跟随

5.1.1 跟随状态

每个跟随状态可以定义以下元素:

- 凸轮，使用前凸轮必须传送到ACOPOS 中，然后凸轮可以在任意状态中使用。
- 补偿齿轮是一个自动计算的曲线，它补偿状态切换时或凸轮的连续连接时的位置差，这里有许多种的变化。

曲线特性已经预先定义在一个状态中:

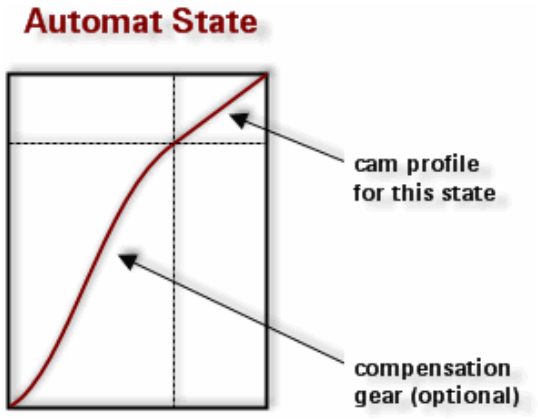


图. 29 跟随状态

状态从左到右运行，补偿齿轮执行一段补偿运动(e.g. 位置补偿)，然后切换到这个状态中的凸轮。

> 备注:
>
> 可以屏蔽补偿齿轮。如果这样做了,状态只包括凸轮。
>
> 如果在跟随状态中应用到了补偿齿轮,它将会在状态中相应的凸轮前执行。

每个状态必须定义切换事件来建立跟随状态的连续进程。

5.1.2 切换事件

切换事件可以用来定义多个事件来在每个独立的状态中触发状态，每个事件需要以下参数:

- 事件类型：决定什么时间触发状态改变，可以是一个"外部"触发信号或当前凸轮的终点, etc.
- 事件属性：决定何时执行状态改变，这意味着实际状态改变可以通过使用触发作为事件切换放在凸轮的结束,根据凸轮特性中的环境发生。
- 切换的目标状态(i.e.下一个要激活的状态)：同样可以选择重复当前状态。

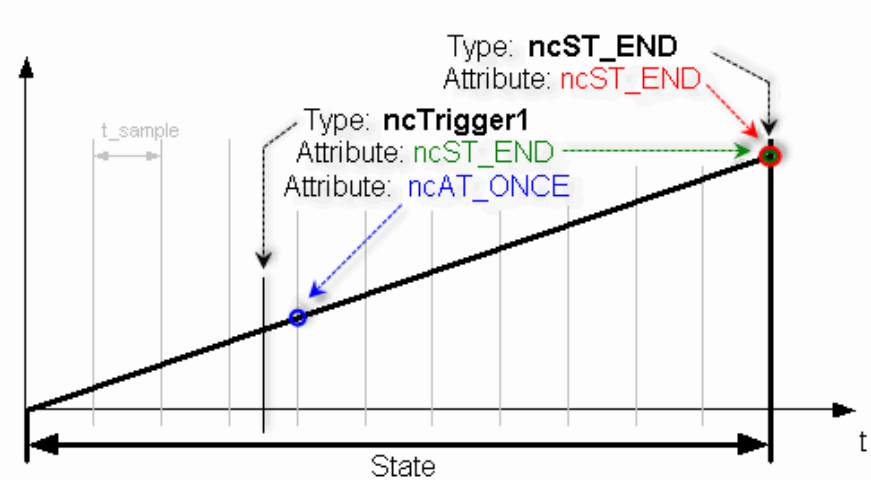


图. 30 不同时间和属性的动作点

上图显示关于事件切换中事件属性如何工作，从左到右显示一个线形曲线特性。让我们假设先前定义的一个改变事件"Trigger1"在此状态中发生，使用事件属性"At Once"来立即改变到预先定义的状态(考虑采样周期)。结果,把接下来的曲线特性确切的放到实际触发时间的位置(触发信号由累加的定时器决定)。这样,由于采样周期定位进程不会发生错误。

当使用"State End"属性时,在当前凸轮周期结束发生改变。

当使用"State End"属性时, 改变在当前凸轮周期结束发生，可以在这里定义后续的状态。

> 备注:
>
> 当创建特殊形式的状态时，基本状态是启始点，该状态的状态索引号为0，该状态没有补偿(补偿齿轮)和凸轮。
>
> 基本状态只需要定义期望的改变事件。在某种程度上,它作为等待步骤。
>
> 基本状态可以用来执行e.g.一段指定的同步。同步信号可以很快的作为改变事件定义(e.g.: 触发, etc.).

这就是现在我们如何使用跟随设置来得到指定的连续的凸轮进程，根据主轴定位从轴，有很多方式来连接每个独立的状态。

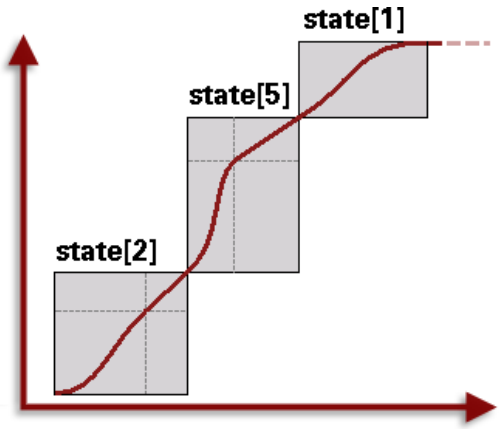


图. 31 跟随状态的进程

在跟随参数设定好后,跟随就可以在任意状态启动并根据定义的切换事件和后续状态在独立状态间跳转。

为了使我们对凸轮跟随的实际使用有一个大致的概念,让我们看一下下面这个例子:

印标纠偏

需要做什么?

有一个带有印刷筒的传送带(从轴)以恒定速度运行，我们的目的是总是准确的"击中"传送带上的循环印标。由于圆筒的圆周速率和传送带的速度差异总会引起背离，如果该值超过了一个指定的值,于是产生信号(触发)并执行纠偏。

下图显示使用凸轮跟随针对该任务的一个解决方案:

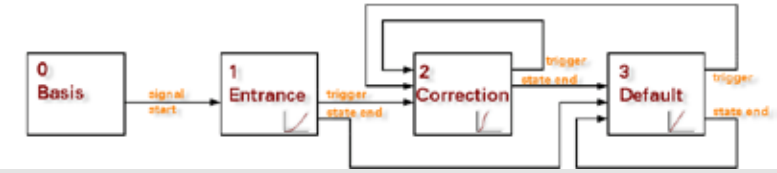


图. 32 印刷标记纠偏流程图

步骤:

从基本状态启动(等待信号，启动连接)， 系统通过进入状态(->进入曲线)按一定规则进入位置操作,传送带启动。根据不同情况,当得到触发信号("超过背离容许值")后，在后续步骤中执行一个纠偏的过程(纠正从轴路径的曲线)。如果在没有任何触发信号的情况下，凸轮结束后状态总是回到标准步骤("一般"定位)。

可以立即使用凸轮跟随来解决该任务，就象我们已经看到的一样,同样可以"从外部"介入凸轮跟随进程，来实现在跟随运行时灵活的过程调节。

5.2 凸轮跟随的准备

ACP10_MC 库同样提供函数来初始化(设定参数,创建) ACOPOS上的凸轮跟随，不同的功能块和我们先前用过函数使用方法是几乎一样的。

在以下章节我们将涉及如何使用这些功能块，除了Automation Studio 在线帮助,同样会提供不同凸轮跟随变化和设置可能的不同观念。

5.2.1 下载凸轮

我们在跟随状态中使用凸轮前，首先必须通过"MC_BR_DownloadCamProfObj"函数传送到ACOPOS 驱动(从轴)中。

凸轮按一个指定的索引号存储在ACOPOS 中，不同跟随状态的凸轮可以通过索引号选择。

> 警告:
>
> 当定义状态时，跟随状态选择凸轮轨迹。所以这一步只有在ACOPOS 上有相应的索引号时才能进行。

5.2.2 定义状态

"MC_BR_InitAutState"函数用来设定跟随状态的参数。

使用索引号来指定处理的状态，索引号1到14对应此功能块可以对应的"常规"跟随状态。除标准参数以外,例如主轴和从轴,状态选择凸轮和已经定义的主轴和从轴凸轮拉伸一样。

就象我们先前看到的,可以使用补偿齿轮来进入一个状态。"MC_BR_InitAutState"功能块提供需要的参数。

补偿齿轮是一个自动计算的曲线，它补偿状态切换时或凸轮的连续连接时的位置差。

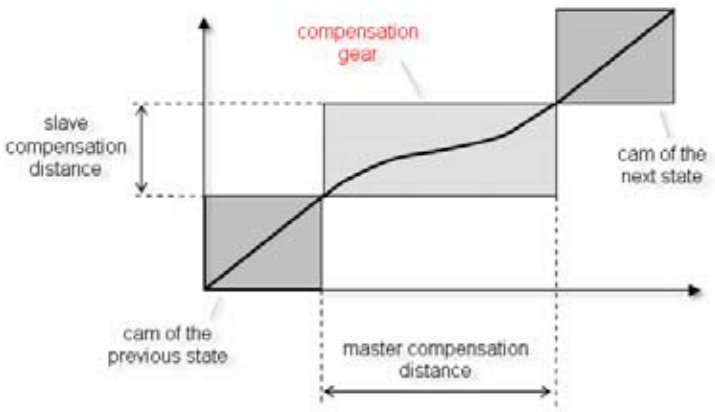


图. 33 补偿齿轮

上图显示两个连续状态(凸轮)间的一段补偿，该补偿属于后一个状态(右边的凸轮)，设置的参数用来计算和执行。

有些定义补偿的基本参数:

- 补偿模式
- 主轴补偿路径
- 从轴补偿路径

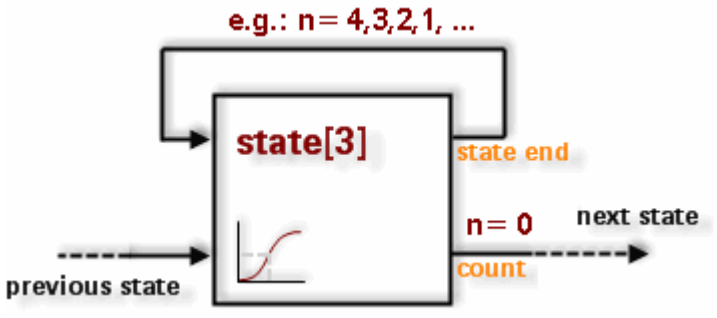
不同的补偿齿轮模式提供不同的补偿路径和补偿速度。

> 备注:
>
> 补偿路径根据特定的变化指定，e.g.基于两个靠近的凸轮的中心。这样使我们能够更简单的处理特定应用。
>
> "桥接"凸轮之间定义的"闲置相位"(指定的距离)时，补偿齿轮的效率非常地高。

提供附加的参数来高级设置补偿齿轮(不同的极限值, etc.)。在状态初始化中这些参数的应用可以屏蔽。所以,并不需要过多考虑这些参数。

> 备注:
>
> 关于补偿齿轮的详细信息可以在Automation Studio 在线帮助中找到。

此外，可以通过使用"MC_BR_InitAutState"来定义一个特殊的改变事件，"计数"("ncCOUNT")，该事件可以用来退出循环往复的状态。原则上,这意味着一个预先定义的值("RepeatCounterInit") 在每次状态退出时计数减一次，如果计数器到零了,那就触发"count"事件，切换到另一个状态。



除了这次切换事件,需要一些状态来用于这类功能。

5.2.3 定义切换事件

"MC_BR_InitAutEvent"函数用来定义跟随状态的切换时间并有效的连接(状态的进程)。

每15种跟随状态(索引号 0 to 14)可以定义5种改变事件，相应的索引号在功能块指定。

这些是基本参数:

- 事件类型
- 事件属性("动作点")
- 后续状态的索引号

在凸轮跟随中用何种时间来切换状态?

- "start" ("ncS_START")事件可以用来同步电机连接基本状态，必须为周期往复的事件，定义一个主轴的启动位置和位置间隔。当主轴通过相应的位置(或间隔往复)时产生该事件。
- "count"事件(参考先前)。
- "trigger"事件("ncTRIGGER1" 或 "ncTRIGGER2")能够对外部传感器的硬件信号起反映。
- "ParID"事件("ncPAR_ID")可以评估ACOPOS参数ID，切换事件按相应的级别生成。
- 应用程序可以通过使用"signal"事件("ncSIGNAL1/2/3 或 4")来生成一个改变(参考后面)。
- 此外,同样可以从两个前面提到的事件通过逻辑"and"操作生成切换事件。

当切换事件发生时"at once" ("ncAT_ONCE")或"state end" ("ncST_END")属性可以定义成状态切换的"动作点"。

> 备注:
>
> 当离开状态时，可以用"action"参数来创建"重要的事件"。该属性用来当凸轮跟随运行时(在激活运行时)，改变跟随结构或改变凸轮数据。传输后,改变的数据通过这种类型的事件同步的应用。
>
> 可以在Automation Studio在线帮助中找到详细信息(切换事件,跟随的在线参数修改, etc)。

就象你看到的一样,有大量的可能性会影响跟随进程的处理。

5.2.4 定义全局变量

使用"全局变量"也就是说该设定原则上对所有跟随状态都有效。

这些设定包括:

- 主轴, 从轴和驱动连接参数(主轴信号, e.g.: 主轴的设置或实际位置, etc.)
- 设置"start"切换事件(→可选)
- 为"ParID"切换事件定义事件参数(→ 可选)
- 设置"direct start" (→ 可选)
- 附加轴和动态调节曲线的参数(→ 可选)

"MC_BR_InitAutPar"功能块用来定义相应的参数。

默认,指定的主轴(或指定的驱动连接参数"MasterParID")使用于跟随中的所有状态，"start"切换事件(参考先前)的设定和期望的"ParID"切换事件参数都在这里选择。

为凸轮跟随启动指定期望的启动状态，同样可以在凸轮中启动("direct start"):

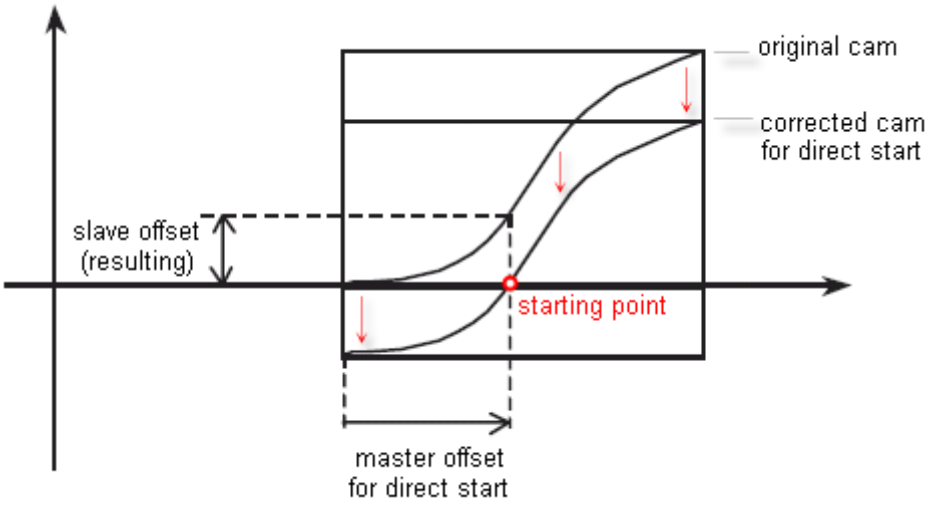


图. 34 直接启动演示

如上图所示,曲线被纠正后从轴位置位于相应点等待直接启动，驱动连接以及激活。

此外,同样可以指定"附加轴"参数，附加主轴如何工作已经在章节"3.3. 动态状态切换"中阐述过。

对于凸轮跟随来说,同样可以打开一个从轴元素("附加从轴"):

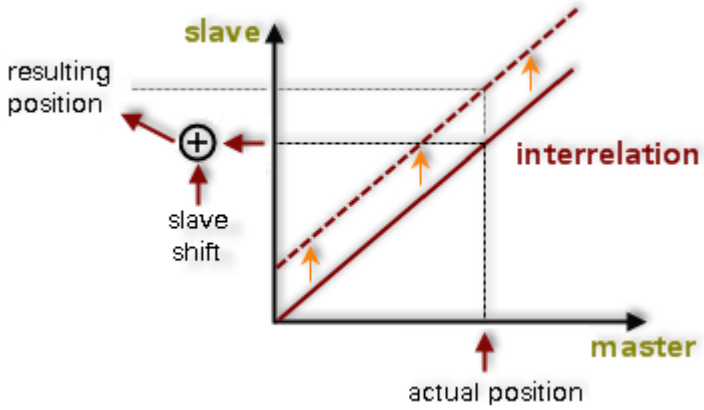


图. 35 主轴 – 驱动连接关系 – 附加轴 – 从轴

如上图所示,有驱动连接产生一段附加值添加到从轴侧位置，这样附加从轴在垂直方向偏移了凸轮轨迹，两轴相对应位置也同样改变。

提供更多的参数用来定义ParID进行动态调节曲线，这些设定提供动态拉伸凸轮以及特殊补偿齿轮应用的可能性。

指定参数的值构成出其指定的不同元素的数量。

> 备注:
>
> 主轴的最大速度("MaxMasterVelocity")用来计算补偿齿轮，从曲线特性得到的从轴速度在任何时候都不能超过该定义值。

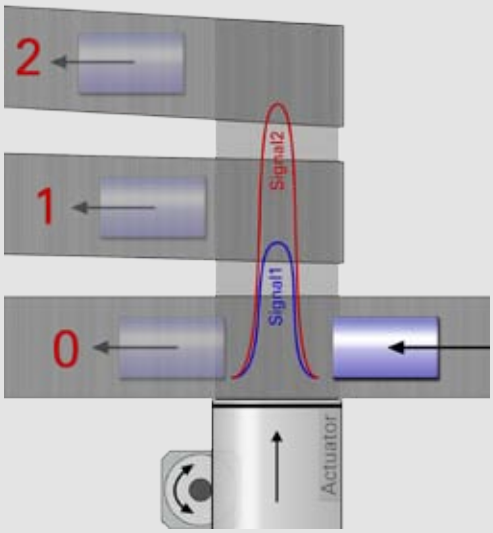
凸轮跟随

任务: "设定凸轮跟随的参数"

在这个例子中,我们将使用凸轮跟随执行一个简单的任务。

默认:

产品按一个正常的进程步骤在传送带上传输并到达了一个交叉点。这部分产品必须分成三部分以被后续加工。当他们通过滑动送料时从传送带上"推"离到相应的位置。滑动送料状态的驱动由ACOPOS 控制。

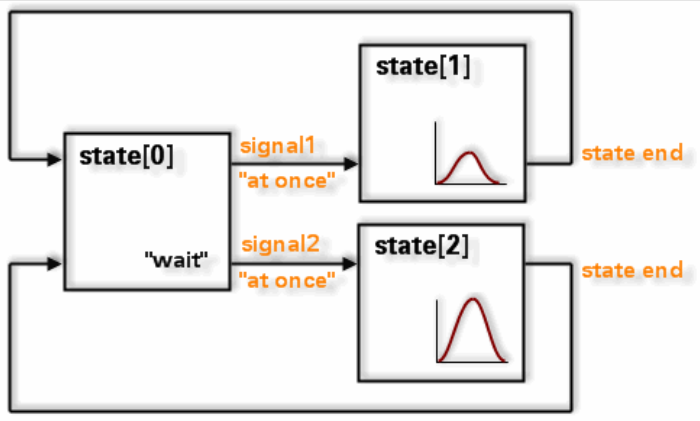


根据先前通过的处理部分,滑动送料装置必须在相应的线上执行向前或向后运动，三个通道中的一部分允许通过缺口，不需要一侧位置偏移运动。

该运动必须在传送带移动单位特定距离执行，这样,它必须通过凸轮连接滑动送料驱动的传送带移动单位来执行。基于我们的目的,我们可以创建一个合适的凸轮并根据必要的位置偏移拉伸。

处理过程中的信号指定，需要一个切换。为了测试目的,我们将使用"Signal1"和 "Signal2"事件,事件由程序产生。当信号到达时,必须立即执行相应的偏移运动。

下图显示凸轮跟随的一个可能进程:



基本状态可以用来等待信号(Index 0)。作为一个选择,可以选择一个水平的曲线(斜率 0)或可以选择值 0 作为从轴拉伸值。在任何这些情况下从轴不执行运动。

当得到一个信号后立即切换到相应的状态。两个状态使用相同的凸轮。可以通过从轴侧拉伸凸轮来定义偏移运动的范围。这样, 必须按比例决定每个凸轮尤其是状态2。如果到达了凸轮的结束(并启动点)，到达后切换回基本状态。

我们现在可以使用先前阐述过的功能块创建结构。

设置跟随参数:

可以很简单的测试定义跟随参数功能块的使用，使用循环任务("功能块调用")来定义固定设定。这些函数后面可以使用Watch窗口连续的激活。以下摘录的程序显示相应进程的例子:

```
(* Cyclic program section *)
(* Function block calls *)
(* Download Cam Profile *)
MC_BR_DownloadCamProfileObj_0.Axis:= Axis2Obj;
MC_BR_DownloadCamProfileObj_0.DataObjectName:= ‘profile’;
MC_BR_DownloadCamProfileObj_0.Index:= 1;
MC_BR_DownloadCamProfileObj_0();

(* Cam Automat Definition *)
        (* Global Automat Parameters *)
        MC_BR_InitAutPar_0.Master:= Axis1Obj;
        MC_BR_InitAutPar_0.Slave:= Axis2Obj;
        MC_BR_InitAutPar_0();

        (* State 0 *)
                ...                     (* "InitAutState" not necessary *)
        (* State 0, Event 0 *)
        MC_BR_InitAutEvent_00.Slave:= Axis2Obj;
        MC_BR_InitAutEvent_00.StateIndex:= 0;
        MC_BR_InitAutEvent_00.EventIndex:= 0;
        MC_BR_InitAutEvent_00.Type:= ncSIGNAL1;
        MC_BR_InitAutEvent_00.Attribute:= ncAT_ONCE;
        MC_BR_InitAutEvent_00.NextState:= 1;

MC_BR_InitAutEvent_00();
        (* State 0, Event 1 *)
        MC_BR_InitAutEvent_00.Slave:= Axis2Obj;
        MC_BR_InitAutEvent_00.StateIndex:= 0;
        MC_BR_InitAutEvent_00.EventIndex:= 1;
        MC_BR_InitAutEvent_00.Type:= ncSIGNAL2;
        MC_BR_InitAutEvent_00.Attribute:= ncAT_ONCE;
        MC_BR_InitAutEvent_00.NextState:= 2;

MC_BR_InitAutEvent_00();
        (* State 1 *)
        MC_BR_InitAutState_1.Master:= Axis1Obj;
        MC_BR_InitAutState_1.Slave:= Axis2Obj;
        MC_BR_InitAutState_1.StateIndex:= 1;
        MC_BR_InitAutState_1.CamProfileIndex:= 1;
        MC_BR_InitAutState_1.MasterFactor:= 1;
        MC_BR_InitAutState_1.SlaveFactor:= 1;
        MC_BR_InitAutState_1.CompMode:= ncOFF;
        MC_BR_InitAutState_1();
        (* State 1, Event 0 *)
                ...
```

使用缺少的进程(为状态2改变的"SlaveFactor")，来完成任务的准备。
"MC_BR_AutControl"功能块同样也要添加，该函数用来控制(启动,停止,etc.)凸轮跟随。
(* Automat Control *)

MC_BR_AutControl_0.Slave:= Axis2Obj;
MC_BR_AutControl_0.Enable:= 1;
MC_BR_AutControl_0();
备注: 为了增加程序的结构化和清晰度,推荐使用状态索引号并且功能块中使用实例名称来设置事件:
- State[7] Change event[3]:
MC_BR_InitAutEvent_73.Slave:= ...

执行以上指定的任务并传送改变的部分，每个函数可以在Watch 窗口启动，在运行时注意检测显示的状态。
备注:
跟随参数的顺序是没有关系的，当设定状态的参数时，只允许下载凸轮。
在后续章节一些简单描述如何激活跟随后，我们马上就可以测试我们功能。

备注:

在以上例子中,事件"Signal1/2"用来仿真的状态切换，可以在应用任务中使用"MC_BR_AutControl"函数来生成。在这里可以运用"触发"事件(硬件触发)做一个实际的应用，这些直接在ACOPOS上计算，这样可以达到最小响应时间。

5.3 控制凸轮跟随

可以使用"MC_BR_AutControl"功能块来控制凸轮跟随。原则上,当凸轮传送并跟随初始化后就可以启动系统了。包括的驱动必须在启动跟随前相应的准备好(i.e. 初始化完成并寻参完成)。

"MC_BR_AutControl"功能块可以通过"Enable"输入激活。可以通过剩下的控制器输入给定大量的凸轮跟随命令。这些命令包括:

- 跟随激活:启动,停止和重起
- 控制切换事件"信号1/2/3/4"
- 使用可选的应用数据结构有选择性的初始化跟随
- 锁定一致的在线参数改变

5.3.1 跟随激活

使用"Start"命令激活凸轮，默认情况下,以基本状态启动处理(索引号0)。否则,启动根据由"MC_BR_InitAutPar"影响的设定进行直接启动(参考章节"5.2.4. 定义全局参数")。

"stop"命令结束凸轮跟随处理，取消驱动连接。在轴定位进程中要避免抖动。("平滑"的进入停止,在曲线结束时以0斜率曲线推出, etc.).

> 备注:
>
> 在任意时刻可以通过停止从轴运动来停止跟随操作，可以切换到状态255来退出跟随。

在失败后可以通过执行"重起"命令把从轴重新回到跟随操作。

是如何工作的呢?

发生错误，对从轴执行定位命令或切换到一个独立运动会导致从轴"退出"驱动连接。

当跟随启动一次后,直到从轴退出前一直进行处理。在跟随"等待"模式下完成计算事件状态处理。

这使从轴在其实际位置重新连接成为了可能。

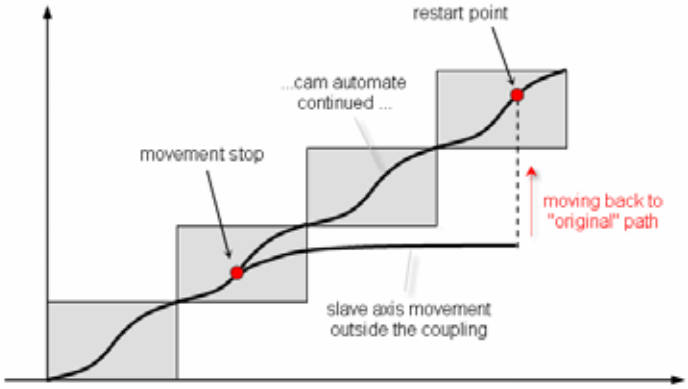


图. 36 从轴出错后重起

5.3.2 控制改变事件"信号1/2/3/4"

切换事件"信号1/2/3/4"可以通过设定相应的输入激活。如果该类型事件一度被用做状态切换后内部就会复位信号。事件在再次使用前信号必须被重新设置。

任务: "激活凸轮跟随"

使用已知的方法是连接轴准备好运动。

"basic"任务中的慢速进程可以用来作为主轴的匀速运动，使用命令启动先前设置好的凸轮跟随，激活状态切换信号。

注意要观察驱动状态。

凸轮跟随

备注:

ACP10_MC 例子项目中的"automat"任务显示在完整进程框架下设置和控制凸轮跟随的功能块应用。所有驱动准备和处理错误的进程(同"basic"一样)都包括在程序进程中，提供相似的控制结构来操作进程。

在"automat"任务中启动跟随创建"迷你标签机器"功能 – 参考Automation Studio在线帮助 → Cam Profile Automat → Examples.

5.3.3 跟随可选择的初始化

"MC_BR_AutControl"功能块提供一个选择来设置和初始化凸轮跟随，于是专门提供数据类型"MC_AUTDATA_TYP"，该数据结构包括所有凸轮跟随参数。

可以在应用程序中创建一个该类型变量并用来设置跟随数据 – e.g.:

可以在应用程序中创建一个该类型变量并用来设置跟随数据 – e.g.:

```
aut_data.Master:= Axis1Obj;
aut_data.StartPosition:= 1000;
...
aut_data.state[0].event[0].Type:=
ncSignal;
aut_data.state[0].event[0].
NextSate:= 1;
...
aut_data.state[1].CamProfileIndex:=
1;
aut_data.state[1].MasterFactor:=
1000;
aut_data.state[1].SlaveFactor:=
2000;
...
aut_data.state[1].event[0].
Type:=ncST_END
...
```

数据结构的地址必须传输到"MC_BR_AutControl"函数中来初始化跟随参数("AdrAutData")。于是可以使用"InitAutData"函数输入启动进程。

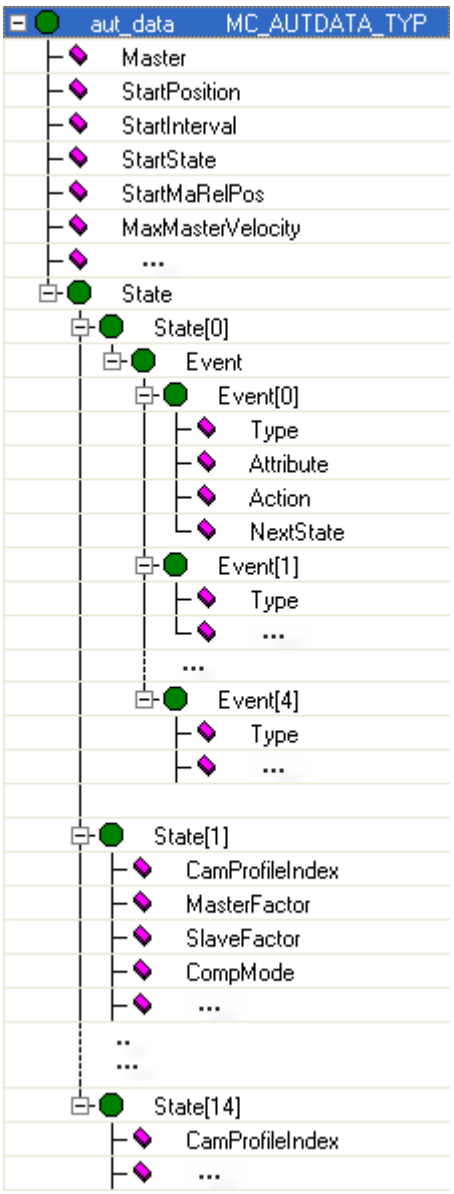


图. 37 示意跟随数据结构

5.3.4 在线参数修改

在凸轮跟随操作时，可以改变特定参数和凸轮数据。这就意味着新计算出的凸轮和特定凸轮参数(补偿路径,倍增系数, etc.)的改变可以在运行时执行。

> 备注:
>
> 除参数外它还决定跟随的基本结构(跟随状态的行进)。

可以使用一个锁定机构("ParLock")抑制改变的参数传输:

可以通过锁机制("ParLock")来抑制传输改变的数据:

同步传输修改的数据:

在传输修改的数据前(新的参数值, etc.),已经通过"ParLock"命令把直接传输跟随操作值锁定了。可以传输新值, e.g. 特定状态的修改参数。在"ParLock"撤回后，修改就同步的初始化。当运行时需要注意以下凸轮功能:

- 当状态进入下一个周期时特定跟随状态的修改设定以及生效。
- 此外,同样可以只在选择的状态转换("重大事件")时执行此基本(同步)修改数据初始化。

那种方式的凸轮跟随设定正好可以在一个ACOPOS 循环中准备和激活。

> 备注:
>
> 该进程的相信信息可以在Automation Studio 在线帮助中找到。

6、小结

ACP10_MC库提供大量的函数来连接轴对象，不同的功能块的设计是基于PLCopen 运动控制 标准并基于功能应用形成统一的设计。

使用"电子齿轮"可以执行线形位置连接,如果需要甚至可以用一个预先定义的启动点。

同样提供简单功能通过电子凸轮来动态位置连接。这样,可以用应用程序来控制不同位置轮廓的连接。

Automation Studio 提供凸轮编译器来创建凸轮轨迹。多种设定是调节凸轮处理变的十分简单。

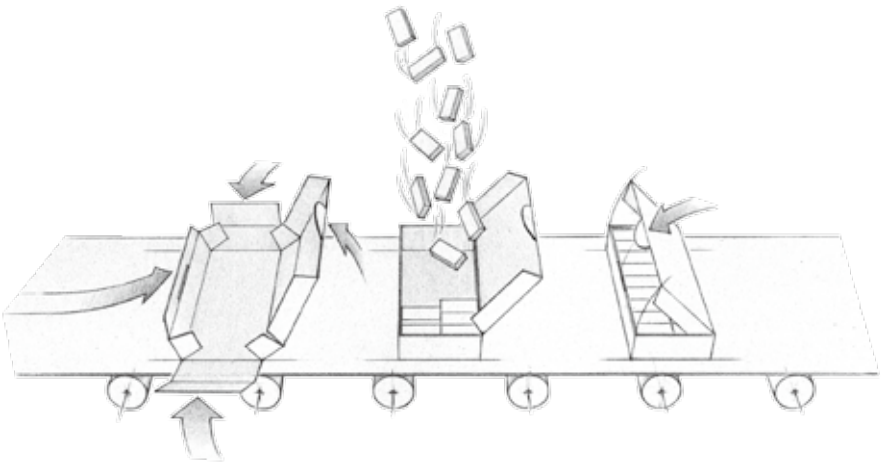


图. 38 纸盒成型

ACOPOS凸轮编译器是一个有效连接定位轮廓十分强有力的工具。需要的进程完全通过定义完成，可以使用清晰结构化的函数来完成跟随结构的初始化和跟随模式的控制。当凸轮跟随启动后,定义的进程完全独立的在ACOPOS 中处理。这样介绍应用程序的负载并提供一个快速事件控制的定位进程。

ACP10_MC多轴函数服从运动控制状态流程中状态的影响。这里用户可以得到必要的信息来设计进程。

ACP10_MC例子项目包括单独连接应用的例子任务，并可以用来作为创建完整定位应用的模版。

7、附录

运动控制基本函数(ACP10_MC):

驱动准备:

MC_Power

MC_Home

MC_BR_BrakeOperation

基本运动:

MC_MoveAbsolute

MC_MoveAdditive

MC_MoveVelocity

MC_BR_MoveAbsoluteTriggStop

MC_BR_MoveAdditiveTriggStop

MC_BR_MoveVelocityTriggStop

MC_Stop

MC_Halt

MC_SetOverride

读取驱动的状态：

MC_ReadStatus

MC_ReadActualPosition

MC_ReadActualVelocity

MC_ReadActualTorque

读取和确认驱动的错误：

MC_ReadAxisError

MC_Reset

开关量输入/输出信号：

MC_ReadDigitalInput

MC_ReadDigitalOutput

MC_WriteDigitalOutput

MC_DigitalCamSwitch

位置测量：

MC_TouchProbe

MC_AbortTrigger

Motion Control Multi-axis Functions (ACP10_MC):

电子齿轮:

MC_GearIn

MC_GearInPos

MC_GearOut

MC_Phasing

电子凸轮:

MC_CamTableSelect

MC_CamIn

MC_CamOut

凸轮跟随:

MC_BR_DownloadCamProfObj

MC_BR_InitAutPar

MC_BR_InitAutState

MC_BR_InitAutEvent

MC_BR_AutControl

Notes

附录

Notes

Notes

附录

培训模块综述

TM200 – 贝加莱B&R 公司介绍**
TM201 – 贝加莱B&R 产品系列**
TM210 – Automation Studio™ 基础
TM211 – Automation Studio™ 在线通信
TM212 – 自动化对象（Target）**
TM213 – 自动化运行（Runtime）系统
TM220 – 维护信息*
TM221 – 自动化组件和出错信息查询*
TM223 – Automation Studio™ 诊断
TM230 – 结构化软件编程
TM231 – 面向机器设备的Automation Studio™ *
TM240 – 梯形图(LAD)
TM241 – 功能块图 (FBD)*
TM242 – 连续功能图 (CFC)*
TM243 – 顺序功能图 (SFC)*
TM245 – 指令表 (IL)*
TM246 – 结构文本 (ST)
TM247 – Automation Basic (AB)*
TM248 – ANSI C
TM250 – 内存管理和数据存贮
TM260 – Automation Studio™ 函数库I
TM261 – Automation Studio™ 函数库 II*
TM264 – 定时处理单元（TPU）*

TM400 – 运动控制的基础
TM401 – 贝加莱B&R 运动控制产品**
TM402 – 运动控制系统的计算*
TM410 – ASiM 的基础
TM440 – ASiM的基本功能
TM441 – ASiM多轴运动功能
TM445 – ACOPOS ACP10 软件
TM446 – 电子凸轮*
TM447 – ACOPOS 智能过程技术（SPT）*
TM450 – ACOPOS 控制理念和控制器设置
TM460 – 启动B&R 电机*
TM461 – 启动第三方电机*
TM470 – CNC*

TM600 – 图文显示的基础
TM601 – 贝加莱人机界面产品**
TM610 – ASiV 的基础
TM620 –ASiV 的维护*
TM630 – 图文显示的编程规则
TM640 –ASiV报警系统
TM650 –ASiV的国际化操作
TM660 –ASiV 的远程操作
TM670 –ASiV 高级应用

TM700 – Automation Net PVI
TM701 – PVI 通信*
TM710 – PVI DLL 编程
TM711 – PVI的服务
TM712 – PVIControl.NET
TM720 – PVI 维护和诊断*
TM730 – PVI OPC

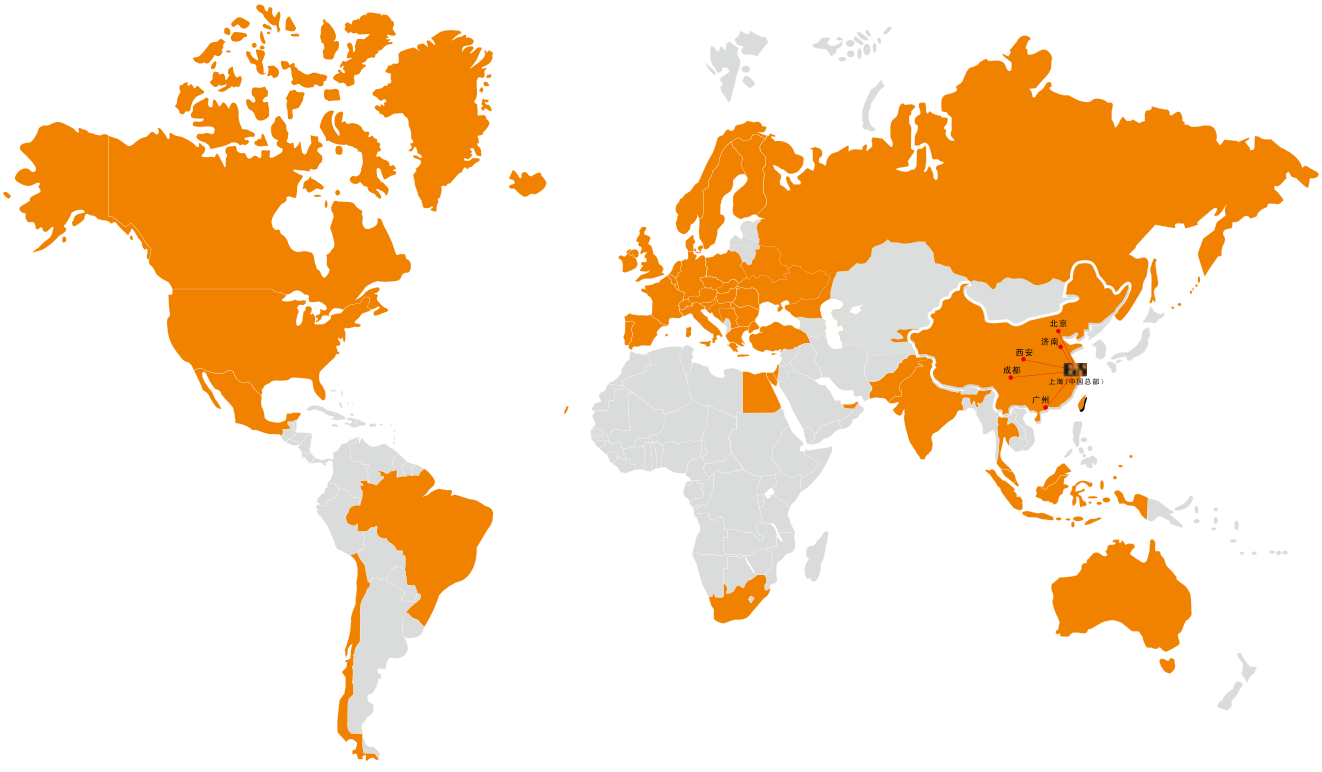
TM800 – APROL 系统概念
TM801 – APROL 工程设计基础
TM810 – APROL 安装，配置和恢复*
TM811 – APROL运行(Runtime)系统*
TM812 – APROL 操作员管理
TM813 – APROL XML 查询*
TM814 – APROL 审计追踪*
TM820 – APROL 维护*
TM830 – APROL 项目工程设计
TM840 – APROL 参数管理和配方
TM850 – APROL 控制器配置和INA 通讯
TM860 – APROL 库设计
TM861 – APROL 通讯互联*
TM865 – APROL 库指导手册
TM870 – APROL Python编程*
TM880 – APROL 报表*

** 查看产品目录
* 即将出版

全球总部
Bernecker+Rainer Industrie-Elektronik Ges.m.b.H.
B&R Straße 1
A-5142 Eggelsberg　奥地利
Tel.：+43(0)7748/6586-0
Fax：+43(0)7748/6586-26
info@br-automation.com
www.br-automation.com

中国总部
贝加莱工业自动化（上海）有限公司
上海市漕宝路70号光大会展中心C座16楼
Tel.：+86/(0)21/6432 6000
Fax：+86/(0)21/6432 6108
info.cn@br-automation.com
www.br-automation.cn

全球50多个国家超过120个分支机构 **www.br-automation.com/contact**



中国总部　● 中国办事处

Austria · Australia · Belgium · Belarus · Brazil · Bulgaria · Canada · Chile · China · Croatia · Cyprus · Czech Republic
Denmark · Egypt · Emirates · Finland · France · Germany · Greece · Hungary · India · Indonesia · Ireland · Israel · Italy · Korea
Kyrgyzstan · Malaysia · Mexico · The Netherlands · Norway · Pakistan · Poland · Portugal · Romania · Russia · Singapore
Slovakia · Slovenia · South Africa · Spain · Sweden · Switzerland · Thailand · Turkey · Ukraine · United Kingdom · USA